Query Answering Based on Modal Correspondence Theory

Evgeny Zolin

School of Computer Science
University of Manchester
Manchester, UK
ezolin@cs.man.ac.uk

Abstract. We propose a query answering technique applicable to a wide family of cyclic conjunctive queries with one distinguished variable. It exploits the fact that, given such a query, one can build in linear time a concept of some Description Logic (DL) such that to answer the query w.r.t. any knowledge base (KB) is the same as to find all instances of that concept. Notably, the method is uniform: the concept does not depend on a KB and even on a DL in which the KB is formulated. Thus, for these conjunctive queries, the problem of query answering is reduced to concept instance retrieval, which in turn is reducible to KB (un)satisfiability. The technique presented here is based on the modal correspondence theory (and close relationship between modal and description logics) and can be regarded as a practical application of that theory.

1 Introduction

Description Logics are family of knowledge representation formalisms closely related to first-order and modal logics. Most of them can be viewed as decidable fragments of the first-order logic. Their complexity was thoroughly investigated, and practical reasoning algorithms were designed (e.g., [9, 11, 14]) and implemented as highly optimised reasoning systems, e.g., Racer [7], FaCT [8], and Pellet [18]. Typical reasoning services provided by these systems are checking KB satisfiability, concept subsumption (and building a taxonomy, i.e., a computing subsumption relationships between all concept names occurring in a KB), concept instances retrieval etc. (see [1, Chap. 8] for an overview).

Among reasoning tasks, querying is a fundamental mechanism for extracting information from a KB. Two most important reasoning services involving queries—query answering and query containment, also called subsumption—were extensively explored [3–6, 10, 12, 17], but have not got a complete solution. A typical obstacle is the presence of cycles in a query (cf. [6]). In general, query containment and answering are reducible to each other and are at least as hard as concept subsumption or satisfiability problems (whence their lower complexity bounds). However, to the best of our knowledge, no tight upper bounds for their complexity are obtained so far. To illustrate the situation, recall that the Description Logic $\mathcal{SHIQ}$ is known to be ExpTime-complete [11]. At the same time, the complexity of query answering over $\mathcal{SHIQ}$ KBs is shown in [5] to be 3coNExpTime, if a KB has no transitive roles, and 4coNExpTime otherwise.

In this paper, we concentrate on the query answering problem. We show that that for a wide family of conjunctive queries (with one distinguished variable) one can build in linear time a concept (usually, in a very simple DL) whose instances are exactly the answers to the query (w.r.t. any KB). As a consequence, within this family, the query answering problem is reduced to concept instance retrieval, which in turn is reducible to KB unsatisfiability. Note that the reduction is applicable to many cyclic queries, and the resulting concept is not in general equivalent to the query. The technique presented here is based on the modal correspondence theory; this theory explores the relation between the properties of frames expressible in modal and first-order languages.

To illustrate the method, consider the following instance retrieval request: $\mathcal{KB} \models a\colon (\neg X \sqcup \exists R.X)$, where the concept name X does not occur in the knowledge base $\mathcal{KB}$. The task is to find all individuals a (called constants in first-order logic) occurring in $\mathcal{KB}$ that belong to the concept $\neg X \sqcup \exists R.X$ in any model of $\mathcal{KB}$. It is not hard to show that exactly thos individuals a will be retrieved that satisfy (in all models of $\mathcal{KB}$) the condition aRa. Hence, the concept $\neg X \sqcup \exists R.X$ can be used for answering the cyclic query $q(x) \leftarrow xRx$. Similarly, the request $\mathcal{KB} \models a\colon (\exists R.\neg X \sqcup \exists S.X)$, where X does not occur in $\mathcal{KB}$, returns exactly those individuals a that are answers of the query $q(x) \leftarrow \exists y(xRy \wedge xSy)$. Thus, concepts involving *fresh* concept names can be used to compute the answers of even cyclic queries.

Now recall from modal logic that the formula $p \to \Diamond p$ is *valid at a point* e of a frame iff this point is reflexive, i.e., eRe holds. Similarly, the formula $\Box_R p \to \Diamond_S p$ is valid at a point e iff there is a point d such that eRd and eSd holds, i.e., e satisfies the first-order condition $\alpha(x) := \exists y(xRy \wedge xSy)$. Comparing this with the above observations, we can generalise it as follows: whenever the validity of a modal formula at a point of a frame is equivalent to some first-order condition (and this condition has the form of conjunctive query), we can use this fact for query answering. This is the main result of this paper (Theorem 7), and an open question is whether the converse implication holds. An important feature of this approach is its uniformity: a concept used to answers a query is not only independent of a KB against which the query is answered, but also independent of a DL in which a KB is formulated. Therefore, extending the expressivity of a DL does not destroy our query answering algorithm (in contrast to other approaches where, e.g., introducing transitive roles can invalidate an algorithm).

In the next section we recall some notions concerning DLs and query answering. Section 3 contains the main theorem of the paper, which enables us to transfer results from modal correspondence theory to query answering. As an application, Sect. 4 and 5 present two families of queries that can be handled by this approach. As a by-product, in Sect. 5 we obtain an (at least syntactic) extension to the well-known in modal logic Sahlqvist theorem. Numerous examples of queries captured by this technique are shown in Sect. 6. We give conclusions and an outlook in Sect. 7. An Appendix contains all the proofs.

$$\begin{array}{lll} \top^{\mathcal{I}} = \Delta^{\mathcal{I}} & \{a\}^{\mathcal{I}} = \{a^{\mathcal{I}}\} & (C \sqcap D)^{\mathcal{I}} = C^{\mathcal{I}} \cap D^{\mathcal{I}} \\ (\neg C)^{\mathcal{I}} = \Delta^{\mathcal{I}} \setminus C^{\mathcal{I}} & & (R^-)^{\mathcal{I}} = \{\langle e,d\rangle \mid \langle d,e\rangle \in R^{\mathcal{I}}\} \\ \multicolumn{3}{l}{(\geqslant n\, R.C)^{\mathcal{I}} = \{\ e \in \Delta^{\mathcal{I}} \ \mid\ \#\{d \in C^{\mathcal{I}} \colon\ \langle e,d\rangle \in R^{\mathcal{I}}\} \geqslant n\ \}} \end{array}$$

Fig. 1. Semantics for Description Logics.

2 Preliminaries

First let us recall the notions related to the DL $\mathcal{ALC}$ and its extensions.

Definition 1. (Syntax) The vocabulary consists of finite sets of *concept names* CN, *role names* RN, and *individual names* IN (also called as *constants*). Concepts of the logic $\mathcal{ALC}$ are defined by the following syntax:

$$C ::= \quad \top \quad | \quad A \quad | \quad \neg C \quad | \quad C \sqcap D \quad | \quad \exists R.C,$$

where A is a concept name, R a role name, and C and D are concepts. Other connectives are taken as customary abbreviations, e.g., $(C \sqcup D) := \neg(\neg C \sqcap \neg D)$, $(C \to D) := \neg(C \sqcap \neg D)$, $\bot := \neg\top$, $\forall R.C := \neg\exists R.\neg C$.

A *terminology* (or a TBox) $\mathcal{T}$ is a finite set of *axioms* of the form $C \sqsubseteq D$, where C, D are arbitrary concepts. An ABox $\mathcal{A}$ is a finite set of *assertions* of the form $a{:}\,C$ and aRb, where $a, b \in \mathsf{IN}$, C is a concept and R a role. Finally, a *knowledge base* $\mathcal{KB} = \langle \mathcal{T}, \mathcal{A}\rangle$ consists of a TBox $\mathcal{T}$ and an ABox $\mathcal{A}$.

Appending the letters $\mathcal{I}$, $\mathcal{Q}$, $\mathcal{O}$, or $\mathcal{H}$ to the name of the logic refers to the following syntax extensions:

$\mathcal{I}$: *inverse roles*: if R is a role, then R^- is a role;
$\mathcal{Q}$: *qualified number restrictions*: if R is a role, C a concept, and $n > 0$, then $(\geqslant n\, R.C)$ is a concept; then we can use an abbreviation $\exists R.C := (\geqslant 1\, R.C)$;
$\mathcal{O}$: *nominals*: if $a \in \mathsf{IN}$, then $\{a\}$ is a concept;
$\mathcal{H}$: *role hierarchy*: axioms of the form $R \sqsubseteq S$ are allowed in a TBox.

