Journal of Artificial Intelligence Research 31 (2008) 157-204 Submitted 06/07; published 01/08

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

Birte Glimm BIRTE.GLIMM@COMLAB.OX.AC.UK
Ian Horrocks IAN.HORROCKS@COMLAB.OX.AC.UK
Oxford University Computing Laboratory, UK

Carsten Lutz CLU@TCS.INF.TU-DRESDEN.DE
Dresden University of Technology, Germany

Ulrike Sattler SATTLER@CS.MAN.AC.UK
The University of Manchester, UK

Abstract

Conjunctive queries play an important role as an expressive query language for Description Logics (DLs). Although modern DLs usually provide for transitive roles, conjunctive query answering over DL knowledge bases is only poorly understood if transitive roles are admitted in the query. In this paper, we consider unions of conjunctive queries over knowledge bases formulated in the prominent DL $\mathcal{SHIQ}$ and allow transitive roles in both the query and the knowledge base. We show decidability of query answering in this setting and establish two tight complexity bounds: regarding combined complexity, we prove that there is 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, which is optimal. Regarding data complexity, we prove containment in co-NP.

1. Introduction

Description Logics (DLs) are a family of logic based knowledge representation formalisms (Baader, Calvanese, McGuinness, Nardi, & Patel-Schneider, 2003). Most DLs are fragments of First-Order Logic restricted to unary and binary predicates, which are called concepts and roles in DLs. The constructors for building complex expressions are usually chosen such that the key inference problems, such as concept satisfiability, are decidable and preferably of low computational complexity. 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 contrast to databases, however, DL knowledge bases adopt an open world semantics, i.e., they represent information about the domain in an incomplete way.

Standard DL reasoning services include testing concepts for satisfiability and retrieving certain instances of a given concept. The latter retrieves, for a knowledge base consisting of an ABox $\mathcal{A}$ and a TBox $\mathcal{T}$, all (ABox) individuals that are instances of the given (possibly complex) concept expression C, i.e., all those individuals a such that $\mathcal{T}$ and $\mathcal{A}$ entail that a is an instance of C. The underlying reasoning problems are well-understood, and it is known that the combined complexity of these reasoning problems, i.e., the complexity measured in the size of the TBox, the ABox, and the query, is ExpTime-complete for $\mathcal{SHIQ}$ (Tobies, 2001). The data complexity of a reasoning problem is measured in the size of the ABox only. Whenever the TBox and the query are small compared to the ABox, as is often the case in practice, the data complexity gives a more useful performance estimate. For $\mathcal{SHIQ}$, instance retrieval is known to be data complete for co-NP (Hustadt, Motik, & Sattler, 2005).

Despite the high worst case complexity of the standard reasoning problems for very expressive DLs such as $\mathcal{SHIQ}$, there are highly optimized implementations available, e.g., FaCT++ (Tsarkov & Horrocks, 2006), KAON2[1], Pellet (Sirin, Parsia, Cuenca Grau, Kalyanpur, & Katz, 2006), and RacerPro[2]. These systems are used in a wide range of applications, e.g., configuration (McGuinness & Wright, 1998), bio informatics (Wolstencroft, Brass, Horrocks, Lord, Sattler, Turi, & Stevens, 2005), and information integration (Calvanese, De Giacomo, Lenzerini, Nardi, & Rosati, 1998b). Most prominently, DLs are known for their use as a logical underpinning of ontology languages, e.g., OIL, DAML+OIL, and OWL (Horrocks, Patel-Schneider, & van Harmelen, 2003), which is a W3C recommendation (Bechhofer, van Harmelen, Hendler, Horrocks, McGuinness, Patel-Schneider, & Stein, 2004).

In data-intensive applications, querying KBs plays a central role. Instance retrieval is, in some aspects, a rather weak form of querying: although possibly complex concept expressions are used as queries, we can only query for tree-like relational structures, i.e., a DL concept cannot express arbitrary cyclic structures. This property is known as the tree model property and is considered an important reason for the decidability of most Modal and Description Logics (Grädel, 2001; Vardi, 1997). Conjunctive queries (CQs) are well known in the database community and constitute an expressive query language with capabilities that go well beyond standard instance retrieval. For an example, consider a knowledge base that contains an ABox assertion $(\exists \text{hasSon}.(\exists \text{hasDaughter}.\top))(\text{Mary})$, which informally states that the individual (or constant in FOL terms) Mary has a son who has a daughter; hence, that Mary is a grandmother. Additionally, we assume that both roles hasSon and hasDaughter have a transitive super-role hasDescendant. This implies that Mary is related via the role hasDescendant to her (anonymous) grandchild. For this knowledge base, Mary is clearly an answer to the conjunctive query $\text{hasSon}(x, y) \wedge \text{hasDaughter}(y, z) \wedge \text{hasDescendant}(x, z)$, when we assume that x is a distinguished variable (also called answer or free variable) and y, z are non-distinguished (existentially quantified) variables.

