# Conjunctive Query Answering for the Description Logic $\mathcal{SHIQ}$

**Birte Glimm**[1], **Ian Horrocks**[1], **Carsten Lutz**[2], **Uli Sattler**[1]

[1] School of Computer Science
University of Manchester, UK

[2] Institute for Theoretical Computer Science
TU Dresden, Germany

## Abstract

Conjunctive queries play an important role as an expressive query language for Description Logics (DLs). Although modern DLs usually provide for transitive roles, it was an open problem whether conjunctive query answering over DL knowledge bases is decidable if transitive roles are admitted in the query. In this paper, we consider conjunctive queries over knowledge bases formulated in the popular DL $\mathcal{SHIQ}$ and allow transitive roles in both the query and the knowledge base. We show that query answering is decidable and establish the following complexity bounds: regarding combined complexity, we devise a deterministic algorithm for query answering that needs time single exponential in the size of the KB and double exponential in the size of the query. Regarding data complexity, we prove co-NP-completeness.

## 1 Introduction

Description Logics (DLs) [Baader *et al.*, 2003] are a well-established family of logic-based knowledge representation formalisms that have recently gained increased attention due to their usage as the logical underpinning of ontology languages such as DAML+OIL and OWL [Horrocks *et al.*, 2003]. A DL knowledge base (KB) consists of a TBox, which contains intensional knowledge such as concept definitions and general background knowledge, and an ABox, which contains extensional knowledge and is used to describe individuals. Using a database metaphor, the TBox corresponds to the schema, and the ABox corresponds to the data.

In data-intensive applications, querying KBs plays a central role. Essentially, there are two forms of querying. The first one is *instance retrieval*, which allows the retrieval of all certain instances of a given (possibly complex) concept $C$, i.e., it returns all individuals from the ABox that are an instance of $C$ in every model of the KB. Technically, instance retrieval is well-understood. For the prominent DL $\mathcal{SHIQ}$, which underlies DAML+OIL and OWL Lite, it is ExpTime-complete [Tobies, 2001], and, despite this high worst-case complexity, efficient implementations are available. On the other hand, instance retrieval is a rather poor form of querying: concepts are used as queries, and thus we can only query for structures that are invariant under (guarded) bisimulations. For this reason, many applications require *conjunctive query answering* as a stronger form of querying, i.e., computing the certain answers to a conjunctive query over a DL knowledge base.

Until now it was an open problem whether conjunctive query answering is decidable in $\mathcal{SHIQ}$. In particular, the presence of transitive and inverse roles makes the problem rather tricky [Glimm *et al.*, 2006], and results were only available for two restricted cases. The first case is obtained by stipulating that the variables in queries can only be bound to individuals that are explicitly mentioned in the ABox. The result is a form of closed-domain semantics, which is different from the usual open-domain semantics in DLs. It is easily seen that conjunctive query answering in this setting can be reduced to instance retrieval. In the second case, the binary atoms in conjunctive queries are restricted to roles that are neither transitive nor have transitive sub-roles, and it is known that conjunctive query answering in this setting is decidable and co-NP-complete regarding data complexity [Ortiz *et al.*, 2006].

In this paper, we show that unrestricted conjunctive query answering in $\mathcal{SHIQ}$ is decidable. More precisely, we devise a decision procedure for the *entailment* of a conjunctive query by a $\mathcal{SHIQ}$ knowledge base, which is the decision problem corresponding to conjunctive query answering. It is well-known that decidability and complexity results carry over from entailment to answering. Our decision procedure for query entailment consists of a rather intricate reduction to KB consistency in $\mathcal{SHIQ}^{\sqcap}$, i.e., $\mathcal{SHIQ}$ extended with role conjunction. The latter is easily seen to be decidable. The resulting (deterministic) algorithm for conjunctive query entailment in $\mathcal{SHIQ}$ needs time double exponential in the size of the query and single exponential in the size of the KB. This result concerns the combined complexity, i.e., it is measured in the size of all inputs: the query, the ABox, and the TBox. Since query and TBox are usually small compared to the ABox, the data complexity (measured in the size of the ABox, only) is also a relevant issue. We show that (the decision problem corresponding to) conjunctive query answering in $\mathcal{SHIQ}$ is co-NP-complete regarding data complexity, and thus not harder than instance retrieval [Hustadt *et al.*, 2005].

This paper is accompanied by a technical report which contains full proofs [Glimm *et al.*, 2006].

## 2 Preliminaries

### Syntax and Semantics of $\mathcal{SHIQ}$

Let $\mathsf{N_C}$, $\mathsf{N_R}$, and $\mathsf{N_I}$ be sets of *concept names*, *role names*, and *individual names*. We assume that the role names contain a subset $\mathsf{N_{tR}} \subseteq \mathsf{N_R}$ of *transitive role names*. A *role* is an element of $\mathsf{N_R} \cup \{r^- \mid r \in \mathsf{N_R}\}$, where roles of the form $r^-$ are called *inverse roles*. A *role inclusion* is of the form $r \sqsubseteq s$ with $r, s$ roles. A *role hierarchy* $\mathcal{H}$ is a finite set of role inclusions.

An *interpretation* $\mathcal{I} = (\Delta^\mathcal{I}, \cdot^\mathcal{I})$ consists of a non-empty set $\Delta^\mathcal{I}$, the *domain* of $\mathcal{I}$, and a function $\cdot^\mathcal{I}$, which maps every concept name $A$ to a subset $A^\mathcal{I} \subseteq \Delta^\mathcal{I}$, every role name $r \in \mathsf{N_R}$ to a binary relation $r^\mathcal{I} \subseteq \Delta^\mathcal{I} \times \Delta^\mathcal{I}$ such that $r^\mathcal{I}$ is transitive for every $r \in \mathsf{N_{tR}}$, and every individual name $a$ to an element $a^\mathcal{I} \in \Delta^\mathcal{I}$. An interpretation $\mathcal{I}$ *satisfies* a role inclusion $r \sqsubseteq s$ if $r^\mathcal{I} \subseteq s^\mathcal{I}$, and a role hierarchy $\mathcal{H}$ if it satisfies all role inclusions in $\mathcal{H}$. We use the following standard notation:

(1) $\mathsf{Inv}(r) := r^-$ if $r \in \mathsf{N_R}$ and $\mathsf{Inv}(r) := s$ if $r = s^-$ for a role name $s$.

