# Revisiting the Hardness of Query Answering in Expressive Description Logics[⋆]

Magdalena Ortiz and Mantas Šimkus

Institute of Information Systems,
Vienna University of Technology, Austria
ortiz@kr.tuwien.ac.at, simkus@dbai.tuwien.ac.at

**Abstract.** Answering conjunctive queries over *Description Logic* (DL) knowledge bases is known to be 2ExpTime-hard for the DLs $\mathcal{ALCI}$, $\mathcal{SH}$, and their extensions. In this technical note, we revisit these results to identify other equally hard settings. In particular, we show that a simple adaptation of the proof for $\mathcal{SH}$ proves that query answering is 2ExpTime-hard already for $\mathcal{ALC}$ if we consider more expressive query languages such as positive existential queries and (restricted classes of) conjunctive regular path queries.

## 1 Introduction

Ontology-based data access, and the related setting of query answering over *Description Logic* (DL) *knowledge bases* (KBs), has received considerable attention in the DL community. Most work has been devoted to the so-called *lightweight DLs* of the DL-Lite and $\mathcal{EL}$ families, but *expressive DLs* like $\mathcal{ALC}$ and its extentions have also been considered, cf. [13] and its references. The first query answering algorithms for the latter kind of DLs had the common feature of requiring double exponential time [8,4,3], and the question of whether this was worst-case optimal remained open for a while. For all extensions of $\mathcal{ALC}$ that support inverse roles, this gap was closed by Lutz, who proved 2EXPTIME-hardness of answering conjunctive queries (CQs) in $\mathcal{ALCI}$ [9]. An orthogonal 2EXPTIME-hardness result was later shown for CQs over $\mathcal{SH}$ knowledge bases [5], closing the gap for all DLs that support transitive roles and role hierarchies. Recently, 2EXPTIME-hardness was also proved for *DL-Lite*$^{\mathcal{H}}_{bool}$, a DL that does include full $\mathcal{ALC}$, but does support inverse roles and role hierarchies [2]. In contrast, answering unions of CQs (UCQs) is feasible in single exponential time for $\mathcal{ALCH}$ and $\mathcal{ALCHQ}$, and even for $\mathcal{SH}$ if suitable restrictions on the occurrences of transitive roles in the queries are imposed [12,9,6]. This shows that, for plain CQs and UCQs, the culprits for 2EXPTIME-hardness are indeed inverse roles, and transitive roles in combination with role hierarchies.

⋆ This is a pre-print of an article published in IFM 2014, LNCS 8739, pp. 3–17. The final authenticated version is available online at https://doi.org/10.1007/978-3-319-10181-1_1. This research has been partially supported by the Austrian Science Fund (FWF) projects T515 and P25518, and by the Vienna Science and Technology Fund (WWTF) project ICT12-015.

The aforementioned lower bounds are for plain CQs. However, recent works have gone beyond CQs and unions thereof, by investigating more expressive query languages like *regular path queries* and their extensions [4,3,11,1], or restricted classes of Datalog queries [10]. To our knowledge, it had not been investigated whether for these more expressive query languages, 2EXPTIME-hardness holds already in the absence of inverse roles, transitive roles, and role hierarchies. To tackle the issue, in this this technical note we revisit the proof in [5] to show that 2EXPTIME-hardness holds already for $\mathcal{ALC}$ in the following extensions of UCQs:

- positive existential queries (PQs),
- conjunctive regular path queries (CRPQs) without the Kleene star, and
- conjunctive *2-way* regular path queries (C2RPQs) without the Kleene star and with only 2 variables.

## 2 Query Languages

We assume familiarity with DLs, and in particular with $\mathcal{ALC}$ KBs. Their semantics is given by interpretations $\mathcal{I} = \langle \Delta^{\mathcal{I}}, \cdot^{\mathcal{I}} \rangle$. We call $\mathcal{I}$ *tree-shaped* if the graph with nodes $\Delta^{\mathcal{I}}$ and edges $(d, e)$ for all $(d, e) \in r^{\mathcal{I}}$ for a role name $r$, is a tree in the usual sense.

