Answering Regular Path Queries Under Approximate Semantics in Lightweight Description Logics

Oliver Fernández Gil, Anni-Yasmin Turhan
Institute of Theoretical Computer Science, TU Dresden, Germany
{oliver.fernandez, anni-yasmin.turhan}@tu-dresden.de

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* two-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.

Introduction

Regular path queries (RPQs) are a well-investigated query language that attracted much attention in research on semi-structured data and graph databases due to its capabilities to navigate graph-structured data (Mendelzon and Wood 1995; Florescu, Levy, and Suciu 1998). This interest has revived in recent years since in many application areas data is graph-structured and represented in graph database models. RPQs and its extensions are part of SPARQL, which is the standard language recommended by the W3C to query RDF data. In general, a graph database can be viewed as a labeled directed graph, where edge labels state relations between data items. An 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 *two-way* RPQs (2RPQ) can traverse graph edges backwards and *conjunctive* 2RPQs (C2RPQ) allow conjunctions of 2RPQs that share variables.

In scenarios where an RPQ yields no answer over a particular database, it can be useful to relax the query to retrieve more than the classical answers, i.e., pairs that are connected by paths that approximate those paths required by the query. This can be practical 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 structure and to formulate adequate queries. In such situations, queries that over-approximate the set of answers can be very helpful.

Several approaches have been investigated to address approximate semantics for RPQs, as for instance, (Jagadish, Mendelzon, and Milo 1995; Kanza and Sagiv 2001; Grahne and Thomo 2006; Poulovassilis, Selmer, and Wood 2016). In particular, (Grahne and Thomo 2006) proposes a tractable solution that uses *weighted finite-state transducers* (WFTs) to define the approximate semantics. Intuitively, such a transducer transforms input words into corresponding output words, and computes a weight quantifying the cost of this so-called distortion operation. The idea for RPQs is that a transducer specifies (i) which paths are allowed as distortions of the "ideal" paths required by the query, and (ii) the corresponding distortion costs. Approximate answers are then tuples (a, b, η), where η is the cost of distorting a path satisfying the query into one leading from a to b in the data.

Path queries have also been investigated in the setting of ontology-mediated query answering (OMQA), in which semantic knowledge provided in an ontology is used to enrich the data (Poggi et al. 2008; Bienvenu and Ortiz 2015). In contrast to query answering over (graph) databases, OMQA usually adopts the open world assumption by computing certain answers, which are answers that hold in all possible models of the ontology and the data. Ontologies are often formulated in Description Logics (DLs), which are a well-investigated family of fragments of first-order logic that can be used to represent the conceptual knowledge of an application domain in a structured and formal way (Baader et al. 2003, 2017). Answering C2RPQs has been studied for very expressive DLs (Calvanese, Eiter, and Ortiz 2014), and for families of lightweight DLs (Calvanese et al. 2007; Bienvenu, Ortiz, and Simkus 2015; Stefanoni et al. 2014). 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 (Ecke, Peñaloza, and Turhan 2015) and on C2RPQs in the restricted setting of acyclic ontologies formulated in a fragment of RDFs (Poulovassilis, Selmer, and Wood 2016).

The goal of this paper is to define approximate semantics for C2RPQs in DLs and to devise computation algorithms for answering them in the DLs $\mathcal{ELH}$ and DL-Lite$_{\mathcal{R}}$. These two DLs are part of the $\mathcal{EL}$ (Baader, Brandt, and Lutz 2005) and DL-Lite (Calvanese et al. 2007) families of DLs, which underlie the OWL 2 EL and OWL 2 QL profiles.[1] As approximation mechanism we adopt the transducer-based approach provided by (Grahne and Thomo 2006) since, on the one hand, it yields a tractable approximate query answering problem, whereas the solutions proposed by (Kanza and Sagiv 2001) and (Jagadish, Mendelzon, and Milo 1995) have intractable and undecidable query answering problems, respectively. On the other hand, WFTs are a preferred tool to define transformations of regular languages, that can additionally compute weights to express how costly (likely, etc...) these transformations are. Since RPQs can be defined by *non-deterministic finite automata* (NFAs) and WFTs are essentially NFAs with output (and weights), WFTs are thus a natural choice to provide approximate semantics for RPQs. In fact, (Poulovassilis, Selmer, and Wood 2016) already motivates (and gives positive results for) the use of WFTs to define approximate semantics for the OMQA setting. They use the *word edit distance* as approximation tool, which can be expressed using a WFT.

Our contributions are 1) to extend the transducer-based approximate semantics from RPQs to the more general query language of C2RPQs in the graph database setting; 2) to define approximate semantics for answering C2RPQs over DL ontologies. We define the notion of *certain approximate answers* as a generalization of the classical *certain answers*; 3) to investigate two reasoning problems for certain approximate answers: a) the "τ-entailment" problem which asks, given a threshold value μ and a tuple $\bar{a}$, is $\bar{a}$ a certain approximate answer with approximation cost of at most μ?, and b) the computation problem for the exact approximation cost of $\bar{a}$. For 2RPQs, we show that τ-entailment is in NL (PTime) in data (combined) complexity for DL-Lite$_{\mathcal{R}}$, and in PTime for $\mathcal{ELH}$. As for C2RPQs, the problem can be decided in PTime (PSpace) in data (combined) complexity for both DLs. Finally, the computational problem can be solved in PTime for 2RPQs, and PTime (ExpTime) for C2RPQs in data (combined) complexity. Except for the data complexity of C2RPQs in DL-Lite$_{\mathcal{R}}$, all our upper bounds match the lower bounds inherited from the classical semantics, presented in (Bienvenu, Ortiz, and Simkus 2015).

Due to space restrictions, we give the full proofs of the results in the accompanying technical report (Fernández Gil and Turhan 2020).

Preliminaries

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

We introduce the syntax and semantics of the DLs $\mathcal{ELH}$ and DL-Lite$_{\mathcal{R}}$ and notions relevant for query answering.

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

In DL-Lite, two additional constructors are available: *inverse role* (r^-) and *negation* ($\neg$). Complex concepts and roles are built according to the rules: $B ::= A \mid \exists P$; $C ::= B \mid \neg B$; $P ::= r \mid r^-$; $S ::= P \mid \neg P$. We call concepts (roles) of the form B (P) *basic* and of the form C (S) *general*. We define $\mathsf{N_R^-} := \{r^- \mid r \in \mathsf{N_R}\}$, $\mathsf{N_R^\pm} := \mathsf{N_R} \cup \mathsf{N_R^-}$, and write P^- for $P^- = r$ if $P = r^-$ and $P^- = r^-$ if $P = r$.

A DL *knowledge base* (KB) is a pair $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ consisting of a TBox $\mathcal{T}$ and an ABox $\mathcal{A}$. A *TBox* is a finite set of inclusions, which can be of two types, either *general concept inclusions* (GCIs) or *role inclusions* (RIs), and their forms vary according to the DL in use. In $\mathcal{EL}$, inclusions are GCIs of the form $C \sqsubseteq D$ where $C, D \in \mathcal{C}_{\mathcal{EL}}$. In DL-Lite GCIs have the form $B \sqsubseteq C$, where B and C are as above. The DLs $\mathcal{ELH}$ and DL-Lite$_{\mathcal{R}}$ are the extensions of $\mathcal{EL}$ and DL-Lite additionally allowing in the TBox RIs of the form $r \sqsubseteq s$ and $P \sqsubseteq S$, respectively, where $r, s \in \mathsf{N_R}$ and P, S are as above. An *ABox* $\mathcal{A}$ is a finite set of assertions 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 $a, b \in \mathsf{N_I}$. We denote as $\mathsf{Ind}(\mathcal{A})$ the set of individuals occurring in $\mathcal{A}$.

The semantics for $\mathcal{EL}$ and DL-Lite is given by means of *first-order* logic interpretations. An *interpretation* $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ consists of a non-empty domain $\Delta^{\mathcal{I}}$ and an interpretation function $\cdot^{\mathcal{I}}$ that maps each $A \in \mathsf{N_C}$ to $A^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$, each $r \in \mathsf{N_R}$ to $r^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$ and each $a \in \mathsf{N_I}$ to $a^{\mathcal{I}} \in \Delta^{\mathcal{I}}$. The function $\cdot^{\mathcal{I}}$ is inductively extended to arbitrary roles and concepts in the following way:[2]

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

An interpretation $\mathcal{I}$ satisfies an inclusion $G \sqsubseteq H$ if $G^{\mathcal{I}} \subseteq H^{\mathcal{I}}$, and an assertion $A(a)$ ($r(a, b)$) if $a^{\mathcal{I}} \in A^{\mathcal{I}}$ ($(a^{\mathcal{I}}, b^{\mathcal{I}}) \in r^{\mathcal{I}}$). An interpretation is a *model* of $\mathcal{T}$ if it satisfies all inclusions in $\mathcal{T}$, a *model* of $\mathcal{A}$ if it satisfies all assertions in $\mathcal{A}$, and a *model* of $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ if it satisfies $\mathcal{T}$ and $\mathcal{A}$. A KB $\mathcal{K}$ is *satisfiable* if it has at least one model. We write $\mathcal{I} \models \mathcal{X}$ to denote that $\mathcal{I}$ is a model of $\mathcal{X} \in \{\mathcal{T}, \mathcal{A}, \mathcal{K}\}$. Finally, let α be an inclusion or an ABox assertion, we say that $\mathcal{X}$ *entails* α (denoted as $\mathcal{X} \models \alpha$) if $\mathcal{I}$ satisfies α for all models $\mathcal{I}$ of $\mathcal{X}$.