Replacing '$\mathcal{ALC}$' with '$\mathcal{S}$' in the name of a logic refers to allowing transitive roles, i.e., axioms of the form $\mathsf{Trans}(R)$ in a TBox. To maintain decidability, in presence of $\mathcal{S}$, $\mathcal{H}$, and $\mathcal{Q}$ together, the restriction is imposed on the concept syntax: expressions of the form $(\geqslant n\, R.C)$ are regarded as well-formed concepts only if the role R has no transitive subroles w.r.t. given TBox (cf. [9]).

Definition 2. (Semantics) An *interpretation* $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ consists of non-empty *domain* $\Delta^{\mathcal{I}}$ and an interpretation function $\cdot^{\mathcal{I}}$ that maps:

– each constant $a \in \mathsf{IN}$ to an element $a^{\mathcal{I}} \in \Delta^{\mathcal{I}}$,
– each concept name $C \in \mathsf{CN}$ to a subset $C^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$,
– each role name $R \in \mathsf{RN}$ to a binary relation $R^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$;

and is extended to all concepts and roles by inductive clauses shown in Fig. 1. The *unique name assumption* (UNA) is customary made, i.e., only interpretations mapping distinct constants to distinct elements of the domain $\Delta^{\mathcal{I}}$ are considered.

An interpretation $\mathcal{I}$ *satisfies* an axiom $C \sqsubseteq D$, $R \sqsubseteq S$, or $\mathsf{Trans}(R)$ resp., if $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$, $R^{\mathcal{I}} \subseteq S^{\mathcal{I}}$, or the relation $R^{\mathcal{I}}$ is transitive, resp.; $\mathcal{I}$ satisfies an assertion $a{:}\,C$ or aRb if $a^{\mathcal{I}} \in C^{\mathcal{I}}$ or $\langle a^{\mathcal{I}}, b^{\mathcal{I}}\rangle \in R^{\mathcal{I}}$, resp. An interpretation is a *model* of a knowledge base if it satisfies all its TBox axioms and ABox assertions. The fact '$\mathcal{I}$ satisfies $\varPhi$' is written as $\mathcal{I} \models \varPhi$. A knowledge base $\mathcal{KB}$ *entails* $\varPhi$ (written as $\mathcal{KB} \models \varPhi$) if, for all models $\mathcal{I}$ of $\mathcal{KB}$, we have $\mathcal{I} \models \varPhi$.

Definition 3. (Queries) A *conjunctive query* is an expression of the form[1]

$$q(\vec{x}) \;\leftarrow\; t_1(\vec{x},\vec{y}) \wedge \ldots \wedge t_n(\vec{x},\vec{y}),$$

where $\vec{x}, \vec{y}$ are tuples of (*distinguished*, resp., *non-distinguished*) variables, and each *atom* $t_i(\vec{x},\vec{y})$ is of the form $w{:}\,C$ (*concept atom*) or wRz (*role atom*), where C is a concept, R a role, and w, z are either variables from $\vec{x}, \vec{y}$ or constants. The *arity* of a query $q(\vec{x})$ is the number of its distinguished variables: $\mathsf{ar}(q) := |\vec{x}|$. Queries of arity 1 are called *unary*. Given an interpretation $\mathcal{I} = \langle \varDelta, \cdot^{\mathcal{I}}\rangle$, a query q of arity m is interpreted as follows:

$$q^{\mathcal{I}} := \{\ \vec{e} \in \varDelta^m \ \mid\ \mathcal{I} \models \exists \vec{y}\ (\, t_1(\vec{e},\vec{y}) \wedge \ldots \wedge t_n(\vec{e},\vec{y})\,)\ \}.$$

The *answer set* of a query $q(\vec{x})$ w.r.t. a knowledge base $\mathcal{KB}$ is defined as the set of tuples of constants satisfying the query q in all models of $\mathcal{KB}$:

$$\mathsf{ans}_{\mathcal{KB}}(q) := \{\ \vec{a} \in \mathsf{IN}\ \mid\ \mathcal{KB} \models q(\vec{a})\ \}.$$

Note that the answer set is always finite: $|\,\mathsf{ans}_{\mathcal{KB}}(q)| \leqslant |\mathsf{IN}|^{\mathsf{ar}(q)}$, and corresponds, in a model $\mathcal{I}$ of $\mathcal{KB}$, to a subset (a *named* part) of $q^{\mathcal{I}}$, i.e., the inclusion $\mathsf{ans}_{\mathcal{KB}}(q)^{\mathcal{I}} \subseteq q^{\mathcal{I}}$ always holds.

It is worth saying that the query answering problem is closely related to the query subsumption problem addressed by many researches [3, 10]. Recall that a query $q(\vec{x})$ *subsumes* a query $q'(\vec{x})$ w.r.t. a $\mathcal{KB}$ (written as $\mathcal{KB} \models q \sqsupseteq q'$) if $q^{\mathcal{I}} \supseteq q'^{\mathcal{I}}$ holds for every model $\mathcal{I}$ of $\mathcal{KB}$. In fact, these two reasoning problems are reducible to each other. Indeed, a tuple $\vec{a}$ belongs to the answer set of $q(\vec{x})$ iff the subsumption $\mathcal{KB} \models \top \sqsubseteq q(\vec{a})$ holds. Conversely, given a subsumption $\mathcal{KB} \models q(\vec{x}) \sqsubseteq q'(\vec{x})$ to be tested, let $\vec{y}$ be the list of non-distinguished variables in $q(\vec{x})$. Then we introduce tuples of *fresh* constants $\vec{a_x}$ and $\vec{a_y}$ corresponding to $\vec{x}$ and $\vec{y}$, resp., and define the *canonical* ABox $\mathcal{A}_q$ for q as the set of atoms of $q(\vec{x})$ with constants $\vec{a_x}$ and $\vec{a_y}$ substituted for variables $\vec{x}$ and $\vec{y}$:

$$\text{if}\ \ q(\vec{x}) = \textstyle\bigwedge_{i=1}^{n} t_i(\vec{x},\vec{y}) \quad \text{then}\ \ \mathcal{A}_q := \{\ t_i(\vec{a_x},\vec{a_y})\ \mid\ 1 \leqslant i \leqslant n\ \}.$$

Then it is not hard to show that $\mathcal{KB} \models q \sqsubseteq q'$ iff the tuple $\vec{a_x}$ belongs to the answer set of $q'(\vec{x})$ w.r.t. the knowledge base $\mathcal{KB} \cup \mathcal{A}_q$, i.e., $\mathcal{KB} \cup \mathcal{A}_q \models q'(\vec{a_x})$.

[1] The existential quantifier $\exists \vec{y}$ in front of the r.h.s. of the query is customary omitted (and so we did), though implicitly assumed, as follows from the semantics below.

3 Queries answered by concepts

In the remainder of the paper, we will be concerned with unary queries only. The traditional query answering algorithms are usually based on the so called *rolling-up* technique (see [17] for definitions). In its basic form, when applied to a unary query $q(x)$, it can be viewed as a method of building a concept C that is equivalent to $q(x)$ (in the sense that $q^{\mathcal{I}} = C^{\mathcal{I}}$ for all models $\mathcal{I}$ of $\mathcal{KB}$), so that to answer the query $q(x)$ is the same as to retrieve all instances of the concept C. However, the equivalence of a concept C to a query $q(x)$ is sufficient, but not necessary for query answering purposes. Moreover, a rather simple query $q(x) \leftarrow xRx$ is not equivalent to any $\mathcal{ALC}$-concept; this can be easily shown using the tree model property for $\mathcal{ALC}$. Rolling-up techniques work particularly good for tree-like queries, but face certain problems when applied to cyclic ones (see, e.g., [6]). To overcome these difficulties, we introduce the following main notion of our paper.

Definition 4. A unary query $q(x)$ is *answered by* a concept C (written as $q(x) \approx C$) if $q(x)$ and C (considered as a query $x{:}C$) have the same answer sets w.r.t. any knowledge base $\mathcal{KB}$: $\mathsf{ans}_{\mathcal{KB}}(q) = \mathsf{ans}_{\mathcal{KB}}(x{:}C)$. In other words, $q(x) \approx C$ iff, for any $\mathcal{KB}$ and any $a \in \mathsf{IN}$, we have: $\mathcal{KB} \models q(a) \Leftrightarrow \mathcal{KB} \models a{:}C$.