We focus here on Boolean queries of the form $\exists \boldsymbol{x}.\varphi(\boldsymbol{x})$, where $\boldsymbol{x}$ is a tuple of variables and $\varphi(\boldsymbol{x})$ is a formula whose syntax depends on the considered query language. In *positive queries (PQs)*, $\varphi(\boldsymbol{x})$ is built using $\wedge$ and $\vee$ from atoms of the forms $A(x)$ and $r(x, y)$, where $A$ is a concept name, $r$ a role name, and $x, y$ are variables from $\boldsymbol{x}$. A *conjunctive query* (CQ) is a positive query built using only $\wedge$, and a *union of CQs* is a positive query that is in DNF, that is, it is a disjunction of conjunctions. *Conjunctive regular path queries (CRPQs)* are defined analogously to CQs, but atoms may additionally take the form $\mathcal{E}(x, y)$, where $\mathcal{E}$ is a regular expression over the alphabet of role names. *Conjunctive 2-way regular path queries (C2RPQs)* are similar, but regular expressions are over the alphabet of role names $r$ and their inverses $r^{-}$. In this technical note we also consider a restricted class of C(2)RPQs that we call $(\circ, \cup)$-queries, which only allow for concatenation $\circ$ and union $\cup$ in complex roles, but disallows the Kleene star $*$. We note that $(\circ, \cup)$-queries are closely related to PQs. Indeed, every $(\circ, \cup)$-query can be rewritten as a PQ by using sequences of binary atoms in the place of $\circ$, and disjunction in the place of $\cup$. However, this requires the use of additional variables and may result in a larger query.

The semantics of queries is defined in terms of *matches*, which are mappings from the variables in $\boldsymbol{x}$ to objects in $\Delta^{\mathcal{I}}$ that make the query true; the latter notion is defined in the natural way, see e.g.,[4]. Here we consider the *query non-entailment problem*, which consists on deciding whether there exists a model of a given KB $\mathcal{K}$ that admits no match for a given query $q$, in symbols $\mathcal{K} \not\models q$. It is well known that every satisfiable $\mathcal{ALC}$ KB $\mathcal{K}$ with only one ABox individual has a tree-shaped model, and that this extends to query non-entailment: $\mathcal{K} \not\models q$ iff there is a tree-shaped model of $\mathcal{K}$ that admits no match for $q$ (cf. [5,9]). All complexity bounds mentioned here are for *combined complexity*, i.e., the complexity measured in terms of the combined sizes of $\mathcal{K}$ and $q$.

## 3 2ExpTime-hardness of CQs in $\mathcal{SH}$ revisited

We recall the proof of 2ExpTime-hardness of answering CQs over $\mathcal{SH}$ KBs in [5]. It is done by a reduction from the word problem of an exponentially space bounded Alternating Turing Machine $\mathcal{M}$, that is, the problem of deciding for each input word $w$ to $\mathcal{M}$, whether there is an accepting computation of $\mathcal{M}$ on $w$ that uses at most $2^{|w|}$ tape cells. The reduction builds a KB $\mathcal{K}_w$ and a query $q_w$ such that $\mathcal{M}$ accepts $w$ iff $\mathcal{K}_w \not\models q_w$. Since a computation of an ATM is naturally represented as a tree of configurations, the KB $\mathcal{K}_w$ is such that its tree-shaped models resemble accepting computations of $\mathcal{M}$ on $w$. We next recall the construction of $\mathcal{K}_w$ and $q_w$ from [5]. The construction uses a very simple ABox of the form $A(a)$, for $a$ an individual and $A$ a concept name, hence we can restrict our attention to tree-shaped models. We relax slightly the construction of $\mathcal{K}_w$ to be an $\mathcal{ALC}$ KB, by omitting the (sole) use of a transitive role to connect a node and its child to a common successor, for some nodes of the tree-shaped models of $\mathcal{K}_w$. Hence our description of $\mathcal{K}_w$ uses a single role $r$.