If all variables in the query are non-distinguished, the query answer is just true or false and the query is called a Boolean query. Given a knowledge base $\mathcal{K}$ and a Boolean CQ q, the query entailment problem is deciding whether q is true or false w.r.t. $\mathcal{K}$. If a CQ contains distinguished variables, the answers to the query are those tuples of individual names for which the knowledge base entails the query that is obtained by replacing the free variables with the individual names in the answer tuple. The problem of finding all answer tuples is known as query answering. Since query entailment is a decision problem and thus better suited for complexity analysis than query answering, we concentrate on query entailment. This is no restriction since query answering can easily be reduced to query entailment as we illustrate in more detail in Section 2.2.

1. `http://kaon2.semanticweb.org`
2. `http://www.racer-systems.com`

Devising a decision procedure for conjunctive query entailment in expressive DLs such as SHIQ is a challenging problem, in particular when transitive roles are admitted in the query (Glimm, Horrocks, & Sattler, 2006). In the conference version of this paper, we presented the first decision procedure for conjunctive query entailment in $\mathcal{SHIQ}$. In this paper, we generalize this result to unions of conjunctive queries (UCQs) over $\mathcal{SHIQ}$ knowledge bases. We achieve this by rewriting a conjunctive query into a set of conjunctive queries such that each resulting query is either tree-shaped (i.e., it can be expressed as a concept) or grounded (i.e., it contains only constants/individual names and no variables). The entailment of both types of queries can be reduced to standard reasoning problems (Horrocks & Tessaris, 2000; Calvanese, De Giacomo, & Lenzerini, 1998a).

The paper is organized as follows: in Section 2, we give the necessary definitions, followed by a discussion of related work in Section 3. In Section 4, we motivate the query rewriting steps by means of an example. In Section 5, we give formal definitions for the rewriting procedure and show that a Boolean query is indeed entailed by a knowledge base $\mathcal{K}$ iff the disjunction of the rewritten queries is entailed by $\mathcal{K}$. In Section 6, we present a deterministic algorithm for UCQ entailment in $\mathcal{SHIQ}$ that runs in time single exponential in the size of the knowledge base and double exponential in the size of the query. Since the combined complexity of conjunctive query entailment is already 2ExpTime-hard for the DL $\mathcal{ALCI}$ (Lutz, 2007), it follows that this problem is 2ExpTime-complete for $\mathcal{SHIQ}$. This shows that conjunctive query entailment for $\mathcal{SHIQ}$ is strictly harder than instance checking, which is also the case for simpler DLs such as $\mathcal{EL}$ (Rosati, 2007b). We further 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.

The presented decision procedure gives not only insight into query answering; it also has an immediate consequence on the field of extending DL knowledge bases with rules. From the work by Rosati (2006a, Thm. 11), the consistency of a $\mathcal{SHIQ}$ knowledge base extended with (weakly-safe) Datalog rules is decidable iff the entailment of unions of conjunctive queries in $\mathcal{SHIQ}$ is decidable. Hence, we close this open problem as well.

This paper is an extended version of the conference paper: Conjunctive Query Answering for the Description Logic $\mathcal{SHIQ}$. Proceedings of the Twentieth International Joint Conference on Artificial Intelligence (IJCAI'07), Jan 06 - 12, 2007.

2. Preliminaries

We introduce the basic terms and notations used throughout the paper. In particular, we introduce the DL $\mathcal{SHIQ}$ (Horrocks, Sattler, & Tobies, 2000) and (unions of) conjunctive queries.

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

Let N_C, N_R, and N_I be countably infinite sets of *concept names*, *role names*, and *individual names*. We assume that the set of role names contains a subset $N_{tR} \subseteq N_R$ of *transitive role names*. A *role* is an element of $N_R \cup \{r^- \mid r \in N_R\}$, 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* $\mathcal{R}$ 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 $r \in N_R$ to a binary relation $r^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$, every role name $r \in N_{tR}$ to a transitive binary relation $r^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$, 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 $\mathcal{R}$ if it satisfies all role inclusions in $\mathcal{R}$.

We use the following standard notation:

1. We define the function Inv over roles as $\mathsf{Inv}(r) := r^-$ if $r \in N_R$ and $\mathsf{Inv}(r) := s$ if $r = s^-$ for a role name s.