Notice the quantification over *any* KB in this definition. Strictly speaking, we should formulate the definition as follows: a query $q(x)$ is answered by a concept C *over a description logic* $\mathcal{L}$ if $\mathsf{ans}_{\mathcal{KB}}(q) = \mathsf{ans}_{\mathcal{KB}}(x{:}C)$ for any $\mathcal{KB}$ formulated in the logic $\mathcal{L}$. However, we will not complicate the matters in this way, since all the results in our paper will have the form: a certain query $q(x)$ is answered by a concept C over *any* DL $\mathcal{L}$ (at least containing the syntax necessary for formulating C and q) and even over any first-order theory etc. The following lemma will be useful for us later.

Lemma 5. *If the queries $q_1(x) \leftarrow \alpha(x, \vec{y})$ and $q_2(x) \leftarrow \beta(x, \vec{z})$ are answered by concepts C and D, resp., and $\vec{y}, \vec{z}$ are disjoint lists of non-distinguished variables, then the query $q(x) \leftarrow \alpha(x, \vec{y}) \wedge \beta(x, \vec{z})$ is answered by the concept $C \sqcap D$.*

Proof. Follows from: $\exists \vec{y} \exists \vec{z}\, (\alpha(x, \vec{y}) \wedge \beta(x, \vec{z})) \;\equiv\; \exists \vec{y}\, \alpha(x, \vec{y}) \wedge \exists \vec{z}\, \beta(x, \vec{z})$. $\dashv$

Our task is to determine what kind of queries can be answered by concepts and when these concepts can be found efficiently (and preferably in the same language as the query). We will see that this can be done for a wide range of unary queries, provided that we are allowed to use "fresh" concept names not occurring in KBs and queries under consideration, i.e., specially reserved for query answering purposes. This is a rather customary assumption: in the rolling-up technique, new ("representative") concept names are also freely introduced (cf. [17]). But, unlike rolling-up, we do not need to add assertions involving these fresh concept names to a $\mathcal{KB}$. Fresh concept names will be denoted by capital letters from the end of alphabet (X, Y, Z, etc.), possibly sub- or superscripted.

The remainder of the section is devoted to formulation and proof of the main theorem of our paper. This theorem allows us to transfer results from the *modal* *correspondence theory* to the query answering field. The background information on that theory can be found in [2, Chap. 3]; here we will recall its basic notions.

It is known (cf. [16]) that the DL $\mathcal{ALC}$ with role names $R_1, \dots, R_n$ is a notational variant of the propositional multi-modal language with modalities $\Box_1, \dots, \Box_n$, whose formulas are defined inductively: $\bot$ and propositional letters $p_0, p_1, \dots$ are formulas; if φ and ψ are formulas, then so are $\varphi \to \psi$ and $\Box_i \varphi$. Exploiting this fact, whenever φ is a modal formula, we denote by C_φ the concept obtained from φ by replacing $\Box_i$ with $\forall R_i$ and p_j with *fresh* concept name X_j. Kripke semantics for the modal language is introduced as usually (cf. [2]). A modal formula φ is *valid* at the point e of a frame F (written as $F, e \Vdash \varphi$) if, for any valuation ν on F, φ is true at e in a model $\langle F, \nu \rangle$.

Consider also the first-order language (with equality) having only binary predicate symbols $R_1, \dots, R_n$. Note that conjunctive queries containing no concept atoms (we will call them *relational* queries) belong to this first-order language. The following is the key notion in modal correspondence theory.

Definition 6. A first-order formula $\alpha(x)$ with a single free variable *locally corresponds* to a modal formula φ (written as $\alpha(x) \leftrightsquigarrow \varphi$) if, for any frame F and its point e, the equivalence holds: $F \models \alpha(e) \Leftrightarrow F, e \Vdash \varphi$.

Now we are ready to prove the theorem that establishes a connection between the notion of local correspondence and the query answering problem for relational queries. Later on, we will extend the family of queries covered by this approach to the ones containing concept atoms as well.

Theorem 7 (Reduction). *If a unary relational query $q(x)$ locally corresponds to a modal formula φ, then the query $q(x)$ is answered by the $\mathcal{ALC}$-concept C_φ. In symbols: $q(x) \leftrightsquigarrow \varphi \Longrightarrow q(x) \approx C_\varphi$.*

The proof can be found in the Appendix. This theorem allows us to reuse *positive* results from the modal correspondence theory for query answering purposes. For example, it implies that the query $q(x) \leftarrow xRx$ is answered by the concept $X \to \exists R.X$ (where X is a fresh concept name). Other applications will be demonstrated in the subsequent sections. A natural question is whether the converse of this theorem holds; it would enable also to transfer *negative* results. We have succeeded to prove only a half of the converse, in the following sense (the proof is again in the Appendix).

Lemma 8. *Suppose that a unary relational query $q(x)$ is answered by a concept C_φ, i.e., $q(x) \approx C_\varphi$, for some modal formula φ. Then for any frame F and its point e, the condition $F \models q(e)$ implies*[2] *$F, e \Vdash \varphi$.*

In what follows, we consider only queries with no constants in role atoms, since the general query answering problem is reducible to this case, at the price of introducing nominals in concept atoms: we can equivalently rewrite an atom yRa as $yRz \wedge z\colon\{a\}$, for a new non-distinguished variable z, and similarly for aRy.

[2] Replacing this implication by equivalence would yield the converse of Theorem 7. After this paper has been accepted, the author has succeeded to prove the converse implication, but only for *finitely branching* frames.

4 Application 1: Queries and the Sahlqvist fragment

The notion of local correspondence is extensively explored in modal logic. The most prominent in this direction is the result that a wide family of modal formulas (known as *Sahlqvist formulas*) locally correspond to first-order (FO) conditions, and moreover, there is an algorithm that takes a Sahlqvist formula and returns the corresponding FO condition (see [2, §3.5–3.6] for details; originally in [15]). We will use here a sort of converse of this result, obtained by M. Kracht (see [2, §3.7]; originally in [13]), who described syntactically the class of all those FO formulas (now called *Kracht formulas*) that locally correspond to Sahlqvist formulas, and provided the corresponding algorithm. We will apply this result to those Kracht formulas that have the form of a (unary relational) conjunctive query and then extend it to queries containing also concept atoms.

First, recall the definition of Kracht formulas. Consider a FO language having only binary relation symbols $R_1, \dots, R_n$. Formulas of the form $\exists y(xRy \wedge \alpha(y))$ and $\forall y(xRy \to \alpha(y))$ will be abbreviated[3] as $(\exists y\overline{R}x)\alpha(y)$ and $(\forall y\overline{R}x)\alpha(y)$, resp. A formula is called *clean* if no variable occurs both free and bound in it, and no two distinct occurrences of quantifiers bind the same variable. The formulas $\top$, $\bot$, and $xR_\sigma y$ are called *quasi-atomic*, where $\sigma = (i_1, \dots, i_r)$ is a sequence of indices from $\{1, \dots, n\}$, and $xR_\sigma y$ stands for $\exists z_2 \dots \exists z_r(xR_{i_1}z_2 \wedge z_2R_{i_2}z_3 \wedge \dots \wedge z_rR_{i_r}y)$.

Definition 9. A formula is *restrictedly positive* if it is built up from quasi-atomic ones using conjunction $\wedge$, disjunction $\vee$, and restricted quantifiers $(\exists y\overline{R}x)$ and $(\forall y\overline{R}x)$. A variable y in a *clean* formula is called *inherently universal* if either y is free, or y is bound by a restricted universal quantifier $(\forall y\overline{R}x)$ which is not in the scope of any existential quantifier. Finally, a *Kracht formula* is a clean, restrictedly positive formula such that every of its quasi-atomic subformulas contains at least one inherently universal variable.

For example, $(\forall y\overline{R}x)(\forall z\overline{R}x)(\exists w\overline{R}y)zRw$ is a Kracht formula, whereas the formula $(\exists y\overline{R}x)(\forall z\overline{R}x)yRz$ is not. Now we present a family of relational queries that belong to the Kracht fragment. We will not prove that this family is exhaustive, in the sense that it covers, modulo equivalence, exactly the intersection of the Kracht fragment and conjunctive queries (although this is likely to be true).

