The Complexity of Conjunctive Query Answering in Expressive Description Logics

Carsten Lutz

Institut für Theoretische Informatik
TU Dresden, Germany
lutz@tcs.inf.tu-dresden.de

Abstract. Conjunctive query answering plays a prominent role in applications of description logics (DLs) that involve instance data, but its exact complexity was a long-standing open problem. We determine the complexity of conjunctive query answering in expressive DLs between $\mathcal{ALC}$ and $\mathcal{SHIQ}$, and thus settle the problem. In a nutshell, we show that conjunctive query answering is 2ExpTime-complete in the presence of inverse roles, and only ExpTime-complete without them.

1 Introduction

Description logics (DLs) originated in the late 1970ies as knowledge representation (KR) formalisms, and nowadays play an important role as ontology languages [1]. Traditionally, DLs are used for the representation of and reasoning about the conceptual modeling of an application domain. Most KR applications of DLs are of this kind, and also the majority of ontologies focusses on conceptual modeling. In contrast, more recent applications of DLs additionally involve (potentially large amounts of) instance data. In particular, instance data plays an important role when using DL ontologies for data-integration and in ontology-mediated data access.

In DLs, a TBox is used to represent conceptual information, and instance data is stored in the ABox. Consequently, traditional DL research has mainly concentrated on TBox reasoning, where the main reasoning services are subsumption and satisfiability. In the presence of ABoxes, additional reasoning services are required to query the instance data. The most basic such service is *instance retrieval*, i.e., to return all certain answers to a query that has the form of a DL concept. Instance retrieval can be viewed as a well-behaved generalization of subsumption and satisfiability: it is usually possible to adapt algorithms in a straightforward way, and the computational complexity coincides in almost all cases (but see [20] for an exception). A more powerful way to query ABoxes is *conjunctive query answering*, as first studied in the context of DLs by Calvanese et al. in 1998 [2]. Roughy speaking, conjunctive query answering generalizes instance retrieval by admitting also queries whose relational structure is not tree-shaped. This generalization is both natural and useful because the relational structure of ABoxes is usually not tree-shaped either.

Conjunctive queries have been studied extensively in the DL literature, see for example [2–4, 6, 7, 9–11, 17, 21]. In contrast to the case of instance retrieval, developing algorithms for conjunctive query answering requires novel techniques. In particular, all hitherto known algorithms for conjunctive query answering in the basic propositionally closed DL $\mathcal{ALC}$ and its extensions require double exponential time. In contrast, subsumption, satisfiability, and instance checking (the decision problem corresponding to instance retrieval) are ExpTime-complete even in the expressive DL $\mathcal{SHIQ}$, which is a popular extension of $\mathcal{ALC}$ [8]. It follows that, in DLs between $\mathcal{ALC}$ and $\mathcal{SHIQ}$, the complexity of conjunctive query entailment (the decision problem corresponding to conjunctive query answering) is between ExpTime and 2ExpTime. However, the exact complexity of this important problem has been open for a long time. In particular, it was unclear whether the generalization of instance retrieval to conjunctive query answering comes with an increase in computational complexity.

In this paper, we settle the problem and determine the exact complexity of conjunctive query entailment in DLs between $\mathcal{ALC}$ and $\mathcal{SHIQ}$. More precisely, we show that

(1) Conjunctive query entailment in $\mathcal{ALCI}$, the extension of $\mathcal{ALC}$ with inverse roles, is 2ExpTime-hard. With the upper bound from [7], conjunctive query answering is thus 2ExpTime-complete for any DL between $\mathcal{ALCI}$ and $\mathcal{SHIQ}$.

(2) Conjunctive query entailment in $\mathcal{SHQ}$ is in ExpTime. With the ExpTime lower bound for instance checking in $\mathcal{ALC}$, conjunctive query entailment is thus ExpTime-complete for any DL between $\mathcal{ALC}$ and $\mathcal{SHQ}$.

In short, conjunctive query entailment is one exponential harder than instance checking in the presence of inverse roles, but not without them. Result (2) was proved independently and in parallel for the DL $\mathcal{ALCH}$ in [18], and generalized to also include transitive roles (under some restrictions) in [19].

We also consider the special case of conjunctive query entailment where the query is *rooted*, i.e., it is connected and contains at least one answer variable. We prove matching lower and upper bounds to show that

(3) Rooted conjunctive query entailment is NExpTime-complete for any DL between $\mathcal{ALCI}$ and $\mathcal{SHIQ}$.

Thus, rootedness reduces the complexity of query entailment in the presence of inverse roles (but not without them). In the upper bounds of (2) and (3), we disallow transitive and other so-called non-simple roles in the query. We also show that rooted conjunctive query entailment in $\mathcal{ALCI}$ with transitive roles becomes 2ExpTime-complete if transitive roles are admitted in the query.

This paper is organized as follows. In Section 2, we briefly review some preliminaries. We then establish the lower bounds, starting with the NExpTime one of (3) in Section 3. The 2ExpTime lower bound of (1) builds on that, but we have to confine ourselves to a brief sketch in Section 4. This section also establishes 2ExpTime-hardness of $\mathcal{ALCI}$ with transitive roles in the query. In Section 5, we prove the ExpTime upper bound of (2). In Section 6, we give some further discussion of transitive roles in the query. This paper is based on the workshop papers [15] and [16].

2 Preliminaries

We assume standard notation for the syntax and semantics of $\mathcal{SHIQ}$ knowledge bases [8]. In particular, $\mathsf{N_C}$, $\mathsf{N_R}$, and $\mathsf{N_I}$ are countably infinite and disjoint sets of *concept names*, *role names*, and *individual names*. A *TBox* is a set of concept inclusions $C \sqsubseteq D$, role inclusions $r \sqsubseteq s$, and transitivity statements $\mathsf{Trans}(r)$, and a *knowledge base (KB)* is a pair $(\mathcal{T}, \mathcal{A})$ consisting of a TBox $\mathcal{T}$ and an ABox $\mathcal{A}$. We write $\mathcal{K} \models s \sqsubseteq r$ if the role inclusion $s \sqsubseteq r$ is true in all models of $\mathcal{K}$, and similarly for $\mathcal{K} \models \mathsf{Trans}(r)$. It is easy to see and well-known that "$\mathcal{K} \models s \sqsubseteq r$" and "$\mathcal{K} \models \mathsf{Trans}(r)$" are decidable in polytime [8]. As usual, a role is called *simple* if there is no role s such that $\mathcal{K} \models s \sqsubseteq r$, and $\mathcal{K} \models \mathsf{Trans}(s)$. We write $\mathsf{Ind}(\mathcal{A})$ to denote the set of all individual names in an ABox $\mathcal{A}$. Throughout the paper, the number n inside number restrictions $(\geq n\, r\, C)$ and $(\leq n\, r\, C)$ is assumed to be coded in binary. $\mathcal{ALC}$ is the fragment of $\mathcal{SHIQ}$ that disallows role hierarchies, transitive roles, inverse roles, and number restrictions.

