TECHNISCHE UNIVERSITÄT DRESDEN

Technische Universität Dresden
Institute for Theoretical Computer Science
Chair for Automata Theory

LTCS–Report

Answering Regular Path Queries Under Approximate Semantics in Lightweight Description Logics

Oliver Fernández Gil Anni-Yasmin Turhan

LTCS-Report 20-05

Postal Address:
Lehrstuhl für Automatentheorie
Institut für Theoretische Informatik
TU Dresden
01062 Dresden

http://lat.inf.tu-dresden.de

Visiting Address:
Nöthnitzer Str. 46
Dresden

Answering Regular Path Queries Under Approximate Semantics in Lightweight Description Logics

Oliver Fernández Gil*[1] and Anni-Yasmin Turhan[1]

[1]Theoretical Computer Science, TU Dresden, Germany

Abstract

Classical regular path queries (RPQs) can be too restrictive for some applications and answering such queries under approximate semantics to relax the query is desirable. While for answering regular path queries over graph databases under approximate semantics algorithms are available, such algorithms are scarce for the ontology-mediated setting. In this paper we extend an approach for answering RPQs over graph databases that uses weighted transducers to approximate paths from the query in two ways. The first extension is to answering approximate *conjunctive* 2-way regular path queries (C2RPQs) over graph databases and the second is to answering C2RPQs over $\mathcal{ELH}$ and DL-Lite$_\mathcal{R}$ ontologies. We provide results on the computational complexity of the underlying reasoning problems and devise approximate query answering algorithms.

*Supported by DFG grant BA 1122/20-1.

Contents

1 Introduction

Regular path queries (RPQs) is a well-investigated query language that dates back to the early 90's, where its capabilities to navigate graph-structured data attracted much attention in research on semistructured data and graph databases [MW95, FLS98]. This interest was revived in recent years, since in many application areas data is graph-structured and represented in graph database models. Notable examples of applications of RPQs are querying biological networks, the semantic web and social networks. Moreover, RPQs and its extensions are part of SPARQL, which is the standard language recommended by the W3C to query RDF data. Formally, a graph database consists of a labeled directed graph, where edge labels correspond to binary predicates stating relations between data items. A RPQ consists of a regular language over these labels, and retrieves pairs of data items (a, b) that are connected by paths complying to the specified regular language. The extension of *two-way* RPQs (2RPQ) allows to traverse edges backwards, and the more expressive language of *conjunctive* 2RPQs (C2RPQ) allows conjunctions of 2RPQs that can share variables.

In scenarios where a RPQ yields no answers over a particular database, it can be fruitful to relax the query to retrieve more than the classical answers, i.e., pairs that are connected by paths that are "close enough" or that approximate the paths required by the query. This can be useful to provide feasible alternatives in applications where data is gathered automatically from heterogeneous data sources and exact semantics need not yield the expected results. Similarly,

in applications where the data is irregular and evolves in structure and content, it can be hard for users to have full knowledge of its vocabulary and structure. In this situation, queries that approximate/relax the set of answers may be helpful.

Several approaches have been considered to address this problem, as for instance, [JMM95, KS01, GT06, PSW16]. In particular, [GT06] proposes an elegant and tractable solution that uses a weighted finite-state transducers to define the approximation semantics. Roughly speaking, such a transducer is a mechanism that transforms input words into corresponding output words, and computes a weight quantifying the cost of the transformation. The idea is to use a transducer as a means to specify which paths are allowed to be considered approximations/distortions of the "ideal" paths specified by the query, and to specify their distortion costs. Approximate answers are then tuples (a, b, η), where η is the minimal cost of distorting a path that is complying with the query into a path leading from a to b.

Path queries have also been investigated for ontology-mediated query answering (OMQA), in which semantic knowledge provided in a background ontology is used to enrich the data. The employed ontologies are then often formulated in a description logic (DL). Description logics are a family of logic-based knowledge representation formalisms, which can be used to represent the conceptual knowledge of an application domain in a structured and formally well-understood way. Compared to query answering over a (graph) database, OMQA usually adopts the open world assumption where all possible extensions of the ontology and the data are considered when computing the answers. Answering C2RPQs has been studied for very expressive DLs [CEO14], and for the families $\mathcal{EL}$ and DL-Lite of lightweight DLs [CDL$^+$07, BOS15]. However, approaches for query answering under approximate semantics in the OMQA setting are scarce. There is prior work on the simple case of instance queries [EPT15] and on C2RPQs in the restricted setting of acyclic ontologies using RDFs schema [PSW16] or non-gradual variants of conjunctive queries [PTT18].

The goal of this paper is to define approximate semantics for answering C2RPQs in the DLs $\mathcal{ELH}$ and DL-Liteand to devise computation algorithms for answering them. These two DLs are of particular relevance, since they are computationally well-behaved which makes them the languages of choice for OMQA and the core languages of two of the ontology language profiles standardized by the W3C: OWL 2 EL and OWL 2 QL, respectively.

Our contributions are i) to extend the transducer-based approximate semantics from RPQs to the more general case of C2RPQs in the graph database setting. To this end, we need to take into account that, differently from RPQs, C2RPQs may contain more than one query atom and quantified variables. Once this is defined, we move forward into the OMQA setting and ii) define approximate semantics for answering C2RPQs over ontologies formulated in the lightweight DLs $\mathcal{ELH}$ and DL-Lite$_\mathcal{R}$. We define the notion of *certain approximate answers* as a generalization of the classical notion of *certain answers*. Last, iii) we investigate two related computational problems: the decision problem that asks, given a threshold value τ and a tuple $\bar{a}$, is $\bar{a}$ a certain approximate answer with approximation cost at most τ?, and the computational problem of computing the exact approximation cost of $\bar{a}$. For 2RPQs, we devise a polynomial time algorithm that can be used to solve both problems. Regarding C2RPQs, we prove that a) both problems can be solved in polynomial time in *data complexity*, b) the decision problem is in NExptime in combined complexity, and c) provide a double exponential time algorithm (in combined complexity) to compute the approximation cost.

This report is structured as follows. After introducing preliminary notions on DLs in Section 2 and on RPQs in Section 3, we start in to introduce the semantics for approximate answers of RPQs and extend it to C2RPQs over graph databases and DL ontologies in Section 4. In Sections 5 and 6 we study the mentioned computational problems for 2RPQs and C2RPQs, respectively. We conclude the paper with a brief summary.

2 The Description Logics $\mathcal{ELH}$ and DL-Lite$_\mathcal{R}$

We start by introducing the syntax and semantics of the DLs $\mathcal{ELH}$ and DL-Lite$_\mathcal{R}$, as well as some related technical notions that are needed in the rest of the paper. Afterwards, we recall the definition of canonical model, which is used later to obtain our results.

2.1 Syntax and semantics

The logics $\mathcal{ELH}$ and DL-Lite$_\mathcal{R}$ are members of the $\mathcal{EL}$- and DL-Lite families of DLs, respectively. In the following, we introduce the basic concept languages underlying $\mathcal{EL}$ and DL-Lite, and identify the extensions corresponding to $\mathcal{ELH}$ and DL-Lite$_\mathcal{R}$.

Let $\mathsf{N_C}$ and $\mathsf{N_R}$ be countable sets of *concept* and *role* names. Let furthermore $A \in \mathsf{N_C}$, $r \in \mathsf{N_R}$ and $C \in \mathcal{C}_{\mathcal{EL}}$. The *set of $\mathcal{EL}$ concept descriptions* $\mathcal{C}_{\mathcal{EL}}$ is inductively built from $\mathsf{N_C}$ using the concept constructors *conjunction* ($C \sqcap D$), *existential restriction* ($\exists r.C$) and *top* ($\top$), according to the following syntactic rule:

$$C ::= \top \mid A \mid C \sqcap C \mid \exists r.C.$$

Let $A \in \mathsf{N_C}$ and $r \in \mathsf{N_R}$. As for DL-Lite, two additional constructors are available: *inverse roles* (r^-) and *negation* ($\neg$). Complex concepts and roles are built according to the following syntax:

$$\begin{array}{ll} B ::= A \mid \exists P & P ::= r \mid r^- \\ C ::= B \mid \neg B & S ::= P \mid \neg P. \end{array} \tag{1}$$

We call concepts (roles) of the form B (P) *basic* and those of the form C (S) *general* concepts (roles). We use $\mathsf{N_R^-}$ to denote the set $\{r^- \mid r \in \mathsf{N_R}\}$ and $\mathsf{N_R^\pm}$ to denote $\mathsf{N_R} \cup \mathsf{N_R^-}$. In addition, we sometimes write P^- with the meaning that: $P^- = r$ if $P = r^-$ and $P^- = r^-$ if $P = r$.

Knowledge about the domain of interest can be expressed in a description logic *knowledge base* (KB). A DL KB is a pair $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ consisting of a TBox $\mathcal{T}$ and an ABox $\mathcal{A}$. The ABox contains information about specific individuals (represented by a countably infinite set of individual names $\mathsf{N_I}$). More precisely, an *ABox* $\mathcal{A}$ consists of a finite set of assertions. Assertions can be of the form $A(a)$ (*concept assertion*) and $r(a, b)$ (*role assertion*), where $A \in \mathsf{N_C}$, $r \in \mathsf{N_R}$ and $a, b \in \mathsf{N_I}$. Observe that we are using *simple* ABoxes, i.e. all concept assertions use only concept names. We denote as $\mathsf{Ind}(\mathcal{A})$ the set of individuals occurring in $\mathcal{A}$. Regarding the TBox, it consists of a finite set of inclusions, which state general knowledge about the application domain. These inclusions can be of two types, namely *general concept inclusions* (GCIs) and *role inclusions* (RIs), and their forms vary depending on the considered DL. More precisely, in $\mathcal{EL}$, inclusions are GCIs of the form $C \sqsubseteq D$ where $C, D \in \mathcal{C}_{\mathcal{EL}}$. In DL-Lite inclusions have the form $B \sqsubseteq C$, where B and C are as described in (1). Further, permitting RIs in the TBox yields the DLs $\mathcal{ELH}$ and DL-Lite$_\mathcal{R}$. More precisely, $\mathcal{ELH}$ and DL-Lite$_\mathcal{R}$ are the extensions of $\mathcal{EL}$ and DL-Lite allowing RIs of the form $r \sqsubseteq s$ and $P \sqsubseteq S$ in the TBox, respectively, where $r, s \in \mathsf{N_R}$ and P, S are as defined in (1).

The semantics for $\mathcal{EL}$ and DL-Lite is given by means of *first-order* logic interpretations. An *interpretation* $\mathcal{I} = (\Delta^\mathcal{I}, .^\mathcal{I})$ consists of a non-empty domain $\Delta^\mathcal{I}$ and an interpretation function $.^\mathcal{I}$ that assigns subsets of $\Delta^\mathcal{I}$ to concept names in $\mathsf{N_C}$, binary relations over $\Delta^\mathcal{I}$ to role names in $\mathsf{N_R}$ and domain elements $a^\mathcal{I}$ to individual names $a \in \mathsf{N_I}$. The function $.^\mathcal{I}$ is inductively extended

to arbitrary roles and concept descriptions in the following way:[1]

$$\top^{\mathcal{I}} := \Delta^{\mathcal{I}}, \quad (C \sqcap D)^{\mathcal{I}} := C^{\mathcal{I}} \cap D^{\mathcal{I}}, \quad (\neg B)^{\mathcal{I}} := \Delta^{\mathcal{I}} \setminus B,$$
$$(\neg P)^{\mathcal{I}} := (\Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}) \setminus P^{\mathcal{I}}, \ (r^-)^{\mathcal{I}} := \{(x,y) \,|\, (y,x) \in r^{\mathcal{I}}\},$$
$$(\exists r.C)^{\mathcal{I}} := \{x \in \Delta^{\mathcal{I}} \mid \exists y.((x,y) \in r^{\mathcal{I}} \wedge y \in C^{\mathcal{I}})\}.$$

We say that $\mathcal{I}$ satisfies an assertion $A(a)$ iff $a^{\mathcal{I}} \in A^{\mathcal{I}}$, and $r(a,b)$ iff $(a^{\mathcal{I}}, b^{\mathcal{I}}) \in r^{\mathcal{I}}$. In addition, $\mathcal{I}$ satisfies an inclusion $G \sqsubseteq H$ iff $G^{\mathcal{I}} \subseteq H^{\mathcal{I}}$. Further, $\mathcal{I}$ is a model of an ABox $\mathcal{A}$ (in symbols $\mathcal{I} \models \mathcal{A}$) iff it satisfies all the assertions in $\mathcal{A}$, a model of a TBox $\mathcal{T}$ (written as $\mathcal{I} \models \mathcal{T}$) iff $\mathcal{I}$ satisfies all inclusions in $\mathcal{T}$, and a model of a KB $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ (denoted as $\mathcal{I} \models \mathcal{K}$) iff $\mathcal{I} \models \mathcal{T}$ and $\mathcal{I} \models \mathcal{A}$. Then, a KB $\mathcal{K}$ is *satisfiable* iff it has at least one model. While $\mathcal{ELH}$ KBs are always satisfiable, this is not the case for DL-Lite since, for instance, the KB $(\{A \sqsubseteq \neg B\}, \{A(a), B(a)\})$ has clearly no model. Finally, let α be an inclusion $G \sqsubseteq H$ or an ABox assertion $A(a)$ (or $r(a,b)$), we say that $\mathcal{K}$ entails α (denoted as $\mathcal{K} \models \alpha$) iff $\mathcal{I} \models \alpha$ for all models $\mathcal{I}$ of $\mathcal{K}$.

We continue by introducing a normal form for $\mathcal{ELH}$ TBoxes. This is convenient later on to simplify the presentation of technical notions and decision procedures. An $\mathcal{ELH}$ TBox is in *normal form* if all its concept inclusions are of one of the following forms:

$$A \sqsubseteq B \qquad A_1 \sqcap A_2 \sqsubseteq B \qquad A \sqsubseteq \exists r.B \qquad \exists r.A \sqsubseteq B,$$

where $A, B, A_1, A_2 \in \mathsf{N_C} \cup \{\top\}$. As shown in [BBL05], by introducing new concept names, any $\mathcal{ELH}$ TBox $\mathcal{T}$ can be transformed in linear time into a normalized $\mathcal{ELH}$ TBox $\mathcal{T}'$ that is a *model conservative extension* of $\mathcal{T}$, i.e., every model of $\mathcal{T}'$ is also a model of $\mathcal{T}$, and every model of $\mathcal{T}$ can be extended to a model of $\mathcal{T}'$ by appropriately interpreting the additional concept names [LW10].

2.2 Canonical models

The canonical model of a KB $\mathcal{K}$ (sometimes also called a prime model) is a kind of model that can be embedded into any other model of $\mathcal{K}$ by a homomorphism. Canonical models have been widely studied for the $\mathcal{EL}$ and DL-Lite families, since they are essential in the development of decision procedures for answering different types of queries in lightweight DLs [CDL+07, KL07, BOS15]. As the decision procedures developed in this paper also use them, we recall their definitions for the DLs $\mathcal{ELH}$ and DL-Lite$_{\mathcal{R}}$ (as presented in [BOS15]).

Given an $\mathcal{ELH}$ or a satisfiable DL-Lite$_{\mathcal{R}}$ knowledge base $\mathcal{K} = (\mathcal{T}, \mathcal{A})$, we denote the *canonical model* of $\mathcal{K}$ as $\mathcal{U}_{\mathcal{K}} = (\Delta^{\mathcal{U}_{\mathcal{K}}}, \cdot^{\mathcal{U}_{\mathcal{K}}})$. The domain $\Delta^{\mathcal{U}_{\mathcal{K}}}$ consists of sequences e of the form $aP_1C_1 \dots P_nC_n$ $(n \geq 0)$, where $a \in \mathsf{Ind}(\mathcal{A})$, $P_i \in \mathsf{N_R^{\pm}}$ and C_i is a concept description. For $\mathcal{ELH}$, each role P_i is of the form $r_i \in \mathsf{N_R}$ and each concept C_i is just a concept name $A_i \in \mathsf{N_C}$. In addition, e is required to satisfy:

- $\mathcal{K} \models \exists r_1.A_1(a)$, if $n \geq 1$,
- $\mathcal{T} \models A_i \sqsubseteq \exists r_{i+1}.A_{i+1}$, for all $1 \leq i < n$.

As for DL-Lite$_{\mathcal{R}}$, each C_i is of the form $\exists P_i^-$ and e must fulfill the following conditions:

- $\mathcal{K} \models \exists P_1(a)$, if $n \geq 1$,
- $\mathcal{T} \models \exists P_i^- \sqsubseteq \exists P_{i+1}$, for all $1 \leq i < n$.