Lemma 10. *Any relational conjunctive query of the following form is (equivalent to a Kracht formula and hence) answered by an $\mathcal{ALC}$-concept:*

$$q(x) \;\leftarrow\; \exists\vec{y}\,\big(\,\mathsf{Tree}(x,\vec{y}) \;\wedge\; \textstyle\bigwedge_{i,j} xR_{\sigma_i}y_j \;\wedge\; \textstyle\bigwedge_{k,\ell} y_kR_{\tau_\ell}x\,\big),$$

where $\mathsf{Tree}(x,\vec{y})$ *is a conjunction of role atoms forming an oriented x-rooted tree.*[4] *In other words, $q(x)$ states the existence of an x-rooted oriented tree with additional oriented chains linking x with other nodes (in any direction).*

[3] This is similar to designating a formula $\exists y(x \leqslant y \wedge \alpha(y))$ by $(\exists y \geqslant x)\alpha(y)$.

[4] An *oriented tree* is a connected oriented graph without cycles (and loops $\langle e, e\rangle$) s.t. each node has at most one incoming edge. Its *root* is a node with no incoming edges.

PROOF. The conjunct $\mathsf{Tree}(x, \vec{y})$ allows to equivalently rewrite the quantifier prefix $\exists\vec{y}$ into a restricted one:

$$\exists\vec{y}\,(\mathsf{Tree}(x,\vec{y}) \wedge \alpha(x,\vec{y})) \equiv (\exists y_1 \overline{R}_1 y_1')\ldots(\exists y_r \overline{R}_r y_r')\,\alpha(x,\vec{y}),$$

where the condition $y_i' \in \{x, y_1, \ldots, y_{i-1}\}$ holds, since the formula is clean and x is the only free variable. It is easily seen that the r.h.s. is a Kracht formula, and hence locally corresponds to a modal formula. Now we apply Theorem 7 to conclude that the query $q(x)$ is answered by some $\mathcal{ALC}$-concept. $\dashv$

Next we describe an algorithm for answering these queries; it also captures queries with concept atoms. By Lemma 5, we can consider queries $q(x)$ without atoms of the form $x{:}\,A$ and xRx, since adding these atoms is equivalent to adding the conjuncts A and $X \to \exists R.X$ to the concept that answers the query.

The algorithm for answering queries within the Kracht fragment

Assume that we are given a conjunctive query of the form

$$q(x) \;\leftarrow\; \exists\vec{y}\,\big(\,\mathsf{Tree}(x,\vec{y}) \;\wedge\; \underbrace{\textstyle\bigwedge_{i,j} xR_{\sigma_i}y_j}_{\alpha(x,\vec{y})} \;\wedge\; \underbrace{\textstyle\bigwedge_{k,\ell} y_kR_{\tau_\ell}x}_{\beta(x,\vec{y})} \;\wedge\; \underbrace{\textstyle\bigwedge_t y_t{:}\,B_t}_{\gamma(\vec{y})}\big), \qquad (\mathcal{K})$$

where $\mathsf{Tree}(x,\vec{y})$ is a conjunction of role atoms forming an oriented x-rooted tree; expressions of the form $yR_\sigma z$ are role chains linking y with z; the components of the query are denoted by α, β, γ for easy reference. We assume that the conjuncts in the above formula $(\mathcal{K})$ are distinct. Then we proceed as follows:

1) Introduce *fresh* concept names: $Y_{i,j}$ for each conjunct $xR_{\sigma_i}y_j$ in $\alpha(x,\vec{y})$, and $X_{k,\ell}$ for each conjunct $y_kR_{\tau_\ell}x$ in $\beta(x,\vec{y})$.

2) Build two conjunctions PRE and CON (for 'premise' and 'conclusion', resp.) by first putting PRE $:= x{:}\,\top$ and CON $:= y_1{:}\,\top \wedge \ldots \wedge y_r{:}\,\top$ and then proceed as follows (here $\forall R_\sigma.Y$ stands for $\forall R_{i_1}.\cdots\forall R_{i_r}.Y$, if $\sigma = (i_1, \ldots, i_r)$, and similarly for $\exists R_\tau.X$):
   - for each $xR_{\sigma_i}y_j$ in α, add $x{:}\forall R_{\sigma_i}.Y_{i,j}$ to PRE and $y_j{:}\,Y_{i,j}$ to CON;
   - for each $y_kR_{\tau_\ell}x$ in β, add $x{:}\,X_{k,\ell}$ to PRE and $y_k{:}\,\exists R_{\tau_\ell}.X_{k,\ell}$ to CON;
   - for each $y_t{:}\,B_t$ in γ, add the conjunct $y_t{:}\,B_t$ to CON.

3) In PRE and CON obtained, merge conjuncts related to the same variables, i.e., replace $z{:}\,C \wedge z{:}\,D$ by $z{:}\,(C \sqcap D)$. After this, PRE will be of the form $x{:}\,C$ and CON of the form: $y_1{:}\,D_1 \wedge \ldots \wedge y_r{:}\,D_r$, for some concepts C and D_i.

4) Form the expression PRE $\to \exists\vec{y}\,(\mathsf{Tree}(x,\vec{y}) \wedge$ CON $\wedge\, x{:}\,\top)$, or explicitly:

$$x{:}\,C \to (\exists y_1\overline{R}_1y_1')\ldots(\exists y_r\overline{R}_ry_r')\,(\,y_1{:}\,D_1 \wedge \ldots \wedge y_r{:}\,D_r \wedge x{:}\,\top\,)$$

where $y_i' \in \{x, y_1, \ldots, y_{i-1}\}$. Now apply the standard rolling-up technique to turn the restricted quantifiers $(\exists y_i\overline{R}_iy_i')$ into DL-quantifiers, starting from the innermost one. For example, assuming for notational simplicity that $y_r' \equiv y_1$ (the case $y_r' \equiv x$ is similar), the variable y_r is eliminated as follows:

$$\begin{aligned}&(\exists y_r\overline{R}_ry_1)\,(\,y_1{:}\,D_1 \wedge \ldots \wedge y_r{:}\,D_r \wedge x{:}\,E\,) \;\equiv\\ &y_1{:}\,(D_1 \sqcap \exists R_r.D_r) \wedge y_2{:}\,D_2 \wedge \ldots \wedge y_{r-1}{:}\,D_{r-1} \wedge x{:}\,E.\end{aligned}$$

After eliminating all the variables y_i, CON will be of the form $x\colon D$. Hence the expression considered at the beginning of Item 4 is equivalently transformed into $x\colon C \to x\colon D$. The desired concept answering the query $q(x)$ is $(C \to D)$, or, in the traditional DL notation, $\neg C \sqcup D$.

Note that the resulting concept belongs to the minimal DL containing the concepts B_t occurring in the original query. In particular, for a relational query we obtain an $\mathcal{ALC}$-concept. We will illustrate how this algorithm works in Ex. 5 (see Sect. 6). The algorithm described above is taken from the general Kracht's Theorem (see [2, §3.7]) and simplified for the case of FO formulas having the form of conjunctive queries (and slightly modified to capture concept atoms). One can follow the proof of Kracht's Theorem to show that the algorithm indeed returns a concept that answers the original query. We sum up with the following

Theorem 11. *The query answering problem for unary conjunctive queries from the family $\mathcal{K}$ is linearly reduced to the problem of concept instance retrieval (which in turn is reducible to knowledge base unsatisfiability).*

5 Application 2: Parallel-serial queries

It is known that the Sahlqvist fragment does not cover all modal formulas having FO correspondents, as well as the Kracht fragment does not contain all FO formulas that locally correspond to modal formulas. In this section we show that the family $\mathcal{K}$ of queries (which are equivalent to some Kracht formulas) does not cover all the conjunctive queries answered by $\mathcal{ALC}$-concepts. This result can be regarded as *syntactic* extension to Kracht's theorem. Since we did not prove that the family $\mathcal{K}$ is exactly the restriction of the Kracht fragment to the set of conjunctive queries, we cannot conclude that the queries considered below form a *semantically* proper extension to the Kracht fragment. At the end of this section we also describe a wider family of queries answered by $\mathcal{ALCI}$-concepts.