(2) For a role hierarchy $\mathcal{H}$, $\sqsubseteq^*_\mathcal{H}$ is the reflexive transitive closure of $\sqsubseteq$ over $\mathcal{H} \cup \{\mathsf{Inv}(r) \sqsubseteq \mathsf{Inv}(s) \mid r \sqsubseteq s \in \mathcal{H}\}$, and we use $r \equiv^*_\mathcal{H} s$ as an abbreviation for $r \sqsubseteq^*_\mathcal{H} s$ and $s \sqsubseteq^*_\mathcal{H} r$.

(3) For a role hierarchy $\mathcal{H}$ and a role $s$, we define the set $\mathsf{Trans}_\mathcal{H}$ of transitive roles as

$$\{s \mid \exists \text{ role } r \text{ with } r \equiv^*_\mathcal{H} s \text{ and } r \in \mathsf{N_{tR}} \text{ or } \mathsf{Inv}(r) \in \mathsf{N_{tR}}\}.$$

(4) A role $r$ is called *simple* w.r.t. a role hierarchy $\mathcal{H}$ if, for each role $s$ such that $s \sqsubseteq^*_\mathcal{H} r$, $s \notin \mathsf{Trans}_\mathcal{H}$.

The subscript $\mathcal{H}$ of $\sqsubseteq^*_\mathcal{H}$ and $\mathsf{Trans}_\mathcal{H}$ is dropped if clear from the context. $\mathcal{SHIQ}$*-concepts* (or concepts for short) are built inductively using the following grammar, where $A \in \mathsf{N_C}$, $n \in \mathbb{N}$, $r$ is a role, and $s$ is a simple role:

$$\begin{array}{rl} C ::= & \top \mid \bot \mid A \mid \neg C \mid C_1 \sqcap C_2 \mid C_1 \sqcup C_2 \mid \\ & \forall r.C \mid \exists r.C \mid \leqslant ns.C \mid \geqslant ns.C. \end{array}$$

The semantics of $\mathcal{SHIQ}$-concepts is defined as usual, see e.g. [Horrocks *et al.*, 2000] for details.

A *general concept inclusion* (GCI) is an expression $C \sqsubseteq D$, where both $C$ and $D$ are concepts. A finite set of GCIs is called a *TBox*. An *assertion* is an expression of the form $A(a), \neg A(a), r(a,b), \neg r(a,b)$, or $a \not\approx b$, where $A$ is a concept name, $r$ is a role, and $a, b \in \mathsf{N_I}$. An *ABox* is a finite set of assertions. We use $\mathsf{Ind}(\mathcal{A})$ to denote the set of individual names occurring in $\mathcal{A}$. An interpretation $\mathcal{I}$ *satisfies* a GCI $C \sqsubseteq D$ if $C^\mathcal{I} \subseteq D^\mathcal{I}$. Satisfaction of assertions is defined in the obvious way, e.g. $A(a)$ is satisfied if $a^\mathcal{I} \in A^\mathcal{I}$. An interpretation $\mathcal{I}$ *satisfies* a TBox (ABox) if it satisfies each GCI (assertion) in it.

Observe that, in ABox assertions $C(a)$, we require $C$ to be a (possibly negated) concept name. This is a standard assumption when the data complexity of DLs is analyzed, see e.g. [Donini *et al.*, 1994]. In this paper, we will sometimes also allow ABox assertions $C(a)$, where $C$ is an arbitrary concept. To make this explicit, we will call such ABoxes *generalized*.

A *knowledge base* (KB) is a triple $(\mathcal{T}, \mathcal{H}, \mathcal{A})$ with $\mathcal{T}$ a TBox, $\mathcal{H}$ a role hierarchy, and $\mathcal{A}$ an ABox. Let $\mathcal{K} = (\mathcal{T}, \mathcal{H}, \mathcal{A})$ be a KB and $\mathcal{I} = (\Delta^\mathcal{I}, \cdot^\mathcal{I})$ an interpretation. An interpretation $\mathcal{I}$ *satisfies* $\mathcal{K}$ if it satisfies $\mathcal{T}$, $\mathcal{H}$, and $\mathcal{A}$. If $\Gamma$ is a TBox, ABox, or KB and $\mathcal{I}$ satisfies $\Gamma$, we say that $\mathcal{I}$ is a *model* of $\Gamma$ and write $\mathcal{I} \models \Gamma$. A knowledge base $\mathcal{K}$ *is consistent* if it has a model.

### Conjunctive Queries

Let $\mathsf{N_V}$ be a countably infinite set of *variables* disjoint from $\mathsf{N_C}$, $\mathsf{N_R}$, and $\mathsf{N_I}$. An *atom* is an expression $A(v)$ (*concept atom*) or $r(v, v')$ (*role atom*), where $A$ is a concept name, $r$ is a role, and $v, v' \in \mathsf{N_V}$. A *conjunctive query* $q$ is a non-empty set of atoms. Intuitively, such a set represents the conjunction of its elements. We use $\mathsf{Var}(q)$ ($\mathsf{Var}(at)$) to denote the set of variables occurring in the query $q$ (atom $at$). Let $\mathcal{I}$ be an interpretation, $q$ a conjunctive query, and $\pi : \mathsf{Var}(q) \to \Delta^\mathcal{I}$ a total function. We write

- $\mathcal{I} \models^\pi C(v)$ if $(\pi(v)) \in C^\mathcal{I}$;
- $\mathcal{I} \models^\pi r(v, v')$ 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 $\pi$ 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 $\pi$ 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 defined as follows: given a knowledge base $\mathcal{K}$ and a query $q$, decide whether $\mathcal{K} \models q$. It is well-known that query entailment and query answering can be mutually reduced and that decidability and complexity results carry over [Calvanese *et al.*, 1998; Horrocks and Tessaris, 2000]. In the remainder of this paper, we concentrate on query entailment.