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

3. For a role hierarchy $\mathcal{R}$ and a role s, we define the set $\mathsf{Trans}_{\mathcal{R}}$ of transitive roles as $\{s \mid \text{there is a role } r \text{ with } r \equiv_{\mathcal{R}} s \text{ and } r \in N_{tR} \text{ or } \mathsf{Inv}(r) \in N_{tR}\}$.

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

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

$$C ::= \top \mid \bot \mid A \mid \neg C \mid C_1 \sqcap C_2 \mid C_1 \sqcup C_2 \mid \forall r.C \mid \exists r.C \mid \leqslant n\; s.C \mid \geqslant n\; s.C.$$

Given an interpretation $\mathcal{I}$, the semantics of $\mathcal{SHIQ}$-concepts is defined as follows:

$$\begin{array}{rll}
\top^{\mathcal{I}} = \Delta^{\mathcal{I}} & (C \sqcap D)^{\mathcal{I}} = C^{\mathcal{I}} \cap D^{\mathcal{I}} & (\neg C)^{\mathcal{I}} = \Delta^{\mathcal{I}} \setminus C^{\mathcal{I}} \\
\bot^{\mathcal{I}} = \emptyset & (C \sqcup D)^{\mathcal{I}} = C^{\mathcal{I}} \cup D^{\mathcal{I}} & \\
(\forall r.C)^{\mathcal{I}} = & \multicolumn{2}{l}{\{d \in \Delta^{\mathcal{I}} \mid \text{ if } (d, d') \in r^{\mathcal{I}}, \text{ then } d' \in C^{\mathcal{I}}\}} \\
(\exists r.C)^{\mathcal{I}} = & \multicolumn{2}{l}{\{d \in \Delta^{\mathcal{I}} \mid \text{ there is a } (d, d') \in r^{\mathcal{I}} \text{ with } d' \in C^{\mathcal{I}}\}} \\
(\leqslant n\; s.C)^{\mathcal{I}} = & \multicolumn{2}{l}{\{d \in \Delta^{\mathcal{I}} \mid \sharp(s^{\mathcal{I}}(d, C)) \leq n\}} \\
(\geqslant n\; s.C)^{\mathcal{I}} = & \multicolumn{2}{l}{\{d \in \Delta^{\mathcal{I}} \mid \sharp(s^{\mathcal{I}}(d, C)) \geq n\}}
\end{array}$$

where $\sharp(M)$ denotes the cardinality of the set M and $s^{\mathcal{I}}(d, C)$ is defined as

$$\{d' \in \Delta^{\mathcal{I}} \mid (d, d') \in s^{\mathcal{I}} \text{ and } d' \in C^{\mathcal{I}}\}.$$

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 interpretation $\mathcal{I}$ *satisfies* a GCI $C \sqsubseteq D$ if $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$, and a TBox $\mathcal{T}$ if it satisfies each GCI in $\mathcal{T}$.

An (ABox) *assertion* is an expression of the form $C(a)$, $r(a, b)$, $\neg r(a, b)$, or $a \neq b$, where C is a concept, r is a role, $a, b \in N_I$. An *ABox* is a finite set of assertions. We use $\mathsf{Inds}(\mathcal{A})$ to denote the set of individual names occurring in $\mathcal{A}$. An interpretation $\mathcal{I}$ *satisfies* an assertion $C(a)$ if $a^{\mathcal{I}} \in C^{\mathcal{I}}$, $r(a, b)$ if $(a^{\mathcal{I}}, b^{\mathcal{I}}) \in r^{\mathcal{I}}$, $\neg r(a, b)$ if $(a^{\mathcal{I}}, b^{\mathcal{I}}) \notin r^{\mathcal{I}}$, and $a \neq b$ if $a^{\mathcal{I}} \neq b^{\mathcal{I}}$. An interpretation $\mathcal{I}$ *satisfies* an ABox if it satisfies each assertion in $\mathcal{A}$, which we denote with $\mathcal{I} \models \mathcal{A}$.

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

2.1.1 Extending $\mathcal{SHIQ}$ to $\mathcal{SHIQ}^{\sqcap}$

In the following section, we show how we can reduce a conjunctive query to a set of ground or tree-shaped conjunctive queries. During the reduction, we may introduce concepts that contain an intersection of roles under existential quantification. We define, therefore, the extension of $\mathcal{SHIQ}$ with role conjunction/intersection, denoted as $\mathcal{SHIQ}^{\sqcap}$ and, in the appendix, we show how to decide the consistency of $\mathcal{SHIQ}^{\sqcap}$ knowledge bases.