First, we define the family of *parallel-serial* queries (or ps-queries, for short) with two *poles* x and y (where x is the distinguished and y a non-distinguished variable) inductively. A query $q(x) \leftarrow xRy$ is an *atomic* ps-query with the poles x and y. If $q_1(x) \leftarrow \alpha(x, y, \vec{u})$ and $q_2(x) \leftarrow \beta(x, y, \vec{v})$ are ps-queries with the poles x and y and the tuples $\vec{u}$ and $\vec{v}$ are disjoint, then $q(x) \leftarrow \alpha(x, y, \vec{u}) \wedge \beta(x, y, \vec{v})$ is a ps-query with the same poles x and y called a *parallel connection* of q_1 and q_2 and denoted as $q_1 \| q_2$. If $q_1(x) \leftarrow \alpha(x, y, \vec{u})$ is a ps-query with the poles x and y, $q_2(y) \leftarrow \beta(y, z, \vec{v})$ is a ps-query with the poles y and z, and α and β have only y as a common variable, then $q(x) \leftarrow \alpha(x, y, \vec{u}) \wedge \beta(y, z, \vec{v})$ is a ps-query with the poles x and z called a *serial connection* of q_1 and q_2 and denoted as $q_1 \circ q_2$.

To a ps-query $q(x)$ we associate a role expression $\mathcal{R}(q)$ built from role names using intersection and composition: for an atomic query $q(x) \leftarrow xRy$, we put $\mathcal{R}(q) := R$; and inductive steps are: $\mathcal{R}(q_1 \| q_2) := \mathcal{R}(q_1) \sqcap \mathcal{R}(q_2)$, $\mathcal{R}(q_1 \circ q_2) := \mathcal{R}(q_1) \circ \mathcal{R}(q_2)$. By induction, it is straightforward to show that any ps-query $q(x)$ is answered by (and even equivalent to) the $\mathcal{ALC}(\sqcap, \circ)$-concept $\exists \mathcal{R}(q).\top$ (note that this concept contains no concept names).

Unfortunately, for most DLs used in practice, this solution to query answering is unsatisfactory, since adding role intersection and composition to a logic (together with arbitrary concept and role inclusion axioms in a TBox) leads to undecidability. Therefore, our next aim is to find conditions under which a ps-query is answered by an $\mathcal{ALC}$-concept. For convenience, we extend Definition 4 to the case of two concepts: we say that concepts C and D are *similar* (written as $C \approx D$) if they have the same instances w.r.t. any knowledge base $\mathcal{KB}$, i.e., $\mathcal{KB} \models a{:}\,C \Leftrightarrow \mathcal{KB} \models a{:}\,D$, for any $a \in \mathsf{IN}$.

Lemma 12 (Elimination Lemma). *The concept $\exists(R \sqcap S).C$ is similar to the concept $\forall R.Y \to \exists S.(Y \sqcap C)$ (where Y is a fresh concept name). In general:*

$$\exists(R_0 \sqcap \ldots \sqcap R_n).C \;\approx\; \big(\forall R_1.Y_1 \sqcap \ldots \sqcap \forall R_n.Y_n \;\to\; \exists R_0.(Y_1 \sqcap \ldots \sqcap Y_n \sqcap C)\big).$$

Proof. The concept $\exists(R \sqcap S).C$ is equivalent to the conjunctive query $q(x) \leftarrow (xRy \land xSy \land x{:}\,C)$, which belongs to the family $\mathcal{K}$ from Sect. 4. Applying the algorithm described there, we obtain that $q(x)$ is answered by the concept $\forall R.Y \to \exists S.(Y \sqcap C)$. The proof for several roles is completely analogous. $\dashv$

Now we introduce a new family $\mathcal{Z}$ of queries: *it consists of queries whose relational part is a ps-query built up from atomic ones using parallel connections and restricted serial connections: $q_1 \circ q_2$ is allowed only if q_2 is atomic.*

Theorem 13. *Any query from the family $\mathcal{Z}$ is answered by a concept that can be built in linear time. Relational queries from $\mathcal{Z}$ are answered by $\mathcal{ALC}$-concepts.*

The proof is in the Appendix. The families $\mathcal{K}$ and $\mathcal{Z}$ are incomparable w.r.t. inclusion: indeed, the following query belongs to $\mathcal{K} \setminus \mathcal{Z}$:

Query:	$q(x) \;\leftarrow\; xRy \land yRz \land yRw \land xPz \land xQw$
Concept:	$\forall P.Z \sqcap \forall Q.W \to \exists R.(\exists R.Z \sqcap \exists R.W)$

whereas the following one belongs to $\mathcal{Z} \setminus \mathcal{K}$ (but we do not know whether it is equivalent to any Kracht formula beyond the family $\mathcal{K}$):

Query:	$q(x) \;\leftarrow\; xRy \land xSy \land yPw \land xRz \land xSz \land zQw$
Concept:	$\forall R.(Y \sqcap Z) \sqcap \forall S.(Y \to \forall P.W) \to \exists S.(Z \sqcap \exists Q.W)$

It is straightforward to "lift" Theorem 13 to modal logic, thus obtaining the statement: *any first-order formula $q(x)$ from the family $\mathcal{Z}$ locally corresponds to a modal formula that can be obtained efficiently.* This is a *syntactic* extension to Kracht's theorem (as already pointed out, we do not know if this is a *semantically* proper extension). At the same time, this can be regarded as an argument in favour of the (unproved) converse of Theorem 7: the family $\mathcal{Z}$ did not bring us a counterexample to the converse of Theorem 7.

Notice that all relational queries considered so far were answered by $\mathcal{ALC}$-concepts. What if we allow inverse roles? If we "forget" the direction of edges in the queries from the families $\mathcal{K}$ and $\mathcal{Z}$, then we arrive to the following family $\mathcal{Y}$ of queries (formulated in terms of the underlying graph of a query, whose nodes are variables and edges correspond to role atoms): *$\mathcal{Y}$ is the family of connected* *queries $q(x)$ without cycles (even non-oriented) consisting of non-distinguished variables only, i.e., every cycle in $q(x)$ must contain the distinguished variable x.* We conclude with the following theorem (its proof is in the Appendix).

Theorem 14. *Any query from the family $\mathcal{Y}$ is answered by a concept that can be built in linear time. Relational queries from $\mathcal{Y}$ are answered by $\mathcal{ALCI}$-concepts. The family $\mathcal{Y}$ includes both families $\mathcal{K}$ and $\mathcal{Z}$, i.e., $\mathcal{K} \cup \mathcal{Z} \subset \mathcal{Y}$.*

6 The harvest

Here we present concrete examples of the results obtained. As already mentioned above, we can consider queries $q(x)$ without an atom of the form $x\colon A$.

Example 1. Since reflexivity is expressible by the modal formula $p \to \Diamond p$, we conclude that the query $q(x) \leftarrow xRx$ is answered (over any DL knowledge base) by the concept $X \to \exists R.X$. Similarly, the results from Sect. 4 yield the following (cyclic) queries and corresponding concepts:

Query	Concept
$q_1(x) \leftarrow xRy \wedge ySx$	$X \to \exists R.\exists S.X$
$q_2(x) \leftarrow xRy \wedge xSy$	$\forall R.Y \to \exists S.Y$
$q_3(x) \leftarrow xRy \wedge ySx \wedge y\colon B$	$X \to \exists R.(B \sqcap \exists S.X)$
$q_4(x) \leftarrow xRy \wedge xSy \wedge y\colon B$	$\forall R.Y \to \exists S.(B \sqcap Y)$

The query $q_1(x)$ can be easily generalised to the case of an oriented cycle of the length n starting and ending at x, and iterating $\exists R$ in the corresponding concept. Similarly, one can generalise the query $q_2(x)$, replacing R and S by chains of the length n and m both going from x to y, as shown in our next example.

Example 2. For any $m \geqslant 0$ and $n \geqslant 0$ with $m + n > 0$, the query[5]

$$\begin{aligned} q(x) \leftarrow\ & xR_1y_2 \wedge y_2R_2y_3 \wedge \ldots \wedge y_mR_my \wedge \\ & xS_1z_2 \wedge z_2S_2z_3 \wedge \ldots \wedge z_nS_ny \end{aligned}$$

is answered by an $\mathcal{ALC}$-concept $\forall R_1.\cdots\forall R_m.Y \to \exists S_1.\cdots\exists S_n.Y$, and the query

$$\begin{aligned} q(x) \leftarrow\ & xR_1y_2 \wedge y_2R_2y_3 \wedge \ldots \wedge y_mR_my \ \wedge\ y_2\colon B_2 \wedge \ldots \wedge y_m\colon B_m \wedge \\ & xS_1z_2 \wedge z_2S_2z_3 \wedge \ldots \wedge z_nS_ny \ \wedge\ z_2\colon C_2 \wedge \ldots \wedge z_n\colon C_n \ \wedge y\colon D \end{aligned}$$

(with B_i, C_j, D being concepts in a language $\mathcal{L}$) is answered by the $\mathcal{L}$-concept

$$\begin{aligned} & \exists R_1.(B_2 \sqcap \exists R_2.(B_3 \sqcap \ldots (B_m \sqcap \exists R_m.\neg Y)\ldots)) \sqcup \\ & \exists S_1.(C_2 \sqcap \exists S_2.(C_3 \sqcap \ldots (C_n \sqcap \exists S_n.(Y \sqcap D))\ldots)). \end{aligned}$$

The query $q_2(x)$ in Example 1 can be generalised in another direction: instead of two roles linking x with y, we can take several ones. This leads to the following example; and it is the first time where we need more than one fresh concept name.