For convenience, we assume that conjunctive queries are closed under inverses, i.e., if $r(v, v') \in q$, then $\mathsf{Inv}(r)(v', v) \in q$. If we add or remove atoms from a query, we silently assume that we do this such that the resulting query is again closed under inverses. We will also assume that queries are connected. Formally, a query $q$ (closed under inverses) is *connected* if, for all $v, v' \in \mathsf{Var}(q)$, there exists a sequence $v_0, \ldots, v_n$ such that $v_0 = v$, $v_n = v'$, and for all $i < n$, there exists a role $r$ such that $r(v_i, v_{i+1}) \in q$. A collection $q_0, \ldots, q_k$ of queries is a *partitioning* of $q$ if $q = q_0 \cup \cdots \cup q_k$, $\mathsf{Var}(q_i) \cap \mathsf{Var}(q_j) = \emptyset$ for $i < j \leq k$, and each $q_i$ is connected. The following lemma shows that connectedness can be assumed w.l.o.g.

**Lemma 1.** *Let $\mathcal{K}$ be a knowledge base, $q$ a conjunctive query, and $q_0, \ldots, q_n$ a partitioning of $q$. Then $\mathcal{K} \models q$ iff $\mathcal{K} \models q_i$ for all $i \leq n$.*

## 3 Forests and Trees

In this section, we carefully analyze the entailment of queries by knowledge bases and establish a set of general properties that will play a central role in our decision procedure. We start by showing that, in order to decide whether $\mathcal{K} \models q$, it suffices to check whether $\mathcal{I} \models q$ for all models $\mathcal{I}$ of $\mathcal{K}$ that are of a particular shape. Intuitively, these models are shaped like a forest (in the graph-theoretic sense), modulo the fact that transitive roles have to be interpreted in transitive relations.

Let $\mathbb{N}^*$ be the set of all (finite) words over the alphabet $\mathbb{N}$. A *tree* $T$ is a non-empty, prefix-closed subset of $\mathbb{N}^*$. For

$w, w' \in T$, we call $w'$ a *successor* of $w$ if $w' = w \cdot c$ for some $c \in \mathbb{N}$, where "$\cdot$" denotes concatenation. We call $w'$ a *neighbor* of $w$ if $w'$ is a successor of $w$ or $w$ is a successor of $w'$.

**Definition 2.** A *forest base for $\mathcal{K}$* is an interpretation $\mathcal{I}$ that interpretes transitive roles in unrestricted (i.e., not necessarily transitive) relations and, additionally, satisfies the following conditions:

T1 $\Delta^\mathcal{I} \subseteq \mathsf{Ind}(\mathcal{A}) \times \mathbb{N}^*$ such that, for all $a \in \mathsf{Ind}(\mathcal{A})$, the set $\{w \mid (a, w) \in \Delta^\mathcal{I}\}$ is a tree;

T2 if $((a, w), (a', w')) \in r^\mathcal{I}$, then either $w = w' = \varepsilon$ or $a = a'$ and $w'$ is a neighbor of $w$;

T3 for all $a \in \mathsf{Ind}(\mathcal{A})$, $a^\mathcal{I} = (b, \varepsilon)$ for some $b \in \mathsf{Ind}(\mathcal{A})$.

A model $\mathcal{I}$ of $\mathcal{K}$ is *canonical* if there exists a forest base $\mathcal{J}$ for $\mathcal{K}$ such that $\mathcal{J}$ is identical to $\mathcal{I}$ except that, for all non-simple roles $r$, we have

$$r^\mathcal{I} = r^\mathcal{J} \cup \bigcup_{s \sqsubseteq^* r,\ s \in \mathsf{Trans}} (s^\mathcal{J})^+.$$

In this case, we say that $\mathcal{J}$ is a forest base *for* $\mathcal{I}$. $\triangle$

Observe that, in canonical models $\mathcal{I}$, each individual $a$ is mapped to a pair $(b, \varepsilon)$, where $a = b$ does not necessarily hold. We need this since we do not adopt the *uniqe name assumption (UNA)*: if $a, b \in \mathsf{N_I}$ with $a \neq b$, then we allow that $a^\mathcal{I} = b^\mathcal{I}$. If desired, the UNA can easily be adopted by adding an assertion $a \neq b$ for each pair of individual names in $\mathsf{Ind}(\mathcal{A})$ to the ABox.

**Lemma 3.** *$\mathcal{K} \not\models q$ iff there exists a canonical model $\mathcal{I}$ of $\mathcal{K}$ such that $\mathcal{I} \not\models q$.*

**Proof.** Using standard unravelling (see e.g. [Tobies, 2001]), each model of $\mathcal{K}$ can be converted into a canonical model of $\mathcal{K}$. Moreover, if $\mathcal{I} \models \mathcal{K}$ and $\mathcal{I}'$ is the canonical model obtained by unravelling $\mathcal{I}$, then it is not hard to show that $\mathcal{I} \not\models q$ implies $\mathcal{I}' \not\models q$, for all conjunctive queries $q$. ❑

Lemma 3 shows that, when deciding whether $\mathcal{K} \models q$, it suffices to check whether $\mathcal{I} \models q$ for all canonical models $\mathcal{I}$ of $\mathcal{K}$. As a next step, we would like to show that, for canonical models $\mathcal{I}$, to check whether $\mathcal{I} \models q$, we can restrict our attention to a particular kind of match $\pi$ for $\mathcal{I}$ and $q$. A match $\pi$ for $\mathcal{I}$ and $q$ is a *forest match* if, for all $r(v, v') \in q$, we have one of the following:

- $\pi(v), \pi(v') \in \mathsf{N_I} \times \{\varepsilon\}$;
- $\pi(v), \pi(v') \in \{a\} \times \mathbb{N}^*$ for some $a \in \mathsf{Ind}(\mathcal{A})$.