*Configuration nodes.* Intuitively, the tree-shaped models $\mathcal{K}_w$ are trees of nodes representing configurations of $\mathcal{M}$. For a configuration $K_h$ in which $\mathcal{M}$ is in state $q$ and its head in position $i$ reading a symbol $a$, there is a $(q, a, i)$-*configuration node* $n$ that represents $K_h$. The node $n$ stores $K_h$, and as a technical trick, it additionally stores another configuration $K_p$ such that $\mathcal{M}$ may move from $K_p$ to $K_h$; that is, $n$ contains both the current configuration $K_h$ and a possible previous configuration $K_p$. To properly store the current and previous configurations $K_h$ and $K_p$, each $(q, a, p)$-configuration node $n$ is in turn the root of a binary tree of depth $|w|$ whose $2^{|w|}$ leaves correspond to the tape cell positions, and store their $|w|$-bit address using a set of concept names $\mathbf{B} = \{B_1, \dots B_{|w|}\}$. All arcs in this tree are $r$-arcs. Each leaf $\ell$ with address $i$, which is called an $i$-*cell* (or just a *cell* if $i$ is unimportant) has a child $\ell_h$ and a child $\ell_p$ that store the symbol on tape position $i$ in $K_h$ and in $K_p$, respectively If $i$ is the position of the head of $\mathcal{M}$ in $K_h$, then $\ell_h$ also stores the state $q$ of $\mathcal{M}$, otherwise it stores a special marker $nil$ which intuitively means 'the head is not on this position'. Similarly for $K_p$ and $\ell_p$. For these labels we use concept names for alphabet symbols, states of $\mathcal{M}$, and the special marker $nil$.

$\mathcal{K}_w$ contains axioms that ensure that the configurations $K_h$ and $K_p$ are described correctly: e.g., there is exactly one symbol on each tape position, the head is at exactly one position, and $\mathcal{M}$ is in exactly one state. There are also axioms in $\mathcal{K}_w$ to ensure that $K_h$ is the result of correctly applying from $K_p$ a transition of $\mathcal{M}$.

*Computation trees.* A tree-shaped model $\mathcal{I}_c$ of $\mathcal{K}_w$ is called a *computation tree* and it is a tree of configuration nodes connected via the role $r$. Its root $n_0$ has an $r$-successor that is a $(q_0, a_0, 0)$-configuration node describing (as current configuration) the initial configuration of $\mathcal{M}$. Each $(q, a, p)$-configuration node $n$ with $q$ an existential state has a 2-step $r$-successor $n'$ that is a $(q', a', p + M)$-configuration node, for some transition $(q, a, q', a', M)$ of $\mathcal{M}$ (here the transition is read as follows: $\mathcal{M}$ is in state $q$ and reading $a$, it writes $a$, moves to state $q'$, and the head moves in direction $M \in \{-1, 0, 1\}$). Similarly, each $(q, a, p)$-configuration node $n$ with $q$ a universal state, has a 2-step $r$-successor $n'$ that is a $(q', a', p + M)$-configuration node for each transition $(q, a, q', a', M)$ of $\mathcal{M}$. The axioms of $\mathcal{K}_w$ also ensure that every configuration node