We assume that $\mathcal{ELH}$ TBoxes are in *normal form*, i.e., all its GCIs are of one of the following forms: $A \sqsubseteq B$, $A_1 \sqcap A_2 \sqsubseteq B$, $A \sqsubseteq \exists r.B$ or $\exists r.A \sqsubseteq B$, where $A, B, A_1, A_2 \in \mathsf{N_C} \cup \{\top\}$. This is without loss of generality by results shown in (Baader, Brandt, and Lutz 2005). Last, we denote as $\mathsf{sig}(\mathcal{T})$ the signature of $\mathcal{T}$, and as $\mathfrak{C}_{\mathcal{T}}$ the set $\mathsf{N_C} \cap \mathsf{sig}(\mathcal{T})$ if $\mathcal{T}$ is an $\mathcal{ELH}$ TBox, or $(\mathsf{N_C} \cap \mathsf{sig}(\mathcal{T})) \cup \{\exists r, \exists r^- \mid r \in \mathsf{N_R} \cap \mathsf{sig}(\mathcal{T})\}$ if $\mathcal{T}$ is a DL-Lite$_{\mathcal{R}}$ TBox.

Canonical Models. We recall the definition of the canonical model of an $\mathcal{ELH}$ or a DL-Lite$_{\mathcal{R}}$ KB, as presented in (Bienvenu, Ortiz, and Simkus 2015). The *canonical model* $\mathcal{U}_{\mathcal{K}} = (\Delta^{\mathcal{U}_{\mathcal{K}}}, \cdot^{\mathcal{U}_{\mathcal{K}}})$ of an $\mathcal{ELH}$ or a DL-Lite$_{\mathcal{R}}$ KB $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ has domain $\Delta^{\mathcal{U}_{\mathcal{K}}}$, which consists of sequences e of

[1]see https://www.w3.org/TR/owl2-profiles/

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

the form $aP_1C_1 \dots P_nC_n$ ($n \geq 0$), where $a \in \mathsf{Ind}(\mathcal{A})$, $P_i \in \mathsf{N}_\mathsf{R}^{\pm}$ and C_i is a concept. For $\mathcal{ELH}$, each role P_i is of the form $r_i \in \mathsf{N}_\mathsf{R}$ and each concept C_i is a concept name $A_i \in \mathsf{N}_\mathsf{C}$. In addition, e is required to satisfy:

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

It remains to fix the interpretation function $\cdot^{\mathcal{U}_\mathcal{K}}$ of $\mathcal{U}_\mathcal{K}$. Given $e \in \Delta^{\mathcal{U}_\mathcal{K}}$, $\mathsf{tail}(e)$ denotes the final concept C_n in e, i.e., either A_n or $\exists P_n^-$. Then, $\cdot^{\mathcal{U}_\mathcal{K}}$ 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 sub-structure representing $\mathcal{A}$ and of (possibly infinite) trees rooted at each $a \in \mathsf{Ind}(\mathcal{A})$ containing anonymous individuals. Given $e, e' \in \Delta^{\mathcal{U}_\mathcal{K}}$, T_e denotes the subtree rooted at e in $\mathcal{U}_\mathcal{K}$ and $e' \in T_e$ means 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 as $\mathfrak{d}(e') + 1$ if $e = e'PC$. Finally, $\mathsf{T}(\mathcal{U}_\mathcal{K})$ denotes the set of tails of $\mathcal{U}_\mathcal{K}$. Notice that this set consists of elements A or $\exists P^-$ occurring in $\mathcal{T}$.

Regular Path Queries

We introduce 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 *regular expression* (*r.e.*) or a *non-deterministic finite automaton*. Since our complexity results (provided in later sections) consist of upper bounds, and *r.e.* can be polynomially translated into equivalent NFAs, we adopt without loss of generality the NFA representation. As usual, an NFA $\mathfrak{N}$ is a tuple $\mathfrak{N} = (Q_\mathfrak{N}, \Sigma, \delta_\mathfrak{N}, I_\mathfrak{N}, F_\mathfrak{N})$ and $\mathcal{L}(\mathfrak{N})$ denotes the regular language defined by $\mathfrak{N}$.

A *conjunctive two-way regular path query* is of the form $q(\bar{x}) = \exists \bar{y}.\varphi(\bar{x}, \bar{y})$, where $\bar{x}, \bar{y}$ are disjoint tuples of variables, and $\varphi(\bar{x}, \bar{y})$ is a conjunction of atoms of the form $A(t)$ and $\mathfrak{N}(t, t')$, where $A \in \mathsf{N}_\mathsf{C}$, $t, t' \in \bar{x} \cup \bar{y} \cup \mathsf{N}_\mathsf{I}$, and $\mathfrak{N}$ is an NFA with Σ a finite subset of $\mathsf{N}_\mathsf{R}^{\pm} \cup \{A? \mid A \in \mathsf{N}_\mathsf{C}\}$. Variables and individuals 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 sets of terms, variables, answer variables and quantified variables of q as $\mathsf{terms}(q)$, $\mathsf{vars}(q)$, $\mathsf{avars}(q)$ and $\mathsf{qvars}(q)$, respectively. If $\mathsf{avars}(q) = \emptyset$, q is a *Boolean query*. We sometimes write $\mathsf{at} \in q$ to refer to an atom at of q, and $\wedge_{j=1}^{p} \mathfrak{N}_j(t_j, t'_j)$ to denote a C2RPQ with p atoms.

We consider two sublanguages of C2RPQs. *Two-way regular path queries* are of the form $q(x, z) = \mathfrak{N}(x, z)$, where $x, z \in \mathsf{avars}(q)$. *Regular path queries* are the special case of 2RPQs that do not use symbols from N_R^-.

Next we define the semantics of C2RPQs. Let $\mathcal{I} = (\Delta^\mathcal{I}, \cdot^\mathcal{I})$ be an interpretation and $d, d' \in \Delta^\mathcal{I}$. A *path* π from d to d' in $\mathcal{I}$ is a sequence $d_0u_1d_1u_2d_2 \dots u_md_m$ 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 $\ell(\pi) := u_1 \dots u_m$ if $m > 0$, and as the *empty word* ε if $\pi = d_0$. We write $d \xrightarrow{\mathcal{I},u} d'$ to state that there is a path from d to d' with label u in $\mathcal{I}$. For $\pi = d_0$, we simply write $d_0 \xrightarrow{\mathcal{I},\varepsilon} d_0$. A *match* for a C2RPQ q in $\mathcal{I}$ is a mapping $h : \mathsf{terms}(q) \to \Delta^\mathcal{I}$ 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 $\mathfrak{N}(t, t') \in q$: $\exists u \in \mathcal{L}(\mathfrak{N})$ such that $h(t) \xrightarrow{\mathcal{I},u} h(t')$.

Given a KB $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ and a C2RPQ q with answer variables $x_1, \dots, x_k$, a tuple of individual names $(a_1, \dots, a_k)$ from $\mathsf{Ind}(\mathcal{A})$ is a *certain answer* of q w.r.t. $\mathcal{K}$ if for each model $\mathcal{I}$ of $\mathcal{K}$ there is a match h for q in $\mathcal{I}$ such that $h(x_i) = a_i^\mathcal{I}$. 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, then $\mathsf{cert}(q, \mathcal{K}) = \{()\}$, if q has a match in every model of $\mathcal{K}$. Due to the semantics, atoms $A(t)$ are equivalent to $A?(t, t)$ and we can assume w.l.o.g. that C2RPQs have only atoms of the form $\mathfrak{N}(t, t')$.