[1] The constructor $\exists P$ is an abbreviation for $\exists P.\top$.

It remains to fix the interpretation of concept, role and individual names for $\mathcal{U}_\mathcal{K}$. Given $e \in \Delta^{\mathcal{U}_\mathcal{K}}$, we denote as $\mathsf{tail}(e)$ the final concept C_n in e, i.e., either A_n or $\exists P_n^-$. Then, the interpretation function $.^{\mathcal{U}_\mathcal{K}}$ is defined as follows:

- $A^{\mathcal{U}_\mathcal{K}} := \{a \in \mathsf{Ind}(\mathcal{A}) \mid \mathcal{K} \models A(a)\} \cup$
  $\{e \in \Delta^{\mathcal{U}_\mathcal{K}} \setminus \mathsf{Ind}(\mathcal{A}) \mid \mathcal{T} \models \mathsf{tail}(e) \sqsubseteq A\}$,
- $r^{\mathcal{U}_\mathcal{K}} := \{(a, b) \mid \mathcal{K} \models r(a, b)\} \cup \{(e, ePC) \mid$
  $\mathcal{T} \models P \sqsubseteq r\} \cup \{(ePC, e) \mid \mathcal{T} \models P \sqsubseteq r^-\}$,
- $a^{\mathcal{U}_\mathcal{K}} := a$, for all $a \in \mathsf{Ind}(\mathcal{A})$.

Notice that $\mathcal{U}_\mathcal{K}$ consists of a small graph representing $\mathcal{A}$, and (possibly) infinite trees rooted at each $a \in \mathsf{Ind}(\mathcal{A})$ containing anonymous individuals. Given $e, e' \in \Delta^{\mathcal{U}_\mathcal{K}}$, we denote as T_e the subtree in $\mathcal{U}_\mathcal{K}$ rooted at e and write $e' \in T_e$ to say that e' is an element in T_e. An important property of $\mathcal{U}_\mathcal{K}$ is that if $e, e' \in \Delta^{\mathcal{U}_\mathcal{K}} \setminus \mathsf{Ind}(\mathcal{A})$ and $\mathsf{tail}(e) = \mathsf{tail}(e')$, then T_e and $T_{e'}$ are isomorphic. Further, the depth $\mathfrak{d}(e)$ of e in $\mathcal{U}_\mathcal{K}$ is defined as 0 if $e \in \mathsf{Ind}(\mathcal{A})$ and $\mathfrak{d}(e') + 1$ if $e = e'PC$. Then, we say that $e' \in T_e$ has depth i in T_e if $\mathfrak{d}(e') - \mathfrak{d}(e) = i$. Last, we denote as $\mathsf{T}(\mathcal{U}_\mathcal{K})$ the set of tails of $\mathcal{U}_\mathcal{K}$. Notice that this set consists of elements A or $\exists P^-$ occurring in $\mathcal{T}$.

3 Regular path queries

We introduce the syntax and semantics of conjunctive two-way regular path queries and some of its sublanguages for which we define approximate semantics later on. Regular path queries are defined using *regular languages* represented either by a *non-deterministic finite automata* (NFA) or a *regular expressions* (*r.e.*). We assume the reader is familiar with these notions and proceed right away to introduce the consider query languages.

A *conjunctive 2-way regular path query* (C2RPQ) is of the form $q(\bar{x}) = \exists \bar{y}.\varphi(\bar{x}, \bar{y})$, where $\bar{x}, \bar{y}$ are disjoint tuples from a set of variables $\mathsf{N_V}$. The formula $\varphi(\bar{x}, \bar{y})$ is a conjunction of atoms of the form $A(t)$ and $\mathfrak{R}(t, t')$ where t and t' are variables in $\bar{x}, \bar{y}$ or individual names in $\mathsf{N_I}$, $A \in \mathsf{N_C}$, and $\mathfrak{R}$ is an NFA or a *r.e.* defining a regular language (denoted $\mathcal{L}(\mathfrak{R})$) over the alphabet $\mathsf{N_R^\pm} \cup \{A? \mid A \in \mathsf{N_C}\}$. Variables and individuals names in q are called *terms*. Variables in $\bar{x}$ are the *answer variables* and those in $\bar{y}$ are the *quantified* variables of q. We denote the set of terms as $\mathsf{terms}(q)$, the set of variables as $\mathsf{vars}(q)$ and the set of answer and quantified variables of q as $\mathsf{avars}(q)$ and $\mathsf{qvars}(q)$, respectively. If $\mathsf{avars}(q) = \emptyset$, then q is called a *Boolean query*. We sometimes write $\mathsf{at} \in q$ to refer to an atom at of q. In this paper we consider the following specializations of C2RPQs :

Conjunctive (one-way) regular path queries (CRPQs) are C2RPQs that do not use any symbol from $\mathsf{N_R^-}$.

Two-way regular path queries (2RPQs) are obtained by restricting C2RPQ to queries of the form $q(x, z) = \mathfrak{R}(x, z)$, where $x, z \in \mathsf{avars}(q)$.

Regular path queries (RPQs) are the special case of 2RPQs that do not use any symbol from $\mathsf{N_R^-}$.

Next we define the semantics of C2RPQs and thus of their specializations. Let $\mathcal{I} = (\Delta^\mathcal{I}, .^\mathcal{I})$ be an interpretation and $d, d' \in \Delta^\mathcal{I}$. A *path* π from d to d' in $\mathcal{I}$ is a sequence of the form $d_0 u_1 d_1 u_2 d_2 \dots u_m d_m$ such that $m \geq 0$, $d_0 = d$, $d_m = d'$ and for all $1 \leq j \leq m$:

- $d_j \in \Delta^{\mathcal{I}}$ and $u_j \in \mathsf{N}_\mathsf{R}^{\pm} \cup \{A? \mid A \in \mathsf{N}_\mathsf{C}\}$,
- $u_j = A?$ implies $d_{j-1} = d_j$ and $d_j \in A^{\mathcal{I}}$, and
- $u_j \in \mathsf{N}_\mathsf{R}^{\pm}$ implies $(d_{j-1}, d_j) \in (u_j)^{\mathcal{I}}$.

The *label* of π is defined as $\ell(\pi) := u_1 \dots u_m$ and we write $d \xrightarrow{\mathcal{I},u} d'$ to indicate that there is a path from d to d' with label u in $\mathcal{I}$. A *match* for a C2RPQ q in $\mathcal{I}$ is a mapping $h : \mathsf{terms}(q) \mapsto \Delta^{\mathcal{I}}$ such that:

- $h(a) = a^{\mathcal{I}}$ for all $a \in \mathsf{terms}(q) \cap \mathsf{N}_\mathsf{I}$,
- $h(t) \in A^{\mathcal{I}}$ for all $A(t) \in q$, and
- for all $\mathfrak{R}(t, t') \in q$: there exists $u \in \mathcal{L}(\mathfrak{R})$ such that $h(t) \xrightarrow{\mathcal{I},u} h(t')$.

Given a knowledge base $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ and a C2RPQ q with answer variables $x_1, \dots, x_k$, a tuple of individuals $(a_1, \dots, a_k)$ from $\mathsf{Ind}(\mathcal{A})$ is a *certain answer* of q w.r.t. $\mathcal{K}$ iff for each model $\mathcal{I}$ of $\mathcal{K}$ there exists a match h for q in $\mathcal{I}$ such that $h(x_i) = a_i^{\mathcal{I}}$ $(1 \leq i \leq k)$. The set of certain answers of q w.r.t. $\mathcal{K}$ is denoted as $\mathsf{cert}(q, \mathcal{K})$. If q is a Boolean query, $\mathsf{cert}(q, \mathcal{K})$ consists of the empty tuple $()$, if q has a match in every model of $\mathcal{K}$. The semantics tells us that, one can assume without loss of generality that C2RPQs have only atoms of the form $\mathfrak{R}(t, t')$, since every atom $A(t)$ can be equivalently replaced with $A?(t, t)$.

As shown in [BOS15], for DL-Lite$_\mathcal{R}$ and $\mathcal{ELH}$, the set $\mathsf{cert}(q, \mathcal{K})$ can be characterized by only considering matches of q in the canonical model $\mathcal{U}_\mathcal{K}$.

Lemma 1 (Lemma 3.2, [BOS15])**.** *Let $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ be an $\mathcal{ELH}$ or a satisfiable DL-Lite$_\mathcal{R}$ KB, $q(\bar{x})$ a C2RPQ of arity k and $\bar{a}$ a k-tuple of individuals from $\mathcal{A}$. Then, $\bar{a} \in \mathsf{cert}(q, \mathcal{K})$ iff there is a match h for q in $\mathcal{U}_\mathcal{K}$ such that $h(\bar{x}) = \bar{a}$.*

We have defined the semantics of C2RPQs over DL interpretations, but they were first introduced and are often considered w.r.t. graph databases. DL interpretations and basic forms of graph databases are actually very similar notions, since they both can be seen as relational structures over predicates of arity at most 2. For the development of the next section, it is convenient to establish this correspondence more formally. Let Σ and Λ be disjoint alphabets. A graph database is a tuple $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \ell_\mathcal{V})$, where $(\mathcal{V}, \mathcal{E})$ is a directed labeled graph: $\mathcal{E} \subseteq \mathcal{V} \times \Sigma \times \mathcal{V}$ is a set of edges labeled with symbols from Σ and $\ell_\mathcal{V} : \mathcal{V} \mapsto \Lambda$ assigns labels from Λ to vertices in $\mathcal{V}$ [CM90]. Then, $\mathcal{G}$ can be translated into a finite interpretation $\mathcal{I}_\mathcal{G}$ over $\mathsf{N}_\mathsf{I} = \mathcal{V}$, $\mathsf{N}_\mathsf{C} = \Lambda$, and $\mathsf{N}_\mathsf{R} = \Sigma$ with domain $\Delta^{\mathcal{I}_\mathcal{G}} := \mathcal{V}$, where:[2]

- $A^{\mathcal{I}_\mathcal{G}} := \{d \in \mathcal{V} \mid \ell_\mathcal{V}(d) = A\}$ for all $A \in \Lambda$, and
- $r^{\mathcal{I}_\mathcal{G}} := \{(d, e) \mid (d, r, e) \in \mathcal{E}\}$ for all $r \in \Sigma$.
- $a^{\mathcal{I}_\mathcal{G}} := a$ for all $a \in \mathcal{V}$.

The semantics of a C2RPQ q over $\mathcal{G}$ is defined by considering only matches into $\mathcal{I}_\mathcal{G}$. Using the translation, the notions of a path in $\mathcal{G}$ and a match of q in $\mathcal{G}$ can now be formalized w.r.t. $\mathcal{I}_\mathcal{G}$. We denote as $\mathsf{ans}(q, \mathcal{G})$ the set of answers of q in $\mathcal{G}$.

[2]The opposite translation is defined as expected by considering $\Sigma = \mathsf{N}_\mathsf{R}$ and $\Lambda = 2^{\mathsf{N}_\mathsf{C}}$.

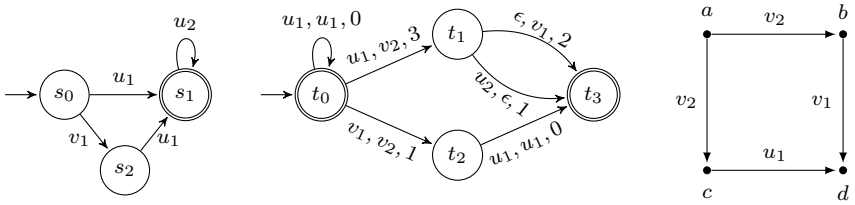

Figure 1: Approximate semantics.

4 Approximate semantics for regular path queries

Our notion of approximate semantics for regular path queries over DL ontologies is an extension of such semantics of RPQs over graph databases proposed in [GT06]. Regular path queries are specified by finite automata and for their approximate variants it is a natural idea to employ finite transducers, which are essentially, finite automata with an output. In case of weighted transducers, input words can be associated with output words together with a cost. For giving regular path queries approximate semantics, the idea is now to accept also tuples as answers that may not satisfy the query restrictions, but are connected by paths that approximate those required by the query with a low cost. The transducer defines which paths represent approximations and their approximation cost.

We recall the original transducer-based approach from [GT06] for RPQs over graph databases. We extend this approach to answering conjunctive two-way regular path queries under approximate semantics over graph databases which are then in turn extended to define the approximate semantics for answering C2RPQs over DL knowledge bases.

4.1 Approximate semantics for RPQs by transducers

We start by formally introducing the form of weighted finite-state transducer used in [GT06]. In general, weighted transducers are defined over a *semiring*, which are weighted structures of the form $(\mathbb{K}, \oplus, \otimes, \mathsf{0}, \mathsf{1})$ where $\mathsf{0}, \mathsf{1}$ are two constant elements from the set $\mathbb{K}$, and $\oplus, \otimes$ are two binary operations over $\mathbb{K}$ satisfying certain properties [Meh04]. The weighted transducer considered in [GT06] are those defined over the *tropical semiring* $(\mathbb{N} \cup \{\infty\}, \min, +, \infty, 0)$ [Sim78]. In this paper, we refer to them as distortion transducers. They are defined as follows.

Definition 2. A *distortion transducer* (dT) is a tuple $\mathfrak{T} = (\Sigma, Q, \delta, I, F)$ where Q is a finite state set, Σ is a finite input/output alphabet, $I \subseteq Q$ the set of initial states, $F \subseteq Q$ set of final states, and $\delta \subseteq Q \times \Sigma \cup \{\epsilon\} \times \Sigma \cup \{\epsilon\} \times \mathbb{N} \times Q$ is the transition relation.

Given a dT $\mathfrak{T} = (\Sigma, Q, \delta, I, F)$, a *run* of $\mathfrak{T}$ on a word $u \in \Sigma^*$ is a sequence of tuples

$$\rho = (q_1, u_1, v_1, w_1, q_2), \ldots, (q_n, u_n, v_n, w_n, q_{n+1})$$

such that $u = u_1 \ldots u_n$, $q_1 \in I$, $q_n \in F$ and each $(q_i, u_i, v_i, w_i, q_{i+1}) \in \delta$ $(i < n)$. The *weight of a run* ρ is defined as $wt(\rho) := w_1 + \ldots + w_n$. A run ρ distorts u into v with *cost* $wt(\rho)$. Let $\mathcal{R}(\mathfrak{T}, u, v)$ be the set of all pairs $(\rho, wt(\rho))$ such that ρ is a run of $\mathfrak{T}$ distorting u into v. The cost of distorting u into v through $\mathfrak{T}$ is defined as:

$$c_{\mathfrak{T}}(u, v) := \begin{cases} \infty, & \text{if } \mathcal{R}(\mathfrak{T}, u, v) = \emptyset \\ \min\{wt(\rho) \mid (\rho, wt(\rho)) \in \mathcal{R}(\mathfrak{T}, u, v)\}, & \text{otherwise.} \end{cases}$$

The idea proposed in [GT06] is to use a distortion transducer to declare what distortions of

the words required by a RPQ atom $\mathfrak{R}(x, z)$ are allowed as "acceptable" and the cost of the corresponding distortion. Let us illustrate this with the following example.

Example 3. Figure 1 depicts an an NFA $\mathfrak{R}$ with $\mathcal{L}(\mathfrak{R}) = \{u_1u_2^*\} \cup \{v_1u_1u_2^*\}$, a dT $\mathfrak{T}$, and a graph database $\mathcal{G}$. One can see that, querying $\mathcal{G}$ through the RPQ $q(x, z)$ defined by $\mathfrak{R}$ yields only one classical answer, i.e., $\mathsf{ans}(q, \mathcal{G}) = \{(c, d)\}$.

By using $\mathfrak{T}$ inbetween $\mathfrak{R}$ and $\mathcal{G}$, the set $\mathsf{ans}(q, \mathcal{G})$ can be approximated. For instance, $\mathfrak{T}$ states that v_2 is an allowed distortion of the word u_1u_2 and using this distortion is "penalized" with cost 4. Then, since $u_1u_2 \in \mathcal{L}(\mathfrak{R})$ and a path with label v_2 connects a to b in $\mathcal{G}$, instead of dismissing (a, b) as an answer one can now see it as an approximate answer with cost 4. A further example is the tuple (a, d) where two distortions are possible: v_1u_1 into v_2u_1 and u_1 into v_2v_1 with costs 1 and 5, respectively. In this case, the smallest distortion cost gives the distortion cost of (a, d). Finally, notice that $\mathfrak{T}$ gives no way to distort a word in $\mathcal{L}(\mathfrak{R})$ into v_1. Consequently, (b, d) is not an approximate answer (or with distortion cost ∞).