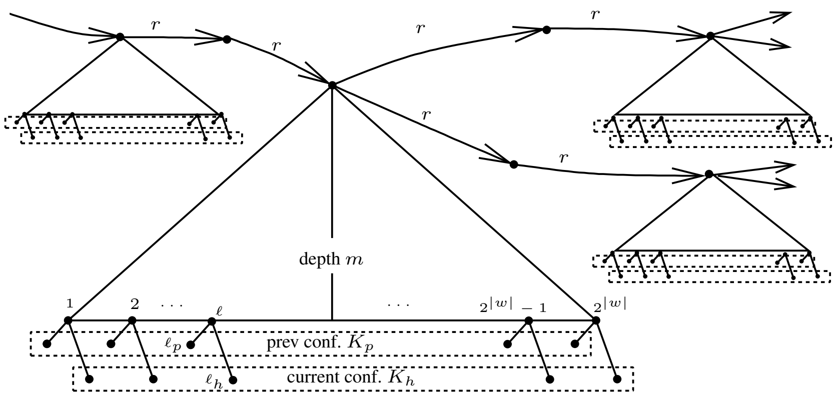

**Fig. 1.** A fragment of a computation tree

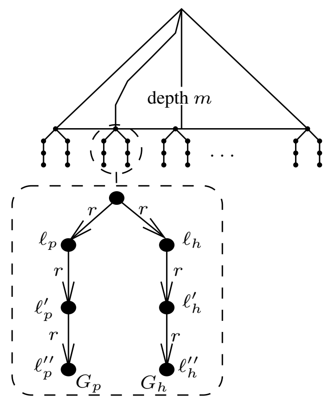

**Fig. 2.** A configuration tree with a magnified cell

with no successors is in an accepting state. Figure 1 illustrates a fragment of a computation tree with four configuration trees. A configuration tree with a magnified $i$-cell is illustrated in Figure 2.

*Proper computation trees.* To have a one to one correspondence between the tree-shaped models $\mathcal{I}_c$ of $\mathcal{K}_w$ and the accepting computations $c$ of $\mathcal{M}$ on input $w$, it suffices to ensure that for each pair $n, n'$ of successive configuration nodes, the current configuration of $n$ coincides with the previous configuration of $n'$. This is captured by the notion of *properness*, which states that for every counter $i$ value up to $2^{|w|}$, the node $\ell_h$ of the $i$-cell of $n$ and the node $\ell_p$ of the $i$-cell of $n'$ satisfy exactly the same concepts corresponding to head position, written symbol, and state of $\mathcal{M}$. Properness is not guaranteed by the axioms of $\mathcal{K}_w$ alone, and a tree-shaped model $\mathcal{I}_c$ of $\mathcal{K}_w$ (i.e., a computation tree) may be proper or not. Here where the query comes into play, by testing a computation tree is proper. More precisely, $q_w$ should have a match in a computation tree $\mathcal{I}_c$ iff $\mathcal{I}_c$ is *not* proper. In this way each computation tree with no match corresponds to an accepting computation of $\mathcal{M}$ on $w$, and we obtain that there is a tree-shaped model $\mathcal{I}_c$ of $\mathcal{K}_w$ where there is no match for $q_w$ iff there is an accepting computation of $\mathcal{M}$ on $w$. This (together with the tree-model property of $\mathcal{ALC}$) suffices to ensure that $\mathcal{K}_w \not\models q$ iff $\mathcal{M}$ accepts $w$ as desired.

By using suitable auxiliary nodes and labels, properness can be characterized in such a way that it can easily tested by the query. We use a set $\mathbf{Z}$ of concept names $Z_{a,q}$ for $a$ an alphabet symbol and $q$ either a state $\mathcal{M}$ or the special marker $nil$. We have said that every $i$-cell $\ell$ has two children $\ell_p$ and $\ell_h$, which respectively correspond to the $i$-th tape position in the previous and in the current configuration. By adding auxiliary $r$-children $\ell'_f$ to $\ell_f$ and $\ell''_f$ to $\ell'_f$ for $f \in \{h,p\}$, and labeling all $\ell_f$ and $\ell'_f$ with suitable values for the concepts $\mathbf{B} \cup \mathbf{Z}$ (exactly as done in [5]), one can obtain the following characterization. Two cells $\ell$ and $m$ are called *$A$-conspicuous*, where $A$ is a concept name, if (c1) $A$ is true at the $\ell_h$-node of $n$ and the $m_p$-node of $n'$, or (c2) $A$ is true at the $\ell'_h$-node of $n$ and the $m'_p$-node of $n'$. In Proposition 4 of [5] it is proved that a computation tree is not proper iff the following property $(*)$ holds:
$(*)$ There exits a cell $\ell$ in a configuration tree, and a cell $m$ in a successive configuration tree, such that $\ell$ and $m$ are $A$-conspicuous for all $A \in \mathbf{B} \cup \mathbf{Z}$.