An important result shown in (Bienvenu, Ortiz, and Simkus 2015), for DL-Lite$_\mathcal{R}$ and $\mathcal{ELH}$, is that 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. *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 KBs, but initially these were defined over graph databases. Both, DL interpretations and basic forms of graph databases can be seen as relational structures over (unary and) binary predicates (Consens and Mendelzon 1990). For the rest of the paper we consider graph databases as finite DL interpretations (see (Fernández Gil and Turhan 2020) for a formal correspondence between the two notions). The semantics of C2RPQs over graph databases adopt the *closed world assumption*, i.e., answers to a C2RPQ q are simply matches in the graph database $\mathcal{I}_\mathcal{G}$. In this setting, N_I is $\Delta^{\mathcal{I}_\mathcal{G}}$ and $a^{\mathcal{I}_\mathcal{G}} = a$ for all $a \in \Delta^{\mathcal{I}_\mathcal{G}}$. We denote by $\mathsf{ans}(q, \mathcal{I}_\mathcal{G})$ the set of answers of q in $\mathcal{I}_\mathcal{G}$.

Approximate Semantics for Path Queries

Our approximate semantics for RPQs over DL KBs extends those of RPQs over graph databases proposed in (Grahne and Thomo 2006). We recall their original transducer-based approach for RPQs over graph databases, and extend it to answering C2RPQs under approximate semantics over graph databases, which are then in turn extended to approximate semantics for answering C2RPQs over DL KBs.

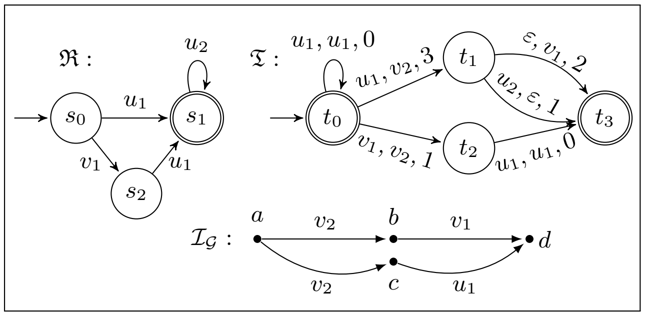

Figure 1: NFA $\mathfrak{R}$, dT $\mathfrak{T}$, and graph DB $\mathcal{I}_{\mathcal{G}}$ from Example 1.

Approximate Semantics for RPQs

In general, weighted transducers are defined for arbitrary *semirings* (Mehryar 2004), while the ones used in (Grahne and Thomo 2006) are defined over the *tropical semiring* (Simon 1978). We refer to these as distortion transducers.

Definition 1. A *distortion transducer* (dT) is a tuple $\mathfrak{T} = (\Sigma, Q, \delta, I, F)$ where Q is a finite set of states, Σ is a finite input/output alphabet, $I, F \subseteq Q$ are the sets of initial and final states, respectively. The transition relation is $\delta \subseteq Q \times \Sigma \cup \{\varepsilon\} \times \Sigma \cup \{\varepsilon\} \times \mathbb{N} \times Q$. 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 $u = u_1 \ldots u_n$, $q_1 \in I$, $q_{n+1} \in F$, and each $(q_i, u_i, v_i, w_i, q_{i+1}) \in \delta$ (for $i \leq n$). The *weight* of the run ρ is $wt(\rho) := w_1 + \ldots + w_n$. A run ρ *distorts* u into $v = v_1 \ldots v_n$ 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:[3]

$$c_{\mathfrak{T}}(u, v) := \min\{wt(\rho) \mid (\rho, wt(\rho)) \in \mathcal{R}(\mathfrak{T}, u, v)\}.$$

We illustrate the idea from (Grahne and Thomo 2006) of how to use a distortion transducer to model "acceptable" distortions of the words required by an RPQ atom and the corresponding distortion cost, by the following example.

Example 1. Figure 1 depicts an NFA $\mathfrak{R}$ with $\mathcal{L}(\mathfrak{R}) = \{u_1 u_2^*\} \cup \{v_1 u_1 u_2^*\}$, a dT $\mathfrak{T}$, and a graph database $\mathcal{I}_{\mathcal{G}}$. One can see that, querying $\mathcal{I}_{\mathcal{G}}$ through the RPQ $q(x, z)$ defined by $\mathfrak{R}$ yields $\mathsf{ans}(q, \mathcal{I}_{\mathcal{G}}) = \{(c, d)\}$. By using $\mathfrak{T}$ "in between" $\mathfrak{R}$ and $\mathcal{I}_{\mathcal{G}}$, the set $\mathsf{ans}(q, \mathcal{I}_{\mathcal{G}})$ can be (over) approximated. For instance, $\mathfrak{T}$ states that v_2 is an allowed distortion of the word $u_1 u_2$ and the use of this distortion is "penalized" with cost 4. Then, since $u_1 u_2 \in \mathcal{L}(\mathfrak{R})$ and a path with label v_2 connects a to b in $\mathcal{I}_{\mathcal{G}}$, instead of dismissing (a, b) as an answer one can now obtain it as an approximate answer with cost 4. The tuple (a, d) admits two distortions: $v_1 u_1$ into $v_2 u_1$ with costs 1 and u_1 into $v_2 v_1$ with costs 5. In this case, the smallest distortion cost gives the distortion cost of (a, d). Since $\mathfrak{T}$ gives no way to distort a word in $\mathcal{L}(\mathfrak{R})$ into v_1, the tuple (b, d) is an approximate answer with distortion cost ∞.

Based on this idea, we introduce approximate answers for RPQs formally.[4] Given an RPQ $q(x, z) = \mathfrak{R}(x, z)$ and a graph database $\mathcal{I}_{\mathcal{G}}$. The *set of approximate answers* of q in $\mathcal{I}_{\mathcal{G}}$, through a dT $\mathfrak{T}$ with $\Sigma \subseteq \mathsf{N_R}$, is defined as:

$$\begin{aligned} \tilde{\mathsf{ans}}_{\mathfrak{T}}(q, \mathcal{I}_{\mathcal{G}}) := \big\{(a, b, \eta_{a,b}) \mid a, b \in \Delta^{\mathcal{I}_{\mathcal{G}}} \text{ and } \\ \eta_{a,b} = \min\{c_{\mathfrak{T}}(u, v) \mid u \in \mathcal{L}(\mathfrak{R}) \wedge a \xrightarrow{\mathcal{I}_{\mathcal{G}}, v} b\}\big\}. \end{aligned} \quad (1)$$

We say that (a, b) is an approximate answer with distortion cost $\eta_{a,b}$. In Example 1 approximate answers with $\eta < \infty$ are the tuples $(c, d, 0)$, $(a, d, 1)$, $(a, b, 4)$, and $(a, c, 4)$.

In principle, the approach allows to relax and/or restrain the classical semantics of RPQs. Intuitively, a relaxation would preserve classical answers of q in $\mathcal{I}_{\mathcal{G}}$ (with cost 0) in the approximation. To achieve this, it suffices to add to $\mathfrak{T}$ a state t_0' that is initial and final with neutral transitions $(t_0', u, u, 0, t_0')$ for all $u \in \Sigma$ (Grahne and Thomo 2006). Moreover, the approach can implement a variety of approximations. For instance, one can use ε-transitions to build transducers whose distortion costs correspond to the *word edit distance*, see (Poulovassilis, Selmer, and Wood 2016).

Approximate Semantics for C2RPQs

As for RPQs, we define the approximate answers of a C2RPQ q of arity k as pairs of the form $(\bar{a}, \eta_{\bar{a}})$, where $\bar{a}$ is a k-tuple of elements in $\Delta^{\mathcal{I}_{\mathcal{G}}}$ and $\eta_{\bar{a}}$ is the approximation cost for $\bar{a}$. Intuitively, $\eta_{\bar{a}}$ expresses "how distant" $\bar{a}$ is to be an answer of q in $\mathcal{I}_{\mathcal{G}}$. In contrast to RPQs, C2RPQs may contain symbols from $\mathsf{N_R^-}$, quantified variables and more than one atom. The use of inverse roles in the NFA simply means for the approximate semantics, to use them in the dT as well. However, the two last differences require adaptations in the definition of $\eta_{\bar{a}}$ given for RPQs in Equation (1).

First, we take into account the quantified variables by considering all possible mappings $h : \mathsf{terms}(q) \to \Delta_{\mathcal{G}}^{\mathcal{I}}$ such that $h(\bar{x}) = \bar{a}$. For each such mapping we define an approximation cost h_c that measures how distant h is to be a match for q in $\mathcal{I}_{\mathcal{G}}$. The lowest such cost h_c of a mapping is then the cost value $\eta_{\bar{a}}$ in $(\bar{a}, \eta_{\bar{a}})$. Second, the value h_c is obtained by: i) computing for each $\mathfrak{R}_j(t_j, t_j') \in q$ the distortion cost η_j of the pair $(h(t_j), h(t_j'))$ using (1), and ii) combining all these values into h_c using an appropriate function.