[5] For $m = 0$ the first line in q is dropped and y replaced by x; and similarly for $n = 0$.

Example 3. A query $q(x) \leftarrow xR_0y \wedge \ldots \wedge xR_ny$ is answered by the $\mathcal{ALC}$-concept

$$\forall R_1.Y_1 \sqcap \ldots \sqcap \forall R_n.Y_n \;\rightarrow\; \exists R_0.(Y_1 \sqcap \ldots \sqcap Y_n).$$

Now if we stretch each R_i into a chain linking x with y, we obtain the following.

Example 4. The query of the form $q(x) \leftarrow q_0(x) \wedge \ldots \wedge q_n(x)$, where each $q_i(x)$ is a chain of m_i edges linking x with y:

$$q_i(x) \leftarrow \;\; xR^i_1y^i_2 \wedge y^i_2R^i_2y^i_3 \wedge \ldots \wedge y^i_{m_i}R^i_{m_i}y,$$

(superscripts are indices not powers) is answered by the $\mathcal{ALC}$-concept

$$\forall R_1.Y_1 \sqcap \ldots \sqcap \forall R_n.Y_n \;\rightarrow\; \exists R_0.(Y_1 \sqcap \ldots \sqcap Y_n),$$

where $\forall R_i$ stands for $\forall R^i_1. \cdots \forall R^i_{m_i}$ and similarly for $\exists R_0$. We leave adding concept atoms to this query as an exercise.

All these queries belong to the family $\mathcal{K}$ from introduced in Sect. 4: in the last example, the chain $q_0(x)$ can be regarded as a $\mathsf{Tree}(x,\vec{y})$, while the other chains $q_i(x)$ together constitute the conjunct $\alpha(x,\vec{y})$. An example of the query from the family $\mathcal{Z}$ (to be more exact, from $\mathcal{Z} \setminus \mathcal{K}$) was given after Theorem 13. Now we will illustrate how the algorithm presented in Sect. 4 works.

Example 5. Consider the query $q(x) \leftarrow \mathsf{Tree}(x,\vec{y}) \wedge \alpha(x,\vec{y}) \wedge \beta(x,\vec{y})$, where

$$\begin{aligned}
\mathsf{Tree}(x,\vec{y}) &\equiv xR_1y_1 \wedge y_1R_2y_2 \wedge y_1R_3y_3 \wedge y_1R_4y_4 \wedge y_4R_5y_5 \wedge y_4R_6y_6,\\
\alpha(x,\vec{y}) &\equiv xS_1y_1 \wedge xS_4y_6,\\
\beta(x,\vec{y}) &\equiv y_2S_2x \wedge y_5S_3x.
\end{aligned}$$

We introduce fresh concept names Y_{11} and Y_{46} for atoms in $\alpha(x,\vec{y})$, and X_{22} and X_{53} for atoms in $\beta(x,\vec{y})$. Next we build the expressions PRE and CON; they contain conjuncts that correspond to atoms in $\alpha(x,\vec{y})$ and $\beta(x,\vec{y})$:

$$\begin{array}{ll}
\text{PRE:} & x{:}\,\forall S_1.Y_{11} \wedge x{:}\,\forall S_4.Y_{46} \wedge x{:}\,X_{22} \wedge x{:}\,X_{53},\\
\text{CON:} & y_1{:}\,Y_{11} \wedge y_6{:}\,Y_{46} \wedge y_2{:}\,\exists S_2.X_{22} \wedge y_5{:}\,\exists S_3.X_{53}.
\end{array}$$

Then the expression PRE $\rightarrow \exists\vec{y}\,(\mathsf{Tree}(x,\vec{y}) \wedge$ CON) will look as follows:

$$\begin{array}{l}
x{:}\,\big(\,\forall S_1.Y_{11} \sqcap \forall S_4.Y_{46} \sqcap X_{22} \sqcap X_{53}\,\big) \;\rightarrow\\
(\exists y_1\overline{R}_1x)(\exists y_2\overline{R}_2y_1)(\exists y_3\overline{R}_3y_1)(\exists y_4\overline{R}_4y_1)(\exists y_5\overline{R}_5y_4)(\exists y_6\overline{R}_6y_4)\\
\big(\,y_1{:}\,Y_{11} \wedge y_6{:}\,Y_{46} \wedge y_2{:}\,\exists S_2.X_{22} \wedge y_5{:}\,\exists S_3.X_{53}\,\big).
\end{array}$$

Now we roll up the restricted quantifiers, starting from the innermost one. After rolling-up the quantifiers first over y_6 and then over y_5, we will obtain:

$$\begin{array}{l}
x{:}\,\big(\,\forall S_1.Y_{11} \sqcap \forall S_4.Y_{46} \sqcap X_{22} \sqcap X_{53}\,\big) \;\rightarrow\\
(\exists y_1\overline{R}_1x)(\exists y_2\overline{R}_2y_1)(\exists y_3\overline{R}_3y_1)(\exists y_4\overline{R}_4y_1)\\
\big(\,y_1{:}\,Y_{11} \wedge y_2{:}\,\exists S_2.X_{22} \wedge y_4{:}\,(\exists R_6.Y_{46} \sqcap \exists R_5.\exists S_3.X_{53})\,\big).
\end{array}$$

When all quantifiers are rolled-up, we end up with an expression of the form $x{:}\,(C \rightarrow D)$, where the concept $(C \rightarrow D)$ answering our original query $q(x)$ is:

$$\begin{array}{l}
\big(\,\forall S_1.Y_{11} \sqcap \forall S_4.Y_{46} \sqcap X_{22} \sqcap X_{53}\,\big) \;\rightarrow\\
\exists R_1.\big(\,Y_{11} \sqcap \exists R_2.\exists S_2.X_{22} \sqcap \exists R_3.\top \sqcap \exists R_4.(\exists R_6.Y_{46} \sqcap \exists R_5.\exists S_3.X_{53})\,\big).
\end{array}$$

7 Conclusions and outlook

One of the achievements of this paper is the established relationship between the notion of local correspondence from modal logic and the notion 'a concept answers a query' (Theorem 7). As an application, this enabled us to reuse the results from modal correspondence theory for answering a wide range of conjunctive queries with one distinguished variable. We have also found a syntactic (and probably semantic) extension to Kracht's theorem. Please note that, although at the first glace, only $\mathcal{ALC}$ or $\mathcal{ALCI}$ occur explicitly in Theorems 11,13,14, these should not be misunderstood as giving a query answering method for $\mathcal{ALC}$ or $\mathcal{ALCI}$ knowledge bases only. In fact, as we pointed out earlier, the results obtained provide us with algorithms for query answering over KBs formulated in any DL (extending $\mathcal{ALC}$), using as minimal means as possible (i.e., the resulting concept belongs to a simple logic $\mathcal{ALC}$ or $\mathcal{ALCI}$).

Analysing these families of queries, one can observe that the inherent complexity of query answering lies in the relational structure (i.e., the underlying graph) of the query. This structure was captured by concepts of $\mathcal{ALC}$ (Theorems 11 and 13) or $\mathcal{ALCI}$ (Theorem 14). A natural extension would be to invoke into this framework the correspondence theory for richer modal logics. The usage of qualified number restrictions (i.e., $\mathcal{ALCQ}$-concepts) for answering relational queries would involve the correspondence theory for graded modal logic. However, to the best of our knowledge, no general results similar to Sahlqvist's theorem are known for graded modal logic. Another possible direction is to extend this technique to DLs with relations of arbitrary arity, thus involving the correspondence theory for polyadic modal logics (which is explored extensively).