Based on this idea, the notion of approximate answers for RPQs is formalized as follows. Given are a RPQ $q(x, z) := \mathfrak{R}(x, z)$[3] and a graph database $\mathcal{G} = (\mathcal{V}, \mathcal{E})$. The *set of approximate answers* of q on $\mathcal{G}$, through a distortion transducer $\mathfrak{T}$ with $\Sigma = \mathsf{N}_\mathsf{R}$, is defined as:

$$\tilde{\mathsf{ans}}_{\mathfrak{T}}(q, \mathcal{G}) := \{(a, b, \eta) \mid a, b \in \mathcal{V} \text{ and } \eta = \min\{c_{\mathfrak{T}}(u, v) \mid u \in \mathcal{L}(\mathfrak{R}) \wedge a \xrightarrow{\mathcal{I}_\mathcal{G}, v} b\}\}. \tag{2}$$

We say that (a, b) is in the approximate answers with distortion cost η. The approximate answers in the previous example with $\eta < \infty$ are the tuples $(c, d, 0), (a, d, 1), (a, b, 4)$ and $(a, c, 4)$.

Some general observations about this approximation mechanism follow. On an abstract level, the approach allows to relax and/or restrain the classical semantics of RPQs. Intuitively, a relaxation would require that classical answers of q in $\mathcal{G}$ are preserved (with cost 0) in the approximation. As pointed out in [GT06], to achieve this it suffices to add an initial and final state t_0' to $\mathfrak{T}$ with neutral transitions $(t_0', u, u, 0, t_0')$ for all $u \in \Sigma$. At the approximation level, one can define a variety of approximation schemas. For instance, as mentioned in [PSW16], one can use transitions with ϵ to build transducers whose distortion costs corresponds to the *edit distance* between two words. In addition, one can express that the cost of approximating a symbol u by v differs depending on the context (i.e., the path) they occur in.

4.2 Approximate semantics for C2RPQs over a graph database

As for RPQs, we define the approximate answers of a C2RPQ $q(\bar{x}) = \exists \bar{y}.\varphi(\bar{x}, \bar{y})$ of arity k as pairs of the form $(\bar{a}, \eta)$, where $\bar{a} = (a_1, \ldots, a_k)$ is a tuple of nodes in $\mathcal{G}$ and η is the approximation cost for $\bar{a}$. The idea is for η to express "how close" $\bar{a}$ is to be an answer tuple of q in $\mathcal{G}$. However, in contrast to RPQs, a C2RPQ q may now contain quantified variables and more than one atom. Both aspects require adaptations in the definition of η given in $\tilde{\mathsf{ans}}_{\mathfrak{T}}(q, \mathcal{G})$ for RPQs in equation (2).

First, to accommodate the quantified variables we proceed as usual for CQs and consider all possible matches h for q in $\mathcal{I}_\mathcal{G}$ such that $h(\bar{x}) = \bar{a}$. For each of such match we define an approximation cost h_c that measures how close h is to be a match for q in $\mathcal{I}_\mathcal{G}$. The lowest such cost h_c of a match is then the cost value η in $(\bar{a}, \eta)$. Second, the value h_c is obtained by: i) computing for each atom $\mathfrak{R}(t_j, t_j') \in q$ the distortion cost η_j of the pair $(h(t_j), h(t_j'))$ using (2), and ii) combining all these values into h_c using an appropriate function.

A function $\mathrm{f} : \mathbb{N}^p \cup \{\infty\} \to \mathbb{N} \cup \{\infty\}$ is a *p-ary combining function* if it is:

[3]Symbols of the form $A?$ are not considered in [GT06].

- *commutative*, i.e., $\mathrm{f}(c_1, \ldots, c_p) = \mathrm{f}(c_{\sigma(1)}, \ldots, c_{\sigma(p)})$, where σ is any permutation of the indices $1 \ldots p$.
- *monotonic*, i.e., $c_1 \leq d_1, \ldots, c_p \leq d_p$ implies $\mathrm{f}(c_1, \ldots, c_p) \leq \mathrm{f}(d_1, \ldots, d_p)$, and
- *zero closed*, i.e., $\mathrm{f}(c_1, \ldots, c_p) = 0$ iff $c_i = 0$ for all $1 \leq i \leq p$.

Commutativity of f ensures that the order of atoms in q does not influence the value of h_c, whereas zero closedness implies $h_c = 0$ if h is a match in the classical sense. In addition to these properties, we restrict the attention to combining functions that can be computed in polynomial time. Examples of such functions are the *sum*, *average*, *minimum* and *maximum*.

Using a combining function f, we can now define the notion of approximate match. An *approximate match* $h^{q,\mathcal{I}_\mathcal{G}}_{\mathfrak{T},\mathrm{f}}$ for q in $\mathcal{I}_\mathcal{G}$, through a distortion transducer $\mathfrak{T}$ and combining function f, is a pair $h^{q,\mathcal{I}_\mathcal{G}}_{\mathfrak{T},\mathrm{f}} = (h, h_c)$ where:

- $h : \mathsf{terms}(q) \to \Delta^{\mathcal{I}_\mathcal{G}}$ is a mapping such that $h(a) = a$ for all $a \in \mathsf{terms}(q) \cap \Delta^{\mathcal{I}_\mathcal{G}}$; and
- the approximation cost $h_c \in \mathbb{N}$ is defined as

$$h_c := \fint_{\mathfrak{R}(t,t') \in q} \min \left\{ c_\mathfrak{T}(u, v) \mid u \in \mathcal{L}(\mathfrak{R}) \wedge h(t) \xrightarrow{\mathcal{I}_\mathcal{G}, v} h(t') \right\}. \tag{3}$$

Approximate answers to a C2RPQ need to take into account all approximate matches. For an arbitrary k-tuple $\bar{a}$ of nodes in $\Delta^{\mathcal{I}_\mathcal{G}}$, we denote by $H^{q,\mathcal{I}_\mathcal{G}}_{\mathfrak{T},\mathrm{f}}(\bar{a})$ the *set of approximate matches* satisfying $h(\bar{x}) = \bar{a}$. We are now ready to extend the notion of approximate answer to C2RPQs.

Definition 4. Let $q(\bar{x}) = \exists \bar{y}.\varphi(\bar{x}, \bar{y})$ be a C2RPQ with p many atoms, $\mathcal{G}$ a graph database, $\mathfrak{T}$ a distortion transducer with $\Sigma = \mathsf{N}^{\pm}_{\mathsf{R}} \cup \{A? \mid A \in \mathsf{N}_{\mathsf{C}}\}$ and f a p-ary combining function. Then, the *set of approximate answers* of q on $\mathcal{G}$, through $\mathfrak{T}$ and f, is defined as:

$$\tilde{\mathsf{ans}}_{\mathfrak{T},\mathrm{f}}(q, \mathcal{G}) := \left\{ (\bar{a}, \eta) \mid \bar{a} \in \mathcal{V} \wedge \eta = \min\{h_c \mid (h, h_c) \in H^{q,\mathcal{I}_\mathcal{G}}_{\mathfrak{T},\mathrm{f}}(\bar{a})\} \right\}.$$

We say that $\bar{a}$ is in the approximate answers with approximation cost η. Notice that this definition is an extension of the approximate semantics of RPQs, whenever the combining function f is the identity function. For the rest of the paper, we will assume that this is the case and in the presence of 2RPQs we will just write $\tilde{\mathsf{ans}}_\mathfrak{T}$. Moreover, requiring f to be zero closed ensures that as for RPQs, we can relax the classical semantics of C2RPQs by modifying $\mathfrak{T}$ as explained above.

4.3 Approximate semantics for C2RPQs over a DL KB

While in graph database settings query answering adopts the closed world assumption and regards one model, in OMQA it adopts the open world assumption and regards what holds in all models of the ontology. Our approximate semantics that extends the classical one follows this idea.

The notion of approximate match extends naturally from graph databases to DL interpretations. More precisely, in an approximate match $h^{q,\mathcal{I}}_{\mathfrak{T},\mathrm{f}} = (h, h_c)$ of a C2RPQ q in an interpretation $\mathcal{I}$, h is now a mapping into $\Delta^\mathcal{I}$ such that $h(a) = a^\mathcal{I}$ for all terms $a \in \mathsf{N_I}$ in q, and h_c is defined in terms of $\xrightarrow{\mathcal{I}, v}$ (instead of $\xrightarrow{\mathcal{I}_\mathcal{G}, v}$). For a k-tuple $\bar{a}$ of individual names in $\mathcal{A}$, $H^{q,\mathcal{I}}_{\mathfrak{T},\mathrm{f}}(\bar{a})$ denotes the *set of approximate matches* satisfying $h(x_i) = a_i^\mathcal{I}$. Now, in order to get the certain approximate answers to the C2RPQ, the semantics uses an upper bound on the approximation costs in any

of the models. More precisely, an answer tuple $\bar{a}$ of ABox individuals incurs in each model a certain (minimal) cost for the combination of the distortions of the query atoms. The most costly of these approximations supplies an upper bound on the approximation costs η for this tuple over all models.

Definition 5. Let $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ be a knowledge base and $q(\bar{x})$ a C2RPQ with p many atoms. The set of *certain approximate answers* of q w.r.t. $\mathcal{K}$, through a distortion transducer $\mathfrak{T}$ with $\Sigma = \mathsf{N}_\mathsf{R}^\pm \cup \{A? \mid A \in \mathsf{N}_\mathsf{C}\}$ and a p-ary combining function f, is defined as:

$$\tilde{\mathsf{cert}}_{\mathfrak{T},\mathrm{f}}(q,\mathcal{K}) := \Big\{(\bar{a},\eta) \mid \bar{a} \in \mathsf{Ind}(\mathcal{A}) \text{ and } \eta = \sup_{\mathcal{I}\models\mathcal{K}} \big\{ \min\{h_c \mid (h,h_c) \in H^{q,\mathcal{I}}_{\mathfrak{T},\mathrm{f}}(\bar{a})\}\big\}\Big\}.$$

To see that the supremum in this definition always exists, we show that similar to $\mathsf{cert}(q,\mathcal{K})$, the set $\tilde{\mathsf{cert}}_{\mathfrak{T},\mathrm{f}}(q,\mathcal{K})$ can be characterized by looking only at approximate matches in the canonical model $\mathcal{U}_\mathcal{K}$. The following lemma generalizes the result in Lemma 1 to the approximate semantics.

Lemma 6. *Let $\mathcal{K} = (\mathcal{T},\mathcal{A})$ be an $\mathcal{ELH}$ or a satisfiable DL-Lite$_\mathcal{R}$ KB, $q(\bar{x})$ a C2RPQ of arity k with p atoms, $\mathfrak{T}$ a dT and* f *a p-ary combining function. Then, $(\bar{a},\eta) \in \tilde{\mathsf{cert}}_{\mathfrak{T},\mathrm{f}}(q,\mathcal{K})$ iff $\eta = min\{h_c \mid (h,h_c) \in H^{q,\mathcal{U}_\mathcal{K}}_{\mathfrak{T},\mathrm{f}}(\bar{a})\}$.*

Proof. Let $(h,h_c) \in H^{q,\mathcal{U}_\mathcal{K}}_{\mathfrak{T},\mathrm{f}}(\bar{a})$ and $\mathcal{I}$ be an arbitrary model of $\mathcal{K}$. We show that there is an approximate match in $\mathcal{I}$ with the cost at most h_c. To this end, we consider a homomorphism hom from $\mathcal{U}_\mathcal{K}$ into $\mathcal{I}$ and prove that $(hom \circ h, (hom \circ h)_c)$ is such a match.

First, it is clear that $(hom \circ h, (hom \circ h)_c) \in H^{q,\mathcal{I}}_{\mathfrak{T},\mathrm{f}}(\bar{a})$. Second, let $v \in (\mathsf{N}_\mathsf{R}^\pm \cup \{A?\})^*$ such that $h(t) \xrightarrow{\mathcal{U}_\mathcal{K},v} h(t')$. The use of hom implies $hom(h(t)) \xrightarrow{\mathcal{I},v} hom(h(t'))$. This means that for all $\mathfrak{R}(t,t') \in q$, the minimum in (3) for $\mathfrak{R}$ w.r.t. $hom \circ h$ and $\mathcal{I}$ is not greater than w.r.t. h and $\mathcal{U}_\mathcal{K}$. Thus, since f is monotonic, it follows that $(hom \circ h)_c \leq h_c$. □

Finally, we state the computational problems that we are interested in. We consider the *computational* problem: given a tuple $\bar{a}$, compute the value η such that $(\bar{a},\eta) \in \tilde{\mathsf{cert}}_{\mathfrak{T},\mathrm{f}}(q,\mathcal{K})$. In addition, we investigate the associated *decision* problem (called τ-entailment) that asks: given a tuple $\bar{a}$ and a threshold value $\tau \in \mathbb{N}$, whether $\bar{a}$ is a certain approximate answer of q w.r.t. $\mathcal{K}$ with approximation cost $\eta \leq \tau$.

5 Answering approximate 2RPQs in $\mathcal{ELH}$ and DL-Lite$_\mathcal{R}$

In this section, we provide a polynomial time algorithm to compute the certain approximate answers of 2RPQs over $\mathcal{ELH}$ and DL-Lite$_\mathcal{R}$ ontologies. Our algorithm combines and extends the approaches to answer approximate RPQs over graph databases from [GT06] and 2RPQs in $\mathcal{ELH}$ and DL-Lite$_\mathcal{R}$ under classical semantics from [BOS15].

[GT06] provide a polynomial time algorithm to answer RPQs under approximate semantics, which amounts to finding the shortest path in a weighted directed graph. This graph is obtained from the Cartesian product of the NFA defining the query, the distortion transducer, and the queried database. We lift this approach to the OMQA setting. Applying this procedure directly on the data in the ABox need not be *complete*, since models of $\mathcal{K}$ may contain anonymous individuals induced by GCIs in the TBox. To deal with this, our first step is to consider $\mathcal{U}_\mathcal{K}$ instead of just the data to build a suitable weighted graph.

A *weighted graph* is a pair $G = (V,E)$, where V is a set of vertices and E is a set of edges labeled with a numerical weight, which in our setting are of the form $E \subseteq V \times \mathbb{N} \times V$. A path

π in G is a sequence $\mathrm{v}_1 w_1 \mathrm{v}_2 \ldots \mathrm{v}_{n-1} w_{n-1} \mathrm{v}_n$ where $\mathrm{v}_i \in V$, $w_i \in \mathbb{N}$ and $(\mathrm{v}_i, w_i, \mathrm{v}_{i+1}) \in E$. The *cost* $c(\pi)$ of π is the sum of all its weights.

Now, let $\mathfrak{R}(x, z)$ be a 2RPQ with $\mathfrak{R} = (Q_\mathfrak{R}, \Sigma, \delta_\mathfrak{R}, I_\mathfrak{R}, F_\mathfrak{R})$ and $\mathfrak{T} = (Q_\mathfrak{T}, \Sigma, \delta_\mathfrak{T}, I_\mathfrak{T}, F_\mathfrak{T})$ a dT, where $\Sigma = \mathsf{N}_\mathsf{R}^\pm \cup \{A? \mid A \in \mathsf{N}_\mathsf{C}\}$. In addition, let $\mathcal{K}$ be an $\mathcal{ELH}$ or DL-Lite$_\mathcal{R}$ KB. Using $\mathfrak{R}$, $\mathfrak{T}$ and $\mathcal{U}_\mathcal{K}$, the *weighted graph* $G_{\mathfrak{R}\times\mathfrak{T}\times\mathcal{U}_\mathcal{K}} = (V, E)$ is defined as:

- $V := \{(s, t, e) \in Q_\mathfrak{R} \times Q_\mathfrak{T} \times \Delta^{\mathcal{U}_\mathcal{K}}\}$.
- $E := \{((s,t,e), w, (s',t',e')) \mid (s,u,s') \in \delta_\mathfrak{R}, (t,u,v,w,t') \in \mathfrak{T},$
  $v \in \mathsf{N}_\mathsf{R}^\pm \Rightarrow (e,e') \in v^{\mathcal{U}_\mathcal{K}}, v = A? \Rightarrow (e \in A^{\mathcal{U}_\mathcal{K}} \wedge e = e')\} \cup$
  $\{((s,t,e), w, (s,t',e')) \mid (t,\epsilon,v,w,t') \in \mathfrak{T}, v \in \mathsf{N}_\mathsf{R}^\pm \Rightarrow (e,e') \in v^{\mathcal{U}_\mathcal{K}},$
  $v = A? \Rightarrow (e \in A^{\mathcal{U}_\mathcal{K}} \wedge e = e')\} \cup$
  $\{((s,t,e), w, (s',t',e)) \mid (s,u,s') \in \delta_\mathfrak{R}, (t,u,\epsilon,w,t') \in \mathfrak{T}\}$.