Which combining function is appropriate depends highly on the application at hand. We deliberately treat such a function as a parameter of the approximation formalism, so that our approximation semantics is flexible enough to accommodate the needs of different applications. Namely, a function $\mathsf{f} : (\mathbb{N} \cup \{\infty\})^p \to \mathbb{N} \cup \{\infty\}$ is a *p-ary combining function* if it is *monotonic*,

- *commutative*, i.e., $\mathsf{f}(c_1, \ldots, c_p) = \mathsf{f}(c_{\sigma(1)}, \ldots, c_{\sigma(p)})$, where σ is any permutation of the indices $1 \ldots p$, and
- *zero closed*, i.e., $\mathsf{f}(c_1, \ldots, c_p) = 0$ if $c_i = 0$ $(1 \leq i \leq p)$.

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 classical match. For efficiency reasons, we restrict our attention to polynomial time computable functions, e.g. the *sum*, *minimum* and *maximum*.

We now define the notion of approximate match for a C2RPQ q in an interpretation $\mathcal{I}$. An *approximate match* $h_{\mathfrak{T},\mathsf{f}}^{q,\mathcal{I}}$

[3] The minimum of the empty set is defined as ∞.

[4] (Grahne and Thomo 2006) do not cover symbols such as $A?$.

for q in $\mathcal{I}$, through a dT $\mathfrak{T}$ and combining function f, is a pair $h^{q,\mathcal{I}}_{\mathfrak{T},\mathrm{f}} = (h, h_c)$ where $h : \mathsf{terms}(q) \to \Delta^{\mathcal{I}}$ is a mapping s.t.:

- $h(a) = a^{\mathcal{I}}$ for all $a \in \mathsf{terms}(q) \cap \mathsf{N_I}$; and
- the approximation cost $h_c \in \mathbb{N} \cup \{\infty\}$ is defined as

$$h_c := \underset{\mathfrak{R}(t,t')\in q}{\mathrm{f}} \min\left\{c_{\mathfrak{T}}(u,v) \,|\, u \in \mathcal{L}(\mathfrak{R}) \wedge h(t) \xrightarrow{\mathcal{I},v} h(t')\right\}.$$

For C2RPQs, approximate answers need to take into account all of the approximate matches. For an arbitrary k-tuple $\bar{a}$ of elements in $\Delta^{\mathcal{I}}$, we denote by $H^{q,\mathcal{I}}_{\mathfrak{T},\mathrm{f}}(\bar{a})$ 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 2. Let q be a C2RPQ with p atoms, $\mathcal{I}_{\mathcal{G}}$ a graph database, $\mathfrak{T}$ a dT with $\Sigma \subseteq \mathsf{N}^{\pm}_{\mathsf{R}} \cup \{A? \mid A \in \mathsf{N_C}\}$ and f a p-ary combining function. Then, the *set of approximate answers* of q in $\mathcal{I}_{\mathcal{G}}$, through $\mathfrak{T}$ and f, is defined as:

$$\tilde{\mathsf{ans}}_{\mathfrak{T},\mathrm{f}}(q, \mathcal{I}_{\mathcal{G}}) := \big\{(\bar{a}, \eta_{\bar{a}}) \mid \bar{a} \in \Delta^{\mathcal{I}_{\mathcal{G}}} \text{ and } \eta_{\bar{a}} = \min\{h_c \mid (h, h_c) \in H^{q,\mathcal{I}_{\mathcal{G}}}_{\mathfrak{T},\mathrm{f}}(\bar{a})\}\big\}.$$

We say that $\bar{a}$ is an approximate answer with approximation cost $\eta_{\bar{a}}$. This definition is an extension of the approximate semantics for RPQs, in the following sense: if f is the identity function id, both give the same results. Since 2RPQs (like RPQs) consist of one atom, we assume from now on that f $= id$ when considering 2RPQs. Moreover, requiring f to be zero closed ensures that as for RPQs, relaxation would preserve classical answers of C2RPQs by modifying $\mathfrak{T}$ by a new state as explained above.

Approximate Semantics over DL KBs

Classical query answering over graph databases adopts the closed world assumption and regards one model, while over KBs it adopts the open world assumption and regards all models by computing the certain answers. Our approximate semantics that extends the classical one pursuits this idea.

In order to get the certain approximate answers to C2RPQs, the approximate semantics uses an upper bound on the approximation costs that holds w.r.t. all 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 $\eta_{\bar{a}}$ for this tuple over all models.

Definition 3. Let $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ be a knowledge base and $q(\bar{x})$ a C2RPQ with p atoms. The set of *certain approximate answers* of q w.r.t. $\mathcal{K}$, through a dT $\mathfrak{T}$ with $\Sigma \subseteq \mathsf{N}^{\pm}_{\mathsf{R}} \cup \{A? \mid A \in \mathsf{N_C}\}$ and a p-ary combining function f, is defined as:

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

Similarly to $\mathsf{cert}(q, \mathcal{K})$, in $\mathcal{ELH}$ and DL-Lite$_{\mathcal{R}}$ the set $\tilde{\mathsf{cert}}_{\mathfrak{T},\mathrm{f}}(q, \mathcal{K})$ can be characterized by considering approximate matches in the canonical model $\mathcal{U}_{\mathcal{K}}$ only. The following lemma generalizes the result in Lemma 1.

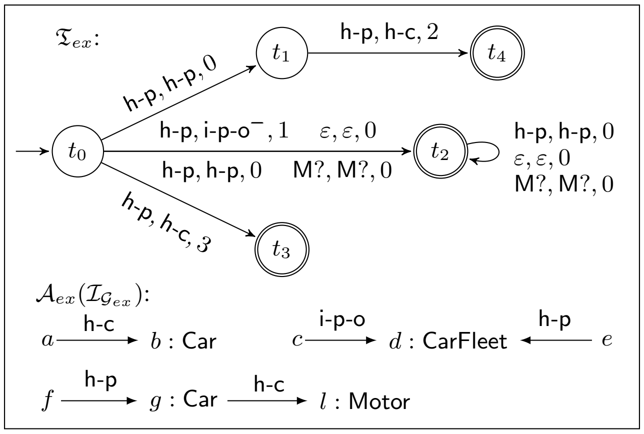
$\mathfrak{T}_{ex}$:
t_0
t_1
t_2
t_3
t_4
h-p, h-p, 0
h-p, h-c, 2
h-p, i-p-o$^-$, 1
$\varepsilon, \varepsilon, 0$
h-p, h-p, 0
M?, M?, 0
h-p, h-c, 3
h-p, h-p, 0
$\varepsilon, \varepsilon, 0$
M?, M?, 0
$\mathcal{A}_{ex}(\mathcal{I}_{\mathcal{G}_{ex}})$:
$a \xrightarrow{\text{h-c}} b$: Car
$c \xrightarrow{\text{i-p-o}} d$: CarFleet
h-p
e
$f \xrightarrow{\text{h-p}} g$: Car $\xrightarrow{\text{h-c}} l$: Motor

Figure 2: The dT $\mathfrak{T}_{ex}$ and ABox $\mathcal{A}_{ex}$.

	classical s.	approximate s. (f $= +$)
$\mathcal{I}_{\mathcal{G}_{ex}}$	$\emptyset$	$\{((f,l),2),((g,l),3),((f,g),5),((g,f),5)\}$
$\mathcal{K}_{ex}$	$\{(a,b),(b,a),(f,g),(g,f),(f,l),(g,l)\}$	$S \ \cup \{((d,c),1),((c,d),1)\}$

Table 1: Answers to q_{ex} from Example 2.

Lemma 2. *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, $(\bar{a}, \eta_{\bar{a}}) \in \tilde{\mathsf{cert}}_{\mathfrak{T},\mathrm{f}}(q, \mathcal{K})$ iff $\eta_{\bar{a}} = min\{h_c \mid (h, h_c) \in H^{q,\mathcal{U}_{\mathcal{K}}}_{\mathfrak{T},\mathrm{f}}(\bar{a})\}$.*

We illustrate the effects of using classical vs. approximate semantics in the graph database vs. the OMQA settings.