Let $\mathsf{N_V}$ be a countably infinite set of *variables*. An *atom* is an expression $C(v)$ or $r(v, v')$, where C is a $\mathcal{SHIQ}$ concept, r is a simple (but possibly inverse) role, and $v, v' \in \mathsf{N_V}$. A *conjunctive query* q is a finite set of atoms. We use $\mathsf{Var}(q)$ to denote the set of variables occurring in the query q. For each query q, the set $\mathsf{Var}(q)$ is partitioned into *answer variables* and (existentially) *quantified variables*. Let $\mathcal{A}$ be an ABox, $\mathcal{I}$ a model of $\mathcal{A}$, q a conjunctive query, and $\pi : \mathsf{Var}(q) \to \Delta^{\mathcal{I}}$ a total function. such that for every answer variable $v \in \mathsf{Var}(q)$, there is an $a \in \mathsf{N_I}$ such that $\pi(v) = a^{\mathcal{I}}$. We write $\mathcal{I} \models^{\pi} C(v)$ if $\pi(v) \in C^{\mathcal{I}}$ and $\mathcal{I} \models^{\pi} r(v, v')$ if $(\pi(v), \pi(v')) \in r^{\mathcal{I}}$. If $\mathcal{I} \models^{\pi} at$ for all $at \in q$, we write $\mathcal{I} \models^{\pi} q$ and call π a *match* for $\mathcal{I}$ and q. We say that $\mathcal{I}$ *satisfies* q and write $\mathcal{I} \models q$ if there is a match π for $\mathcal{I}$ and q. If $\mathcal{I} \models q$ for all models $\mathcal{I}$ of a KB $\mathcal{K}$, we write $\mathcal{K} \models q$ and say that $\mathcal{K}$ *entails* q.

The *query entailment problem* is, given a knowledge base $\mathcal{K}$ and a query q, to decide whether $\mathcal{K} \models q$. This is the decision problem corresponding to query answering, see e.g. [7] for details. Observe that we do not admit the use of individual constants in conjunctive queries. This assumption is only for simplicity, as such constants can easily be simulated by introducing additional concept names [7]. We speak of *rooted query entailment* when the query q is *rooted*, i.e., when q is connected and contains at least one answer variable.

3 Rooted Query Entailment in $\mathcal{ALCI}$ and $\mathcal{SHIQ}$

$\mathcal{ALCI}$ is the extension of $\mathcal{ALC}$ with inverse roles, and thus a fragment of $\mathcal{SHIQ}$. The aim of this section is to show that rooted query entailment in $\mathcal{ALCI}$ is NExpTime-complete in all DLs between $\mathcal{ALCI}$ and $\mathcal{SHIQ}$. To comply with space limitations, we concentrate on the lower bound. It applies even to the case where TBoxes are empty.

Let $\mathcal{ALC}^{\mathsf{rs}}$ be the variation of $\mathcal{ALC}$ in which all roles are interpreted as reflexive and symmetric relations. Our proof of the NExpTime lower bound proceeds by first polynomially reducing rooted query entailment in $\mathcal{ALC}^{\mathsf{rs}}$ w.r.t.

the empty TBox to rooted query entailment in $\mathcal{ALCI}$ w.r.t. the empty TBox. Then, we prove co-NExpTime-hardness of rooted query entailment in $\mathcal{ALC}^{\mathsf{rs}}$. Regarding the first step, the idea is to replace each symmetric role s with the composition of r^- and r, with r a role of $\mathcal{ALCI}$. Although r is not interpreted in a symmetric relation, the composition of r^- and r is clearly symmetric. To achieve reflexivity, we ensure that $\exists r^-.\top$ is satisfied by all relevant individuals and for all relevant roles r. Then, every domain element can reach itself by first travelling r^- and then r, which corresponds to a reflexive s-loop. Since we are working without TBoxes and thus cannot use statements such as $\top \sqsubseteq \exists r^-.\top$, a careful manipulation of the ABox and query is needed. Details are given in [15].

Before we prove co-NExpTime-hardness of rooted query entailment in $\mathcal{ALC}^{\mathsf{rs}}$ with empty TBoxes, we discuss a preliminary. An interpretation $\mathcal{I}$ of $\mathcal{ALC}^{\mathsf{rs}}$ is *tree-shaped* if there is a bijection f from $\Delta^{\mathcal{I}}$ into the set of nodes of a finite undirected tree (V, E) such that $(d, e) \in s^{\mathcal{I}}$, for some role name s, implies that $d = e$ or $\{f(d), f(e)\} \in E$. The proof of the following result is standard, using unravelling.

Lemma 1. *If $\mathcal{A}$ is an $\mathcal{ALC}^{\mathsf{rs}}$-ABox and q a conjunctive query, then $\mathcal{A} \not\models q$ implies that there is a tree-shaped model $\mathcal{I}$ of $\mathcal{A}$ such that $\mathcal{I} \not\models q$.*

Thus, we can concentrate on tree-shaped interpretations throughout the proof. We now give a reduction from a NExpTime-complete variant of the tiling problem to rooted query *non*-entailment in $\mathcal{ALC}^{\mathsf{rs}}$.

Definition 1 (Domino System). *A* domino system $\mathfrak{D}$ *is a triple* (T, H, V)*, where* $T = \{0, 1, \ldots, k-1\}$*,* $k \geq 0$*, is a finite set of* tile types *and* $H, V \subseteq T \times T$ *represent the* horizontal and vertical matching conditions*. Let $\mathfrak{D}$ be a domino system and* $c = c_0, \ldots, c_{n-1}$ *an* initial condition*, i.e. an n-tuple of tile types. A mapping* $\tau : \{0, \ldots, 2^{n+1}-1\} \times \{0, \ldots, 2^{n+1}-1\} \to T$ *is a* solution *for $\mathfrak{D}$ and c iff for all $x, y < 2^{n+1}$, the following holds (where $\oplus_i$ denotes addition modulo i): (i) if $\tau(x, y) = t$ and $\tau(x \oplus_{2^{n+1}} 1, y) = t'$, then $(t, t') \in H$; (ii) if $\tau(x, y) = t$ and $\tau(x, y \oplus_{2^{n+1}} 1) = t'$, then $(t, t') \in V$; (iii) $\tau(i, 0) = c_i$ for $i < n$.*

For NExpTime-hardness of this problem see, e.g., Corollary 4.15 in [13]. We show how to translate a given domino system $\mathfrak{D}$ and initial condition $c = c_0 \cdots c_{n-1}$ into an ABox $\mathcal{A}_{\mathfrak{D},c}$ and query $q_{\mathfrak{D},c}$ such that each tree-shaped model $\mathcal{I}$ of $\mathcal{A}_{\mathfrak{D},c}$ that satisfies $\mathcal{I} \not\models q_{\mathfrak{D},c}$ encodes a solution to $\mathfrak{D}$ and c, and conversely, each solution to $\mathfrak{D}$ and c gives rise to a (tree-shaped) model of $\mathcal{A}_{\mathfrak{D},c}$ with $\mathcal{I} \not\models q_{\mathfrak{D},c}$. The ABox $\mathcal{A}_{\mathfrak{D},c}$ contains only the assertion $C_{\mathfrak{D},c}(a)$, with $C_{\mathfrak{D},c}$ a conjunction $C^1_{\mathfrak{D},c} \sqcap \cdots \sqcap C^7_{\mathfrak{D},c}$ whose conjuncts we define in the following. For convenience, let $m = 2n+2$. The purpose of the first conjunct $C^1_{\mathfrak{D},1}$ is to enforce a binary tree of depth m whose leaves are labelled with the numbers $0, \ldots, 2^m-1$ of a binary counter implemented by the concept names $A_0, \ldots, A_{m-1}$. We use concept names $L_0, \ldots, L_m$ to distinguish the different levels of the tree. This is necessary because we work with reflexive and symmetric roles. In the following