To test $(*)$ with the query, we ensure that each $\ell''_f$ satisfies a special concept $G_f$, but we relax condition (c) in Definition 1 of [5], which requires that $\ell''_f$ is also a child of $\ell_f$, and that the arcs from $\ell_f$ to $\ell''_f$ and from $\ell'_f$ to $\ell''_f$ are $t$-arcs for a transitive role $t$. Instead, we only require $\ell''_f$ to be an $r$-child of $\ell'_f$. The queries testing for $(*)$ are similar to those in [5], but differ slightly according to the query language being considered.

**Positive Queries.** We can define the query $q_w$ that tests $(*)$ as a PQ as follows. As in [5], we obtain $q_w$ by taking the conjunction of a CQ $q(A,u,v)$ for each $A \in \mathbf{B} \cup \mathbf{Z}$, where the variables $u, v$ are shared by all $q(A,u,v)$, and the remaining variables are disjoint. That is, $q_w = \exists u, v. \bigwedge_{A \in \mathbf{B} \cup \mathbf{Z}} q(A,u,v)$, where $q(A,u,v)$ is as follows:

$$\begin{aligned} q(A,u,v) = \exists x_1^A, x_2^A, x_3^A, y_0^A, \ldots y_{|w|+1}^A, z_0^A, \ldots z_{|w|+3}^A\,.\\ & r(x_1^A, y_0^A) \wedge r(x_1^A, z_0^A) \wedge \\ & r(y_0^A, y_1^A) \wedge \cdots \wedge r(y_{|w|}^A, y_{|w|+1}^A) \wedge A(y_{|w|+1}^A) \wedge \\ & r(z_0^A, z_1^A) \wedge \cdots \wedge r(z_{|w|+2}^A, z_{|w|+3}^A) \wedge A(z_{|w|+3}^A) \wedge \\ & \big(r(y_{|w|+1}^A, u) \vee \big(r(y_{|w|+1}^A, x_2^A) \wedge r(x_2^A, u)\big)\big) \wedge G_h(u) \wedge \\ & \big(r(z_{|w|+3}^A, v) \vee \big(r(z_{|w|+3}^A, x_3^A) \wedge r(x_3^A, v)\big)\big) \wedge G_p(v) \end{aligned}$$

The basic query $q(A,u,v)$ is illustrated in the left-hand-side of Figure 3, and the full query $q_w$ on the right-hand-side. For readability, most labels have been omitted in the depiction of $q_w$. Each arc represents an atom of the form $r(x,y)$. The double dashed arcs between $y_{|w|+1}^A$ and $u$, and between $z_{|w|+3}^A$ and $v$, represent a disjunction. The only difference between this $q(A,u,v)$ and the one in [5] is that the disjunction of atoms $r(y_{|w|+1}^A, u) \vee \big(r(y_{|w|+1}^A, x_2^A) \wedge r(x_2^A, u)\big)$ replaces the atom $t(y_{|w|+1}^A, u)$, and similarly $r(z_{|w|+3}^A, v) \vee \big(r(z_{|w|+3}^A, x_3^A) \wedge r(x_3^A, v)\big)$ replaces $t(z_{|w|+3}^A, v)$.

Intuitively, $q(A,u,v)$ deals with $A$-conspicuousness, and $q_w$ tests $(*)$ by taking the conjunction for all $A \in \mathbf{B} \cup \mathbf{Z}$. Note that the shared variables $u$, $v$ are needed to ensure that all the components $q(A,u,v)$ speak about the same pair of cells $\ell$, $m$.