To simplify notation, we use $G_{\mathcal{U}_\mathcal{K}}$ to refer to $G_{\mathfrak{R}\times\mathfrak{T}\times\mathcal{U}_\mathcal{K}}$, if $\mathfrak{R}$ and $\mathfrak{T}$ are clear from the context. In addition, we assume that $\mathfrak{T}$ does not have transitions with ϵ-labels. This does not affect our results, but eases considerably their technical presentation. Notice that, the third set defining $G_{\mathcal{U}_\mathcal{K}}$ can be simulated by using a *dummy* concept B such that $\mathcal{K} \models B \equiv \top$, whereas edges in the second set can be treated as the ones in the first set. The following lemma uses $G_{\mathcal{U}_\mathcal{K}}$ to characterize $\tilde{\mathsf{cert}}_\mathfrak{T}(q, \mathcal{K})$.

Lemma 7. *Let $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ be an $\mathcal{ELH}$ or a satisfiable DL-Lite$_\mathcal{R}$ KB, $q(x,z) := \mathfrak{R}(x,z)$ a 2RPQ, $\mathfrak{T}$ a dT, and $a, b \in \mathsf{Ind}(\mathcal{A})$. Then, $(a,b,\eta) \in \tilde{\mathsf{cert}}_\mathfrak{T}(q,\mathcal{K})$ iff the minimal cost c^* of a path in $G_{\mathcal{U}_\mathcal{K}}$ from a vertex (s_0, t_0, a) to a vertex (s_f, t_f, b) is equal to η, where $s_0 \in I_\mathfrak{R}$, $t_0 \in I_\mathfrak{T}$, $s_f \in F_\mathfrak{R}$, and $t_f \in F_\mathfrak{T}$.*

Proof. By Lemma 6, $(a,b,\eta) \in \tilde{\mathsf{cert}}_\mathfrak{T}(q,\mathcal{K})$ implies the following for 2RPQs:

$$\eta = \inf\{c_\mathfrak{T}(u,v) \mid u \in \mathcal{L}(\mathfrak{R}) \wedge a \xrightarrow{\mathcal{U}_\mathcal{K}, v} b\}.$$

In the following, we show that $c^* \leq \eta$ and $\eta \leq c^*$.

$\eta \leq c^*$: if no path of the form $(s_0,t_0,a)\ldots(s_f,t_f,b)$ exists in $G_{\mathcal{U}_\mathcal{K}}$, then $c^* = \infty$ and $\eta \leq c^*$. Otherwise, let $\pi = (s_0,t_0,a)\ldots(s_f,t_f,b)$ be a path in $G_{\mathcal{U}_\mathcal{K}}$. Following the definition of $G_{\mathcal{U}_\mathcal{K}}$, words $u', v' \in \Sigma^*$ can be obtained from π such that: $u' \in \mathcal{L}(\mathfrak{R})$, $a \xrightarrow{\mathcal{U}_\mathcal{K}, v'} b$ and there is a run ρ of $\mathfrak{T}$ that distorts u' into v' with $wt(\rho) = c(\pi)$. This means that $c_\mathfrak{T}(u',v') \leq c(\pi)$. Hence, since π is arbitrarily chosen, it follows that $\eta \leq c^*$.

$c^* \leq \eta$: if for all pair of words $u', v' \in \Sigma^*$ either $u' \notin \mathcal{L}(\mathfrak{R})$, $\mathcal{R}(\mathfrak{T}, u', v') = \emptyset$ or $a \xrightarrow{\mathcal{U}_\mathcal{K}, v'} b$ does not hold, then $\eta = \infty$. Hence, $c^* \leq \eta$. Otherwise, let $u' \in \mathcal{L}(\mathfrak{R})$ and $v' \in \Sigma^*$ such that $\mathcal{R}(\mathfrak{T}, u', v') \neq \emptyset$ and $a \xrightarrow{\mathcal{U}_\mathcal{K}, v'} b$. In addition, let $(\rho, wt(\rho)) \in \mathcal{R}(\mathfrak{T}, u', v')$. By definition of a run of $\mathfrak{R}$ and $\mathfrak{T}$, one can build a path $\pi = (s_0,t_0,a)\ldots(s_f,t_f,b)$ in $G_{\mathcal{U}_\mathcal{K}}$ such that $c(\pi) = wt(\rho)$. By definition of $c_\mathfrak{T}$ it follows that $c^* \leq c_\mathfrak{T}(u',v')$. Thus, $c^* \leq \eta$. □

Clearly, since in general $\mathcal{U}_\mathcal{K}$ can be infinite, running a shortest path algorithm directly on $G_{\mathcal{U}_\mathcal{K}}$ would not yield an algorithm to compute $\tilde{\mathsf{cert}}_\mathfrak{T}(q,\mathcal{K})$. To overcome this problem, we extend the ideas used in [BOS15] to decide whether $(a,b) \in \mathsf{cert}(q,\mathcal{K})$, which consist on applying a *symbolic* computation to solve a reachability problem in the possibly infinite graph $G_{\mathfrak{R}\times\mathcal{U}_\mathcal{K}}$. We continue by describing how to adapt these ideas to obtain algorithms for our more general settings.

5.1 A polynomial time algorithm

Let $G_{\mathfrak{R}\times\mathfrak{T}\times\mathcal{U}_\mathcal{A}}$ (abbreviated as $G_{\mathcal{U}_\mathcal{A}}$) be the sub-graph of $G_{\mathcal{U}_\mathcal{K}}$ restricted to vertices (s,t,a) such that $a \in \mathsf{Ind}(\mathcal{A})$. The idea is to reduce the search of paths of minimal cost in $G_{\mathcal{U}_\mathcal{K}}$ to searching in an extension of $G_{\mathcal{U}_\mathcal{A}}$ that has polynomial size. In order to explain this, we first need to introduce the notion of an e-path in $G_{\mathcal{U}_\mathcal{K}}$.

Definition 8. Let $e \in \Delta^{\mathcal{U}_\mathcal{K}}$. An e-path in $G_{\mathcal{U}_\mathcal{K}}$ is a path of the form $(s,t,e)\,\gamma\,(s',t',e)$ such that:

- $\gamma \in \mathbb{N}$ implies $e \in \Delta^{\mathcal{U}_\mathcal{K}} \setminus \mathsf{Ind}(\mathcal{A})$, and
- γ only visits vertices (s'',t'',e') such that $e' \in \Delta^{\mathcal{U}_\mathcal{K}} \setminus \mathsf{Ind}(\mathcal{A})$ and $e' \in T_e$.

Notice that an e-path may visit more than two vertices of the form $(\_,\_,e)$ if $e \in \Delta^{\mathcal{U}_\mathcal{K}} \setminus \mathsf{Ind}(\mathcal{A})$, but not if $e = a \in \mathsf{Ind}(\mathcal{A})$.

Let now π be a path from (s,t,a) to (s',t',b) in $G_{\mathcal{U}_\mathcal{K}}$ with $a,b \in \mathsf{Ind}(\mathcal{A})$. This path can be decomposed as follows:

$$(s_1,t_1,a_1)\,\gamma_1\,(s_2,t_2,a_2)\,\gamma_2 \ldots \gamma_{n-1}\,(s_n,t_n,a_n), \tag{4}$$

where $n \geq 1$, $s_1 = s$, $t_1 = t$, $s_n = s'$, $t_n = t'$, $a_i \in \mathsf{Ind}(\mathcal{A})$, $a_1 = a$, $a_n = b$, and for all γ_i either i) $\gamma_i \in \mathbb{N}$, or ii) $a_i = a_{i+1}$ and $(s_i,t_i,a_i)\gamma_i(s_{i+1},t_{i+1},a_i)$ is an a_i-path. One observation follows for those $\gamma_i \notin \mathbb{N}$:

- Let c_i be the cost of $(s_i,t_i,a_i)\gamma_i(s_{i+1},t_{i+1},a_i)$. If all edges $((s_i,t_i,a_i),c_i,(s_{i+1},t_{i+1},a_i))$ are added to $G_{\mathcal{U}_\mathcal{A}}$, then a path from (s,t,a) to (s',t',b) with the same cost as (4) can be found in the augmented $G_{\mathcal{U}_\mathcal{A}}$.

This correspondence also extends to paths of minimal cost. More precisely,

Proposition 9. *Let v_1,v_2 be vertices in $G_{\mathcal{U}_\mathcal{A}}$. Further, let $G^*_{\mathcal{U}_\mathcal{A}}$ be the extension of $G_{\mathcal{U}_\mathcal{A}}$ with all the edges $((s,t,a),c^*,(s',t',a))$ such that:*

- *c^* is the minimal cost of an a-path from (s,t,a) to (s',t',a) in $G_{\mathcal{U}_\mathcal{K}}$.*

Then, in $G_{\mathcal{U}_\mathcal{K}}$ and $G^*_{\mathcal{U}_\mathcal{A}}$, the minimal cost of a path from v_1 to v_2 is the same .

Proof. $\Rightarrow$: Let π be a path of minimal cost from (s,t,a) to (s',t',a) in $G_{\mathcal{U}_\mathcal{K}}$. As explained before, it can be decomposed as shown in (4). If $\gamma_i \in \mathbb{N}$, then $(s_i,t_i,a_1)\,w_i\,(s_{i+1},t_{i+1},a_2)$ is an edge in $G^*_{\mathcal{U}_\mathcal{A}}$. Otherwise, $(s_i,t_i,a_1)\,\gamma_i\,(s_{i+1},t_{i+1},a_2)$ is an a-path with cost c. Since $c(\pi)$ is minimal, c must also be minimal. This means that $(s_i,t_i,a_1)\,c\,(s_{i+1},t_{i+1},a_1)$ is an edge in $G^*_{\mathcal{U}_\mathcal{A}}$. Thus, there is a path π^* from (s,t,a) to (s',t',a) in $G^*_{\mathcal{U}_\mathcal{K}}$ such that $c(\pi) = c(\pi^*)$.

$\Leftarrow$: Let π^* be a path of minimal cost from (s,t,a) to (s',t',a) in $G^*_{\mathcal{U}_\mathcal{K}}$. In addition, consider an arbitrary edge $(\_,\_,a_1)\,w\,(\_,\_,a_2)$ in π^*. If such an edge is not in $G_{\mathcal{U}_\mathcal{A}}$, by definition of $G^*_{\mathcal{U}_\mathcal{A}}$ there is a path in $G_{\mathcal{U}_\mathcal{K}}$ of the form $(\_,\_,a_1)\ldots(\_,\_,a_2)$ of cost w. This means that one can successively replace such edges by the corresponding path to obtain a path π in $G_{\mathcal{U}_\mathcal{K}}$ such that $c(\pi^*) = c(\pi)$. □

Hence, to compute $\tilde{\mathsf{cert}}_\mathfrak{T}(q,\mathcal{K})$, one can restrict the attention to the fragment $G_{\mathcal{U}_\mathcal{A}}$ of $G_{\mathcal{U}_\mathcal{K}}$, provided that all minimal costs c^* can be computed. We exploit this in Algorithm 1 above

Algorithm 1 Answering 2RPQs under approximate semantics

Input: A 2RPQ $\mathfrak{R}(x,z)$, a dT $\mathfrak{T}$ and an $\mathcal{ELH}$ or DL-Lite$_\mathcal{R}$ KB $\mathcal{K}=(\mathcal{T},\mathcal{A})$.
Output: $\tilde{\mathsf{cert}}_\mathfrak{T}(q,\mathcal{K})$.

1: **if** $\mathcal{K}$ is unsatisfiable **then** return $\{(a,b,0) \mid a,b \in \mathsf{Ind}(\mathcal{A})\}$;
2: Compute $G_{\mathcal{U}_\mathcal{A}}$ and sp;
3: **for all** $(\mathfrak{p},\mathfrak{q},a) \in (Q_\mathfrak{R} \times Q_\mathfrak{T})^2 \times \mathsf{Ind}(\mathcal{A})$ **do**
4: add edge $(\mathfrak{p},a)$ $sp[\mathfrak{p},\mathfrak{q},a]$ $(\mathfrak{p},a)$ to $G_{\mathcal{U}_\mathcal{A}}$;
5: **end for**
6: Let $S=\{(s_0,t_0,a) \mid s_0 \in I_\mathfrak{R}, t_0 \in I_\mathfrak{T}, a \in \mathsf{Ind}(\mathcal{A})\}$;
7: Let $T=\{(s_f,t_f,b) \mid s_f \in F_\mathfrak{R}, t_f \in F_\mathfrak{T}, b \in \mathsf{Ind}(\mathcal{A})\}$;
8: Run a shortest path algorithm on $G_{\mathcal{U}_\mathcal{A}}$ with source and target sets S and T;
9: **return** $\mathfrak{A} := \{(a,b,\eta) \mid \eta$ is the minimal cost of a path from $(\_,\_,a) \in S$ to $(\_,\_,b) \in T\}$;

to obtain a corresponding algorithm. It uses a table sp containing all such minimal costs c^* in entries of the form $[(s,t),(s',t'),a]$, where $(s,t),(s',t') \in Q_\mathfrak{R} \times Q_\mathfrak{T}$ and $a \in \mathsf{Ind}(\mathcal{A})$. Let us continue by explaining how to compute sp.

By definition of $\mathcal{U}_\mathcal{K}$ and $G_{\mathcal{U}_\mathcal{K}}$, an a-path from (s,t,a) to (s',t',a) in $G_{\mathcal{U}_\mathcal{K}}$ must be of the form:

$$(s,t,a)\ w_1\ (s_1,t_1,aPC)\ \gamma\ (s_2,t_2,aPC)\ w_2\ (s',t',a), \tag{5}$$

where $(s_1,t_1,aPC)\ \gamma\ (s_2,t_2,aPC)$ is an aPC-path. Hence, to compute sp, it is enough to know the minimal cost of such an aPC-path. To this end, we use an additional table spa with entries of the form $[(s,t),(s',t'),C]$ where $C \in \mathsf{T}(\mathcal{U}_\mathcal{K})$. The intention is for such an entry to contain the minimal cost of an e-path from (s,t,e) to (s',t',e), where $e \in \Delta^{\mathcal{U}_\mathcal{K}} \setminus \mathsf{Ind}(\mathcal{A})$ and $\mathsf{tail}(e)=C$. Recall that for different individuals e,e' such that $\mathsf{tail}(e)=\mathsf{tail}(e')$, the trees T_e and $T_{e'}$ are isomorphic. This means that a path π in T_e has a corresponding one π' in $T_{e'}$ such that $\ell(\pi)=\ell(\pi')$. From this, it is not hard to see that $[(s,t),(s',t'),C]$ is well-defined. Based on these ideas, we now describe the computation of sp and spa.[4]

The form of an a-path given in (5) tells us that each value $sp[(s,t),(s',t'),a]$ corresponds to the minimal value of an expression of the form $w_1 + spa[(s_1,t_1),(s_2,t_2),C] + w_2$, such that the following conditions are satisfied, for $\mathcal{ELH}$ and DL-Lite$_\mathcal{R}$, respectively:

C1. $C \in \mathsf{N_C}$, $\mathcal{K} \models \exists r.C(a)$, $\mathcal{T} \models r \sqsubseteq r'$, $\mathcal{T} \models r^- \sqsubseteq r''$, $(s,u,s_1) \in \delta_\mathfrak{R}$, $(t,u,r',w_1,t_1) \in \delta_\mathfrak{T}$ $(s_2,u',s') \in \delta_\mathfrak{R}$, and $(t_2,u',r'',w_2,t') \in \delta_\mathfrak{T}$.

C2. $C = \exists P^-$, $\mathcal{K} \models \exists P(a)$, $\mathcal{T} \models P \sqsubseteq P'$, $\mathcal{T} \models P^- \sqsubseteq P''$, $(s,u,s_1) \in \delta_\mathfrak{R}$, $(t,u,P',w_1,t_1) \in \delta_\mathfrak{T}$ $(s_2,u',s') \in \delta_\mathfrak{R}$, and $(t_2,u',P'',w_2,t') \in \delta_\mathfrak{T}$.