$\forall s^i.C$ denotes the i-fold nesting $\forall s.\cdots\forall s.C$. In particular, $\forall s^0.C$ is C.

$$\begin{aligned} C^1_{\mathfrak{D},c} := L_0 \sqcap \bigsqcap_{i<m} \forall s^i.\big(L_i \rightarrow \big(\exists s.(L_{i+1} \sqcap A_i) \sqcap \exists s.(L_{i+1} \sqcap \neg A_i)\big)\big) \sqcap \\ \bigsqcap_{i<m} \forall s^i. \bigsqcap_{j<i} \big((L_i \sqcap A_j) \rightarrow \forall s.(L_{i+1} \rightarrow A_j) \sqcap \\ (L_i \sqcap \neg A_j) \rightarrow \forall s.(L_{i+1} \rightarrow \neg A_j)\big) \end{aligned}$$

From now on, leafs in this tree are called L_m-nodes. Each L_m-node corresponds to a position in the $2^{n+1} \times 2^{n+1}$-grid that we have to tile: the counter A_x realized by the concept names $A_0, \ldots, A_n$ binarily encodes the horizontal position, and the counter A_y realized by $A_{n+1}, \ldots, A_m$ encodes the vertical position. We now extend the tree with some additional nodes. Every L_m-node gets three successor nodes labelled with F, and each of these F-nodes has a successor node labelled G. To distinguish the three different G-nodes below each L_m-node, we additionally label them with the concept names G_1, G_2, G_3.

$$C^2_{\mathfrak{D},c} := \forall s^m.\big(L_m \rightarrow \big(\bigsqcap_{1\le i\le 3} \exists s.(F \sqcap \exists s.(G \sqcap G_i))\big)\big)$$

We want that each G_1-node represents the grid position identified by its ancestor L_m-node, the sibling G_2 node represents the horizontal neighbor position in the grid, and the sibling G_3-node represents the vertical neighbor.

$$\begin{aligned} C^3_{\mathfrak{D},c} := \forall s^m.\big(L_m \rightarrow \big(& \bigsqcap_{i\le n} \big((A_i \rightarrow \forall s^2.(G_1 \sqcup G_3 \rightarrow A_i)) \sqcap \\ & \quad (\neg A_i \rightarrow \forall s^2.(G_1 \sqcup G_3 \rightarrow \neg A_i))\big) \sqcap \\ & \bigsqcap_{n<i<m} \big((A_i \rightarrow \forall s^2.(G_1 \sqcup G_2 \rightarrow A_i)) \sqcap \\ & \quad (\neg A_i \rightarrow \forall s^2.(G_1 \sqcup G_2 \rightarrow \neg A_i))\big) \sqcap \\ & E_2 \sqcap E_3 \ \big)\big) \end{aligned}$$

where E_2 is an $\mathcal{ALC}$-concept ensuring that the A_x value at each G_2-node is obtained from the A_x-value of its G-node ancestor by incrementing modulo 2^{n+1}; similarly, E_3 expresses that the A_y value at each G_3-node is obtained from the A_y-value of its G-node ancestor by incrementing modulo 2^{n+1}. It is not hard to work out the details of these concepts, see e.g. [14] for more details. The *grid representation* that we have enforced is shown in Figure 1. To represent tiles, we introduce a concept name D_i for each $i \in T$. It is now easy to define concepts $C^4_{\mathfrak{D},c}$ and $C^5_{\mathfrak{D},c}$ which enforce that every G-node is labeled with exactly one tile type, and that the initial condition is satisfied—details are left to the reader. To enforce the matching conditions, we proceed in two steps. First we ensure that they are satisfied locally, i.e., among the three G-nodes below each L_m-node:

$$\begin{aligned} C^6_{\mathfrak{D},c} := \forall s^{m+2}.\big(L_m \rightarrow \big(& \bigsqcap_{i\in T} \big(\exists s^2.(G_1 \sqcap D_i) \rightarrow \forall s^2.(G_2 \rightarrow \bigsqcup_{(i,j)\in H} D_j)\big) \sqcap \\ & \bigsqcap_{i\in T} \big(\exists s^2.(G_1 \sqcap D_i) \rightarrow \forall s^2.(G_3 \rightarrow \bigsqcup_{(i,j)\in V} D_j)\big)\big)\big) \end{aligned}$$

Second, we enforce the following condition, which together with local satisfaction of the matching conditions ensures their global satisfaction:

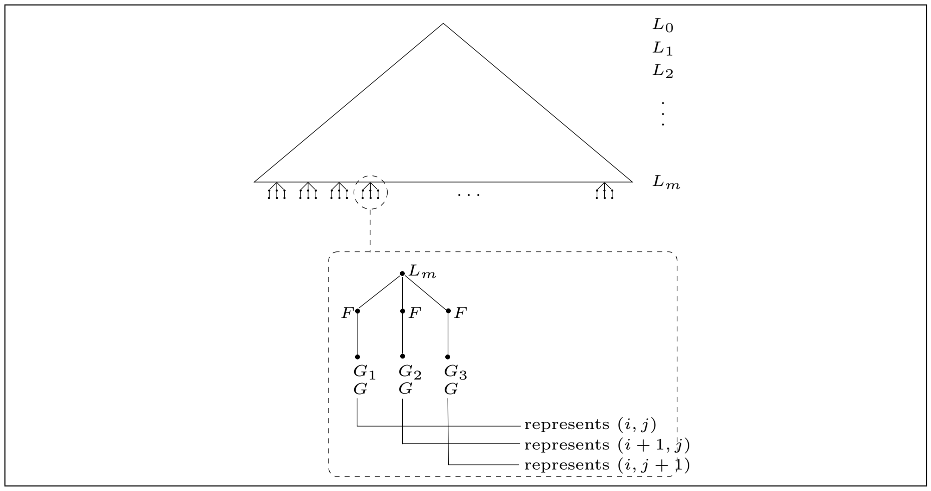

Fig. 1. The structure encoding the $2^{n+1} \times 2^{n+1}$-grid.

($*$) if the A_x and A_y-values of two G-nodes coincide, then their tile types coincide.

In ($*$), a G-node can by any of a G_1-, G_2-, or G_3-node. To enforce ($*$), we use the query. Before we give details, let us finish the definition of the concept $C_{\mathfrak{D},c}$. The last conjunct $C^7_{\mathfrak{D},c}$ enforces two technical conditions that will be explained later: if d is an F-node and e its G-node successor, then

T1 d satisfies A_i iff e satisfies $\neg A_i$, for all $i < m$;
T2 if d satisfies D_j, then e satisfies $D_0, \ldots, D_{j-1}, \neg D_j, D_{j+1}, \ldots, D_{k-1}$, for all $j < k$.

Details of $C^7_{\mathfrak{D},c}$ are left to the reader.