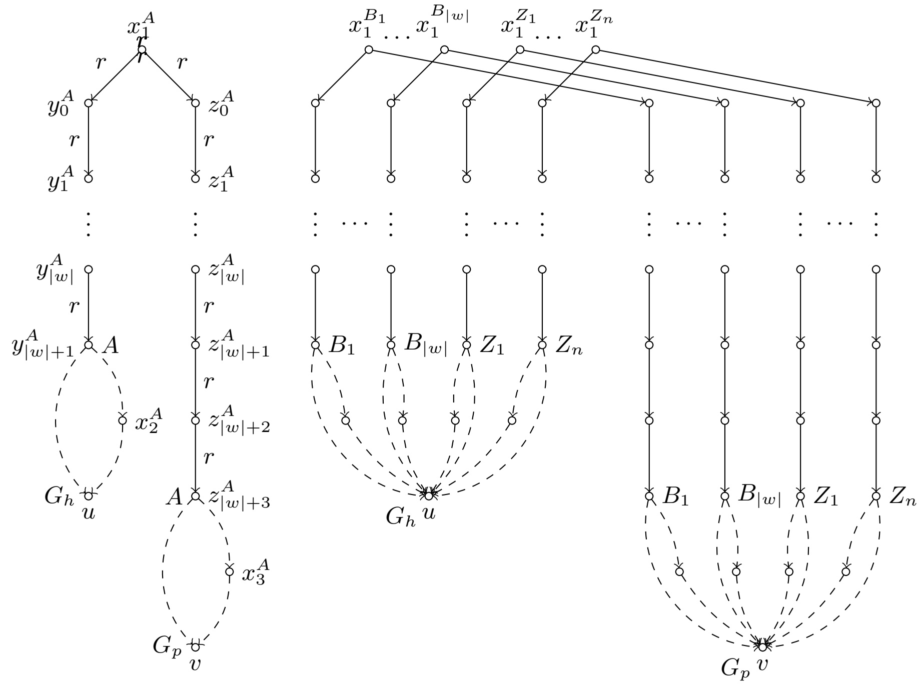

**Fig. 3.** The basic query $q(A, u, v)$ and the full query $q_w$.

To see that $q_w$ has a match iff $(*)$ holds, let $\ell$, $m$ be cells of two successive configurations that are $A$-conspicuous for all $A \in \mathbf{B} \cup \mathbf{Z}$. We can find a match for $q_w$ as follows. First we match $u$ on the $\ell''_h$ node of $\ell$, which satisfies $G_h$, and $v$ on the $m''_p$ node of $m$, which satisfies $G_p$. Consider an arbitrary $A \in \mathbf{B} \cup \mathbf{Z}$. We distinguish two cases:

– If (c1) applies, we match $y^A_{|w|+1}$ on the $\ell_h$ node of $\ell$, $x^A_2$ on the $\ell'_h$ node of $\ell$, $z^A_{|w|+3}$ on $m_p$ node of $m$, and $x^A_3$ on $m'_p$ node of $m$.
– Otherwise, if (c2) applies, we match $y^A_{|w|+1}$ on the $\ell'_h$ node of $\ell$ and $z^A_{|w|+3}$ on $m'_p$ node of $m$. In this case, the matches for $x^A_2$ and $x^A_3$ become irrelevant.

The matches of all other variables are then uniquely determined by conjunctions of atoms $r(z, z')$, in such a way that $x^A_1$ will be matched to the root of the configuration node of $\ell$ in the latter case, and to its parent in the former. As the $z^A_i$ chains are exactly two $r$-arcs longer than the $y^A_i$ chains, $m$ must be a leaf in a configuration node that follows that of $\ell$. We can argue analogously that every match for $q_w$, the variables $u$ and $v$ are respectively matched to the nodes $\ell''_h$ and $m''_p$ of a pair $\ell$, $m$ of cells of two successive configurations that are $A$-conspicuous for all $A \in \mathbf{B} \cup \mathbf{Z}$.

In this way we obtain that, for every computation tree $\mathcal{I}_c$, we have $\mathcal{I}_c$ is proper iff $\mathcal{I}_c \models q_w$. This ends the reduction to PQ entailment in $\mathcal{ALC}$.

Note that converting $q_w$ into a union of CQs (i.e., into DNF) results in an exponentially larger formula. This blow-up may be unavoidable. In fact, for CQs, it has been shown that query entailment is feasible in ExpTime for $\mathcal{ALCH}$ and $\mathcal{ALCHQ}$, and even for $\mathcal{SH}$ if suitable restrictions on the occurrences of transitive roles in the queries are imposed [12,9,6]. It follows from these results that there is no CQ whose size is polynomial in $w$ and $\mathcal{M}$ that can test for properness of computation trees.