Alas, it is not sufficient to only consider forest matches for $\mathcal{I}$ and $q$. Instead, we show the following: we can rewrite $q$ into a set of queries $Q$ such that, for all canonical models $\mathcal{I}$, we have that $\mathcal{I} \models q$ iff $\mathcal{I} \models^\pi q'$ for some $q' \in Q$ and forest match $\pi$. Intuitively, this complication is due to the presence of transitive roles.

**Definition 4.** A query $q'$ is called a *transitivity rewriting* of $q$ w.r.t. $\mathcal{K}$ if it is obtained from $q$ by choosing atoms $r_0(v_0, v_0'), \ldots, r_n(v_n, v_n') \in q$ and roles $s_0, \ldots, s_n \in \mathsf{Trans}_\mathcal{H}$ such that $s_i \sqsubseteq^*_\mathcal{H} r_i$ for all $i \leq n$, and then replacing $r_i(v_i, v_i')$ with

$$s_i(v_i, u_i), s_i(u_i, v_i')$$
or
$$s_i(v_i, u_i), s_i(u_i, u_i'), s_i(u_i', v_i')$$

for all $i \leq n$, where $u_i, u_i' \notin \mathsf{Var}(q)$. We use $\mathsf{tr}_\mathcal{K}(q)$ to denote the set of all transitivity rewritings of $q$ w.r.t. $\mathcal{K}$. $\triangle$

We assume that $\mathsf{tr}_\mathcal{K}(q)$ contains no isomorphic queries, i.e., differences in (newly introduced) variable names are neglected. Together with Lemma 3, the following lemma shows that, to decide whether $\mathcal{K} \models q$, it suffices to check the existence of some canonical model $\mathcal{I}$, some forest match $\pi$, and some $q' \in \mathsf{tr}_\mathcal{K}(q)$ such that $\mathcal{I} \models^\pi q'$.

**Lemma 5.** *Let $\mathcal{I}$ be a model of $\mathcal{K}$.*

1. *If $\mathcal{I}$ is canonical and $\mathcal{I} \models q$, then there is a $q' \in \mathsf{tr}_\mathcal{K}(q)$ such that $\mathcal{I} \models^{\pi'} q'$, with $\pi'$ a forest match.*
2. *If $\mathcal{I} \models q'$ with $q' \in \mathsf{tr}_\mathcal{K}(q)$, then $\mathcal{I} \models q$.*

**Proof.** (1) can be proved by using the match and the canonical structure of $\mathcal{I}$ to guide the rewriting process. (2) holds by definition of transitivity rewritings and the semantics. ❑

The most important property of forest matches is the following: if $\mathcal{I} \models^\pi q$ with $\pi$ a forest match, then $\pi$ splits the query $q$ into several subqueries: the *base subquery* $q_0$ contains all role atoms that are matched to root nodes:

$$q_0 := \{r(v, v') \in q \mid \pi(v), \pi(v') \in \mathsf{N_I} \times \{\varepsilon\}\};$$

Moreover, for each $(a, \varepsilon) \in \mathsf{N_I} \times \{\varepsilon\}$ which occurs in the range of $\pi$, there is an *object subquery* $q_a$:

$$q_a := \{at \mid \forall v \in \mathsf{Var}(at) : \pi(v) \in \{a\} \times \mathbb{N}^*\} \setminus q_0.$$

Clearly, $q = q_0 \cup \bigcup_a q_a$. Although the resulting subqueries are not a partitioning of $q$ in the sense of Section 2, one of the fundamental ideas behind our decision procedure is to treat the different subqueries more or less separately. The main benefit is that the object subqueries can be rewritten into tree-shaped queries which can then be translated into concepts. This technique is also known as "rolling up" of tree conjunctive queries into concepts and was proposed in [Calvanese *et al.*, 1998; Horrocks and Tessaris, 2000].

Formally, a query $q$ is *tree-shaped* if there exists a bijection $\sigma$ from $\mathsf{Var}(q)$ into a tree $T$ such that $r(v, v') \in q$ implies that $\sigma(v)$ is a neighbor of $\sigma(v')$ in $T$. Before we show how to rewrite the object subqueries into tree-shaped queries, let us substantiate our claim that tree-shaped queries can be rolled up into concepts. The result of rolling up is not a $\mathcal{SHIQ}$-concept, but a concept formulated in $\mathcal{SHIQ}^\sqcap$, the extension of $\mathcal{SHIQ}$ with role intersection. More precisely, $\mathcal{SHIQ}^\sqcap$ is obtained from $\mathcal{SHIQ}$ by admitting the concept constructors $\exists \alpha.C$, $\forall \alpha.C$, $\leqslant \alpha.C$, and $\geqslant \alpha.C$, where $\alpha$ is a role conjunction $r_1 \sqcap \cdots \sqcap r_k$ with the $r_i$ (possibly inverse) roles.

Let $q$ be a tree-shaped query and $\sigma$ a bijection from $\mathsf{Var}(q)$ into a tree $T$. Inductively assign to each $v \in \mathsf{Var}(q)$ a $\mathcal{SHIQ}^\sqcap$-concept $C_q(v)$:

- if $\sigma(v)$ is a leaf of $T$, then $C_q(v) := \bigsqcap_{A(v) \in q} A$

- if $\sigma(v)$ has successors $\sigma(v_1), \ldots, \sigma(v_n)$ in $T$, then

$$C_q(v) := \bigsqcap_{A(v) \in q} A \sqcap \bigsqcap_{1 \leq i \leq n} \exists (\bigsqcap_{r(v,v_i) \in q} r).C_q(v_i).$$

Then the rolling up $C_q$ of $q$ is defined as $C_q(v_r)$, where $v_r$ is such that $\sigma(v_r) = \varepsilon$. (Recall that $\sigma$ is a bijection, hence, such a $v_r$ exists.) The following lemma shows the connection between the query and the rolled up concept.