We now construct the query $q_{\mathfrak{D},c}$ that does *not* match the grid representation iff ($*$) is satisfied. In other words, $q_{\mathfrak{D},c}$ matches the grid representation iff there are two G-nodes that agree on the value of the counters A_x and A_y, but are labelled with different tile types.

The construction of $q_{\mathfrak{D},c}$ is in several steps, starting with the query $q^i_{\mathfrak{D},c}$ on the left-hand side of Figure 2, where $i \in \{0, \ldots, m-1\}$. In the queries $q^i_{\mathfrak{D},c}$, all the edges represent the role s and v_{ans} is the only answer variable. The edges are undirected because we are working with symmetric roles. Formally,

$$\begin{aligned}
q^i_{\mathfrak{D},c} := \{ & s(v_{i,0}, v_{i,1}), \ldots, s(v_{i,2m+2}, v_{i,2m+3}), \\
& s(v'_{i,0}, v'_{i,1}), \ldots, s(v'_{i,2m+2}, v'_{i,2m+3}), \\
& s(v_{i,0}, v'_{i,0}), s(v_{i,2m+3}, v'_{i,2m+3}), \\
& s(v, v_{i,0}), s(v, v'_{i,0}), \\
& s(v', v_{i,2m+3}), s(v', v'_{i,2m+3}), \\
& s(v_{\mathsf{ans}}, v_{i,m+1}), s(v_{\mathsf{ans}}, v_{i,m+2}), s(v_{\mathsf{ans}}, v'_{i,m+1}), s(v_{\mathsf{ans}}, v'_{i,m+2}), \\
& G(v), G(v'), A_i(v_{i,0}), \neg A_i(v'_{i,0}), \neg A_i(v_{i,2m+3}), A_i(v'_{i,2m+3}) \}
\end{aligned}$$

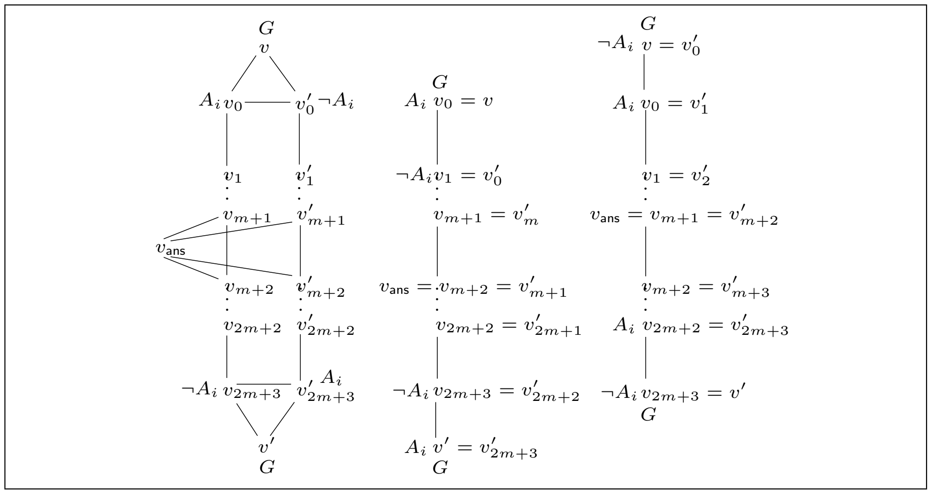

Fig. 2. The query $q^i_{\mathfrak{D},a}$ (left) and two of its collapsings (middle and right).

Observe that we dropped the index "i" to variables in Figure 2. Also observe that all the queries $q^i_{\mathfrak{D},c}$, $i < m$, share the variables v, v', and v_{ans}.

The purpose of the query $q^i_{\mathfrak{D},a}$ is to relate any two G-nodes that agree on the value of the concept name A_i. To explain how this works, we need a few preliminaries. First, a *cycle* in a query is a sequence of distinct nodes $v_0, \ldots, v_n$ such that $n \geq 2$, and $s(v_i, v_{i+1}) \in q$ or $s(v_{i+1}, v_i) \in q$ for all $i \leq n$, where $v_{n+1} := v_0$. A query q' is a *collapsing* of a query q if q' is obtained from q by identifying variables. Each match of $q^i_{\mathfrak{D},c}$ in our *tree-structured* grid representation gives rise to a collapsing of $q^i_{\mathfrak{D},c}$ that does not comprise any cycles. To explain how $q^i_{\mathfrak{D},c}$ works, it is helpful to analyze its cycle-free collapsings. We start with the two cycles v, v_0, v_0' and v', v_{2m+3}, v_{2m+3}'. For eliminating each of these, we have two options:

- to remove the upper cycle, we can identify v with v_0 or v_0';
- to remove the lower cycle, we can identify v' with v_{2m+3} or v_{2m+3}'.

Observe that if we identify v_0 and v_0' (or v_{2m+3} and v_{2m+3}') to collapse the cycle, there will be no matches of the query in any model.

Together, this gives four options for removing the two mentioned length-three cycles. However, two of these options are ruled out because the resulting collapsings have no match in the grid representation. The first such case is when we identify v with v_0 and v' with v_{2m+3}. To see that there is no match, first observe that v_0 and v_{2m+3} have to satisfy G. Then make a case distinction on the two options that we have for eliminating the cycle $\{v_{\mathsf{ans}}, v_{m+1}, v_{m+2}\}$.

Case (1). If we identify v_{ans} and v_{m+1}, the path from the G-variable v_0 to v_{ans} is only of length $m+1$. In our grid representation, all paths from a G-node to an ABox individual (i.e., the root) are of length $m+2$, so there can be no match of this collapsing.

Case (2). If we identify v_{ans} and v_{m+2}, the path from v_{ans} to the G-variable v_{2m+3} is only of length $m+1$ and again there is no match.

We can argue analogously for the case where we identify v with v'_0 and and v' with v'_{2m+3}. Therefore, the two remaining collapsings for eliminating the cycles $\{v, v_0, v'_0\}$ and $\{v', v_{2m+3}, v'_{2m+3}\}$ are the following:

(a) identify v with v_0 and v' with v'_{2m+3};
(b) identify v with v'_0 and v' with v_{2m+3}.

In the first case, we further have to identify v_{ans} with v_{m+2} and v'_{m+1}, for otherwise we can argue as above that there is no match. In the second case, we have to identify v_{ans} with v_{m+1} and v'_{m+2}. After this has been done, there is only one way to eliminate the cycle $v = v_0, \dots, v_{2m+3}, v' = v'_{2m+3}, \dots, v'_0$ such that the result is a chain of length $2m+4$ with the G-variables at both ends and the answer variable exactly in the middle (any other way to collapse means that there are no matches). The reflexive loops at the endpoints of the resulting chain and at v_{ans} can simply be dropped since we work with reflexive roles. The resulting cycle-free queries are shown in the middle and right part of Figure 2.

Note that the middle query has A_i at both ends of the chain, and the right one has $\neg A_i$ at the ends. According to our above argumentation, the original query $q^i_{\mathfrak{D},c}$ has a match in the grid representation iff one of these two collapsings has a match. Thus, every match π of $q^i_{\mathfrak{D},c}$ in the grid representation is such that $\pi(v)$ and $\pi(v')$ are (not necessarily distinct) instances of G that agree on the value of A_i.