Conditions C1 and C2 take into account the definitions of $\mathcal{U}_\mathcal{K}$ and $G_{\mathcal{U}_\mathcal{K}}$ to ensure that the edges $(s,t,a)w_1(s_1,t_1,aPC)$ and $(s_2,t_2,aPC)w_2(s',t',a)$ really exist. As for spa, its computation is described in procedure SPA[5], which applies the four rules listed below. Their definitions use expressions of the form $c \ll d$ meaning that: c is updated with the value of d iff $c < d$. The first two rules play the same role as the expressions use to define sp, i.e., they cover e-paths of the form (5) for $e \in \Delta^{\mathcal{U}_\mathcal{K}} \setminus \mathsf{Ind}(\mathcal{A})$.

S1 $spa[(s,t),(s',t'),C] \ll w_1 + spa[(s_1,t_1),(s_2,t_2),A] + w_2$, if C1* holds.

S2 $spa[(s,t),(s',t'),C] \ll w_1 + spa[(s_1,t_1),(s_2,t_2),\exists P^-] + w_2$, if C2* holds.

[4]These two tables can be seen as generalizations of the tables ALoop_α and Loop_α from [BOS15].
[5]This procedure is not meant to be efficient, but to illustrate that spa can be computed in polynomial time.

The conditions C1* and C2* are the variants of C1 and C2 where $\mathcal{T} \models C \sqsubseteq \exists r.A$ and $\mathcal{T} \models C \sqsubseteq \exists P$ are used instead of $\mathcal{K} \models ...$, respectively. We say that S{1,2} is *applicable* if C{1,2}* holds. The remaining two rules are:

S3 $spa[(s,t),(s',t'),C] \ll w$, if $\mathcal{T} \models C \sqsubseteq A$, $(s,u,s') \in \delta_{\mathfrak{R}}$ and $(t,u,A?,w,t') \in \delta_{\mathfrak{T}}$.

S4 $spa[(s,t),(s',t'),C] \ll spa[(s,t),(s'',t''),C] + spa[(s'',t''),(s',t'),C]$.

The rule S3 is used to take into account paths in $G_{\mathcal{U}_\mathcal{K}}$ of the form $(s,t,e)w(s',t',e)$. As for S4, it considers path that results from composing paths of the previous form and paths of the form (5).

Procedure SPA

1: Initialize $spa[p,p,C] = 0$;
2: Initialize $spa[p,q,C] = \infty$ (if $p \neq q$);
3: Apply rule S3 to all $(\mathfrak{p},\mathfrak{q},C) \in (Q_{\mathfrak{R}} \times Q_{\mathfrak{T}})^2 \times \mathsf{T}(\mathcal{U}_\mathcal{K})$;
4: Apply S4 until *spa* does not change;
5: **repeat**
6: $\quad spa := f(spa)$;
7: **until** *spa* does not change

function f

Apply rule S1(2) to all $(\mathfrak{p},\mathfrak{q},C) \in (Q_{\mathfrak{R}} \times Q_{\mathfrak{T}})^2 \times \mathsf{T}(\mathcal{U}_\mathcal{K})$;
Apply S4 until *spa* does not change;

end function

By looking at the rules, one can see that along a run of SPA an entry of *spa* can only have values that are either 0, ∞ or a sum of weights from $\mathfrak{T}$. Hence, since there are finitely many weights and the value of an entry in *spa* can only decrease, an exhaustive application of S4 always terminates. This means that a fixpoint of f can be reached in finite time, and therefore SPA always terminates. To see that this fixpoint contains the intended values for *spa*, we now prove two lemmas which depend on the following notion. The depth of a path π in $G_{\mathcal{U}_\mathcal{K}}$ is the maximal value $\mathfrak{d}(e') - \mathfrak{d}(e)$ such that $e' \in T_e$ and $(\_,\_,e')$ occurs in π.

Lemma 10. *Let $(s,t),(s',t') \in Q_{\mathfrak{R}} \times Q_{\mathfrak{T}}$, $e \in \Delta^{\mathcal{U}_\mathcal{K}} \setminus \mathsf{Ind}(\mathcal{A})$ and $\mathsf{tail}(e) = C$. After SPA executes line 4, $spa[(s,t),(s',t'),C]$ contains the minimal cost c^* of an e-path of depth 0 from (s,t,e) to (s',t',e).*

Proof. $\Rightarrow$: $spa[(s,t),(s',t'),C] \leq c^*$. An e-path π of depth 0 from (s,t,e) to (s',t',e) is of the form:

$$(s_1,t_1,e)w_1(s_2,t_2,e)w_2 \dots w_{n-1}(s_n,t_n,e),$$

where $n > 1$, $\{s,t\}_1 = \{s,t\}$ and $\{s,t\}_n = \{s',t'\}$. We use induction on the length n of the path to show that

$$spa[(s_1,t_1),(s_n,t_n),C] \leq c(\pi).$$

If $n = 2$, then $(s_1,t_1,e)w_1(s_2,t_2,e)$ is an edge in $G_{\mathcal{U}_\mathcal{K}}$. By definition of $G_{\mathcal{U}_\mathcal{K}}$, it must be that $(s_1,u,s_2) \in Q_{\mathfrak{R}}$, $(t_1,u,v,w_1,t_2) \in Q_{\mathfrak{T}}$, $v = A?$ and $e \in A^{\mathcal{U}_\mathcal{K}}$. The latter must be the case because in $\mathcal{U}_\mathcal{K}$ an anonymous individual has no single self-loops. In addition, by definition of $\mathcal{U}_\mathcal{K}$, $e \in A^{\mathcal{U}_\mathcal{K}}$ implies that $\mathcal{T} \models C \sqsubseteq A$. Hence, rule S3 can be applied to $((s_1,t_1),(s_2,t_2),C)$ w.r.t. A and w_1. This means that $spa[(s_1,t_1),(s_2,t_2),C] \leq c(\pi)$. Assume that $n > 2$ and let π_1 be the sub-path $(s_1,t_1,e)\dots(s_{n-1},t_{n-1},e)$ and π_2 the sub-path $(s_{n-1},t_{n-1},e)w_{n-1}(s_n,t_n,e)$. The application of induction yields $spa[(s_1,t_1),(s_{n-1},t_{n-1}),C] \leq c(\pi_1)$ and $spa[(s_{n-1},t_{n-1}),(s_n,t_n),C] \leq c(\pi_2)$. Hence, since S4 is applied exhaustively, it must

be that $spa[(s_1,t_1),(s_n,t_n),C] \leq c(\pi)$. Thus, since π is chosen arbitrarily, it follows that $spa[(s,t),(s',t'),C] \leq c^*$.

$\Leftarrow$: $c^* \leq spa[(s,t),(s',t'),C]$. Consider an execution of line 3, followed from an exhaustive application of S4 in line 4. Let $\mu_1,\ldots,\mu_k$ be the sequence of updates performed along this execution. We assume the sequence is in order, i.e., μ_1 is the first update and μ_k is the last one. In addition, we denote as $spa[(s^i,t^i),(s'^i,t'^i),C^i]$ the entry corresponding to the update μ_i and ν_i the updated value. Let now $e^i \in \Delta^{\mathcal{U}_\mathcal{K}} \setminus \mathsf{Ind}(\mathcal{A})$ such that $\mathsf{tail}(e^i) = C^i$. We show by induction on i that there exists an e^i-path of depth 0 in $G_{\mathcal{U}_\mathcal{K}}$ of the form $\pi_i = (s^i,t^i,e^i)\ldots(s'^i,t'^i,e^i)$ such that $c(\pi_i) \leq \nu_i$. We consider two cases:

- μ_i corresponds to an application of S3. In this case, we have that $\mathcal{T} \models C^i \sqsubseteq A$, $(s^i,u,s'^i) \in \mathfrak{R}_\delta$ and $(t^i,u,A?,w,t'^i) \in \delta_\mathfrak{T}$. Since $\mathsf{tail}(e^i) = C^i$, by definition of $\mathcal{U}_\mathcal{K}$ we have that $e^i \in A^{\mathcal{U}_\mathcal{K}}$. Hence, the definition of $G_{\mathcal{U}_\mathcal{K}}$ tells us that $(s^i,t^i,e^i)w(s'^i,t'^i,e^i)$ is an edge in $G_{\mathcal{U}_\mathcal{K}}$. This means that we have an e^i-path of cost w. Thus, since $\nu_i = w$, our claim holds.
- μ_i corresponds to an application of S4. This means that $\nu_i = \nu_{j_1} + \nu_{j_2}$, where $j_1, j_2 < i$. In addition, the entries corresponding to updates ν_{j_1} and ν_{j_2} must be of the form:

  $$spa[(s^i,t^i),(s'',t''),C^i] \qquad \text{and} \qquad spa[(s'',t''),(s'^i,t'^i),C^i].$$

  By induction, there are e^i-paths π_1 and π_2 of depth 0 in $G_{\mathcal{U}_\mathcal{K}}$ of the form $(s^i,t^i,e^i)\ldots(s'',t'',e^i)$ and $(s'',t'',e^i)\ldots(s'^i,t'^i,e^i)$, respectively, such that $c(\pi_1) \leq \nu_{j_1}$ and $c(\pi_2) \leq \nu_{j_2}$. Combining these two paths we obtain an e^i-path of depth 0 in $G_{\mathcal{U}_\mathcal{K}}$ of the form $(s^i,t^i,e^i)\ldots(s'^i,t'^i,e^i)$ with cost $c(\pi_1) + c(\pi_2) \leq \nu_{j_1} + \nu_{j_2} = \nu_i$. Thus, our inductive claim is satisfied.

Overall, we have shown that after the execution of line 4, for each entry $spa[(s,t),(s',t'),C]$ there exists an e-path π of depth 0 in $G_{\mathcal{U}_\mathcal{K}}$ of the form $(s,t,e)\ldots(s',t',e)$ such that $\mathsf{tail}(e) = C$ and $c(\pi) \leq spa[(s,t),(s',t'),C]$. Thus, it follows that $c^* \leq spa[(s,t),(s',t'),C]$. □

This result extends to all depths $\mathfrak{d} > 0$ w.r.t. $\mathfrak{d}$ applications of f.

Lemma 11. *Let $\mathfrak{d} \geq 1$, $(s,t),(s',t') \in Q_\mathfrak{R} \times Q_\mathfrak{T}$, $e \in \Delta^{\mathcal{U}_\mathcal{K}} \setminus \mathsf{Ind}(\mathcal{A})$ and $\mathsf{tail}(e) = C$. After $\mathfrak{d}$ applications of f, $spa[(s,t),(s',t'),C]$ contains the minimal cost c^* of an e-path of depth at most $\mathfrak{d}$ from (s,t,e) to (s',t',e).*

Proof. Let us start by establishing a correspondence between applications of the rule S1 and certain edges in $G_{\mathcal{U}_\mathcal{K}}$ (the same correspondence can be obtain for S2 and DL-Lite$_\mathcal{R}$).

- If S1 is applicable to $(s,t),(s',t')$ and C, then condition C1* is satisfied. This means that $\mathcal{T} \models C \sqsubseteq \exists r.A$, $\mathcal{T} \models r \sqsubseteq r'$ and $\mathcal{T} \models r^- \sqsubseteq r''$. Hence, the definition of $\mathcal{U}_\mathcal{K}$ yields the existence of $e' \in \Delta^{\mathcal{U}_\mathcal{K}}$ such that $e \neq e'$, $\mathsf{tail}(e') = A$, $(e,e') \in r'^{\mathcal{U}_\mathcal{K}}$ and $(e',e) \in r''^{\mathcal{U}_\mathcal{K}}$. Then, the remaining requirements of C1* and the definition of $G_{\mathcal{U}_\mathcal{K}}$ yield two edges in $G_{\mathcal{U}_\mathcal{K}}$ of the form

  $$(s,t,e)\ w_1\ (s_1,t_1,e') \text{ and } (s_2,t_2,e')\ w_2\ (s',t',e). \tag{6}$$

- Conversely, assume that $(s,t,e)\ w_1\ (s_1,t_1,e')$ and $(s_2,t_2,e')\ w_2\ (s',t',e)$ are two edges in $G_{\mathcal{U}_\mathcal{K}}$ such that $e \neq e'$. By definition of $G_{\mathcal{U}_\mathcal{K}}$, we have that there are:
  - $(s,u_1,s_1) \in \mathfrak{R}_\delta$ and $(t,u_1,r',w_1,t_1) \in \delta_\mathfrak{T}$ such that $(e,e') \in r'^{\mathcal{U}_\mathcal{K}}$, and
  - $(s_2,u_2,s') \in \mathfrak{R}_\delta$ and $(t_2,u_2,r'',w_2,t') \in \delta_\mathfrak{T}$ such that $(e',e) \in r''^{\mathcal{U}_\mathcal{K}}$.

Let $\mathsf{tail}(e') = A$. By definition of $\mathcal{U}_\mathcal{K}$, there must exists $r \in \mathsf{N_R}$ such that $\mathcal{T} \models r \sqsubseteq r'$, $\mathcal{T} \models r^- \sqsubseteq r''$ and $\mathcal{T} \models C \sqsubseteq \exists r.A$. Hence, one can see that condition C1* is satisfied w.r.t. A, r' and r'' and S1 is applicable with right-hand side of the form:

$$w_1 + spa[(s_1, t_1), (s_2, t_2), A] + w_2. \tag{7}$$

Next, we show by well-founded induction on the number of iterations $\mathfrak{d}$, that $spa[(s,t)(s',t'), C] = c^*$. Let us start by showing that $spa[(s,t)(s',t'), C] \geq c^*$. Trivial cases are when $(s,t) = (s',t')$ or $spa[(s,t)(s',t'), C]$ is never updated. Otherwise, suppose that $spa[(s,t)(s',t'), C]$ is updated by an application of S1 or S4:

- S1. Then, there are two edges in $G_{\mathcal{U}_\mathcal{K}}$ of the form (6) with weights w_1 and w_2. Moreover, the individual e' is such that $\mathsf{tail}(e') = A$. By induction, $spa[(s_1, t_1), (s_2, t_2), A]$ is the minimal cost of an e'-path of depth at most $\mathfrak{d} - 1$ from (s_1, t_1, e') to (s_2, t_2, e'). Hence, there is an e-path of depth at most $\mathfrak{d}$ in $G_{\mathcal{U}_\mathcal{K}}$ of the form $(s,t,e) \dots (s',t',e)$ with cost $w_1 + spa[(s_1, t_1), (s_2, t_2), A] + w_2$. Since the latter is the number assigned to $spa[(s_1, t_1), (s_2, t_2), A]$ by the application of S1, it follows that $spa[(s,t)(s',t'), C] \geq c^*$.
- S4. The proof is the same as in Lemma 10 for paths of depth 0. In this case, we consider an execution of the two lines in f at the $\mathfrak{d}$-th iteration of f.

We continue by showing that $spa[(s,t)(s',t'), C] \leq c^*$. An e-path π of depth at most $\mathfrak{d}$ from (s,t,e) to (s',t',e) can have two possible forms:

- $(s,t,e)\, w_1\, (s_1,t_1,e')\, \gamma\, (s_2,t_2,e')\, w_2\, (s',t',e)$, where $(s_1,t_1,e')\, \gamma\, (s_2,t_2,e')$ is an e'-path of depth at most $\mathfrak{d}-1$ with cost c'. Then, the cost of π is $c = w_1 + c' + w_2$. Let $\mathsf{tail}(e') = A$. As explained above, S1 is applicable with right-hand side of the form (7). Moreover, the application of induction yields that $c' \geq spa[(s_1, t_1), (s_2, t_2), A]$. Thus, it follows that $c \geq spa[(s,t), (s',t'), C]$.
- $(s = s_1, t = t_1, e)\, \gamma_1\, (s_2, t_2, e)\, \gamma_2\, \dots \gamma_{n-1}\, (s' = s_n, t' = t'_n, e)$, where for all $1 \leq i < n$ either $\gamma_i \in \mathbb{N}$ or $(s_i, t_i, e)\, \gamma_i\, (s_{i+1}, t_{i+1}, e)$ is an e-path of the previous form. Let c_i be the cost of each path $(s_i, t_i, e)\gamma_i(s_{i+1}, t_{i+1}, e)$. Then, the cost c of the whole path is $c_1 + \ldots + c_{n-1}$. We consider the following two cases:
  - $\gamma_i \in \mathbb{N}$. Then, $(s_i, t_i, e)\gamma_i(s_{i+1}, t_{i+1}, e)$ has depth 0. Hence, the application of Lemma 10 yields that $c_i \geq spa[(s_i, t_i), (s_{i+1}, t_{i+1}), C]$.
  - $(s_i, t_i, e)\, \gamma_i\, (s_{i+1}, t_{i+1}, e)$ is an e-path with the form considered in the previous case. Then, using the result shown for such paths, we obtain that $c_i \geq spa[(s_i, t_i), (s_{i+1}, t_{i+1}), C]$.