**Lemma 6.** *Let $q$ be a tree-shaped query and $\mathcal{I}$ an interpretation. Then $\mathcal{I} \models q$ iff $C_q^{\mathcal{I}} \neq \emptyset$.*

We now show how to transform a query $q$ into a set $Q$ of tree-shaped ones. To describe the exact goal of this translation, we need to introduce tree matches as a special case of forest matches: a match $\pi$ for a canonical model $\mathcal{I}$ and $q$ is a *tree match* if the range of $\pi$ is a subset of $\{a\} \times \mathbb{N}^*$, for some $a \in \mathsf{N_I}$. Now, our tree transformation should be such that

> $(*)$ whenever $\mathcal{I} \models^{\pi} q$ with $\mathcal{I}$ a canonical model and $\pi$ a tree match, then $\mathcal{I} \models^{\pi'} q'$ for some (tree-shaped) query $q' \in Q$ and tree match $\pi'$.

Recall the splitting of a query into a base subquery and a set of object subqueries $q_a$, induced by a forest match $\pi$. It is not hard to see that for each $q_a$, the restriction of $\pi$ to $\mathsf{Var}(q_a)$ is a tree match for $\mathcal{I}$ and $q_a$. Thus, object subqueries together with their inducing matches $\pi$ satisfy the precondition of $(*)$.

The rewriting of a query into a tree-shaped one is a three stage process. In the first stage, we derive a *collapsing* $q_0$ of the original query $q$ by (possibly) identifying variables in $q$. This allows us, e.g., to transform atoms $r(v,u), r(v,u'), r(u,w), r(u',w)$ into a tree shape by identifying $u$ and $u'$. In the second stage, we derive an *extension* $q_1$ of $q_0$ by (possibly) introducing new variables and role atoms that make certain existing role atoms $r(v,v')$ redundant, where $r$ is non-simple. In the third stage, we derive a *reduct* $q'$ of $q_1$ by (possibly) removing redundant role atoms, i.e., atoms $r(v,v')$ such that there exist atoms $s(v_0,v_1), \ldots, s(v_{n-1},v_n) \in q$ with $v_0 = v$, $v_n = v'$, $s \sqsubseteq^* r$, and $s \in \mathsf{Trans}$. Combining the extension and reduct steps allows us, e.g., to transform a non-tree-shaped "loop" $r(v,v)$ into a tree shape by adding a new variable $v'$ and edges $s(v,v'), s(v',v)$ such that $s \sqsubseteq^* r$ and $s \in \mathsf{Trans}$, and then removing the redundant atom $r(v,v)$.

In what follows, the *size* $|q|$ of a query $q$ is defined as the number of atoms in $q$.

**Definition 7.** A *collapsing* of $q$ is obtained by identifying variables in $q$. A query $q'$ is an *extension* of $q$ w.r.t. $\mathcal{K}$ if:

1. $q \subseteq q'$;
2. $A(v) \in q'$ implies $A(v) \in q$;
3. $r(v,v') \in q' \setminus q$ implies that $r$ occurs in $\mathcal{K}$;
4. $|\mathsf{Var}(q')| \leq 4|q|$;
5. $|\{r(v,v') \in q' \mid r(v,v') \notin q\}| \leq 171|q|^2$.

A query $q'$ is a *reduct* of $q$ w.r.t. $\mathcal{K}$ if:

1. $q' \subseteq q$;
2. $A(v) \in q$ implies $A(v) \in q'$;
3. if $r(v,v') \in q \setminus q'$, then there is a role $s$ such that $s \sqsubseteq^* r$, $s \in \mathsf{Trans}$, and there are $v_0, \ldots, v_n$ such that $v_0 = v$, $v_n = v'$, and $s(v_i, v_{i+1}) \in q'$ for all $i < n$.

A query $q'$ is a *tree transformation* of $q$ w.r.t. $\mathcal{K}$ if there exist queries $q_0$ and $q_1$ such that

- $q_0$ is a collapsing of $q$,
- $q_1$ is an extension of $q_0$ w.r.t. $\mathcal{K}$, and
- $q'$ is a tree-shaped reduct of $q_1$.

We use $\mathsf{tt}_{\mathcal{K}}(q)$ to denote the set of all tree transformations of $q$ w.r.t. $\mathcal{K}$. △

We remark that Condition 5 of query extensions is not strictly needed for correctness, but it ensures that our algorithm is only single exponential in the size of $\mathcal{K}$. As in the case of $\mathsf{tr}_{\mathcal{K}}(q)$, we assume that $\mathsf{tt}_{\mathcal{K}}(q)$ does not contain any isomorphic queries.

The next lemma states the central properties of tree transformations. Before we can formulate it, we introduce two technical notions. Let $q' \in \mathsf{tt}_{\mathcal{K}}(q)$, $\mathcal{I} \models^{\pi} q$, and $\mathcal{I} \models^{\pi'} q'$. Then $\pi$ and $\pi'$ are called *$\varepsilon$-compatible* if the following holds: if $v \in \mathsf{Var}(q)$, $v$ was identified with variable $v' \in \mathsf{Var}(q')$ during the collapsing step, and at least one of $\pi(v), \pi'(v')$ is in $\mathsf{N_I} \times \{\varepsilon\}$, then $\pi(v) = \pi'(v')$. Furthermore, we call $\pi$ an *$a$-tree match* if $\pi(v) \in \{a\} \times \mathbb{N}^*$ for all $v \in \mathsf{Var}(q)$.

**Lemma 8.** *Let $\mathcal{I}$ be a model of $\mathcal{K}$.*

1. *If $\mathcal{I}$ is canonical and $\pi$ an $a$-tree match with $\mathcal{I} \models^{\pi} q$, then there is a $q' \in \mathsf{tt}_{\mathcal{K}}(q)$ and an $a$-tree match $\pi'$ such that $\mathcal{I} \models^{\pi'} q'$ and $\pi, \pi'$ are $\varepsilon$-compatible.*
2. *If $q' \in \mathsf{tt}_{\mathcal{K}}(q)$ and $\mathcal{I} \models^{\pi'} q'$, then there is a match $\pi$ with $\mathcal{I} \models^{\pi} q$ and $\pi, \pi'$ are $\varepsilon$-compatible.*

**Proof.** The proof of (2) is straightforward, but the proof of (1) is quite complex. The basic idea behind the proof of (1) is that, given a canonical model $\mathcal{I}$ of $\mathcal{K}$ and a tree match $\pi$ such that $\mathcal{I} \models^{\pi} q$, we can use $\pi$ and $\mathcal{I}$ to guide the transformation process. The bounds of $4|q|$ and $171|q|^2$ in Conditions 4 and 5 of extensions are derived by computing the maximum number of variables and atoms that might be introduced in the guided transformation process. ❑

Intuitively, using $a$-tree matches and $\varepsilon$-compatibility in Lemma 8 ensures that, if we are given a match for the base subquery and a tree match for a tree transformation of each object subquery, then we can construct a forest match of the original query.