At this point, a technical remark is in order. Observe that, in the two relevant collapsings of $q^i_{\mathfrak{D},c}$, the end nodes of the chain and their immediate neighbors are labeled dually w.r.t. A_i and $\neg A_i$. This is an artifact of query construction and cannot be avoided. To deal with it, we have introduced F-nodes into our grid representation and ensured that they satisfy Property $\mathsf{T1}$.

Now set $q_{\mathsf{cnt}} := \bigcup_{i<m} q^i_{\mathfrak{D},c}$. It is not hard to see that every match π of q_{cnt} in the grid representation is such that $\pi(v)$ and $\pi(v')$ are (not necessarily distinct) instances of G that have the same A_i-value, for *all* $i < m$. The query q_{cnt} is almost the desired query $q_{\mathfrak{D},c}$. Recall that we want to enforce Condition $(*)$ from above, and thus also need to talk about tile types in the query. The query q_{tile} is given in the left-hand side of Figure 3 for the case of three tiles, i.e., $T = \{0, 1, 2\}$. In general, for $T = \{0, \dots, k-1\}$, we define

$$\begin{aligned} q_{\mathsf{tile}} := \bigcup_{i<k} \{ & s(w_{i,0}, w_{i,1}), \dots, s(w_{i,2m+2}, w_{i,2m+3}), \\ & s(v_{\mathsf{ans}}, w_{i,m+1}), s(v_{\mathsf{ans}}, w_{i,m+2}), \\ & s(v, w_{i,0}), s(v', w_{i,2m+3}), \\ & D_i(w_{i,0}), D_i(w_{i,2m+3}) \} \\ \cup \bigcup_{i<j<k} & \{ s(w_{i,0}, w_{j,0}), s(w_{i,2m+3}, w_{j,2m+3}) \} \\ \cup\ & \{ G(v), G(v') \} \end{aligned}$$

Observe that q_{cnt} and q_{tile} share the variables v, v', and v_{ans}. Also observe that q_{tile} is very similar to the queries $q^i_{\mathfrak{D},c}$, the main difference being the number of

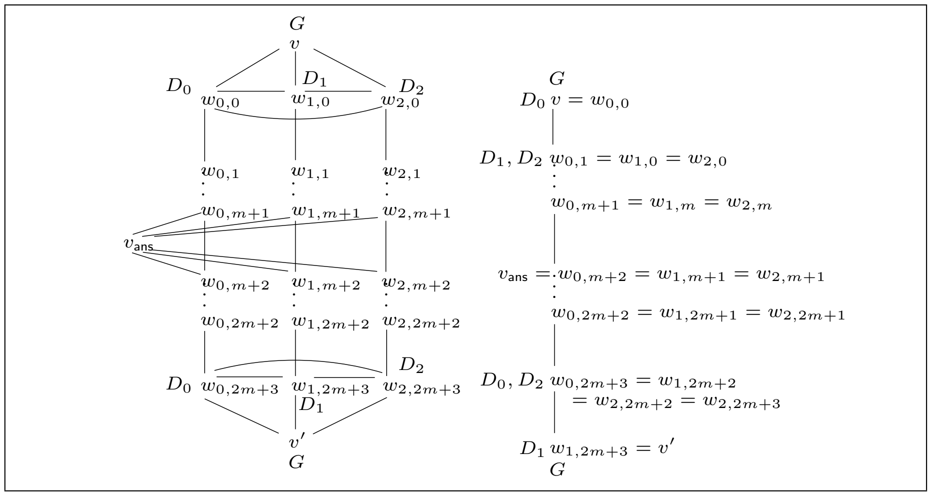

Fig. 3. The query q_{tile} (left) and one of its collapsings (right).

vertical chains. Whereas the queries $q^i_{\mathfrak{D},c}$ have two collapsings that are cycle-free and can have matches in the grid representation, q_{tile} has $k \cdot (k-1)$ such collapsings: for all $i, j \in T$ with $i \neq j$, there is a collapsing into a linear chain of length $2m + 4$ whose two end nodes are labelled D_i and D_j, respectively. An example of such a collapsing is presented on the right-hand side of Figure 3. The arguments for how to obtain these collapsing from q_{tile} and why other collapsings have no match in the grid representation are similar to the line of argumentation used for $q^i_{\mathfrak{D},c}$ and involves Property $\mathsf{T2}$. We refer to [15] for details.

Now, the desired query $q_{\mathfrak{D},c}$ is simply the union of q_{cnt} and q_{tile}. From what was already said about q_{cnt} and q_{tile}, it is easily derived that $q_{\mathfrak{D},c}$ does not match the grid representation iff Property $(*)$ is satisfied. It is possible to show that there is a solution for $\mathfrak{D}$ and c iff $(\emptyset, \mathcal{A}_{\mathfrak{D},c}) \not\models q_{\mathfrak{D},c}$. We have thus proved that rooted query entailment in $\mathcal{ALCI}$ is co-NExpTime-hard. A matching upper bound can be obtained by adapting the techniques in [7]. More details are given in [16].

Theorem 1. *Rooted query entailment in $\mathcal{ALCI}$ is co-*NExpTime*-complete. The lower bound holds even if the TBox is empty and the ABox is of the form $\{C(a)\}$.*

4 2ExpTime-hardness Results

Theorem 1 shows that, already in the case of rooted queries, conjunctive query entailment in DLs between $\mathcal{ALCI}$ and $\mathcal{SHIQ}$ is more difficult than instance checking. In the general case, conjunctive query entailment in these DLs is even 2ExpTime-complete. The proof is by a reduction of the word problem of exponentially space bounded alternating Turing machines (ATMs) [5], and reuses many ideas from the reduction given in Section 3. Because of space limitations, we can only give a very rough sketch of the proof.

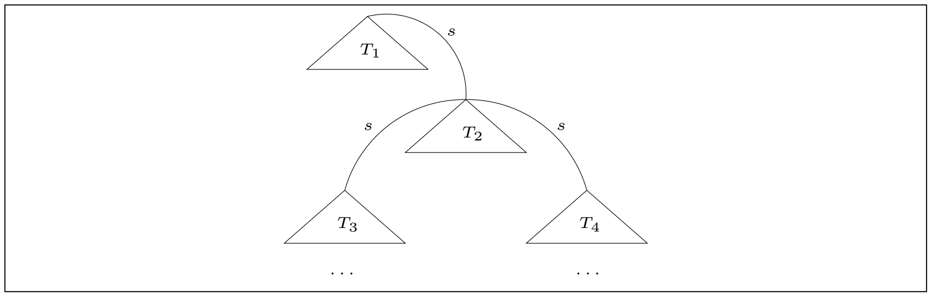

Fig. 4. Representing ATM computations.