  Notice now, that an application of the rule S4 using the entries

  $$spa[(s_1, t_1), (s_2, t_2), C] \text{ and } spa[(s_2, t_2), (s_3, t_3), C]$$

  implies that $c_1 + c_2 \geq spa[(s_1, t_1), (s_3, t_3), C]$. Similarly, extending this to $(s_1, t_1), (s_3, t_3)$ and $(s_3, t_3), (s_4, t_4)$ yields that $c_1 + c_2 + c_3 \geq spa[(s_1, t_1), (s_4, t_4), C]$ Hence, it is not hard to see that $c_1 + \ldots + c_{n+1} \geq spa[(s_1, t_1), (s_n, t_n), C]$. Thus, $c \geq spa[(s_1, t_1), (s_n, t_n), C]$.

Finally, since the e-path π is chosen arbitrarily, it follows that $c^* \geq spa[(s,t), (s',t'), C]$. □

Using the previous two lemmas, it is not hard to show that *spa* and *sp* contain the intended values.

Lemma 12. *Let $(s,t),(s',t') \in Q_{\mathfrak{R}} \times Q_{\mathfrak{T}}$ and $C \in \mathsf{T}(\mathcal{U}_\mathcal{K})$. For all $e \in \Delta^{\mathcal{U}_\mathcal{K}} \setminus \mathsf{Ind}(\mathcal{A})$ such that $\mathsf{tail}(e) = C$, the value $spa[(s,t),(s',t'),C]$ is the minimal cost of an e-path in $G_{\mathcal{U}_\mathcal{K}}$ of the form $(s,t,e)\dots(s',t',e)$.*

Proof. Assume that f performs $i+1$ iterations. By Lemmas 10 and 11, after i iterations of f, the entry $spa[(s,t),(s',t'),C]$ contains the minimal cost of an e-path in $G_{\mathcal{U}_\mathcal{K}}$ of the form $(s,t,e)\dots(s',t',e)$ of depth at most i. Suppose now that there is a path π in $G_{\mathcal{U}_\mathcal{K}}$ of the same form such that $c(\pi) < spa[(s,t),(s',t'),C]$ and π has depth $j > i$. We show, by well-founded induction on j that this cannot be the case.

By definition of an e-path, π must contain a sub-path of the form

$$(s_1,t_1,e)\; w_1\; (s'_1,t'_1,ePC')\; \gamma\; (s'_2,t'_2,ePC')\; w_2\; (s_2,t_2,e),$$

where $\pi' = (s'_1,t'_1,ePC')\; \gamma\; (s'_2,t'_2,ePC')$ is an ePC'-path of depth $j' \geq i$. By Lemmas 10 and 11 if $j' = i$ or induction if $j' > i$, we obtain that $[(s'_1,t'_1),(s'_2,t'_2),C'] \leq c(\pi')$. Hence, $w_1 + [(s'_1,t'_1),(s'_2,t'_2),C'] + w_2 \leq c(\pi) < spa[(s,t),(s',t'),C]$. Thus, it would have been possible to update $spa[(s,t),(s',t'),C]$ by applying S1(2) at iteration $i+1$ of f, contradicting the assumption that f performs $i+1$ iterations. □

Hence, it follows that Algorithm 1 is correct.

Lemma 13. *Algorithm 1 computes the set $\tilde{\mathsf{ans}}_{\mathfrak{T}}(q,\mathcal{K})$.*

Proof. Since *sp* and *spa* contain the right values, we know that after executing the **for** loop at line 3 the augmented graph $G_{\mathcal{U}_\mathcal{A}}$ is the same as $G^*_{\mathcal{U}_\mathcal{A}}$ in Proposition 9. Now, $(a,b,\eta) \in \mathfrak{A}$ iff the minimal cost of a path from a node (s_0,t_0,a) to a node (s_f,t_f,b) in $G^*_{\mathcal{U}_\mathcal{A}}$ is η. By Proposition 9, the latter is true iff η is also the minimal cost of such a path in $G_{\mathcal{U}_\mathcal{K}}$. Thus, the application of Lemma 7 concludes the proof. □

Regarding the running time of Algorithm 1, deciding consistency of a KB and the entailment checks needed to build $G^*_{\mathcal{U}_\mathcal{A}}$ are polynomial time problems in $\mathcal{ELH}$ and DL-Lite$_\mathcal{R}$ [BBL05, CDL+07]. Further, since $G^*_{\mathcal{U}_\mathcal{A}}$ is of size polynomial in the size of $\mathcal{A}$, running a polynomial time shortest path algorithm on $G^*_{\mathcal{U}_\mathcal{A}}$ remains in polynomial time. As for the computation of *spa*, it is only unclear whether a fixpoint of f can be reached in polynomial time. The following lemma proves that this is the case, regardless of how the weights in $\mathfrak{T}$ are encoded.

Lemma 14. *Let $(s,t),(s',t') \in Q_{\mathfrak{R}} \times Q_{\mathfrak{T}}$ and $e_0 \in \Delta^{\mathcal{U}_\mathcal{K}} \setminus \mathsf{Ind}(\mathcal{A})$ such that there is an e_0-path $(s,t,e_0)\dots(s',t',e_0)$ in $G_{\mathcal{U}_\mathcal{K}}$. Then, there is one such path of minimal cost with depth at most $m = (|Q_{\mathfrak{R}}| \times |Q_{\mathfrak{T}}|)^2 \times |\mathsf{T}(\mathcal{U}_\mathcal{K})|$.*

Proof. Let π_0 be an e_0-path of the form $(s,t,e_0)\dots(s',t',e_0)$ of minimal cost. If the depth of π is $\leq m$ we are done. Otherwise, there exists $e_{m+1} \in T_{e_0}$ such that $\mathfrak{d}(e_{m+1}) - \mathfrak{d}(e_0) = m+1$ and π visits a vertex of the form $(\_,\_,e_{m+1})$. Since T_{e_0} is a tree, there must exist individuals $e_1,\dots,e_m \in \Delta^{\mathcal{U}_\mathcal{K}} \setminus \mathsf{Ind}(\mathcal{A})$ such that for all $1 \leq i \leq m+1$:

- $e_i \in T_{e_i-1}$ and e_i has depth i in T_{e_0}, and
- π_0 visits at least one node of the form (s_i,t_i,e_i).

Consider a vertex $\mathrm{v}_{m+1} = (s_{m+1},t_{m+1},e_{m+1})$ in π_0. Since $e_{m+1} \in T_{e_m}$, by construction of $G_{\mathcal{U}_\mathcal{K}}$, v_m must have a predecessor v_m and a successor v'_m in π_0 of the form (s_m,t_m,e_m) and (s'_m,t'_m,e_m), respectively. Let us assume that v_m and v'_m are the closest such vertices to v_{m+1} in

π_0. This means that $\pi_m = \mathrm{v}_m\, \gamma_1\, \mathrm{v}_{m+1}\, \gamma_2\, \mathrm{v}'_m$ is a sub-path of π_0 such that: all vertices $(\_, \_, e')$ occurring in γ_1 or γ_2 satisfy $e' \in T_{e_m}$. Hence, π_m is an e_m-path in $G_{\mathcal{U}_\mathcal{K}}$. Using the same reasoning, one can find vertices $\mathrm{v}_{m-1} = (s_{m-1}, t_{m-1}, e_{m-1})$ and $\mathrm{v}'_{m-1} = (s'_{m-1}, t'_{m-1}, e_{m-1})$ in π_0, such that they are the closest predecessor and successor of v_m and v'_m, respectively. Further, the path $\pi_{m-1} = \mathrm{v}_{m-1} \ldots \mathrm{v}'_{m-1}$ is an e_{m-1}-path containing π_m. Repeating this process until we reach e_0 yields a sequence $\pi_0, \ldots, \pi_m$ of sub-paths in π_0 such that:

- π_i is an e_i-path of the form $\mathrm{v}_i \ldots \pi_{i+1} \ldots \mathrm{v}'_i$ $(0 \leq i < m)$.

Since there are $m+1$ such paths, there must exist $0 \leq i < j \leq m$ such that $(s_i, t_i) = (s_j, t_j)$, $(s'_i, t'_i) = (s'_j, t'_j)$ and $\mathsf{tail}(e_i) = \mathsf{tail}(e_j)$. The latter means that there is an isomorphism *iso* between T_{e_i} and T_{e_j}. Hence, since π_j is an e_j-path we can apply *iso* to transform each vertex (s'', t'', e'') in π_j into the vertex $(s'', t'', iso(e''))$. This yields an e_i-path $\pi'_i = (s_i, t_i, e_i) \ldots (s'_i, t'_i, e_i)$ in $G_{\mathcal{U}_\mathcal{K}}$ with the same length and cost as π_j. As π_j is contained in π_i, replacing π_i by π'_i in π_0 yields an e_0-path $(s, t, e_0) \ldots \pi'_i \ldots (s', t', e_0)$ with smaller length and cost not greater than π_0. Thus, by repeating this process, we either find a contradiction (against π_0 being of minimal cost) or a proper path of depth not greater than m. □

This lemma implies that f performs at most m iterations, which is polynomial in the size of $\mathcal{T}$ and $\mathfrak{T}$. Hence, Algorithm 1 runs in time polynomial in the size of $\mathfrak{R}, \mathfrak{T}$ and $\mathcal{K}$. Thus, we obtain the following results.

Theorem 15. *For 2RPQs, computing the cost η of a certain approximate answer and deciding τ-entailment are in polynomial time for $\mathcal{ELH}$ and DL-Lite$_\mathcal{R}$.*

As explained in [GT06], by using Dijkstra's algorithm, Algorithm 1 can be adapted to compute the top-k certain approximate answers.

6 Answering approximate C2RPQs in DL-Lite$_\mathcal{R}$ and $\mathcal{ELH}$

In [BOS15], a query rewriting procedure is developed to answer C2RPQs under classical semantics in PSpace. One aspect of the rewriting is that the rewritten queries need not preserve the regular languages required by the atoms in the original query. For this reason, it is not clear how to reuse such a procedure to answer C2RPQs under approximate semantics.

Our solution is based on proving that we can restrict our attention to a finite fragment $\mathcal{U}_\mathcal{K}(q)$ of $\mathcal{U}_\mathcal{K}$ to decide τ-entailment of C2RPQs. This is the fragment of $\mathcal{U}_\mathcal{K}$ restricted to individuals $e \in \Delta^{\mathcal{U}_\mathcal{K}}$ of depth at most $g \cdot m + 1$, where:

$$g := p + 2 \qquad m := |\mathsf{T}(\mathcal{U}_\mathcal{K})| \cdot \prod_{j=1}^{p} (|Q_{\mathfrak{R}_j}| \cdot |Q_\mathfrak{T}|)^2.$$

The following lemma represents the main technical result of this section.

Lemma 16. *Let $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ be an $\mathcal{ELH}$ or a satisfiable DL-Lite$_\mathcal{R}$ KB, $q(\bar{x})$ a C2RPQ with p atoms, $\mathfrak{T}$ a dT and* f *a p-ary combining function. Then, $(\bar{a}, \eta) \in \check{\mathsf{cert}}_{\mathfrak{T},\mathrm{f}}(q, \mathcal{K})$ iff $\eta = min\{h_c \mid (h, h_c) \in H^{\mathcal{U}_\mathcal{K}(q)}_{\mathfrak{T},\mathrm{f}}(\bar{a})\}$.*

Before proceeding with the proof of Lemma 16, we need to introduce some notation. For the rest of this section, a C2RPQ with p atoms is a conjunction $\mathfrak{R}_1(t_1, t'_1) \wedge \ldots \wedge \mathfrak{R}_p(t_p, t'_p)$. For each of these atoms, the weighted graph $G_{\mathfrak{R}_j \times \mathfrak{T} \times \mathcal{U}_\mathcal{K}}$ is abbreviated as $G^j_{\mathcal{U}_\mathcal{K}}$. Further, given an approximate match (h, h_c) in $\mathcal{U}_\mathcal{K}$, we use π^h_j to denote a path in $G^j_{\mathcal{U}_\mathcal{K}}$ such that:

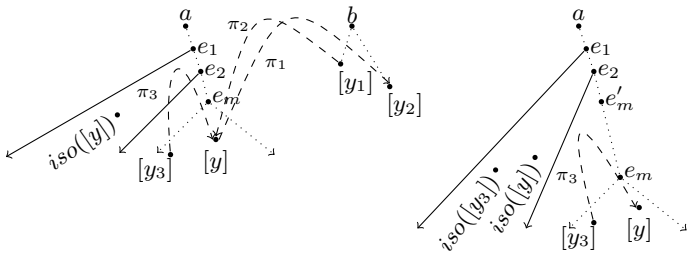

Figure 2: Intuition example.

- π_j^h is of the form $(s_0, t_0, h(t_j)) \dots (s_f, t_f, h(t'_j))$, and
- $c(\pi_j^h)$ is minimal among paths of the previous form.

We will simply write π_j when h is clear from the context. Notice that by (3), $h_c = \mathrm{f}(\pi_1, \dots, \pi_p)$. Finally, we denote as Π_h the set of all selected paths π_j's, and given a set $\mathcal{Y} \subseteq \mathsf{qvars}(q)$ we define $\Pi_h(\mathcal{Y}) := \{\pi_j \in \Pi_h \mid \{t_j, t'_j\} \cap \mathcal{Y} \neq \emptyset\}$.

6.1 Proof of Lemma 16

The main idea to prove Lemma 16 is that each approximate match not restricted to $\mathcal{U}'_{\mathcal{K}}(q)$ can be "made better" in the following sense.

Definition 17. Let $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ be an $\mathcal{ELH}$ or DL-Lite$_{\mathcal{R}}$ KB, $q(\bar{x})$ be a C2RPQ of arity k, $\bar{a}$ a k-tuple of individuals in $\mathcal{A}$, $y \in \mathsf{qvars}(q)$ and $(h, h_c) \in H^{q,\mathcal{U}_{\mathcal{K}}}_{\mathfrak{T},\mathrm{f}}(\bar{a})$. We say that (h, h_c) can be improved w.r.t. y iff there is $(h', h'_c) \in H^{q,\mathcal{U}_{\mathcal{K}}}_{\mathfrak{T},\mathrm{f}}(\bar{a})$ such that $h'_c \leq h_c$ and

- $\mathfrak{d}(h'(y)) < \mathfrak{d}(h(y))$ and $\mathfrak{d}(h'(z)) \leq \mathfrak{d}(h(z))$ for all $z \in \mathsf{qvars}(q)$.

The goal is then to show that every $(h, h_c) \in H^{q,\mathcal{U}_{\mathcal{K}}}_{\mathfrak{T},\mathrm{f}}(\bar{a})$, such that $\mathfrak{d}(h(y)) > g \cdot m + 1$ for some $y \in \mathsf{qvars}(q)$, can be improved w.r.t. y. We proceed in two steps. First, we identify two base cases for which depth $m+1$ and $2m+1$ suffices to improve (h, h_c). Secondly, we show that all other cases can be reduced to the basic ones. The following example illustrates the intuition behind the two base cases.

Example 18. Let $q(\bar{x}) = \mathfrak{R}_1(y_1, y) \wedge \mathfrak{R}_2(y, y_2)$ and $(h, h_c) \in H^{q,\mathcal{U}_{\mathcal{K}}}_{\mathfrak{T},\mathrm{f}}(\bar{a})$. The left-hand side of Figure 2 depicts a fragment of $\mathcal{U}_{\mathcal{K}}$ where $a, b \in \mathsf{Ind}(\mathcal{A})$. Symbols $[y_1], [y_2]$ are the element where h maps variables y_1, y_2 to. Further, the individuals e_1 and e_2 satisfy $\mathsf{tail}(e_1) = \mathsf{tail}(e_2)$, which means that there is an isomorphism *iso* between T_{e_1} and T_{e_2}. These sub-trees are symbolized with the continuous arrows. The *dotted* arrows represent just paths in $\mathcal{U}_{\mathcal{K}}$ and the *dashed lines* "symbolize" the paths π_1 and π_2 in $G^1_{\mathcal{U}_{\mathcal{K}}}$ and $G^2_{\mathcal{U}_{\mathcal{K}}}$. These paths are of the form $(s_0, t_0, h(y)) \dots (s_f, t_f, h(y_2))$ and $(s_0, t_0, h(y_1)) \dots (s_f, t_f, h(y))$, respectively, and they visit nodes of the form $(\_, \_, e_1)$ and $(\_, \_, e_2)$.