Example 2. Consider $\mathcal{L}(\mathfrak{R}_1) = \mathsf{has\text{-}part}^+$ and $\mathcal{L}(\mathfrak{R}_2) = \mathsf{has\text{-}part}^*$, the C2RPQ $q_{ex}(x,y) = \mathfrak{R}_1(x,z) \wedge \mathfrak{R}_2(y,z) \wedge \mathsf{Motor?}(z)$ and the KB $\mathcal{K}_{ex} = (\mathcal{T}_{ex}, \mathcal{A}_{ex})$, with:

$$\mathcal{T}_{ex} = \{\exists\mathsf{is\text{-}part\text{-}of.CarFleet} \sqsubseteq \mathsf{Car}, \; \mathsf{Car} \sqsubseteq \exists\mathsf{has\text{-}part.Motor}, \; \mathsf{has\text{-}component} \sqsubseteq \mathsf{has\text{-}part}\}.$$

The ABox $\mathcal{A}_{ex}$ and the employed dT $\mathfrak{T}_{ex}$ are depicted in Figure 2 with names abbreviated. The graph database $\mathcal{I}_{\mathcal{G}_{ex}}$ is defined by viewing $\mathcal{A}_{ex}$ as an interpretation. The sets of answers to q_{ex} over $\mathcal{I}_{\mathcal{G}_{ex}}$ vs. $\mathcal{K}_{ex}$ and under classical vs. approximate semantics are collected in Table 1, where the set $S := \{((o_1, o_2), 0) \mid (o_1, o_2) \in \mathsf{cert}(q_{ex}, \mathcal{K}_{ex})\}$.[5]

While querying $\mathcal{I}_{\mathcal{G}_{ex}}$ under classical semantics returns no answer, the approximation through $\mathfrak{T}_{ex}$ yields several approximate answers with cost $< \infty$. For example, to obtain $((f,g),5)$, all possible mappings h of the quantified variable z are considered and the distortion costs of $(h(x),h(z))$, $(h(y),h(z))$ and $(h(z),h(z))$ are computed and combined. For instance, by considering $h(z) = l$ the corresponding distortion costs (in the same order) are:

- 2 by transforming h-p h-p into h-p h-c, 3 by transforming h-p into h-c, and 0 since $h(z) = l$ is an instance of Motor and $\mathfrak{T}_{ex}$ has a neutral transition for the symbol Motor.

[5]Approximate answers with infinite cost and answers where both answer variables are mapped to one individual are omitted.

Combining these values with $\mathsf{f} = +$ yields the approximation cost 5, and one can verify that no other mapping for z yields a combined cost smaller than 5.

When switching to the OMQA setting, the axioms in $\mathcal{T}_{ex}$ provide information that is not stated in $\mathcal{A}_{ex}$. This results in new classical answers of q_{ex} w.r.t. $\mathcal{K}_{ex}$ as noted in Table 1, but it can also enable new distortions that incur better approximation costs for some tuples. For instance, for the tuple (d, c) we have that $((d, c), \infty) \in \tilde{\mathsf{ans}}_{\mathfrak{T}_{ex},\mathsf{f}}(q, \mathcal{I}_{\mathcal{G}_{ex}})$. However, the knowledge stated in $\mathcal{K}_{ex}$ gives the following:

- the first GCI in $\mathcal{T}_{ex}$ implies that c is an instance of Car, which then implies (by the second GCI) that c has an anonymous ($\mathsf{has\text{-}part}$)-successor o which is a Motor.

Then, if z is mapped to o, the only atom in q_{ex} not satisfied under classical semantics is $\mathfrak{R}_1(x, z)$. This atom is approximated using $\mathfrak{T}_{ex}$, by transforming $\mathsf{h\text{-}p}\ \mathsf{h\text{-}p}$ into $\mathsf{i\text{-}p\text{-}o}^-\ \mathsf{h\text{-}p}$ which has distortion cost 1, and thus $((d, c), 1)$ is an approximate certain answer of q w.r.t. $\mathcal{K}_{ex}$.

We now state the reasoning problems to be investigated in the next sections. The *τ-entailment decision problem* asks, given a KB $\mathcal{K}$, a dT $\mathfrak{T}$, a C2RPQ q, a combining function f, a tuple $\bar{a}$ and a threshold value $\mu \in \mathbb{N}$, whether $\bar{a}$ is a certain approximate answer of q w.r.t. $\mathcal{K}$, f, and $\mathfrak{T}$ with approximation cost $\eta_{\bar{a}} \leq \mu$. In addition, we consider the *cost computation problem*, which consists of computing $\eta_{\bar{a}}$. To measure the computational complexity of these problems, we consider the usual measures of combined and data complexity (Vardi 1982). The *combined complexity* is calculated by considering $\mathcal{K}, \mathfrak{T}, q, \bar{a}$ and μ as inputs, whereas the *data complexity* takes only $\mathcal{A}$ as input and assumes that all other parameters are fixed. Further, we assume that μ and all numbers in $\mathfrak{T}$ are encoded in binary.

Answering Approximate 2RPQs

(Grahne and Thomo 2006) provide a polynomial time algorithm to answer RPQs under approximate semantics, which amounts to finding the shortest path in a weighted graph obtained from the Cartesian product of the query's NFA, the distortion transducer, and the queried database. Based on Lemma 2, we lift this approach to the OMQA setting by using $\mathcal{U}_{\mathcal{K}}$ instead of just the data in the ABox.

Definition 4. Let $\mathcal{K}$ be an $\mathcal{ELH}$ or a DL-Lite$_{\mathcal{R}}$ knowledge base. Further, 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 $\Sigma \subseteq \mathsf{N}_{\mathsf{R}}^{\pm} \cup \{A? \mid A \in \mathsf{N}_{\mathsf{C}}\}$. The *weighted graph* $G_{\mathfrak{R}\times\mathfrak{T}\times\mathcal{U}_{\mathcal{K}}}$ is a tuple (V, E) where:

- $V := Q_{\mathfrak{R}} \times Q_{\mathfrak{T}} \times \Delta^{\mathcal{U}_{\mathcal{K}}}$ is the set of vertices, and
- $E \subseteq V \times \mathbb{N} \times V$ is a set of weighted edges such that $((s, t, e), w, (s', t', e')) \in E$ iff its components satisfy one of the following set of conditions:
  - $(s, u, s') \in \delta_{\mathfrak{R}}, (t, u, v, w, t') \in \delta_{\mathfrak{T}}$ and $e \xrightarrow{\mathcal{U}_{\mathcal{K}},v} e'$.
  - $s = s', (t, \varepsilon, v, w, t') \in \delta_{\mathfrak{T}}$ and $e \xrightarrow{\mathcal{U}_{\mathcal{K}},v} e'$.

A path π in $G_{\mathfrak{R}\times\mathfrak{T}\times\mathcal{U}_{\mathcal{K}}}$ is a sequence $\mathrm{v}_1 w_1 \mathrm{v}_2 \ldots w_{n-1} \mathrm{v}_n$ where $(\mathrm{v}_i, w_i, \mathrm{v}_{i+1}) \in E$ for all $1 \leq i < n$. The *cost* $c(\pi)$ of π is the sum of all its weights.

For simplicity, we use $G_{\mathcal{U}_{\mathcal{K}}}$ to refer to $G_{\mathfrak{R}\times\mathfrak{T}\times\mathcal{U}_{\mathcal{K}}}$. The following lemma uses $G_{\mathcal{U}_{\mathcal{K}}}$ to characterize $\tilde{\mathsf{cert}}_{\mathfrak{T}}(q, \mathcal{K})$.

Lemma 3. *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 smallest cost 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 η, where $s_0 \in I_{\mathfrak{R}}$, $t_0 \in I_{\mathfrak{T}}$, $s_f \in F_{\mathfrak{R}}$, and $t_f \in F_{\mathfrak{T}}$.*

Since $\mathcal{U}_{\mathcal{K}}$ may be infinite, we cannot use $G_{\mathcal{U}_{\mathcal{K}}}$ directly to obtain algorithms to decide τ-entailment or to compute $\tilde{\mathsf{cert}}_{\mathfrak{T}}(q, \mathcal{K})$. To overcome this, we extend the idea used in (Bienvenu, Ortiz, and Simkus 2015) to decide whether $(a, b) \in \mathsf{cert}(q, \mathcal{K})$, which applies a *symbolic* computation to solve a reachability problem in the (infinite) graph $G_{\mathfrak{R}\times\mathcal{U}_{\mathcal{K}}}$.

Deciding τ-entailment

We start by introducing the notion of an e-path in $G_{\mathcal{U}_{\mathcal{K}}}$.

Definition 5. 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 $e \in \mathsf{Ind}(\mathcal{A})$.

Now, any given path π of the form $(s, t, a) \ldots (s', t', b)$ in $G_{\mathcal{U}_{\mathcal{K}}}$ with $a, b \in \mathsf{Ind}(\mathcal{A})$, can be decomposed as follows:

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

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.

We use this decomposition in Algorithm 1 to decide τ-entailment. More precisely, the algorithm guesses a sequence of vertices $(s_0, t_0, a) \ldots (s_\ell, t_\ell, b)$ where s_0, t_0 (s_ℓ, t_ℓ) are initial (final) states in $\mathfrak{R}$ and $\mathfrak{T}$. For each pair $((s_i, t_i, a_i), (s_{i+1}, t_{i+1}, a_{i+1}))$, it guesses whether $(s_i, t_i, a_i)\, \gamma_i\, (s_{i+1}, t_{i+1}, a_{i+1})$ shall be an edge or an a_i-path in $G_{\mathcal{U}_{\mathcal{K}}}$ forced by a concept $B \in \mathfrak{C}_{\mathcal{T}}$ (lines 6 and 8). If an edge is chosen, its weight w is added to the cost c of the path guessed so far. Otherwise, c is incremented with the minimal cost of an a_i-path of such a form. This cost is stored in the table sp in entries of the form $[(s, t), (s', t'), B]$.