## 4 The Decision Procedure

Let $\mathcal{K}$ be a knowledge base and $q$ a conjunctive query such that we want to decide whether $\mathcal{K} \models q$. A *counter model* for $\mathcal{K}$ and $q$ is a model $\mathcal{I}$ of $\mathcal{K}$ such that $\mathcal{I} \not\models q$. In the following, we show how to convert $\mathcal{K}$ and $q$ into a sequence of knowledge bases $\mathcal{K}_1, \ldots, \mathcal{K}_\ell$ such that (i) every counter model for $\mathcal{K}$ and $q$ is a model of some $\mathcal{K}_i$, (ii) every canonical model of a $\mathcal{K}_i$ is a countermodel for $\mathcal{K}$ and $q$, and (iii) each consistent $\mathcal{K}_i$ has a canonical model. Thus, $\mathcal{K} \models q$ iff all the $\mathcal{K}_i$ are inconsistent. This gives rise to two decision procedures: a deterministic one in which we enumerate all $\mathcal{K}_i$ and which we use to derive an upper bound for combined complexity; and a non-determinstic one in which we guess a $\mathcal{K}_i$ and which yields a tight upper bound for data complexity.

Since the knowledge bases $\mathcal{K}_i$ involve concepts obtained by rolling up tree-shaped queries, they are fomulated in $\mathcal{SHIQ}^\sqcap$ rather than in $\mathcal{SHIQ}$. More precisely, each KB $\mathcal{K}_i$ is of the form $(\mathcal{T} \cup \mathcal{T}_q, \mathcal{H}, \mathcal{A} \cup \mathcal{A}_i)$, where

- $(\mathcal{T}, \mathcal{H}, \mathcal{A})$ is a $\mathcal{SHIQ}$ knowledge base;
- $\mathcal{T}_q$ is a set of GCIs $\top \sqsubseteq C$ with $C$ a $\mathcal{SHIQ}^\sqcap$ concept;
- $\mathcal{A}_i$ is a generalized $\mathcal{SHIQ}^\sqcap$-ABox[1] such that $\mathsf{Ind}(\mathcal{A}_i) \subseteq \mathsf{Ind}(\mathcal{A})$.

In what follows, we call knowledge bases of this form *extended* knowledge bases. Using a standard unravelling argument, it is easy to establish Property (iii) from above, i.e., every consistent extended knowledge base $\mathcal{K}$ has a canonical model.

The actual construction of the extended knowledge bases is based on the analysis in Section 3. To start with, Lemma 3 and 5 imply the following: to ensure that a canonical model of an extended KB is a counter model for $\mathcal{K}$ and $q$, it suffices to prevent forest matches of all queries $q' \in \mathsf{tr}_\mathcal{K}(q)$. In order to prevent such matches, we use the parts $\mathcal{T}_q$ and $\mathcal{A}_i$ of extended knowledge bases.

More precisely, we distinguish between two kinds of forest matches: tree matches and *true* forest matches, i.e., forest matches that are not tree matches. Preventing tree matches of a $q' \in \mathsf{tr}_\mathcal{K}(q)$ in a canonical model is relatively simple: by Lemmas 8 and 6, it suffices to ensure that, for all $q'' \in \mathsf{tt}_\mathcal{K}(q')$, the corresponding concept $C_{q''}$ does not have any instances. Therefore, $\mathcal{T}_q$ is defined as follows:

$$\mathcal{T}_q = \{\top \sqsubseteq \neg C_{q''} \mid q'' \in \mathsf{tt}_\mathcal{K}(q') \text{ for some } q' \in \mathsf{tr}_\mathcal{K}(q)\}.$$

It is easily seen that true forest matches $\pi$ always involve at least one ABox individual (i.e., $\pi(v) \in \mathsf{N_I} \times \{\varepsilon\}$ for at least one variable $v$). In order to prevent true forest matches, we thus use an ABox $\mathcal{A}_i$. Intuitively, we obtain the ABoxes $\mathcal{A}_1, \ldots, \mathcal{A}_\ell$ by considering all possible ways of adding assertions to $\mathcal{K}$ such that, for all queries $q' \in \mathsf{tr}_\mathcal{K}(q)$, all true forest matches are prevented. This gives rise to the knowledge bases $\mathcal{K}_1, \ldots, \mathcal{K}_\ell$.

As suggested in Section 3, we can prevent a true forest match $\pi$ of $q' \in \mathsf{tr}_\mathcal{K}(q)$ by splitting $\pi$ into a base subquery and a number of object subqueries and then making sure that either the base query fails to match (this involves only individual names from the ABox) or at least one of the object subqueries fails to have a tree match. In Section 3, however, we used a concrete forest match $\pi$ to split a query into subqueries. Here, we do not have a concrete $\pi$ available and must consider *all possible ways* in which a forest match can split a query.

Let $q' \in \mathsf{tr}_\mathcal{K}(q)$. A *splitting candidate* for $q'$ is a partial function $\tau : \mathsf{Var}(q') \to \mathsf{Ind}(\mathcal{A})$ with non-empty domain. For $a \in \mathsf{Ind}(\mathcal{A})$, we use $\mathsf{Reach}(a, \tau)$ to denote the set of variables $v \in \mathsf{Var}(q')$ for which there exists a sequence of variables $v_0, \ldots, v_n$, $n \geq 0$, such that

- $\tau(v_0) = a$ and $v_n = v$;
- for all $i \leq n$, $\tau(v_i) = a$ or $\tau(v_i)$ is undefined;
- for all $i < n$, there is a role $r$ s.t. $r(v_i, v_{i+1}) \in q'$.