The main idea is to represent each configuration of an ATM by the leafs of a tree of depth n, similar to the grid representation in Section 3. Trees representing configurations are then interconnected to form a larger tree that represents a computation. This is illustrated in Figure 4. Each of the T_i is a tree of depth n whose leafs represent a configuration. The tree T_1 represents an existential configuration, and thus has only one successor configuration T_2. In contrast, the tree T_2 represents a universal configuration with two successor configurations T_3 and T_4. The difficult part of the reduction is to relate the content of a tape cell in one configuration to the content of the corresponding cell in the successor configurations. The solution is to use queries that are very similar to the query $q_{\mathfrak{D},c}$ employed in the previous section. A few additional technical tricks are needed to achieve directedness (i.e., talking only about successor configurations, but not about predecessor configurations) since we work with symmetric roles. More details of the reduction can be found in [15]. A 2ExpTime upper bound was established in [7] (where also non-simple roles are allowed in the query).

Theorem 2. *Query entailment in $\mathcal{ALCI}$ is 2*ExpTime*-complete. The lower bound holds even for queries without answer variables and for ABoxes of the form $\{C(a)\}$.*

Using Theorem 2, it is also easy to show that admitting transitive roles in the query destroys the better computational properties of rooted query entailment. $\mathcal{ALCI}_{R^+}$ is the extension of $\mathcal{ALCI}$ with transitive roles.

Theorem 3. *Rooted query entailment in $\mathcal{ALCI}_{R^+}$ is 2*ExpTime*-complete if transitive roles are admitted in the query. The lower bound holds even if the TBox contains only transitivity statements and role inclusions, and the ABox is of the form $\{C(a), r(a,a)\}$.*

Proof. (sketch) By Theorem 2, it suffices to establish the lower bound. We reduce non-rooted query entailment in $\mathcal{ALCI}$, which is 2ExpTime-hard by Theorem 2. Let $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ and q be given, with $\mathcal{A} = \{C(a)\}$. Our aim is to construct a knowledge base $\mathcal{K}' = (\mathcal{T}', \mathcal{A}')$ and rooted query q' such that $\mathcal{K} \models q$ iff $\mathcal{K}' \models q'$. Let $C_\mathcal{T} = \bigsqcap_{D \sqsubseteq E \in \mathcal{T}} \neg D \sqcup E$. Fix a role name t not occurring in $\mathcal{K}$ and q, and a variable v_0 not occurring in q. Then set

$$\begin{aligned} \mathcal{T}' &:= \{\mathsf{Trans}(t)\} \cup \{r \sqsubseteq t, r^- \sqsubseteq t \mid r \in \mathsf{N_R} \text{ occurs in } \mathcal{K}\} \\ \mathcal{A}' &:= \{C \sqcap \forall t.C_{\mathcal{T}}(a), t(a,a)\} \\ q' &:= q \cup \{t(v_0, v) \mid v \in \mathsf{N_V} \text{ occurs in } q\}. \end{aligned}$$

We make v_0 an answer variable in q'. It is not hard to prove that $\mathcal{T}'$, $\mathcal{A}'$, and q' are as required. ❏

The results proved in this section and the preceeding one show that conjunctive query entailment is computationally hard in fragments of $\mathcal{SHIQ}$ that contain $\mathcal{ALCI}$. In the next section, we prove that inverse roles are indeed the culprit for the high complexity: in $\mathcal{SHQ}$ ($\mathcal{SHIQ}$ without inverse roles), conjunctive query entailment is only ExpTime-complete and thus of the same complexity as instance checking.

5 Query Entailment in $\mathcal{SHQ}$ is ExpTime-complete

We give an algorithm for query entailment in $\mathcal{SHQ}$ that runs in ExpTime and is inspired by the 2ExpTime algorithm for conjunctive query entailment in $\mathcal{SHIQ}$ given in [7]. The general idea is to (Turing-)reduce query entailment in $\mathcal{SHQ}$ to ABox consistency in $\mathcal{SHQ}^\sqcap$, i.e., $\mathcal{SHQ}$ extended with role conjunction: given a $\mathcal{SHQ}$-knowledge base $\mathcal{K}$ and a query q, we produce $\mathcal{SHQ}^\sqcap$-knowledge bases $\mathcal{K}_1, \ldots, \mathcal{K}_n$ such that $\mathcal{K} \not\models q$ iff any of the $\mathcal{K}_i$ is consisent. The construction ensures that n is exponential in the size of $\mathcal{K}$ and q, and the size of each $\mathcal{K}_i$ is polynomial in the size of $\mathcal{K}$ and q. Since knowledge base consistency in $\mathcal{SHQ}^\sqcap$ can be decided in ExpTime, we obtain the desired ExpTime upper bound for query entailment in $\mathcal{SHQ}$. Proof details for the lemmas presented in this section can be found in [16].

We start with proving an $\mathcal{SHQ}$ counterpart of Lemma 1. Let $\mathcal{J}$ be an interpretation. A *forest base* $\mathcal{J}$ is an interpretation that interprets transitive roles in an arbitrary way (i.e., not necessarily transitively) and where (i) $\Delta^{\mathcal{J}}$ is a prefix-closed subset of $\mathbb{N}^+$ and (ii) if $(d,e) \in r^{\mathcal{J}}$, then $e, d \in \mathbb{N}$ or $e = d \cdot c$ for some $c \in \mathbb{N}$. Elements of $\Delta^{\mathcal{J}} \cap \mathbb{N}$ are called the *roots* of $\mathcal{J}$. An interpretation $\mathcal{I}$ is the $\mathcal{K}$-*closure* of $\mathcal{J}$ if $\mathcal{I}$ is identical to $\mathcal{J}$ except that, for all roles r, we have

$$r^{\mathcal{I}} = r^{\mathcal{J}} \cup \bigcup_{\mathcal{K} \models s \sqsubseteq r \wedge \mathcal{K} \models \mathsf{Trans}(s)} (s^{\mathcal{J}})^+.$$

A model $\mathcal{I}$ of a knowledge base $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ is a *forest model* of $\mathcal{K}$ if (iii) $\mathcal{I}$ is the $\mathcal{K}$-closure of a forest base $\mathcal{J}$, and (iv) for every root d of $\mathcal{J}$, there is an $a \in \mathsf{Ind}(\mathcal{A})$ such that $a^{\mathcal{I}} = d$. The *roots* of $\mathcal{I}$ are defined as the roots of $\mathcal{J}$. The following proposition shows that, when deciding conjunctive query entailment in $\mathcal{SHQ}$, it suffices to concentrate on forest models.

Proposition 1. *Let $\mathcal{K}$ be an $\mathcal{SHQ}$-knowledge base and q a conjunctive query. If $\mathcal{K} \not\models q$, then there is a forest model $\mathcal{I}$ of $\mathcal{K}$ such that $\mathcal{I} \not\models q$.*

Throughout this section, we will sometimes view a conjunctive query as a directed graph $G_q = (V_q, E_q)$ with $V_q = \mathsf{Var}(q)$ and $E_q = \{(v, v') \mid r(v, v') \in q$ for some $r \in \mathsf{N_R}\}$. We call q *tree-shaped* if G_q is a tree. If q is tree-shaped and v is the root of G_q, we call v the root of q.

In the following, we introduce three notions that are central to the construction of the knowledge bases $\mathcal{K}_1, \ldots, \mathcal{K}_n$: fork rewritings, splittings, and spoilers. We start with fork rewritings, and say that

- q' is *obtained from q by fork elimination* if q' is obtained from q by selecting two atoms $r(v', v)$ and $s(v'', v)$ with $v' \neq v''$ and identifying v' and v'';
- q' is a *fork rewriting* of q if q' is obtained from q by repeated (but not necessarily exhaustive) fork elimination;
- q' is a *maximal fork rewriting* of q if q' is a fork rewriting and no further fork elimination is possible in q'.