To obtain sp we exploit that an a-path has the form:

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

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 $C \in \mathsf{T}(\mathcal{U}_{\mathcal{K}})$. Such an entry contains the minimal cost of an e-path $(s, t, e) \ldots (s', t', e)$, where $e \in \Delta^{\mathcal{U}_{\mathcal{K}}} \setminus \mathsf{Ind}(\mathcal{A})$ and $\mathsf{tail}(e) = C$. The tables sp and spa can be seen as generalizations of the tables ALoop_α and Loop_α from (Bienvenu, Ortiz, and Simkus 2015), which store just whether the corresponding paths exist or not. In (Fernández Gil and Turhan 2020), we formally show how to extend the computation of ALoop_α and Loop_α to correctly compute sp and spa. Once we know

Algorithm 1 Answering 2RPQs under approx. semantics.

```
Input: $\mathcal{K}=(\mathcal{T},\mathcal{A})$, $\mathfrak{T}$, $q=\mathfrak{R}(x,z)$, $a,b\in\mathsf{Ind}(\mathcal{A})$ and $\mu\in\mathbb{N}$.
Output: yes iff $(a,b,\eta_{a,b})\in\check{\mathsf{c}}\mathsf{ert}_{\mathfrak{T}}(q,\mathcal{K})$ and $\eta_{a,b}\leq\mu$.
 1: if $\mathcal{K}$ is unsatisfiable then return yes;
 2: $i:=1$; $n:=|Q_{\mathfrak{R}}|\cdot|Q_{\mathfrak{T}}|\cdot|\mathcal{A}|$; $c:=0$;
 3: $(s,t,d):=(s_0,t_0,a)$; $\mathcal{F}:=\{(s,t,b)\mid(s,t)\in F_{\mathfrak{R}}\times F_{\mathfrak{T}}\}$;
 4: while $i<n$ and $c\leq\mu$ and $(s,t,d)\notin\mathcal{F}$ do
 5:     guess $(s',t',d')\in Q_{\mathfrak{R}}\times Q_{\mathfrak{T}}\times\mathsf{Ind}(\mathcal{A})$;
 6:     non-det. choose $(s,t,d)\ w\ (s',t',d')$ in $G_{\mathcal{U}_{\mathcal{K}}}$
 7:         if there is no such edge then $w=\infty$;
 8:     or choose $B\in\mathfrak{C}_{\mathcal{T}}$
 9:         if $d=d'$ and $\mathcal{K}\models B(d)$ then
10:         $w:=sp[(s,t),(s',t'),B]$ else $w:=\infty$;
11:     $c:=c+w$; $(s,t,d):=(s',t',d')$; $i:=i+1$;
12: return yes iff $c\leq\mu$ and $(s,t,d)\in\mathcal{F}$.
```

that *spa* and *sp* contain the intended values, it is not hard to see that Algorithm 1 is correct.

Satisfiability tests of a KB and entailment checks used for the non-deterministic choices are in PTime for $\mathcal{ELH}$ (Baader, Brandt, and Lutz 2005) and in NL for DL-Lite$_{\mathcal{R}}$ (Calvanese et al. 2007). The tables *sp* and *spa* can be computed in polynomial time in the size of the whole input, and in constant time in the size of $\mathcal{A}$. Hence, Algorithm 1 runs in polynomial time, and in non-deterministic logarithmic space in the size of $\mathcal{A}$ for DL-Lite$_{\mathcal{R}}$ (n, i, c are encoded in binary). Finally, it is not hard to transform Algorithm 1 into a polynomial time algorithm (called Algorithm 1c), that uses Dijkstra's *shortest path algorithm* to compute the set $\check{\mathsf{c}}\mathsf{ert}_{\mathfrak{T}}(q,\mathcal{K})$. All the details are provided in (Fernández Gil and Turhan 2020). Thus, together with the lower bounds presented in (Bienvenu, Ortiz, and Simkus 2015) for the classical semantics, we obtain the following results.

Theorem 6. *For 2RPQs, τ-entailment is NL-complete (P-complete) in data (combined) complexity for DL-Lite$_{\mathcal{R}}$, and P-complete for $\mathcal{ELH}$. The cost η of a certain approximate answer is computable in polynomial time for both DLs.*

Answering Approximate C2RPQs

In (Bienvenu, Ortiz, and Simkus 2015), a query rewriting procedure is developed to answer C2RPQs under classical semantics that uses only polynomial space (combined complexity). Such a procedure would be desirable for the approximate semantics as well. However, a rewritten query need not preserve the regular languages required by atoms in the initial query and thus it is unclear how to reuse this 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 for Boolean C2RPQs (bC2RPQs). This fragment is the restriction of $\mathcal{U}_{\mathcal{K}}$ to elements of depth at most $g\cdot m$, where $g:=p+1$ and $m:=|\mathsf{T}(\mathcal{U}_{\mathcal{K}})|\cdot\prod_{j=1}^{p}(|Q_{\mathfrak{R}_j}|\cdot|Q_{\mathfrak{T}}|)^2$. The following lemma is one of the main results of this section.

Lemma 4. *Let $\mathcal{K}=(\mathcal{T},\mathcal{A})$ be an $\mathcal{ELH}$ or a satisfiable DL-Lite$_{\mathcal{R}}$ KB, q a bC2RPQ with p atoms, $\mathfrak{T}$ a dT and f a p-ary combining function. Then, $((),\eta)\in\check{\mathsf{c}}\mathsf{ert}_{\mathfrak{T},\mathrm{f}}(q,\mathcal{K})$ iff $\eta=min\{h_c\mid(h,h_c)\in H^{q,\mathcal{U}_{\mathcal{K}}(q)}_{\mathfrak{T},\mathrm{f}}(())\}$.*

This is then exploited to obtain the second main contribution of this section: PSpace procedures to answer C2RPQs under approximate semantics in both considered DLs. Since classical semantics is subsumed by approximate semantics, these procedures constitute a *worst-case optimal* alternative to the approach from (Bienvenu, Ortiz, and Simkus 2015).

Proof Idea of Lemma 4

The proof idea for Lemma 4 is that each approximate match outside $\mathcal{U}_{\mathcal{K}}(q)$ can be "made better" in the following sense.

Definition 7. Let $\mathcal{K}=(\mathcal{T},\mathcal{A})$ be an $\mathcal{ELH}$ or DL-Lite$_{\mathcal{R}}$ KB, q a bC2RPQ, $y\in\mathsf{vars}(q)$ and $(h,h_c)\in H^{q,\mathcal{U}_{\mathcal{K}}}_{\mathfrak{T},\mathrm{f}}(())$. We say that (h,h_c) can be improved w.r.t. y if there is $(h',h'_c)\in H^{q,\mathcal{U}_{\mathcal{K}}}_{\mathfrak{T},\mathrm{f}}(())$ such that: $h'_c\leq h_c$, $\mathfrak{d}(h'(y))<\mathfrak{d}(h(y))$ and $\mathfrak{d}(h'(z))\leq\mathfrak{d}(h(z))$ for all $z\in\mathsf{vars}(q)$.

The goal is to show that every $(h,h_c)\in H^{q,\mathcal{U}_{\mathcal{K}}}_{\mathfrak{T},\mathrm{f}}(())$ such that $\mathfrak{d}(h(y))>g\cdot m$ for some y, can be improved w.r.t. y. This is shown in our technical report in the following lemma.

Lemma 5. *Let $\mathcal{K}=(\mathcal{T},\mathcal{A})$ be an $\mathcal{ELH}$ or DL-Lite$_{\mathcal{R}}$ KB, q a bC2RPQ, $(h,h_c)\in H^{q,\mathcal{U}_{\mathcal{K}}}_{\mathfrak{T},\mathrm{f}}(())$, and $y\in\mathsf{vars}(q)$ such that $\mathfrak{d}(h(y))>g\cdot m$. Then, (h,h_c) can be improved w.r.t. y.*

From this, it is straightforward to conclude that Lemma 4 holds. Next, we sketch how to use this bound to obtain a NPSpace decision procedure for τ-entailment.

A PSpace Algorithm

For simplicity, we only consider $\mathcal{ELH}$, but all the results also apply for DL-Lite$_{\mathcal{R}}$. The idea is to guess the mapping h and compute the distortion cost for each pair $(h(t_j),h(t'_j))$ *on-the-fly* in a *bottom-up* manner. This is implemented in Algorithm 2, which we continue to explain in more details.