**$(\circ,\cup)$-Queries.** Defining the query $q_w$ as a $(\circ,\cup)$-query is straightforward. We only need to replace in $q(A,u,v)$ the disjunction $r(y^A_{|w|+1},u) \vee \big(r(y^A_{|w|+1},x^A_2) \wedge r(x^A_2,u)\big)$ by the atom $\big(r \cup (r \circ r)\big)(y^A_{|w|+1},u)$, and the disjunction $r(z^A_{|w|+3},v) \vee \big(r(z^A_{|w|+3},x^A_3) \wedge r(x^A_3,v)\big)$ by the atom $\big(r \cup (r \circ r)\big)(z^A_{|w|+3},v)$ (and we can drop the variables $x^A_2$ and $x^A_2$). Then we can define $q_w$ as above, as the conjunction of the (modified) $q(A,u,v)$ for all $A \in \mathbf{B} \cup \mathbf{Z}$. A match for this modified $q_w$ in a computation tree is a match for the positive query above, and vice versa.

An alternative $(\circ,\cup)$-query for testing properness is obtained by replacing in each $q(A,u,v)$ the sequence of atoms $r(x^A_1,y^A_0) \wedge r(y^A_0,y^A_1) \wedge \cdots \wedge r(y^A_{|w|},y^A_{|w|+1})$ by a single atom $r \circ \cdots \circ r(x^A_1,y^A_{|w|+1})$ for a chain $r \circ \cdots \circ r$ of length $|w|+2$, and the sequence $r(x^A_1,z^A_0) \wedge r(z^A_0,z^A_1) \wedge \cdots \wedge r(z^A_{|w|},z^A_{|w|+3})$ by $r \circ \cdots \circ r(x^A_1,z^A_{|w|+3})$ for a chain of length $|w|+4$; note that we can get rid of all but one variable $y^A_i$, and all but one $z^A_i$. Using the *test* constructor $A?$ for a name $A$ sometimes allowed in CRPQs (with semantics $A?^{\mathcal{I}} = \{e,e \mid e \in A^{\mathcal{I}}\}$) we can even replace in $q(A,u,v)$ the whole sequence of atoms from $x^A_1$ to $u$ by a single atom, and the whole sequence from $x^A_1$ to $v$ by another atom, using only one variable $x^A$ additionally to $u$ and $v$. However, the number of variables in the resulting $q_w$ still depends linearly on $|w|$ and $\mathcal{M}$.

Using the inverse roles allowed in C2RPQs, we can even go one step further and write the whole query $q'(A,u,v)$ as one single atom with variables $u$ and $v$:

$$q'(A,u,v) = G_h? \circ (r^- \cup r^- \circ r^-) \circ A? \circ \underbrace{r^- \circ \cdots \circ r^-}_{|w|+2 \text{ times}} \circ \underbrace{r \circ \cdots \circ r}_{|w|+4 \text{ times}} \circ A? \circ \big(r \cup (r \circ r)\big)(u,v)$$

The conjunction of these queries also gives a query $q_w$ that correctly tests properness, but using only two variables. We note that in C2RPQs the tests $A?$ add no expressive power, as they can be simulated by adding a axiom $A \sqsubseteq \exists r_A$ to the KB for a fresh role $r_A$, and replacing $A?$ by $r_A \circ r_A^-$. Summing up, we obtain:

**Theorem 1.** *Query entailment in $\mathcal{ALC}$ is 2ExpTime-hard for:*

1. *positive queries,*
2. *any extension of CQs that allows for atoms of the form $\big(r \cup (r \circ r)\big)(z,z')$ for a role name $r$ and variables $z, z'$,*
3. *the class of $*$-free CRPQs, and*
4. *the class of $*$-free C2RPQs with only two variables.*

We note that for PQs and CRPQs (with no inverses), it is not clear whether the reduction can be done using a bounded number of variables. The same holds for the lower bounds for $\mathcal{ALCI}$ and $\mathcal{SH}$ [9,5]. In contrast, CQ entailment in $\mathcal{SHI}$ is 2ExpTime-hard already for queries with only two variables [7], similarly to C2RPQs in $\mathcal{ALC}$.