The first base case appears if the following are satisfied:

- π_1 visits a node of the form (s_l, t_l, e_1) and the last visited node involving e_2 is of the form (s_l, t_l, e_2).
- π_2 visits a node of the form $(s_\alpha, t_\alpha, e_1)$ and the first visited node involving e_2 is of the form $(s_\alpha, t_\alpha, e_2)$.

One can see that the sub-paths $(s_l, t_l, e_2) \dots (s_f, t_f, h(y))$ and $(s_f, t_f, h(y)) \dots (s_\alpha, t_\alpha, e_2)$ must only visit vertices $(\_, \_, e')$ such that $e' \in T_{e_2}$. Since T_{e_1} and T_{e_2} are isomorphic and $y \in T_{e_2}$, these sub-paths can be reproduced from/to e_1 w.r.t. $iso(h(y))$. This would yield paths π'_1 and π'_2 such that $c(\pi'_{1,2}) \leq c(\pi_{1,2})$, which implies that (h, h_c) can be improved by "moving up" $h(y)$ to $iso(h(y))$. Thus, to ensure that such elements e_1, e_2 exist one could, for example, require $\mathfrak{d}(e_m) = m + 1$.

Clearly, this is a very optimistic situation, since it leaves out many of the cases where h maps a variables related to y into the same sub-tree T_a as $h(y)$. The second base case consists of a particular type of instances of such cases.

Continuation of Example 18. Assume that q contains an additional atom $\mathfrak{R}_3(y_3, y)$ and $h(y_3) = [y_3]$. Further, suppose that π_3 visits a vertex $(\_, \_, e')$ such that $e' \notin T_{e_2}$ but none where $e' = e_1$. The latter means that π_3 need not be reproducible w.r.t. $iso(h(y))$, whereas the former implies that moving $h(y_3)$ up to $iso(h(y_3))$ is also not enough. To avoid this more depth is needed. For instance, suppose now that $\mathfrak{d}(e_m) = 2m + 1$ and $\mathfrak{d}(e'_m) = m + 1$ as depicted in the right-hand side of the figure. Then,

- elements e_1, e_2 can still be found between a and e'_m, and
- π_3 only visits vertices $(\_, \_, e')$ such that $e' \in T_{e_2}$.

Thus, (h, h_c) can be improved by moving up $h(y)$ and $h(y_3)$ to $(iso(h(y)))$ and $(iso(h(y_3)))$.

We now move on to formally prove that the intuition described in the previous example is correct. First, we introduce some needed technical notions.

Definition 19. Let $e, e' \in \Delta^{\mathcal{U}_\mathcal{K}}$ such that $e' \in T_e$ and π a path in $G^j_{\mathcal{U}_\mathcal{K}}$.

- The segment $[e, e']$ in $\mathcal{U}_\mathcal{K}$ consists of all $e'' \in T_e$ such that $e' \in T_{e''}$.
- We say that π covers $[e, e']$ iff π visits a node of the form (s, t, e'') for all $e'' \in [e, e']$.

A segment captures a finite sequence of consecutive elements $e, eP_1C_1, eP_1C_1P_2C_2 \dots$ in $\mathcal{U}_\mathcal{K}$. In the example, the segment $[e_1, e_2]$ is covered by the paths π_1 and π_2 but not by π_3. The following auxiliary lemma shows that shortening a set of paths simultaneously is possible when they satisfy the conditions sketched for the first base case.

Lemma 20. *Let $e_1, e_2, e_3 \in \Delta^{\mathcal{U}_\mathcal{K}} \setminus \mathsf{Ind}(\mathcal{A})$ such that $e_1 \neq e_2$, $\mathsf{tail}(e_1) = \mathsf{tail}(e_2)$, $e_2 \in T_{e_1}$, $e_3 \in T_{e_2}$ and iso an isomorphism between T_{e_1} and T_{e_2}. In addition, let $e_4 \in \Delta^{\mathcal{U}_\mathcal{K}}$ and π be a path in $G^j_{\mathcal{U}_\mathcal{K}}$ from a node (s, t, e_3) to a node (s', t', e_4) (or vice versa) that covers $[e_1, e_2]$. Further, let $(s^{e_i}_\alpha, t^{e_i}_\alpha, e_i)$ and $(s^{e_i}_l, t^{e_i}_l, e_i)$ be the first and last node visited by π of the form $(\_, \_, e_{i=1,2})$, respectively.*

Then, $(s^{e_1}_\alpha, t^{e_1}_\alpha) = (s^{e_2}_\alpha, t^{e_2}_\alpha)$ and $(s^{e_1}_l, t^{e_1}_l) = (s^{e_2}_l, t^{e_2}_l)$ imply the existence of a path π' in $G^j_{\mathcal{U}_\mathcal{K}}$ such that $c(\pi') \leq c(\pi)$ and π' is of the form:

- *$(s, t, iso(e_3)) \dots (s', t', e_4)$ (or vice versa), if $e_3 \neq e_4$, or*
- *$(s, t, iso(e_3)) \dots (s', t', iso(e_4))$ (or vice versa), if $e_4 \in T_{e_2}$.*

Proof. First, assume that $e_4 \notin T_{e_2}$. We consider two cases.

- π goes from (s,t,e_3) to (s',t',e_4). Since $e_3 \in T_{e_2}$, $e_2 \in T_{e_1}$ and π covers $[e_1,e_2]$, π can be decomposed as a path of the form:

$$(s,t,e_3)\gamma(s_\alpha^{e_2},t_\alpha^{e_2},e_2)\dots(s_\alpha^{e_1},t_\alpha^{e_1},e_1)\dots(s',t',e_4)$$

  such that γ only visits nodes of the form $(\_,\_,e'')$ where $e'' \in T_{e_2}$. Now, let γ' be the result of replacing each (s'',t'',e'') in γ by $(s'',t'',iso(e''))$. Since all such e'' are in T_{e_2} and iso is an isomorphism between T_{e_1} and T_{e_2}, we obtain a path $(s,t,iso(e_3))\,\gamma'\,(s_\alpha^{e_2},t_\alpha^{e_2},e_1)$ in $G^j_{\mathcal{U}_\mathcal{K}}$ with the same cost as $(s,t,e_3)\,\gamma\,(s_\alpha^{e_2},t_\alpha^{e_2},e_2)$. Thus, since $(s_\alpha^{e_1},t_\alpha^{e_1}) = (s_\alpha^{e_2},t_\alpha^{e_2})$, this means that $\pi' = (s,t,iso(e_3))\,\gamma'\,(s_\alpha^{e_1},t_\alpha^{e_1},e_1)\dots(s',t',e_4)$ is a path in $G^j_{\mathcal{U}_\mathcal{K}}$ such that $c(\pi') \leq c(\pi)$.

- π goes from (s',t',e_4) to (s,t,e_3). In this case, π can be decomposed as:

$$(s',t',e_4)\dots(s_l^{e_1},t_l^{e_1},e_1)\dots(s_l^{e_2},t_l^{e_2},e_2)\,\gamma\,(s,t,e_3)$$

  such that γ only visits nodes of the form $(\_,\_,e'')$ where $e'' \in T_{e_2}$. Employing similar arguments as above, we obtain a path $(s_l^{e_2},t_l^{e_2},e_1)\,\gamma'\,(s,t,iso(e_3))$ in $G^j_{\mathcal{U}_\mathcal{K}}$ with the same cost as $(s_l^{e_2},t_l^{e_2},e_2)\,\gamma\,(s,t,e_3)$. Again, since $(s_l^{e_1},t_l^{e_1}) = (s_l^{e_2},t_l^{e_2})$, this means that:

$$\pi' = (s',t',e_4)\dots(s_l^{e_1},t_l^{e_1},e_1)\,\gamma'\,(s,t,iso(e_3))$$

  is a path in $G^j_{\mathcal{U}_\mathcal{K}}$ such that $c(\pi') \leq c(\pi)$.

It then remains to look at the case where $e_4 \in T_{e_2}$. The path π has the following form:

$$(s,t,e_3)\gamma_1(s_\alpha^{e_2},t_\alpha^{e_2},e_2)\dots(s_\alpha^{e_1},t_\alpha^{e_1},e_1)\dots(s_l^{e_1},t_l^{e_1},e_1)\dots(s_l^{e_2},t_l^{e_2},e_2)\gamma_2(s',t',e_4),$$

where γ_1,γ_2 only visit nodes of the form (s'',t'',e'') such that $e'' \in T_{e_2}$. Applying the same arguments as for $e_4 \notin T_{e_2}$, we can obtain paths $(s,t,iso(e_3))\gamma_1'(s_\alpha^{e_2},t_\alpha^{e_2},e_1)$ and $(s_l^{e_2},t_l^{e_2},e_1)\gamma_2'(s',t',iso(e_4))$ with the same cost as $(s,t,e_3)\gamma_1(s_\alpha^{e_2},t_\alpha^{e_2},e_2)$ and $(s_l^{e_2},t_l^{e_2},e_2)\gamma_2(s',t',e_4)$, respectively. Hence, $(s_l^{e_1},t_l^{e_1}) = (s_l^{e_2},t_l^{e_2})$ and $(s_\alpha^{e_1},t_\alpha^{e_1}) = (s_\alpha^{e_2},t_\alpha^{e_2})$ yield a path

$$\pi' = (s,t,iso(e_3))\gamma_1'(s_\alpha^{e_1},t_\alpha^{e_1},e_1)\dots(s_l^{e_1},t_l^{e_1},e_1)\gamma_2'(s',t',iso(e_4))$$

such that $c(\pi') \leq c(\pi)$. □

Using the previous lemma, we can now show that an approximate match can be improved for any of the two described bases cases.

Lemma 21. *Let $\mathcal{K} = (\mathcal{T},\mathcal{A})$ be an $\mathcal{ELH}$ or a DL-Lite$_\mathcal{R}$ KB, $q(\bar{x})$ be a C2RPQ of arity k, $\bar{a}$ a k-tuple of individuals in $\mathcal{A}$ and $(h,h_c) \in H^{q,\mathcal{U}_\mathcal{K}}_{\mathfrak{T},\mathfrak{f}}(\bar{a})$. In addition, let $e_0,e_m,e_{2m} \in \Delta^{\mathcal{U}_\mathcal{K}} \setminus \mathsf{Ind}(\mathcal{A})$ such that $e_m \in T_{e_0}$, $e_{2m} \in T_{e_m}$, $\mathfrak{d}(e_m) - \mathfrak{d}(e_0) = \mathfrak{d}(e_{2m}) - \mathfrak{d}(e_m) = m$, and $\mathcal{Y} \subseteq \mathsf{qvars}(q)$ such that $y \in \mathcal{Y} \implies h(y) \in T_{e_{2m}}$. If each $\pi_j \in \Pi_h(\mathcal{Y})$ satisfies that:*

1) *π_j covers $[e_0,e_m]$, or*
2) *π_j does not cover $[e_m,e_{2m}]$ and $\{t_j,t_j'\} \subseteq \mathcal{Y}$,*

then (h,h_c) can be improved w.r.t. all $y \in \mathcal{Y}$.

Proof. Notice that each path in $\Pi_h(\mathcal{Y})$ satisfies exactly one among conditions 1) and 2). Assume that all $\pi_j \in \Pi_h(\mathcal{Y})$ satisfy either condition 1) or 2). Let $\Pi^1_h(\mathcal{Y})$ be the set of paths in $\Pi_h(Y)$ satisfying condition 1) and $\Pi^2_h(\mathcal{Y})$ the set of those satisfying condition 2). We start by considering the case where $\Pi^1_h(\mathcal{Y}),\Pi^2_h(\mathcal{Y}) \neq \emptyset$.

For each path $\pi_j \in \Pi^1_h(\mathcal{Y})$ and $e \in [e_0,e_m]$, we denote as $(s_\alpha^{e,j},t_\alpha^{e,j},e)$ and $(s_l^{e,j},t_l^{e,j},e)$ the first and last node of the form $(\_,\_,e)$, respectively, visited by π_j. Since $[e_0,e_m]$ contains $m+1$ elements, there must exist $e_1,e_2 \in [e_0,e_m]$ such that:

- $e_1 \neq e_2$, $e_2 \in T_{e_1}$, $\mathsf{tail}(e_1) = \mathsf{tail}(e_2)$, and
- $(s_\alpha^{e_1,j}, t_\alpha^{e_1,j}) = (s_\alpha^{e_2,j}, t_\alpha^{e_2,j})$ and $(s_l^{e_1,j}, t_l^{e_1,j}) = (s_l^{e_2,j}, t_l^{e_2,j})$.

Let now iso be an isomorphism between T_{e_1} and T_{e_2} (recall that it exists because $\mathsf{tail}(e_1) = \mathsf{tail}(e_2)$). Since $y \in \mathcal{Y} \implies h(y) \in T_{e_{2m}}$, we have $y \in \mathcal{Y} \implies h(y) \in T_{e_2}$. Hence, we can transform h into a new mapping h' by setting $h'(y) = iso(h(y))$ for all $y \in \mathcal{Y}$ and $h'(y') = h(y')$ for the rest of the variables in $\mathsf{vars}(q)$. We show that (h, h_c) is improved by (h', h'_c) w.r.t. all $y \in \mathcal{Y}$.

First, notice that $e_1, e_2 \in [e_0, e_m]$ and $h(y) \in T_{e_2}$ imply $iso(h(y)) \in T_{e_1}$ for all $y \in \mathcal{Y}$. Secondly, since $e_1 \neq e_2$ and $e_2 \in T_{e_1}$, it follows that $\mathfrak{d}(e_1) < \mathfrak{d}(e_2)$. Hence, $\mathfrak{d}(h'(y)) < \mathfrak{d}(h(y))$ for all $y \in \mathcal{Y}$. Last, by definition, $h'(y') = h(y')$ for all remaining $y' \in \mathsf{vars}(q)$. Consequently, it remains to show that $h'_c \leq h_c$. To this end, we prove that for each $\mathfrak{R}_j(t_j, t'_j) \in q$ there is path π'_j from $(s_0, t_0, h'(t_j))$ to $(s_f, t_f, h'(t'_j))$ such that $c(\pi'_j) \leq c(\pi_j)$. We consider the following three cases.

- $\pi_j \notin \Pi_h(\mathcal{Y})$. This is trivial, since $h(t_j) = h'(t_j)$ and $h(t'_j) = h'(t'_j)$.
- $\pi_j \in \Pi_h^1(\mathcal{Y})$. This case follows from Lemma 20.
- $\pi_j \in \Pi_h^2(\mathcal{Y})$. As $t_j, t'_j \in \mathcal{Y}$ we have that $h(t_j), h(t'_j) \in T_{e_{2m}}$. Therefore, since π_j does not cover $[e_m, e_{2m}]$ and $e_m \in T_{e_2}$, π_j must be of the form $(s_0, t_0, h(t_j))\gamma(s_f, t_f, h(t'_j))$ such that every vertex (s, t, e'') in γ satisfies $e'' \in T_{e_2}$. Hence, the fact that T_{e_1} and T_{e_2} are isomorphic implies that a path $\pi'_j = (s_0, t_0, iso(h(t_j))) \dots (s_f, t_f, iso(h(t'_j)))$ exists in $G^j_{\mathcal{U}_\mathcal{K}}$ such that $c(\pi'_j) \leq c(\pi_j)$.

Thus, since f is a monotonic function, it follows that $h'_c \leq h_c$.

To conclude let us look at the two remaining cases. If $\Pi_h^2(\mathcal{Y}) = \emptyset$, we can just apply Lemma 20 to each $\pi_j \in \Pi_h^1(\mathcal{Y})$ to prove our claim. Otherwise, suppose that $\Pi_h^1(\mathcal{Y}) = \emptyset$. Taking into account the size of $\mathsf{T}(\mathcal{U}_\mathcal{K})$, it is clear that there are $e_1, e_2 \in [e_0, e_m]$ such that $e_1 \neq e_2$, $e_2 \in T_{e_1}$ and $\mathsf{tail}(e_1) = \mathsf{tail}(e_2)$. The same arguments used in the third case above can be applied to prove our claim. □

We are now ready to show that, for the general case, an approximate match can always be improved.