The need to investigate extensions stems from the restricted applicability of the technique we developed so far. For example, we have not succeeded to find an $\mathcal{ALC}$-concept answering the query $q(x) \leftarrow xRy \wedge ySy$. Moreover, if the converse of Theorem 7 holds (which is still an open question), then it would imply that such an $\mathcal{ALC}$-concept does not exist, since the property of frames expressed by $q(x)$ is not modally definable (see [2, Chap. 3], discussion after Corollary 3.16). At the same time, this and other queries shown in the table below can be answered by concepts involving extra role operations (whether they can be answered by any $\mathcal{ALCQI}$-concept is unknown):

Query	Concept
$q(x) \leftarrow xRy \wedge ySy$	$\exists R.\exists(S \sqcap id(\top)).\top$
$q(x) \leftarrow xRy \wedge ySz \wedge yPz$	$\exists R.\exists(S \sqcap P).\top$
$q(x) \leftarrow xRy \wedge xSz \wedge yPz \wedge zQy$	$\forall S.Z \rightarrow \exists R.\exists(P \sqcap Q^{-}).Z$
$q(x) \leftarrow xRyR'w \wedge xSzS'w \wedge yPz$	$\forall S.Z \rightarrow \exists R.\exists(P \sqcap (R' \circ S'^{-})).Z$

Finally, it is interesting whether all conjunctive queries can be answered by some concepts. As an example, for the following "tetrahedron" query

$$q(x) \;\leftarrow\; xRyR'w \wedge xSzS'w \wedge yPz \wedge xQw$$

we have neither found a concept in any DL that would answer it, nor proved that such a concept (in a certain DL) does not exist.

Acknowledgements

The research has been supported by EPSRC, Grant No. GR/S63168/01. The author would like to thank Dr. Ulrike Sattler and Dr. Ian Horrocks for guiding him in Description Logic and the help during the research. The author is also grateful to Birte Glimm for fruitful discussions on the subject of query answering and some useful suggestions.

References

1. F. Baader, D. Calvanese, D. McGuinness, D. Nardi, and P. F. Patel-Schneider, editors. *The Description Logic Handbook.* Cambridge University Press, 2003.
2. P. Blackburn, M. de Rijke, and Y. Venema, *Modal Logic*. Cambridge University Press, Theoretical Tracts in Computer Science, 2001.
3. D. Calvanese, G. De Giacomo, and M. Lenzerini. On the decidability of query containment under constraints. In *Proc. of the 17th ACM SIGACT SIGMOD SIGART Sym. on Principles of Database Systems (PODS'98)*, pp. 149–158, 1998.
4. D. Calvanese, G. De Giacomo, D. Lembo, M. Lenzerini, and R. Rosati. Data Complexity of Query Answering in Description Logics. In *Proc. of the Int. Workshop on Description Logic (DL'05)*, 2005. (`ceur-ws.org`)
5. D. Calvanese, M. Magdalena Ortiz de la Fuente, T. Eiter, and E. Franconi. Data complexity of answering conjunctive queries over $\mathcal{SHIQ}$ knowledge bases. Technical Report. 2005.
6. B. Glimm and I. Horrocks, Handling cyclic conjunctive queries, In *Proc. of the Int. Workshop on Description Logic (DL'05)*, p. 219, 2005. (`ceur-ws.org`)
7. V. Haarslev and R. Möller. Racer: An OWL Reasoning Agent for the Semantic Web. In *Proc. of the Int. Workshop on Applications, Products and Services of Web-based Support Systems, in conj. with the 2003 IEEE/WIC Int. Conf. on Web Intelligence*, Halifax, Canada, pages 91–95, 2003. `http://www.racer-systems.com/`
8. I. Horrocks. FaCT and iFaCT. In *Proc. of the Int. Workshop on Description Logics (DL'99)*, pages 133–135, 1999. (`ceur-ws.org`)
9. I. Horrocks and U. Sattler. A Tableaux Decision Procedure for $\mathcal{SHOIQ}$. In *Proc. of 19th Int. Joint Conf. on Artificial Intelligence (IJCAI 2005)*, 2005. Morgan-Kaufmann Publishers (to appear).
10. I. Horrocks, U. Sattler, S. Tessaris, and S. Tobies. How to decide query containment under constraints using a Description Logic. In *Proc. of the 7th Int. Conf. on Logic for Programming and Automated Reasoning (LPAR'2000)*, Lecture Notes in Artificial Intelligence. Springer-Verlag, 2000. Accompanying technical report: Query containment using a $\mathcal{DLR}$ ABox. LTCS-Report 99-15, LuFG Theoretical Computer Science, RWTH Aachen, Germany, 1999.
11. I. Horrocks, U. Sattler, and S. Tobies. Practical reasoning for very expressive description logics. *Logic Journal of the IGPL*, 8(3):239–263, 2000.
12. I. Horrocks and S. Tessaris. A conjunctive query language for description logic ABoxes. In *Proc. of the 17th Nat. Conf. on Artificial Intelligence (AAAI'2000)*, pp. 399–404, 2000.
13. M. Kracht. How completeness and correspondence theory got married. In de Rijke, editor, *Diamonds and Defaults*, pp. 175–214. Kluwer, 1993.
14. C. Lutz. *The Complexity of Description Logic with Concrete Domains.* PhD Thesis, LuFG Theoretical Computer Science, RWTH Aachen, Germany, 2002.

15. H. Sahlqvist. Correspondence and completeness in the first- and second-order semantics for modal logic. In S. Kanger, editor, *Proc. of the 3rd Scand. Logic Symp.*, Uppsala, 1973. North-Holland Publishing Company, Amsterdam, 1975.
16. K. Schild. A correspondence theory for terminological logics: Preliminary report. In *Proc. of the 12th Int. Joint Conf. on Artificial Intelligence (IJCAI'91)*, pp. 466–471, 1991.
17. S. Tessaris. *Questions and answers: reasoning and querying in Description Logic.* PhD thesis, University of Manchester, 2001.
18. University of Maryland. Pellet OWL reasoner, 2003. Institute for Advanced Computer Studies, MIND LAB, The Semantic Web Research Group. `http://www.mindswap.org/2003/pellet/`

Appendix A: Proofs

Theorem 7 (Reduction). *If a unary relational query $q(x)$ locally corresponds to a modal formula φ, then the query $q(x)$ is answered by the $\mathcal{ALC}$-concept C_φ. In symbols: $q(x) \leftrightsquigarrow \varphi \Longrightarrow q(x) \approx C_\varphi$.*

Proof. Suppose that $q(x) \leftrightsquigarrow \varphi$. Then, given a knowledge base $\mathcal{KB}$ (in any DL containing $\mathcal{ALC}$, see the discussion after Definition 4) and a constant $a \in \mathsf{IN}$, we will prove the following equivalence: $\mathcal{KB} \models q(a) \Leftrightarrow \mathcal{KB} \models a\colon C_\varphi$.

($\Rightarrow$) Take any model $\mathcal{I}$ of $\mathcal{KB}$. By assumption, $\mathcal{I} \models q(a)$. We need to show that $\mathcal{I} \models a\colon C_\varphi$ (independently of how the fresh concept names X_i occurring in C_φ are interpreted in $\mathcal{I}$). Let F be the frame underlying $\mathcal{I}$ and denote $e := a^{\mathcal{I}}$. By definition, from $q(x) \leftrightsquigarrow \varphi$ it follows that, for these F and e, we have: $F \models q(e) \Leftrightarrow F, e \Vdash \varphi$. But we also know that $F \models q(e)$, because $\mathcal{I} \models q(a)$ and $q(x)$ is relational. Hence we conclude that $F, e \Vdash \varphi$, i.e., $e \in \varphi^\nu$ for any valuation ν.

Now take the valuation ν that is "read-off" from our interpretation $\mathcal{I}$ by putting $p_i^\nu := X_i^{\mathcal{I}}$, for all propositional letters p_i occurring in the formula φ. It is easily seen that $\varphi^\nu = C_\varphi^{\mathcal{I}}$, since C_φ is just a notational variant of φ, whereas ν and $\mathcal{I}$ are essentially the same. As shown above, $e \in \varphi^\nu$ for this valuation ν and so $a^{\mathcal{I}} = e \in \varphi^\nu = C_\varphi^{\mathcal{I}}$. Thus we have proven that $\mathcal{I} \models a\colon C_\varphi$.