We call $\tau$ a *split mapping* for $q'$ if, for all $a, b \in \mathsf{Ind}(\mathcal{A})$, $a \neq b$ implies $\mathsf{Reach}(a, \tau) \cap \mathsf{Reach}(b, \tau) = \emptyset$. Intuitively, each split mapping $\tau$ for $q'$ represents the set of forest matches $\pi$ of $q'$ such that $\pi(v) = (\tau(v), \varepsilon)$ for all $v$ in the domain of $\tau$. Each injective split mapping for $q'$ induces a splitting of $q'$ into a base query and object queries. Split mappings $\tau$ need not be injective, however, and thus the general picture is that they induce a splitting of the collapsing $q''$ of $q'$ obtained by identifying all variables $v, v'$ with $\tau(v) = \tau(v')$. This splitting is as follows:

$$\begin{aligned} q_0^\tau &:= \{r(v, v') \in q'' \mid \tau(v), \tau(v') \text{ is defined}\} \\ q_a^\tau &:= \{at \in q'' \mid \mathsf{Var}(at) \subseteq \mathsf{Reach}(a, \tau)\} \setminus q_0^\tau \end{aligned}$$

for all $a \in \mathsf{N_I}$ that are in the range of $\tau$. Observe that the condition which distinguishes splitting candidates and split mappings ensures that $a \neq b$ implies $\mathsf{Var}(q_a^\tau) \cap \mathsf{Var}(q_b^\tau) = \emptyset$. This condition is satisfied by the splittings described in Section 3, and it is needed to independently roll up the subqueries $q_a^\tau$ into concepts.

In the following, we use $\mathsf{sub}(q)$ to denote the set of subqueries of $q$, i.e., the set of non-empty subsets of $q$. Let

$$Q := \{q_3 \mid \exists q_1, q_2 \colon q_1 \in \mathsf{tr}_\mathcal{K}(q), q_2 \in \mathsf{sub}(q_1), q_3 \in \mathsf{tt}_\mathcal{K}(q_2)\}$$

and let $\mathsf{cl}(q)$ be the set of rolled up concepts $C_{q'}$, for all $q' \in Q$. A $\mathcal{SHIQ}^\sqcap$ ABox $\mathcal{A}'$ is called a *q-completion* if it contains only assertions of the form

- $\neg C(a)$ for some $C \in \mathsf{cl}(q)$ and $a \in \mathsf{Ind}(\mathcal{A})$ and
- $\neg r(a, b)$ for a role name $r$ occurring in $Q$ and $a, b \in \mathsf{Ind}(\mathcal{A})$.

Let $\tau$ be a split mapping for $q' \in \mathsf{tr}_\mathcal{K}(q)$ and $\mathcal{A}'$ a $q$-completion. We say that $\mathcal{A}'$ *spoils* $\tau$ if one of the following holds:

(a) there is an $r(v, v') \in q_0^\tau$ such that $\neg r(\tau(v), \tau(v')) \in \mathcal{A}'$;

(b) there is an $a$ in the range of $\tau$ such that $\neg C(a) \in \mathcal{A}'$, where
$$C := \bigsqcup_{q'' \in \mathsf{tt}_\mathcal{K}(q_a^\tau)} C_{q''}.$$

Intuitively, (b) prevents tree matches of the object subqueries, c.f. Lemmas 8 and 6.

Finally, a $q$-completion $\mathcal{A}'$ is called a *counter canidate* for $\mathcal{K}$ and $q$ if for all $q' \in \mathsf{tr}_\mathcal{K}(q)$ and all split mappings $\tau$ for $q'$, $\mathcal{A}'$ spoils $\tau$. Let $\mathcal{A}_1, \ldots, \mathcal{A}_\ell$ be all counter candidates for $\mathcal{K}$ and $q$ and $\mathcal{K}_1, \ldots, \mathcal{K}_\ell$ the associated extended KBs.

**Lemma 9.** *$\mathcal{K} \models q$ iff $\mathcal{K}_1, \ldots, \mathcal{K}_\ell$ are inconsistent.*

We now define the two decision procedures for query entailment in $\mathcal{SHIQ}$: in the deterministic version, we generate all $q$-completions $\mathcal{A}'$ of $\mathcal{A}$ and return "$\mathcal{K} \models q$" if all generated $\mathcal{A}'$ that are a counter candidate give rise to an inconsistent extended KB. Otherwise, we return "$\mathcal{K} \not\models q$". In the non-deterministic version, we guess a $q$-completion $\mathcal{A}'$ of $\mathcal{A}$, return "$\mathcal{K} \not\models q$" if $\mathcal{A}'$ is a counter candidate and the associated

[1]Recall that an ABox is *generalized* if it admits assertions $C(a)$ with $C$ an arbitrary concept.

extended KB is consistent, and "$\mathcal{K} \models q$" otherwise. To show that these algorithms are decision procedures, it remains to show that the consistency of extended knowledge bases is decidable. The following theorem can be proved by a reduction to the DL $\mathcal{ALCQI}b$ and using results from [Tobies, 2001]. In the following, the *size* $|\Gamma|$ of an ABox, TBox, role hierarchy or (extended) knowledge base $\Gamma$ is simply the number of symbols needed to write $\Gamma$ (with numbers written in binary).

**Theorem 10.** *There is a polynomial $p$ such that, given an extended knowledge base $\mathcal{K}' = (\mathcal{T} \cup \mathcal{T}_q, \mathcal{H}, \mathcal{A} \cup \mathcal{A}')$ with $|\mathcal{K}'| = k$, $|\mathcal{A} \cup \mathcal{A}'| = a$, $|\mathcal{T} \cup \mathcal{T}_q \cup \mathcal{H}| = t$, and where the maximum length of concepts in $\mathcal{T}_q$ and $\mathcal{A}'$ is $\ell$, we can decide consistency of $\mathcal{K}'$ in*

- *deterministic time in $2^{p(k)2^{p(\ell)}}$;*
- *non-deterministic time in $p(a) \cdot 2^{2^{p(t)}}$.*

We now discuss the complexity of our algorithms. We start by establishing some bounds on the number and size of transitivity rewritings, tree transformations, etc.