Lemma 22. *Let $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ be an $\mathcal{ELH}$ or DL-Lite$_\mathcal{R}$ KB, $q(\bar{x})$ be a C2RPQ of arity k, $\bar{a}$ a k-tuple of individuals in $\mathcal{A}$, $(h, h_c) \in H^{q,\mathcal{U}_\mathcal{K}}_{\mathfrak{T},\mathrm{f}}(\bar{a})$, and $y \in \mathsf{qvars}(q)$ such that $\mathfrak{d}(h(y)) > g \cdot m + 1$. Then, (h, h_c) can be improved w.r.t. y.*

Proof. Since $\mathfrak{d}(h(y)) > g \cdot m + 1$, there are elements $e_0, e_m, e_{2\cdot m}, \dots e_{g\cdot m}$ in $\Delta^{\mathcal{U}_\mathcal{K}} \setminus \mathsf{Ind}(\mathcal{A})$ such that:

- $e_{\ell\cdot m} \neq e_{i\cdot m}$ and $\mathfrak{d}(e_{i\cdot m}) - \mathfrak{d}(e_{(i-1)\cdot m}) = m$, $(0 \leq \ell < i \leq g)$.
- $e_{i\cdot m} \in T_{e_{(i-1)\cdot m}}$ and $h(y) \in T_{g\cdot m}$, $(0 < i \leq g)$.

For each $2 \leq i \leq g$, we will now define a set $\mathcal{Y}^i \subseteq \mathsf{qvars}(q)$. Except for $\mathcal{Y}^g$, each $\mathcal{Y}^i$ will be defined in terms of $\mathcal{Y}^{i+1}$ with the help of two of the following sets of paths:

- $\Pi_h^1(\mathcal{Y}^i)$ contains all paths in $\Pi_h(\mathcal{Y}^i)$ that cover $[e_{(i-2)\cdot m}, e_{(i-1)\cdot m}]$.
- $\Pi_h^2(\mathcal{Y}^i)$ contains all paths $\pi_j \in \Pi_h(\mathcal{Y}^i)$ such that:

- π_j does not cover $[e_{(i-1)\cdot m}, e_{i\cdot m}]$ and $\{t_j, t'_j\} \subseteq \mathcal{Y}^i$.

- $\Pi^3_h(\mathcal{Y}^i)$ contains all paths $\pi_j \in \Pi_h(\mathcal{Y}^i)$ such that:
  - π_j does not cover $[e_{(i-1)\cdot m}, e_{i\cdot m}]$ and $\{t_j, t'_j\} \not\subseteq \mathcal{Y}^i$.

Coming back to the sets $\mathcal{Y}^i$, we define $\mathcal{Y}^g := \{y\}$ and $\mathcal{Y}^i := \mathcal{Y}^{i+1} \cup \{t_j, t'_j \mid \pi_j \in \Pi^3_h(\mathcal{Y}^{i+1})\}$ for all $2 \leq i < g$. Notice that $\Pi^1_h(\mathcal{Y}^i)$, $\Pi^2_h(\mathcal{Y}^i)$ and $\Pi^3_h(\mathcal{Y}^i)$ are disjoint and their union is $\Pi_h(\mathcal{Y}^i)$. Using these sets, we will now prove that for some $2 \leq i \leq g$, the hypothesis in Lemma 21 are satisfied w.r.t. $\mathcal{Y}^i$. To this end, we show three auxiliary claims.

1. $h(z) \in T_{e_{i\cdot m}}$ for all $z \in \mathcal{Y}^i$. We use induction on $g - i$. For $i = g$, we know by assumption that $h(y) \in T_{g\cdot m}$. Suppose now that $i < g$ and let $z \in \mathcal{Y}^i$. By definition of $\mathcal{Y}^i$ we have to possibilities:
   - $z \in \mathcal{Y}^{i+1}$. Applying induction to $(g - (i+1))$ we obtain that $h(z) \in T_{e_{(i+1)\cdot m}}$. Thus, since $e_{(i+1)\cdot m} \in T_{e_{i\cdot m}}$, it follows that $h(z) \in T_{e_{i\cdot m}}$.
   - $z = t_j$ or $z = t'_j$ for some $\pi_j \in \Pi^3_h(\mathcal{Y}^{i+1})$. Without loss of generality let us assume that $z = t_j$. Since $\Pi^3_h(\mathcal{Y}^{i+1}) \subseteq \Pi_h(\mathcal{Y}^{i+1})$, it must be that $t'_j \in \mathcal{Y}^{i+1}$. Then, applying induction we obtain that $h(t'_j) \in T_{e_{(i+1)\cdot m}}$. Since π_j is a path of the form $(s_0, t_0, h(z)) \dots (s_f, t_f, h(t'_j))$ and it does not covers $[e_{i\cdot m}, e_{(i+1)\cdot m}]$, it must be that $h(z) \in T_{e_{i\cdot m}}$.
2. $\Pi^2_h(\mathcal{Y}^i) \subseteq \Pi^2_h(\mathcal{Y}^{i-1})$ for all $1 \leq i \leq g$. Let $\pi_j \in \Pi^2_h(\mathcal{Y}^i)$. By definition of $\Pi^2_h(\mathcal{Y}^i)$ we have that $t_j, t'_j \in \mathcal{Y}^i$. In addition, from the previous result we also know that $h(t_i), h(t'_i) \in T_{e_{i\cdot m}}$. This means that, since π_j does not cover $[e_{(i-1)\cdot m}, e_{i\cdot m}]$, π_j cannot visit nodes of the form (s, t, f) such that $\mathfrak{d}(f) \leq \mathfrak{d}(e_{(i-1)\cdot m})$. Thus, π_j does not cover $[e_{(i-2)\cdot m}, e_{(i-1)\cdot m}]$, which means that $\pi_j \in \Pi^2_h(\mathcal{Y}^{i-1})$.
3. $\Pi^3_h(\mathcal{Y}^i) \neq \emptyset \implies \Pi^2_h(\mathcal{Y}^i) \subset \Pi^2_h(\mathcal{Y}^{i-1})$. Assume that $\Pi^3_h(\mathcal{Y}^i) \neq \emptyset$ and let $\pi_j \in \Pi^3_h(\mathcal{Y}^i)$. Without loss of generality let us assume that $t_j \in \mathcal{Y}^i$ and $t'_j \notin \mathcal{Y}^i$. From the first claim above we know that $h(t_j) \in T_{e_{i\cdot m}}$. Hence, since π_j does not cover $[e_{(i-1)\cdot m}, e_{i\cdot m}]$, it follows that it does not cover $[e_{(i-2)\cdot m}, e_{(i-1)\cdot m}]$ Now, by definition of $\mathcal{Y}^{i-1}$ we have that $t'_j \in \mathcal{Y}^{i-1}$. Thus, $\pi_j \in \Pi^2_h(\mathcal{Y}^{i-1})$.

Since $\Pi^3_h(\mathcal{Y}^i) \cap \Pi^2_h(\mathcal{Y}^i) = \emptyset$, q has p atoms and $g = p + 2$, claims 2. and 3. above imply that there is $2 \leq i \leq g$ such that $\Pi^3_h(\mathcal{Y}^i) = \emptyset$. By claim 1., we know that $h(z) \in T_{e_{i\cdot m}}$ for all $z \in \mathcal{Y}^i$. Hence, one can see that the hypothesis of Lemma 21 are satisfied w.r.t. $\mathcal{Y}^i, e_{i\cdot m}, e_{(i-1)\cdot m}$ and $e_{(i-2)\cdot m}$. Thus, since $y \in \mathcal{Y}^i$, (h, h_c) can be improved w.r.t. y. □

Using this result one can see that Lemma 16 holds. More precisely, let $(\bar{a}, \eta) \in \tilde{\mathsf{cert}}_{\mathfrak{T},\mathrm{f}}(q, \mathcal{K})$ and $(h, h_c) \in H^{q,\mathcal{U}_\mathcal{K}}_{\mathfrak{T},\mathrm{f}}(\bar{a})$ an approximate match such that $h_c = \eta$. If $\mathfrak{d}(h(y)) > g \cdot m + 1$ for some variables $y \in \mathsf{qvars}(q)$, we can apply Lemma 22 finitely many times until we reach an approximate match $(h', h'_c) \in H^{q,\mathcal{U}_\mathcal{K}}_{\mathfrak{T},\mathrm{f}}(\bar{a})$ such that $h'_c \leq h_c$ and $\mathfrak{d}(h(y)) \leq p \cdot m + 1$ for all variables $y \in \mathsf{qvars}(q)$. This is a match in $\mathcal{U}_\mathcal{K}(q)$ and minimality of h_c implies that $h'_c = \eta$.

Once we have Lemma 16, we can decide τ-entailment as follows. Given $\bar{a} \in \mathsf{Ind}(\mathcal{A})$ and $\tau \in \mathbb{N}$:

1. Build the fragment $\mathcal{U}_\mathcal{K}(q)$ of $\mathcal{U}_\mathcal{K}$.
2. For all mappings $h : \mathsf{vars}(q) \mapsto \Delta^{\mathcal{U}_\mathcal{K}(q)}$ such that $h(\bar{x}) = \bar{a}$:
   (a) compute $c(\pi^h_j)$ for each $\mathfrak{R}_j(t_j, t'_j) \in q$.
   (b) compute $h_c = \mathrm{f}(\pi^h_1, \dots, \pi^h_p)$.

(c) **return** *yes* if $h_c \leq \tau$

3. **return** *no*.

The number of elements in $\mathcal{U}_\mathcal{K}(q)$ is bounded by $\nu := |\mathcal{A}| \cdot |\mathcal{T}|^{g \cdot m+1}$, and hence the size of $\mathcal{U}_\mathcal{K}(q)$ is polynomial in the size of $\mathcal{A}$. The computation of $c(\pi_j^h)$ is done by considering the whole $\mathcal{U}_\mathcal{K}(q)$ as an ABox, and then running Algorithm 1 to find the distortion cost of $(h(t_j), h(t'_j))$. Thus, since there are at most $\nu^{|q|}$ different mappings h, the sketched procedure runs in polynomial time in the size of the ABox. This means that τ-entailment is in PTime w.r.t. data complexity. Notice that a simple modification of the procedure yields an algorithm to compute $\tilde{\mathsf{ans}}_{\mathfrak{T},\mathrm{f}}(q, \mathcal{K})$.

Regarding combined complexity, the previous algorithm yields a *double exponential* time decision procedure for the τ-entailment problem. This upper bound can be improved as follows. First, notice that once a mapping h of q in $\mathcal{U}_\mathcal{K}(q)$ is fixed, the whole $\mathcal{U}_\mathcal{K}(q)$ is not really needed to compute h_c. It suffices to run Algorithm 1 on a fragment $\mathcal{U}_\mathcal{K}(q, h)$ consisting of $\mathcal{A}$ together with the paths from $\mathcal{A}$ to each element $h(y)$ such that $y \in \mathsf{qvars}(q)$. Such a fragment is of size at most $|\mathcal{A}| \cdot |q| \cdot (g \cdot m + 1)$. Secondly, h can be represented as a list of pairs of the form (x, a) or (y, e), where e can be an anonymous individual in $\mathcal{U}_\mathcal{K}(q)$. By definition of $\mathcal{U}_\mathcal{K}$, e can be represented as an unambiguous sequence of length at most $g \cdot m + 1$. This means that a list of such pairs can be guessed in exponential time. Hence, one can build $\mathcal{U}_\mathcal{K}(q, h)$ in exponential time and use steps (a), (b) and (c) to check whether $h_c \leq \tau$. This yields a non-deterministic exponential time decision procedure.

Theorem 23. *In $\mathcal{ELH}$ and DL-Lite$_\mathcal{R}$, τ-entailment of C2RPQs is in PTime and NExpTime, respectively, in data and combined complexity. In addition, the approximation cost η of a certain approximate answer can be computed in polynomial time (double exponential time) in data (combined) complexity.*

7 Conclusions

Approximate semantics are useful for applications requiring flexible query answering. In this paper, we have introduced such semantics for answering C2RPQs over $\mathcal{ELH}$ and DL-Lite$_\mathcal{R}$ ontologies We have extended an approach proposed in [GT06], that uses weighted transducers to define approximate semantics for RPQs in graph databases, to the more general query language of C2RPQs posed over graph databases and, more importantly over DL ontologies. Our approach is flexible to be adapted to different applications—as it can be parameterized with a transducer and a combining function.

We have developed algorithms for computing the certain approximate answers for 2RPQs and C2RPQs over $\mathcal{ELH}$ and DL-Lite$_\mathcal{R}$ontologies. The algorithms for C2RPQs are, to the best of our knowledge, the first for computing answers of such queries under approximate semantics in the presence of DL ontologies.

References

[BBL05] Franz Baader, Sebastian Brandt, and Carsten Lutz. Pushing the $\mathcal{EL}$ envelope. In *Proc. of the 19th Int. Joint Conf. on Artificial Intelligence (IJCAI 2005)*, pages 364–369. Professional Book Center, 2005.

[BOS15] Meghyn Bienvenu, Magdalena Ortiz, and Mantas Simkus. Regular path queries in lightweight description logics: Complexity and algorithms. *J. Artif. Intell. Res.*, 53:315–374, 2015.

[CDL+07] Diego Calvanese, Giuseppe De Giacomo, Domenico Lembo, Maurizio Lenzerini, and Riccardo Rosati. Tractable reasoning and efficient query answering in description logics: The *DL-Lite* family. *J. Autom. Reasoning*, 39(3):385–429, 2007.

[CEO14] Diego Calvanese, Thomas Eiter, and Magdalena Ortiz. Answering regular path queries in expressive description logics via alternating tree-automata. *Inf. Comput.*, 237:12–55, 2014.

[CM90] Mariano P. Consens and Alberto O. Mendelzon. Graphlog: a visual formalism for real life recursion. In *Proc. of the 9th ACM SIGACT SIGMOD SIGART Symp. on Principles of Database Systems (PODS'90)*, pages 404–416. ACM Press, 1990.

[EPT15] Andreas Ecke, Rafael Peñaloza, and Anni-Yasmin Turhan. Similarity-based relaxed instance queries. *Journal of Applied Logic*, 13(4, Part 1):480–508, 2015. Special Issue for the Workshop on Weighted Logics for AI 2013.

[FLS98] Daniela Florescu, Alon Y. Levy, and Dan Suciu. Query containment for conjunctive queries with regular expressions. In Alberto O. Mendelzon and Jan Paredaens, editors, *Proceedings of the Seventeenth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems, June 1-3, 1998, Seattle, Washington, USA*, pages 139–148. ACM Press, 1998.

[GT06] Gösta Grahne and Alex Thomo. Regular path queries under approximate semantics. *Ann. Math. Artif. Intell.*, 46(1-2):165–190, 2006.

[JMM95] H. V. Jagadish, Alberto O. Mendelzon, and Tova Milo. Similarity-based queries. In Mihalis Yannakakis, editor, *Proceedings of the Fourteenth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems, May 22-25, 1995, San Jose, California, USA*, pages 36–45. ACM Press, 1995.

[KL07] Adila Krisnadhi and Carsten Lutz. Data complexity in the $\mathcal{EL}$ family of description logics. In *Proc. of the 14th Int. Conf. on Logic for Programming and Automated Reasoning (LPAR 2007)*, volume 4790 of *Lecture Notes in Computer Science*, pages 333–347. Springer, 2007.

[KS01] Yaron Kanza and Yehoshua Sagiv. Flexible queries over semistructured data. In *PODS*. ACM, 2001.

[LW10] Carsten Lutz and Frank Wolter. Deciding inseparability and conservative extensions in the description logic $\mathcal{EL}$. *J. Symb. Comput.*, 45(2):194–228, 2010.

[Meh04] Mohri Mehryar. Weighted finite-state transducer algorithms. an overview. In Carlos Martín-Vide, Victor Mitrana, and Gheorghe Păun, editors, *Formal Languages and Applications*, pages 551–563. Springer Berlin Heidelberg, 2004.

[MW95] Alberto O. Mendelzon and Peter T. Wood. Finding regular simple paths in graph databases. *SIAM J. Comput.*, 24(6):1235–1258, 1995.

[PSW16] Alexandra Poulovassilis, Petra Selmer, and Peter T. Wood. Approximation and relaxation of semantic web path queries. *J. Web Semant.*, 40:1–21, 2016.

[PTT18] Rafael Peñaloza, Veronika Thost, and Anni-Yasmin Turhan. Query answering for rough $\mathcal{EL}$ ontologies. In Michael Thielscher and Francesca Toni, editors, *Proceedings of 16. International Conference on Principles of Knowledge Representation and Reasoning (KR 2018)*, AAAI, pages 399–408, 2018.

[Sim78] Imre Simon. Limited subsets of a free monoid. In *19th Annual Symposium on Foundations of Computer Science, Ann Arbor, Michigan, USA, 16-18 October 1978*, pages 143–150. IEEE Computer Society, 1978.