($\Leftarrow$) Take any model $\mathcal{I}$ of $\mathcal{KB}$. By assumption, $\mathcal{I} \models a\colon C_\varphi$. Let F be a frame underlying $\mathcal{I}$ and $e := a^{\mathcal{I}}$. We need to show that $\mathcal{I} \models q(a)$; since $q(x)$ is relational, it suffices to show that $F \models q(e)$. Due to the assumption that $q(x) \leftrightsquigarrow \varphi$, it remains to show that $F, e \Vdash \varphi$.

To this end, take any valuation ν and show that $e \in \varphi^\nu$. Let $\mathcal{J}$ be an interpretation differing from $\mathcal{I}$ only in how it interprets the fresh concept names occurring in C_φ, namely it is "read-off" from the valuation ν by putting $X_i^{\mathcal{J}} := p_i^\nu$. Observe that $C_\varphi^{\mathcal{J}} = \varphi^\nu$. Since $\mathcal{J}$ and $\mathcal{I}$ agree on concept, role and individual names occurring in $\mathcal{KB}$, we conclude that $\mathcal{J} \models \mathcal{KB}$. Now we use our assumption $\mathcal{KB} \models a\colon C_\varphi$ to infer that $\mathcal{J} \models a\colon C_\varphi$, from which it follows that $e = a^{\mathcal{I}} = a^{\mathcal{J}} \in C_\varphi^{\mathcal{J}} = \varphi^\nu$.

This completes the proof of Theorem 7. $\dashv$

Lemma 8. *Suppose that a unary relational query $q(x)$ is answered by a concept C_φ, i.e., $q(x) \approx C_\varphi$, for some modal formula φ. Then for any frame F and its point e, the condition $F \models q(e)$ implies $F, e \Vdash \varphi$.*

Proof. Assume that $F \models q(e)$. The query $q(x)$ has the form $\exists \vec{y} \bigwedge_{i=1}^{m} t_i(x, \vec{y})$, where t_i are role atoms. Then there exist $\vec{o} \in \Delta$ such that $F \models t_i(e, \vec{o})$ for all $i \leqslant m$. Now, take the *canonical* ABox $\mathcal{A}_q$ for $q(x)$, i.e., introduce new constants a_x and $\vec{a_y}$ and put $\mathcal{A}_q := \{t_i(a_x, \vec{a_y}) \mid 1 \leqslant i \leqslant m\}$, and consider a knowledge base $\mathcal{KB}_q := \langle \varnothing, \mathcal{A}_q \rangle$. Since $\mathcal{KB}_q \models \bigwedge_{i=1}^{m} t_i(a_x, \vec{a_y})$, we have $\mathcal{KB}_q \models \exists \vec{y} \bigwedge_{i=1}^{m} t_i(a_x, \vec{y})$, and hence $\mathcal{KB}_q \models q(a_x)$. Applying the condition of our Lemma $q(x) \approx C_\varphi$, we obtain that $\mathcal{KB}_q \models a_x : C_\varphi$.

To prove that $F, e \Vdash \varphi$, take an arbitrary valuation ν on the frame F and show that $e \in \varphi^\nu$. Let $\mathcal{I}$ be an interpretation based on F such that it is "read-off" from ν by putting $X_i^{\mathcal{I}} := p_i^\nu$, for all fresh concept names X_i occurring in C_φ, and extended to the new constants by putting $a_x^{\mathcal{I}} := e$ and $\vec{a_y}^{\mathcal{I}} := \vec{o}$. Since $\mathcal{I}$ is based on F and $F \models t_i(o, \vec{e})$, we have $\mathcal{I} \models t_i(a_x, \vec{a_y})$, for all $i \leqslant m$, and hence $\mathcal{I} \models \mathcal{KB}_q$. As shown above, $\mathcal{KB}_q \models a_x : C_\varphi$. Therefore, we conclude that $\mathcal{I} \models a_x : C_\varphi$ and finally $e = a_x^{\mathcal{I}} \in C_\varphi^{\mathcal{I}} = \varphi^\nu$. ⊣

Theorem 13. *Any query from the family $\mathcal{Z}$ is answered by a concept that can be built in linear time. Relational queries from $\mathcal{Z}$ are answered by $\mathcal{ALC}$-concepts.*

Proof. First assume that $q(x)$ is a relational query from $\mathcal{Z}$. Then:

1) Build the role expression $\mathcal{R}(q)$ (see Sect. 5). As already pointed out, the query $q(x)$ is equivalent to the $\mathcal{ALC}(\sqcap, \circ)$-concept $\exists \mathcal{R}(q).\top$.

2) Starting from the concept $\exists \mathcal{R}(q).\top$, repeatedly apply the following rewriting rules (each application of the second rule introduces a new concept name X), until all role operations are eliminated and we arrive to an $\mathcal{ALC}$-concept that answers the query $q(x)$:

$$\begin{array}{rcl} D \sqcup \exists(R \circ S).C & \equiv & D \sqcup \exists R.\exists S.C \\ D \sqcup \exists(R \sqcap S).C & \approx & D \sqcup \exists R.\neg X \sqcup \exists S.(X \sqcap C) \end{array}$$

The second rule follows from the Elimination Lemma and the following observation: *if $C \approx D$ then $(C \sqcup E) \approx (D \sqcup E)$*. Since, in the definition of the family $\mathcal{Z}$, a serial connection $q_1 \circ q_2$ is allowed only if q_2 is atomic, each time the first rule is applied, S will be a role name. Hence we will never obtain an expression of the form $\exists(R \sqcap S).C$ inside another quantifier, and so will be able to eliminate all occurrences of role intersection using the second rule. Therefore, the algorithm always terminates and produces an $\mathcal{ALC}$-concept C such that $C \approx \exists \mathcal{R}(q).\top$. It is easily seen that the procedure works in linear time in size of $q(x)$.

Now suppose that $q(x)$ belongs to $\mathcal{Z}$ and contains concept atoms. Let $y : C$ be one of them, where y is a non-distinguished variable (as already mentioned above, we can always consider queries $q(x)$ without atoms of the form $x : A$). Then we introduce a new non-distinguished variable y' and replace the atom $y : C$ by $y(id(C))y'$ and each atom of the form yRz by $y'Rz$ (recall that $id(C)$ is a notation that goes back to the propositional dynamic logic, and it denotes the role with the semantics: $id(C)^{\mathcal{I}} := \{\langle e, e \rangle \mid e \in C^{\mathcal{I}}\}$). This can be done for all concept atoms in $q(x)$ and results in an equivalent relational query $q'(x)$ (in an extended language) that belongs to $\mathcal{Z}$. Now we apply the above algorithm and finally replace in the resulting concept each occurrence of $\exists id(C).D$ by the equivalent expression $(C \sqcap D)$. ⊣

Theorem 14. *Any query from the family $\mathcal{Y}$ is answered by a concept that can be built in linear time. Relational queries from $\mathcal{Y}$ are answered by $\mathcal{ALCI}$-concepts. The family $\mathcal{Y}$ includes both families $\mathcal{K}$ and $\mathcal{Z}$, i.e., $\mathcal{K} \cup \mathcal{Z} \subset \mathcal{Y}$.*

Proof. Every query $q(x)$ in $\mathcal{K} \cup \mathcal{Z}$ is connected. Furthermore, if we remove from its graph the node x together with the edges incident to x, we will obtain an acyclic graph. This shows that $q(x) \in \mathcal{Y}$, hence the inclusion $\mathcal{K} \cup \mathcal{Z} \subset \mathcal{Y}$.

Now suppose we are given a query $q(x) \in \mathcal{Y}$. We will give a sketch of the proof that $q(x)$ can be represented as a query from "non-oriented" version of the family $\mathcal{K}$ (i.e., with edges oriented arbitrarily). First remove from its graph the node x together with the edges incident to x. This yields an acyclic non-oriented graph, i.e., a forest (a finite set of trees). Since the original graph (before removing x) was connected, it had edges linking x with the trees in this forest; take one edge per tree. Then the node x together with this forest and edges linking x with trees in this forest form a non-oriented tree $\mathsf{Tree}(x, \vec{y})$. Other edges can be considered as chains linking x with some non-distinguished variables. Thus, we have obtained a representation of the query $q(x)$ in the "non-oriented" version of ($\mathcal{K}$). Applying the algorithm described in Sect. 4 yields an $\mathcal{ALCI}$-concept that answers our query. $\dashv$