The following lemma allows us to speak of *the* maximal fork rewriting of a conjunctive query.

Lemma 2. *Modulo variable renaming, every conjunctive query has a unique maximal fork rewriting.*

Now for splittings, which are partitions of the variables in (a fork rewriting of) the input query. Intuitively, a splitting is induced by each match π for some forest model $\mathcal{I}$ of the input KB $\mathcal{K}$ and the input query q. More precisely, each variable $v \in \mathsf{Var}(q)$ is either

(a) mapped to a root $\pi(v)$ of $\mathcal{I}$;
(b) mapped to a non-root $\pi(v)$ of $\mathcal{I}$ such that there is a variable v' mapped to a root $\pi(v')$ of $\mathcal{I}$ and with v reachable from v' in G_q;
(c) mapped to a non-root $\pi(v)$ of $\mathcal{I}$, but does not satisfy Condition (b).

The purpose of splittings is to describe such a partition without reference to a concrete model $\mathcal{I}$ and a concrete match π. Let $\mathcal{K}$ be an $\mathcal{SHQ}$-knowledge base. A *splitting* of q w.r.t. $\mathcal{K}$ is a tuple $\Pi = \langle R, T, S_1, \ldots, S_n, \mu, \nu \rangle$, where $R, T, S_1, \ldots, S_n$ is a partitioning of $\mathsf{Var}(q)$, $\mu : \{1, \ldots, n\} \to R$ assigns to each set S_i a variable $\mu(i)$ in R, and $\nu : R \to \mathsf{Ind}(\mathcal{A})$ assigns to each variable in R an individual in $\mathcal{A}$. A splitting has to satisfy the following conditions, where $q|_V$ denotes the restriction of q to $V \subseteq \mathsf{Var}(q)$:

1. the query $q|_T$ is a variable-disjoint union of tree-shaped queries;
2. the queries $q|_{S_i}$, $1 \leq i \leq n$, are tree-shaped;
3. if $r(v, v') \in q$, then one of the following holds: (i) v, v' belong to the same set $R, T, S_1, \ldots, S_n$ or (ii) $v \in R$, $\mu(i) = v$, and $v' \in S_i$ is the root of $q|_{S_i}$;
4. for $1 \leq i \leq n$, there is an atom $r(\mu(i), v_0) \in q$, with v_0 the root of $q|_{S_i}$.

Intuitively, the R component of a splitting corresponds to Case (a) above, the $S_1, \ldots, S_n$ correspond to Case (b), and T corresponds to Case (c). Before we introduce spoilers, we establish a central lemma about splittings. We start with a preliminary. Let q be a tree-shaped conjunctive query. We define a $\mathcal{SHQ}^\sqcap$-concept $C_{q,v}$ for each variable $v \in \mathsf{Var}(q)$:

– if v is a leaf in G_q, then $C_{q,v} = \bigsqcap_{C(v) \in q} C$;
– otherwise, $C_{q,v} = \bigsqcap_{C(v) \in q} C \sqcap \bigsqcap_{(v,v') \in E_q} \exists(\bigcap_{s(v,v') \in q} s).C_{q,v'})$.

If v is the root of q, we use C_q to abbreviate $C_{q,v}$. Observe that, since we allow only simple roles in a query q, all concepts C_q involve only simple roles inside role conjunction. The following lemma establishes a connection between forest models and splittings of fork rewritings.

Lemma 3. *Let $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ be a knowledge base, $\mathcal{I}$ a forest model of $\mathcal{K}$, and q a conjunctive query. Then $\mathcal{I} \models q$ iff there exists a fork rewriting q' of q and a splitting $\langle R, T, S_1, \ldots, S_n, \mu, \nu \rangle$ of q' w.r.t. $\mathcal{K}$ such that*

1. *for each disconnected component $\widehat{q}$ of T, there is a $d \in \Delta^{\mathcal{I}}$ with $d \in (C_{\widehat{q}})^{\mathcal{I}}$;*
2. *if $C(v) \in q'$ with $v \in R$, then $\nu(v)^{\mathcal{I}} \in C^{\mathcal{I}}$;*
3. *if $r(v, v') \in q'$ with $v, v' \in R$, then $(\nu(v)^{\mathcal{I}}, \nu(v')^{\mathcal{I}}) \in r^{\mathcal{I}}$;*
4. *for $1 \leq i \leq n$, we have $\nu(\mu(i))^{\mathcal{I}} \in \big(\exists(\bigcap_{s(\mu(i),v_0) \in q'} s).C_{q'|_{S_i}}\big)^{\mathcal{I}}$ with v_0 root of the tree-shaped query $q'|_{S_i}$.*

Now for the definition of spoilers, which exploit Lemma 3 to prevent matches of the input query q in forest-models of the input KB $\mathcal{K} = (\mathcal{T}, \mathcal{A})$. We first define spoilers of specific splittings, and then spoilers of the query (i.e., of all splittings). Let $\Pi = \langle R, T, S_1, \ldots, S_n, \mu, \nu \rangle$ be a splitting of q w.r.t. $\mathcal{K}$ such that $q_1, \ldots, q_k$ are the (tree-shaped) disconnected components of $q|_T$. A $\mathcal{SHQ}^{\sqcap}$-knowledge base $(\mathcal{T}', \mathcal{A}')$ is a *spoiler* for q, $\mathcal{K}$, and Π if one of the following conditions hold:

1. $\top \sqsubseteq \neg C_{q_i} \in \mathcal{T}'$, for some i with $1 \leq i \leq k$;
2. there is an atom $C(v) \in q$ with $v \in R$ and $\neg C(\nu(v)) \in \mathcal{A}'$;
3. there is an atom $r(v, v') \in q$ with $v, v' \in R$ and $\neg r(\nu(v), \nu(v')) \in \mathcal{A}'$;
4. $\neg D(\nu(\mu(i))) \in \mathcal{A}'$ for some $i \in \{1, \ldots, n\}$, and where $D = \exists(\bigcap_{s(\mu(i),v_0) \in q} s).C_{q|_{S_i}}$ with v_0 root of $q|_{S_i}$.

A $\mathcal{SHQ}^{\sqcap}$-knowledge base $\mathcal{K}' = (\mathcal{T}', \mathcal{A}')$ is a *spoiler* for q and $\mathcal{K}$ if (i) for every fork rewriting q' of q and every splitting Π of q' w.r.t. $\mathcal{K}$, $\mathcal{K}'$ is a spoiler for q', $\mathcal{K}$, and Π; and (ii) $\mathcal{K}'$ is minimal with Property (i). The proof of the following lemma is based on the correspondence between Conditions 1-4 of spoilers and Conditions 1-4 of Lemma 3.