Let us start by describing how a run ρ of the algorithm guesses a mapping $h_\rho:\mathsf{vars}(q)\to\Delta^{\mathcal{U}_{\mathcal{K}}(q)}$. This is done in the *while-loop* with the help of $\mathfrak{s}$ and $\mathfrak{s}'$. Formally, $\mathfrak{s}$ (and $\mathfrak{s}'$) is a set of tuples $(\mathcal{Y},\lambda)$ such that:

- $\lambda\in S_e\cup\mathsf{Ind}(\mathcal{A})$ where $S_e:=(\mathsf{N_R}\times\mathsf{T}(\mathcal{U}_{\mathcal{K}}))\cup\{()\}$, $\mathcal{Y}\subseteq\mathsf{vars}(q)$, $\mathcal{Y}$ does not appear twice in $\mathfrak{s}$, and
- the family $\mathfrak{s}_{\mathcal{Y}}:=\{\mathcal{Y}\mid(\mathcal{Y},\_)\in\mathfrak{s}\}$ is a partition of $\mathsf{vars}(q)$.

We say that $\mathfrak{s}'$ is *coarser* than $\mathfrak{s}$ if the partition $\mathfrak{s}'_{\mathcal{Y}}$ is coarser than $\mathfrak{s}_{\mathcal{Y}}$, i.e., for all $\mathcal{Y}\in\mathfrak{s}_{\mathcal{Y}}$ there is $\mathcal{Y}'\in\mathfrak{s}'_{\mathcal{Y}}$ s.t. $\mathcal{Y}\subseteq\mathcal{Y}'$.

The intuition behind $\mathfrak{s}$ and $\mathfrak{s}'$ is as follows. At each iteration, the element λ of a tuple $(\mathcal{Y},\lambda)$ guessed in $\mathfrak{s}'$ represents the i^{th}-component of the sequence $h_\rho(y)$ for each variable $y\in\mathcal{Y}$. In case $\lambda=()$, this means that the depth of $h_\rho(y)$ has not yet been reached by the iteration. Otherwise, λ stands for r_iA_i (or a, if $i=0$). Then, ρ induces the mapping $h_\rho(y):=ar_1A_1\ldots r_{n_y}A_{n_y}$, where n_y is the greatest i such that $y\in\mathcal{Y}$ and $\lambda\neq()$ for some $(\mathcal{Y},\lambda)\in\mathfrak{s}'$. In the technical report, we show that each sequence $ar_1A_1\ldots r_{n_y}A_{n_y}$ is well-defined, unique and represents an element in $\mathcal{U}_{\mathcal{K}}(q)$.

Next, we describe how ρ computes the distortion cost η_j of $(h_\rho(t_j),h_\rho(t'_j))$ for each $\mathfrak{R}_j(t_j,t'_j)\in q$. As shown in

Algorithm 2 Deciding τ-entailment for bC2RPQs.

Input: A bC2RPQ $q = \mathfrak{R}_1(t_1, t_1') \land \ldots \land \mathfrak{R}_p(t_p, t_p')$, $\mathcal{K} = (\mathcal{T}, \mathcal{A})$, a dT $\mathfrak{T}$, a p-ary combining function f and $\mu \in \mathbb{N}$.
Output: *yes* **iff** $((), \eta) \in \tilde{\mathsf{a}}\mathsf{ns}_{\mathfrak{T},\mathrm{f}}(q, \mathcal{K})$ and $\eta \leq \mu$.

```
 1: $i := g \cdot m$;  (i is encoded in binary)
 2: $\mathfrak{s} := \{(\{y\}, ()) \mid y \in \mathsf{vars}(q)\}$; $\eta_{\{1\ldots p\}} := \infty$;
 3: while $i \geq 0$ do
 4:     guess $\mathfrak{s}'$ coarser than $\mathfrak{s}$ (with $\lambda \in \mathsf{Ind}(\mathcal{A})$, if $i = 0$);
 5:     for all $(\mathcal{Y}, \lambda) \in \mathfrak{s}$ such that $\lambda = (r, A)$ do
 6:         select $(\mathcal{Y}', \lambda') \in \mathfrak{s}'$ s.t. $\mathcal{Y} \subseteq \mathcal{Y}'$;
 7:         if $\lambda' = ()$ then fail;
 8:         if $i > 0$ and $\mathcal{T} \not\models [\lambda'.A] \sqsubseteq \exists r.A$ then fail;
 9:         if $i = 0$ and $\mathcal{K} \not\models \exists r.A(\lambda')$ then fail;
10:     update $sp^y_\uparrow$ and $sp^y_\downarrow$ (for all $y \in \mathsf{vars}(q)$);
11:     for all $1 \leq j \leq p$ do
12:         if $i = 0$ then update $\eta_j$ applying (5) w.r.t. $\mathfrak{s}'$
13:         if $i > 0$, $(\mathcal{Y}, (r, A)) \in \mathfrak{s}'$ and $t_j, t_j' \in \mathcal{Y}$ then
14:             update $\eta_j$ applying (4) with $\mathsf{tail}(e_i) = A$
15:     $\mathfrak{s} := \mathfrak{s}'$; $i := i - 1$;
16: return success iff $\mathrm{f}(\eta_1, \ldots, \eta_p) \leq \mu$
```

Lemma 3, η_j is the minimal cost of a path π of the form $(s_0, t_0, h(t_j)) \ldots (s_f, t_f, h(t_j'))$ in $G^j_{\mathcal{U}_\mathcal{K}}$, which is of one of the following two (mutually exclusive) forms:

- π visits no node $(\_, \_, b)$ with $b \in \mathsf{Ind}(\mathcal{A})$. This means that there exists $e \in \Delta^{\mathcal{U}_\mathcal{K}} \setminus \mathsf{Ind}(\mathcal{A})$ such that $h(t_j), h(t_j') \in T_e$ and π can be decomposed as:
$$(s_0, t_0, h(t_j))\, \gamma_\uparrow\, (s, t, e)\, \gamma\, (s', t', e)\, \gamma_\downarrow\, (s_f, t_f, h(t_j')) \quad (2)$$
where nodes $(\_, \_, f)$ visited in $\gamma_\uparrow, \gamma_\downarrow$ and γ satisfy that $f \in T_e$, and $f \neq e$ for $\gamma_{\{\uparrow,\downarrow\}}$.
- π does visit a node $(\_, \_, b)$ and can be decomposed as:
$$(s_0, t_0, h(t_j))\gamma_\uparrow(s, t, a)\gamma(s', t', a')\gamma_\downarrow(s_f, t_f, h(t_j')) \quad (3)$$
where $a, a' \in \mathsf{Ind}(\mathcal{A})$, $h(t_j) \in T_a$, $h(t_j') \in T_{a'}$, γ is arbitrary and $\gamma_{\{\uparrow,\downarrow\}}$ are as in the previous case w.r.t. $T_{\{a,a'\}}$.

Hence, η_j is the minimum between the minimal costs of a path of the form (2) and (3). For paths of the form (2), t_j, t_j' must be variables y, z, which means that $h(y) = ar_1A_1 \ldots r_{n_y}A_{n_y}$ and $h(z) = a's_1B_1 \ldots s_{n_z}B_{n_z}$. Further, since $h(y), h(z) \in T_e$, e must be a common prefix $a \ldots r_iA_i = a' \ldots s_iB_i$ of $h(y)$ and $h(z)$, for some $i > 0$. We denote a as e_0, ar_1A_1 as e_1, and so on until $e_{n_y} = h(y)$. Similarly, we use $f_0, \ldots, f_{n_z}$ for $h(z)$. Based on these considerations, we define the tables $sp^y_\uparrow$ and $sp^z_\downarrow$ with entries of the form $[(s, t), i]$ such that:

- $sp^y_\uparrow[(s, t), i]$ and $sp^z_\downarrow[(s, t), i]$ are the minimal cost of a path of the form $(s_0, t_0, h(y))\ \gamma_\uparrow\ (s, t, e_i)$ and $(s, t, f_i)\ \gamma_\downarrow\ (s_f, t_f, h(z))$, respectively.