**Lemma 11.** *Let $|q| = n$ and $|\mathcal{K}| = m$. Then there is a polynomial $p$ such that*

*(a)* $|\mathsf{tr}_{\mathcal{K}}(q)| \leq 2^{p(n) \cdot \log p(m)}$;

*(b)* *for all $q' \in \mathsf{tr}_{\mathcal{K}}(q)$, $|q'| \leq 3n$;*

*(c)* *for all $q' \in \mathsf{tr}_{\mathcal{K}}(q)$, $|\mathsf{tt}_{\mathcal{K}}(q')| \leq 2^{p(n) \cdot \log p(m)}$;*

*(d)* *for all $q' \in \mathsf{tr}_{\mathcal{K}}(q)$ and $q'' \in \mathsf{tt}_{\mathcal{K}}(q')$, $|q''| \leq p(n)$;*

*(e)* $|\mathsf{cl}(q)| \leq 2^{p(n) \cdot \log p(m)}$*; and*

*(f)* *for all $C \in \mathsf{cl}(q)$, $|C| \leq p(n)$.*

Let $k = |\mathsf{cl}(q)|$. We first show that the deterministic version of our algorithm runs in time exponential in $m$ and double exponential in $n$. This follows from Theorem 10 together with the following observations:

(i) The number of $q$-completions is bounded by $2^{k \cdot m + k \cdot m^2}$, which, by Lemma 11(e), is exponential in $m$ and double exponential in $n$;

(ii) Checking whether a $q$-completion is a counter candidate can be done in time exponential in $n$ and polynomial in $m$;

(iii) By Lemma 11, the cardinality of $\mathcal{T}_q$ and of each $q$-completion is exponential in $n$ and polynomial in $m$, and the maximum length of concepts in $\mathcal{T}_q$ and $\mathcal{A}'$ is polynomial in $n$ (and independent of $m$).

Now for the non-deterministic version. Since we aim at an upper bound for data complexity, we only need to verify that the algorithm runs in time polynomial in the size of $|\mathcal{A}|$, and can neglect the contribution of $\mathcal{T}$, $\mathcal{H}$, and $q$ to time complexity. The desired result follows from Theorem 10 and Points (ii) and (iii) above. This bound is tight since conjunctive query entailment is already co-NP-hard regarding data complexity in the $\mathcal{AL}$ fragment of $\mathcal{SHIQ}$ [Donini *et al.*, 1994]. Summing up, we thus have the following.

**Theorem 12.** *Conjunctive query entailment in $\mathcal{SHIQ}$ is data complete for co-*NP*, and can be decided in time exponential in the size of the knowledge base and double exponential in the size of the query.*

## 5 Conclusions

With the decision procedure presented for conjunctive query entailment (and therefore for query answering) in $\mathcal{SHIQ}$, we close a long open problem. It will be part of future work to extend this procedure to $\mathcal{SHOIN}$, which is the DL underlying OWL DL. We will also attempt to find more implementable algorithms for query answering in $\mathcal{SHIQ}$. Carrying out query transformations in a more goal directed way will be crucial to achieving this.

**Acknowledgements**. This work was supported by the EU funded IST-2005-7603 FET Project Thinking Ontologies (TONES). Birte Glimm was supported by an EPSRC studentship.

## References

[Baader *et al.*, 2003] F. Baader, D. Calvanese, D. McGuinness, D. Nardi, and P. F. Patel-Schneider, editors. *The Description Logic Handbook: Theory, Implementation, and Applications*. Cambridge University Press, 2003.

[Calvanese *et al.*, 1998] D. Calvanese, G. De Giacomo, and M. Lenzerini. On the decidability of query containment under constraints. In *Proc. of PODS'98*. ACM Press, 1998.

[Donini *et al.*, 1994] F. Donini, M. Lenzerini, D. Nardi, and A. Schaerf. Deduction in concept languages: From subsumption to instance checking. Journal of Logic and Computation, 4(4):423-452, 1994.

[Glimm *et al.*, 2006] B. Glimm, I. Horrocks, and U. Sattler. Conjunctive query answering for description logics with transitive roles. In *Proc. of DL-06*. CEUR, 2006.

[Glimm *et al.*, 2006] B. Glimm, I. Horrock, C. Lutz, and U. Sattler. Conjunctive query answering for the description logic $\mathcal{SHIQ}$. LTCS-Report 06-01, Theoretical Computer Science, TU Dresden, 2006. See http://lat.inf.tu-dresden.de/research/reports.html.

[Horrocks and Tessaris, 2000] I. Horrocks and S. Tessaris. A conjunctive query language for description logic aboxes. In *Proc. of AAAI-00*, 2000.

[Horrocks *et al.*, 2000] I. Horrocks, U. Sattler, and S. Tobies. Reasoning with Individuals for the Description Logic $\mathcal{SHIQ}$. In *Proc. of CADE-00*, vol. 1831 of *LNAI*. Springer-Verlag, 2000.

[Horrocks *et al.*, 2003] I. Horrocks, P. F. Patel-Schneider, and F. van Harmelen. From SHIQ and RDF to OWL: The making of a web ontology language. *Journ. of Web Semantics*, 1(1), 2003.

[Hustadt *et al.*, 2005] U. Hustadt, B. Motik, and U. Sattler. Data complexity of reasoning in very expressive description logics. In *Proc. of IJCAI-05*, 2005.

[Ortiz *et al.*, 2006] M. M. Ortiz, D. Calvanese, and T. Eiter. Characterizing data complexity for conjunctive query answering in expressive description logics. In *Proc. of AAAI-06*, 2006.

[Tobies, 2001] S. Tobies. *Complexity Results and Practical Algorithms for Logics in Knowledge Representation*. PhD thesis, RWTH Aachen, 2001.