Lemma 4. *Let $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ be a $\mathcal{SHQ}$-knowledge base and q a conjunctive query. Then $\mathcal{K} \not\models q$ iff there is a spoiler $(\mathcal{T}', \mathcal{A}')$ for q and $\mathcal{K}$ such that $(\mathcal{T} \cup \mathcal{T}', \mathcal{A} \cup \mathcal{A}')$ is consistent.*

Lemma 4 suggests the following algorithm for deciding conjunctive query entailment in $\mathcal{SHQ}$: given $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ and q, enumerate all spoilers $(\mathcal{T}', \mathcal{A}')$ for q and $\mathcal{K}$, return "yes" if for all such spoilers, $(\mathcal{T} \cup \mathcal{T}', \mathcal{A} \cup \mathcal{A}')$ is inconsistent, and "no" otherwise. To prove that this algorithm runs in ExpTime, we first note that consistency of $\mathcal{SHQ}^{\sqcap}$-KBs is ExpTime-complete. Since only simple roles occur inside role conjunctions, this can be proved by an easy variation of Lemma 6.19 in [22]. It thus suffices to establish the following.

Lemma 5. *Let $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ be a $\mathcal{SHQ}$-knowledge base and q a conjunctive query. Then the number of spoilers for q and $\mathcal{K}$ is exponential in the size of q and $\mathcal{K}$ and the set of all spoilers can be computed in time exponential in the size of q and $\mathcal{K}$.*

The proof of this lemma is a key ingredient to our ExpTime upper bound. The upper bound on the number of spoilers is established by showing that (i) all individual names and role names occurring in spoilers also occur in the input KB and input query, and (ii) there are only polynomially many different concepts that can occur in spoilers. While (i) is trivial, (ii) is not. Define

$$\mathsf{Trees}(q) := \{q|_{\mathsf{Reach}_q(v)} \mid v \in \mathsf{Var}(q) \text{ and } q|_{\mathsf{Reach}_q(v)} \text{ is tree-shaped}\}.$$

The proof of (ii) proceeds by showing that if C occurs in a spoiler of $\mathcal{K}$ and q and q^* is the maximal fork rewriting of q, then there is a $\widehat{q} \in \mathsf{Trees}(q)$ with $C = C_q$. Details are given in [16].

Summing up, we have established the following result, where the lower bound is trivial by a reduction of instance checking in $\mathcal{SHQ}$.

Theorem 4. *Conjunctive query entailment in $\mathcal{SHQ}$ is* ExpTime*-complete.*

6 Conclusion

We have carried out a detailed investigation of the complexity of conjunctive query entailment in DLs between $\mathcal{ALC}$ and $\mathcal{SHIQ}$. In particular, we have proved that conjunctive query entailment is more complex than instance checking when inverse roles are present (2ExpTime vs ExpTime), and that the complexity coincides without inverse roles (ExpTime). Our two upper bound proofs (Theorem 1 and 4) do not apply to the case where transitive roles are admitted in the query. As shown by Theorem 3, the NExpTime upper bound from Theorem 1 cannot be generalized to this case. It remains an open problem whether or not the ExpTime upper bound in Theorem 4 can be adapted to $\mathcal{SHQ}$ with transitive roles in the query. An ExpTime upper bound for a fragment of this problem is established in [19].

References

1. F. Baader, D. L. McGuiness, D. Nardi, and P. Patel-Schneider. The Description Logic Handbook. Cambridge University Press, 2003.
2. D. Calvanese, G. De Giacomo, and M. Lenzerini. On the decidability of query containment under constraints. In *Proc. of PODS'98*, pages 149–158, 1998.
3. D. Calvanese, G. D. Giacomo, D. Lembo, M. Lenzerini, and R. Rosati. Data complexity of query answering in description logics. In *Proc. of KR'06*, pages 260–270. AAAI Press, 2006.
4. D. Calvanese, T. Eiter, and M. Ortiz. Answering regular path queries in expressive description logics: an automata-theoretic approach. In *Proc. of AAAI'07*. AAAI Press, 2007.

5. A. K. Chandra, D. C. Kozen, and L. J. Stockmeyer. Alternation. *Journal of the ACM*, 28(1):114–133, 1981.
6. B. Glimm, I. Horrocks, I., and U. Sattler. Conjunctive query entailment for $\mathcal{SHOQ}$. In *Proc. of DL'07*, volume 250 of *CEUR-WS*, 2007.
7. B. Glimm, C. Lutz, I. Horrocks, and U. Sattler. Answering conjunctive queries in the $\mathcal{SHIQ}$ description logic. *JAIR*, 31:150–197, 2008.
8. I. Horrocks, U. Sattler, and S. Tobies. Practical reasoning for expressive description logics. In *Proc. of LPAR'99*, number 1705 in LNAI, pages 161–180. Springer, 1999.
9. I. Horrocks, U. Sattler, and S. Tobies. Reasoning with individuals for the description logic SHIQ. In *Proc. of CADE-17*, number 1831 in LNCS, pages 482–496. Springer, 2000.
10. I. Horrocks and S. Tessaris. A conjunctive query language for description logic ABoxes. In *Proc. of AAAI'00*. AAAI Press, 2000.
11. U. Hustadt, B. Motik, and U. Sattler. Data complexity of reasoning in very expressive description logics. In *Proc. of IJCAI'05*, pages 466–471. Professional Book Center, 2005.
12. M. Krötzsch, S. Rudolph, and P. Hitzler. Conjunctive queries for a tractable fragment of OWL 1.1. In *Proc. of ISWC'07*, volume 4825 of LNCS, pages 310-323. Springer, 2007.
13. C. Lutz. *The Complexity of Reasoning with Concrete Domains.* PhD thesis, LuFG Theoretical Computer Science, RWTH Aachen, Germany, 2002.
14. C. Lutz, C. Areces, I. Horrocks, and U. Sattler. Keys, nominals, and concrete domains. *Journal of Artificial Intelligence Research (JAIR)*, 23:667–726, 2005.
15. C. Lutz. Inverse roles make conjunctive queries hard. In *Proc. of DL2007*, volume 250 of *CEUR-WS*, 2007. Full version http://lat.inf.tu-dresden.de/∼clu/papers/
16. C. Lutz. Two upper bounds for conjunctive query answering in $\mathcal{SHIQ}$. In *Proc. of DL2008*, *CEUR-WS*, 2008. Full version http://lat.inf.tu-dresden.de/∼clu/papers/
17. M. Ortiz, D. Calvanese, and T. Eiter. Characterizing data complexity for conjunctive query answering in expressive description logics. In *Proc. of AAAI'06*. AAAI Press, 2006.
18. M. Ortiz, M. Šimkus, and T. Eiter. Worst-case optimal conjunctive query answering for an expressive description logic without inverses. In *Proc. of AAAI'08*. AAAI Press, 2008.
19. M. Ortiz, M. Šimkus, and T. Eiter. Conjunctive query answering in $\mathcal{SH}$ using knots. In *Proc. of DL'08*. CEUR WS, 2008.
20. A. Schaerf. On the complexity of the instance checking problem in concept languages with existential quantification. *JIIS*, 2:265–278, 1993.
21. R. Rosati. On conjunctive query answering in $\mathcal{EL}$. In *Proc. of DL2007*, volume 250 of *CEUR-WS*, 2007.
22. S. Tobies. *Complexity Results and Practical Algorithms for Logics in Knowledge Representation.* PhD thesis, RWTH Aachen, 2001.