## 4 Conclusions

We have seen that query answering in $\mathcal{ALC}$ and its extensions becomes 2ExpTime-hard even for rather restricted settings. However, once this 2ExpTime-hard boundary has been crossed, one can significantly extend both the DL and the query language without an additional increase in complexity. Query entailment remains in 2ExpTime even for *positive* 2-way regular path queries, which extend all the query languages mentioned above, and for $\mathcal{ZIQ}$, $\mathcal{ZOQ}$ ad $\mathcal{ZOI}$, which respectively extend the well known $\mathcal{SHIQ}$, $\mathcal{SHOQ}$ and $\mathcal{SHOI}$ [4,3].

In this paper we have focused on identifying query languages for which query entailment in $\mathcal{ALC}$ is 2ExpTime-hard. It would also be interesting to study which are the minimal DL constructs needed to show 2ExpTime-hardness, similarly as done in [2], but trying to avoid the combined use of role hierarchies and inverse roles. In line with aforementioned paper, it is worth remarking that the presence of disjunction is crucial. Indeed, even C2RPQs in (disjunction-free) Horn-$\mathcal{SHOIQ}$ can be answered in single exponential time [11]. For a more detailed discussion of the topic, and references to other related results, the reader may refer to [13].

## References

1. Bienvenu, M., Calvanese, D., Ortiz, M., Šimkus, M.: Nested regular path queries in description logics. In: Proc. of KR 2014. AAAI Press (2014)
2. Bourhis, P., Morak, M., Pieris, A.: The impact of disjunction on query answering under guarded-based existential rules. In: Proc. of IJCAI 2013. pp. 796–802. IJCAI/AAAI (2013)
3. Calvanese, D., Eiter, T., Ortiz, M.: Regular path queries in expressive description logics with nominals. In: Proc. of IJCAI 2009. pp. 714–720 (2009)
4. Calvanese, D., Eiter, T., Ortiz, M.: Answering regular path queries in expressive description logics via alternating tree-automata. Information and Computation 237(0), 12 – 55 (2014)
5. Eiter, T., Lutz, C., Ortiz, M., Šimkus, M.: Query answering in description logics with transitive roles. In: Proc. of IJCAI 2009. pp. 759–764 (2009)
6. Eiter, T., Ortiz, M., Šimkus, M.: Conjunctive query answering in the Description Logic $\mathcal{SH}$ using knots. Journal of Computer and System Sciences 78(1), 47–85 (2012)
7. Glimm, B., Kazakov, Y.: Role conjunctions in expressive description logics. In: Proc. of LPAR 2008. pp. 391–405 (2008)
8. Glimm, B., Lutz, C., Horrocks, I., Sattler, U.: Conjunctive query answering for the description logic SHIQ. J. Artif. Intell. Res. (JAIR) 31, 157–204 (2008)
9. Lutz, C.: The complexity of conjunctive query answering in expressive description logics. In: Proc. of IJCAR 2008. pp. 179–193. No. 5195 in LNAI (2008)
10. Motik, B., Sattler, U., Studer, R.: Query answering for OWL-DL with rules. J. Web Sem. 3(1), 41–60 (2005)
11. Ortiz, M., Rudolph, S., Šimkus, M.: Query answering in the Horn fragments of the description logics $\mathcal{SHOIQ}$ and $\mathcal{SROIQ}$. In: Proc. of IJCAI 2011. pp. 1039–1044 (2011)
12. Ortiz, M., Šimkus, M., Eiter, T.: Worst-case optimal conjunctive query answering for an expressive description logic without inverses. In: Proc. of AAAI 2008. pp. 504–510 (2008)
13. Ortiz, M., Šimkus, M.: Reasoning and query answering in Description Logics. In: Proc. of Reasoning Web Summer School. LNCS, vol. 7487, pp. 1–53 (2012)