In addition to the constructors introduced for $\mathcal{SHIQ}$, $\mathcal{SHIQ}^{\sqcap}$ allows for concepts of the form

$$C ::= \forall R.C \mid \exists R.C \mid\leqslant n\, S.C \mid\geqslant n\, S.C,$$

where $R := r_1 \sqcap \ldots \sqcap r_n$, $S := s_1 \sqcap \ldots \sqcap s_n$, $r_1, \ldots, r_n$ are roles, and $s_1, \ldots, s_n$ are simple roles. The interpretation function is extended such that $(r_1 \sqcap \ldots \sqcap r_n)^{\mathcal{I}} = {r_1}^{\mathcal{I}} \cap \ldots \cap {r_n}^{\mathcal{I}}$.

2.2 Conjunctive Queries and Unions of Conjunctive Queries

We now introduce Boolean conjunctive queries since they are the basic form of queries we are concerned with. We later also define non-Boolean queries and show how they can be reduced to Boolean queries. Finally, unions of conjunctive queries are just a disjunction of conjunctive queries.

For simplicity, we write a conjunctive query as a set instead of as a conjunction of atoms. For example, we write the introductory example from Section 1 as

$$\{\text{hasSon}(x, y), \text{hasDaughter}(y, z), \text{hasDescendant}(x, z)\}.$$

For non-Boolean queries, i.e., when we consider the problem of query answering, the answer variables are often given in the head of the query, e.g.,

$$(x_1, x_2, x_3) \leftarrow \{\text{hasSon}(x_1, x_2), \text{hasDaughter}(x_2, x_3), \text{hasDescendant}(x_1, x_3)\}$$

indicates that the query answers are those tuples (a_1, a_2, a_3) of individual names that, substituted for x_1, x_2, and x_3 respectively, result in a Boolean query that is entailed by the knowledge base. For simplicity and since we mainly focus on query entailment, we do not use a query head even in the case of a non-Boolean query. Instead, we explicitly say which variables are answer variables and which ones are existentially quantified. We now give a definition of Boolean conjunctive queries.

Definition 1. Let N_V be a countably infinite set of variables disjoint from N_C, N_R, and N_I. A *term* t is an element from $N_V \cup N_I$. Let C be a concept, r a role, and t, t' terms. An *atom* is an expression $C(t)$, $r(t, t')$, or $t \approx t'$ and we refer to these three different types of atoms as *concept atoms, role atoms*, and *equality atoms* respectively. A *Boolean conjunctive query*

q is a non-empty set of atoms. We use $\mathsf{Vars}(q)$ to denote the set of (existentially quantified) variables occurring in q, $\mathsf{Inds}(q)$ to denote the set of individual names occurring in q, and $\mathsf{Terms}(q)$ for the set of terms in q, where $\mathsf{Terms}(q) = \mathsf{Vars}(q) \cup \mathsf{Inds}(q)$. If all terms in q are individual names, we say that q is *ground*. A *sub-query* of q is simply a subset of q (including q itself). As usual, we use $\sharp(q)$ to denote the cardinality of q, which is simply the number of atoms in q, and we use $|q|$ for the size of q, i.e., the number of symbols necessary to write q. A $\mathcal{SHIQ}$ conjunctive query is a conjunctive query in which all concepts C that occur in a concept atom $C(t)$ are $\mathcal{SHIQ}$-concepts.

Since equality is reflexive, symmetric and transitive, we define $\stackrel{*}{\approx}$ as the transitive, reflexive, and symmetric closure of $\approx$ over the terms in q. Hence, the relation $\stackrel{*}{\approx}$ is an equivalence relation over the terms in q and, for $t \in \mathsf{Terms}(q)$, we use $[t]$ to denote the equivalence class of t by $\stackrel{*}{\approx}$.

Let $\mathcal{I} = (\Delta^\mathcal{I}, \cdot^\mathcal{I})$ be an interpretation. A total function $\pi \colon \mathsf{Terms}(q) \to \Delta^\mathcal{I}$ is an *evaluation* if (i) $\pi(a) = a^\mathcal{I}$ for each individual name $a \in \mathsf{Inds}(q)$ and (ii) $\pi(t) = \pi(t')$ for all $t \stackrel{*}{\approx} t'$. We write

- $\mathcal{I} \models^\pi C(t)$ if $\pi(t) \in C^\mathcal{I}$;
- $\mathcal{I} \models^\pi r(t, t')$ if $(\pi(t), \pi(t')) \in r^\mathcal{I}$;
- $\mathcal{I} \models^\pi t \approx t'$ if $\pi(t) = \pi(t')$.