Finally, one can see that $(s, t, e)\ \gamma\ (s', t', e)$ in (2) is an e-path in $G^j_{\mathcal{U}_\mathcal{K}}$. As noted in the previous section, the table spa stores the minimal cost of a path of such a form. Hence, Algorithm 2 updates η_j at line 14 by minimizing the expression
$$sp^y_\uparrow[(s, t), i] + spa[(s, t), (s', t'), \mathsf{tail}(e_i)] + sp^z_\downarrow[(s', t'), i] \quad (4)$$
over all $(s, t), (s', t') \in Q_{\mathfrak{R}_j} \times Q_\mathfrak{T}$. Regarding (3), a simple change in Algorithm 1c allows to compute the minimal cost $sp_\rightsquigarrow[(s, t, a), (s', t', a')]$ of a path $(s, t, a)\ \gamma\ (s', t', a')$ in $G^j_{\mathcal{U}_\mathcal{K}}$. Hence, since $a = e_0$ and $a' = f_0$, when $i = 0$ the algorithm updates η_j by minimizing
$$sp^y_\uparrow[(s, t), 0] + sp_\rightsquigarrow[(s, t, a), (s', t', a')] + sp^z_\downarrow[(s', t'), 0] \quad (5)$$

As for the tables $sp^y_\uparrow$, a path $(s_0, t_0, h(y))\ \gamma_\uparrow\ (s, t, e_i)$ can be decomposed as $(s_0, t_0, h(y))\ \gamma_\uparrow\ (s_1, t_1, e_{i+1})$ $\gamma(s_2, t_2, e_{i+1})\ w\ (s, t, e_i)$. This means that $sp^y_\uparrow[(s, t), i]$ can be expressed in terms of $sp^y_\uparrow[(s_1, t_1), i + 1]$ and $spa[(s_1, t_1), (s_2, t_2), \mathsf{tail}(e_{i+1})]$. Hence, there is no need to store a copy of $sp^y_\uparrow$ for each i, but two copies suffice to perform the update at line 10. The same applies to $sp^y_\downarrow$.

Based on this last observation, it is not hard to see that the algorithm runs in non-deterministic polynomial space. Further, note that any computation requires at most polynomial time in the size of $\mathcal{A}$, including the computation of $sp_\rightsquigarrow$ to update η_j. Hence, the algorithm runs in polynomial time in the size of $\mathcal{A}$. In (Fernández Gil and Turhan 2020), we formally prove that Algorithm 2 decides τ-entailment for bC2RPQs, and also show how to reuse it to compute the approximation costs. Overall, we obtain the following results.

Theorem 8. *Deciding τ-entailment of C2RPQs is in PTime for DL-Lite$_\mathcal{R}$ and P-complete for $\mathcal{ELH}$ in data complexity, and PSpace-complete in combined complexity for both DLs. The cost η of a certain approximate answer is computable in polynomial (exponential) time in data (combined) complexity for both DLs.*

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 the approach from (Grahne and Thomo 2006), that uses WFT 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. Moreover, we have developed algorithms for computing the certain approximate answers for 2RPQs and C2RPQs over $\mathcal{ELH}$ and DL-Lite$_\mathcal{R}$ontologies. Our algorithms for C2RPQs are, to the best of our knowledge, the first ones for computing answers of such queries under approximate semantics in the presence of DL ontologies.

As future work we plan extensions to more expressive DLs, starting with $\mathcal{ELI}$ (Baader, Brandt, and Lutz 2005) and $\mathcal{ALC}$ (Schmidt-Schauß and Smolka 1991). For $\mathcal{ELI}$, we expect the data complexity to remain in PTime and that the canonical model can be bounded in a similar way. The combined complexity would inherit ExpTime-hardness already from instance checking (Baader, Lutz, and Brandt 2008). The DL $\mathcal{ALC}$ does not enjoy the canonical model property and thus it is interesting to investigate whether the cross-product of query NFA, transducer and a tree automaton representing the models of the KB can be used in a similar way.

Acknowledgments

The first author is partially supported by the DFG in grant 389792660 as part of TRR 248 (https://perspicuous-computing.science) and grant BA 1122/20-1. The second author is partially supported by the German Research Foundation (DFG) in the Research Training Group QuantLA (GRK 1763). We would also like to thank the anonymous reviewers for their helpful comments, which helped us to improve this paper substantially.

References

Baader, F.; Brandt, S.; and Lutz, C. 2005. Pushing the $\mathcal{EL}$ Envelope. In *Proc. of the 19th Int. Joint Conf. on Artificial Intelligence (IJCAI 2005)*, 364–369. Professional Book Center.

Baader, F.; Calvanese, D.; McGuinness, D.; Nardi, D.; and Patel-Schneider, P. F., eds. 2003. *The Description Logic Handbook: Theory, Implementation, and Applications*. Cambridge University Press.

Baader, F.; Horrocks, I.; Lutz, C.; and Sattler, U. 2017. *An Introduction to Description Logic*. Cambridge University Press.

Baader, F.; Lutz, C.; and Brandt, S. 2008. Pushing the $\mathcal{EL}$ Envelope Further. In *Proceedings of the Fourth OWLED Workshop on OWL: Experiences and Directions 2008*, volume 496 of *CEUR Workshop Proceedings*. CEUR-WS.org.

Bienvenu, M.; and Ortiz, M. 2015. Ontology-Mediated Query Answering with Data-Tractable Description Logics. In *Reasoning Web. Web Logic Rules - 11th International Summer School 2015, Tutorial Lectures*, volume 9203 of *Lecture Notes in Computer Science*, 218–307. Springer.

Bienvenu, M.; Ortiz, M.; and Simkus, M. 2015. Regular Path Queries in Lightweight Description Logics: Complexity and Algorithms. *J. Artif. Intell. Res.* 53: 315–374.

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

Calvanese, D.; Eiter, T.; and Ortiz, M. 2014. Answering regular path queries in expressive Description Logics via alternating tree-automata. *Inf. Comput.* 237: 12–55.

Consens, M. P.; and Mendelzon, A. O. 1990. 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)*, 404–416. ACM Press.

Ecke, A.; Peñaloza, R.; and Turhan, A.-Y. 2015. Similarity-based Relaxed Instance Queries. *Journal of Applied Logic* 13(4, Part 1): 480–508. Special Issue for the Workshop on Weighted Logics for AI 2013.

Fernández Gil, O.; and Turhan, A.-Y. 2020. Answering Regular Path Queries Under Approximate Semantics in Lightweight Description Logics. LTCS-Report LTCS-20-05, Chair of Automata Theory, Institute of Theoretical Computer Science, Technische Universität Dresden. See http://lat.inf.tu-dresden.de/research/reports.html.

Florescu, D.; Levy, A. Y.; and Suciu, D. 1998. Query Containment for Conjunctive Queries with Regular Expressions. In Mendelzon, A. O.; and Paredaens, J., eds., *Proc. of the 17th ACM SIGACT SIGMOD SIGART Symp. on Principles of Database Systems (PODS'98)*, 139–148. ACM Press.

Grahne, G.; and Thomo, A. 2006. Regular path queries under approximate semantics. *Ann. Math. Artif. Intell.* 46(1-2): 165–190.

Jagadish, H. V.; Mendelzon, A. O.; and Milo, T. 1995. Similarity-Based Queries. In *Proc. of the 14th ACM SIGACT SIGMOD SIGART Symp. on Principles of Database Systems (PODS'95)*, 36–45. ACM Press.

Kanza, Y.; and Sagiv, Y. 2001. Flexible Queries Over Semistructured Data. In *Proc. of the 20th ACM SIGACT SIGMOD SIGART Symp. on Principles of Database Systems (PODS 2001)*. ACM.

Mehryar, M. 2004. Weighted Finite-State Transducer Algorithms. An Overview. In Martín-Vide, C.; Mitrana, V.; and Păun, G., eds., *Formal Languages and Applications*, 551–563. Springer Berlin Heidelberg.

Mendelzon, A. O.; and Wood, P. T. 1995. Finding Regular Simple Paths in Graph Databases. *SIAM J. Comput.* 24(6): 1235–1258.

Poggi, A.; Lembo, D.; Calvanese, D.; Giacomo, G. D.; Lenzerini, M.; and Rosati, R. 2008. Linking Data to Ontologies. *J. Data Semantics* 10: 133–173.

Poulovassilis, A.; Selmer, P.; and Wood, P. T. 2016. Approximation and relaxation of semantic web path queries. *J. Web Semant.* 40: 1–21.

Schmidt-Schauß, M.; and Smolka, G. 1991. Attributive Concept Descriptions with Complements. *Artif. Intell.* 48(1): 1–26.

Simon, I. 1978. Limited Subsets of a Free Monoid. In *Proc. of the 19th Annual Symp. on the Foundations of Computer Science (FOCS'78)*, 143–150. IEEE Computer Society.

Stefanoni, G.; Motik, B.; Krötzsch, M.; and Rudolph, S. 2014. The Complexity of Answering Conjunctive and Navigational Queries over OWL 2 EL Knowledge Bases. *J. Artif. Intell. Res.* 51: 645–705.

Vardi, M. Y. 1982. The Complexity of Relational Query Languages (Extended Abstract). In *Proc. of the 14th ACM SIGACT Symp. on Theory of Computing (STOC'82)*, 137–146. ACM.