If, for an evaluation π, $\mathcal{I} \models^\pi at$ for all atoms $at \in q$, we write $\mathcal{I} \models^\pi q$. We say that $\mathcal{I}$ *satisfies* q and write $\mathcal{I} \models q$ if there exists an evaluation π such that $\mathcal{I} \models^\pi q$. We call such a π a *match* for q in $\mathcal{I}$.

Let $\mathcal{K}$ be a $\mathcal{SHIQ}$ knowledge base and q a conjunctive query. If $\mathcal{I} \models \mathcal{K}$ implies $\mathcal{I} \models q$, we say that $\mathcal{K}$ *entails* q and write $\mathcal{K} \models q$. $\triangle$

The *query entailment problem* is defined as follows: given a knowledge base $\mathcal{K}$ and a query q, decide whether $\mathcal{K} \models q$.

For brevity and simplicity of notation, we define the relation $\bar{\in}$ over atoms in q as follows: $C(t) \mathrel{\bar{\in}} q$ if there is a term $t' \in \mathsf{Terms}(q)$ such that $t \stackrel{*}{\approx} t'$ and $C(t') \in q$, and $r(t_1, t_2) \mathrel{\bar{\in}} q$ if there are terms $t'_1, t'_2 \in \mathsf{Terms}(q)$ such that $t_1 \stackrel{*}{\approx} t'_1$, $t_2 \stackrel{*}{\approx} t'_2$, and $r(t'_1, t'_2) \in q$ or $\mathsf{Inv}(r)(t'_2, t'_1) \in q$. This is clearly justified by definition of the semantics, in particular, because $\mathcal{I} \models r(t, t')$ implies that $\mathcal{I} \models \mathsf{Inv}(r)(t', t)$.

When devising a decision procedure for CQ entailment, most complications arise from cyclic queries (Calvanese et al., 1998a; Chekuri & Rajaraman, 1997). In this context, when we say cyclic, we mean that the graph structure induced by the query is cyclic, i.e., the graph obtained from q such that each term is considered as a node and each role atom induces an edge. Since, in the presence of inverse roles, a query containing the role atom $r(t, t')$ is equivalent to the query obtained by replacing this atom with $\mathsf{Inv}(r)(t', t)$, the direction of the edges is not important and we say that a query is cyclic if its underlying undirected graph structure is cyclic. Please note also that multiple role atoms for two terms are not considered as a cycle, e.g., the query $\{r(t, t'), s(t, t')\}$ is not a cyclic query. The following is a more formal definition of this property.

Definition 2. A query q is *cyclic* if there exists a sequence of terms $t_1, \ldots, t_n$ with $n > 3$ such that

1. for each i with $1 \leq i < n$, there exists a role atom $r_i(t_i, t_{i+1}) \bar{\in} q$,
2. $t_1 = t_n$, and
3. $t_i \neq t_j$ for $1 \leq i < j < n$. $\triangle$

In the above definition, Item 3 makes sure that we do not consider queries as cyclic just because they contain two terms t, t' for which there are more than two role atoms using the two terms. Please note that we use the relation $\bar{\in}$ here, which implicitly uses the relation $\stackrel{*}{\approx}$ and abstracts from the directedness of role atoms.

In the following, if we write that we replace $r(t, t') \bar{\in} q$ with $s(t_1, t_2), \ldots, s(t_{n-1}, t_n)$ for $t = t_1$ and $t' = t_n$, we mean that we first remove any occurrences of $r(\hat{t}, \hat{t}')$ and $\mathsf{Inv}(r)(\hat{t}', \hat{t})$ such that $\hat{t} \stackrel{*}{\approx} t$ and $\hat{t}' \stackrel{*}{\approx} t'$ from q, and then add the atoms $s(t_1, t_2), \ldots, s(t_{n-1}, t_n)$ to q.

W.l.o.g., we assume that queries are connected. More precisely, let q be a conjunctive query. We say that q is *connected* if, for all $t, t' \in \mathsf{Terms}(q)$, there exists a sequence $t_1, \ldots, t_n$ such that $t_1 = t$, $t_n = t'$ and, for all $1 \leq i < n$, there exists a role r such that $r(t_i, t_{i+1}) \bar{\in} q$. A collection $q_1, \ldots, q_n$ of queries is a *partitioning* of q if $q = q_1 \cup \ldots \cup q_n$, $q_i \cap q_j = \emptyset$ for $1 \leq i < j \leq n$, and each q_i is connected.

Lemma 3. *Let $\mathcal{K}$ be a knowledge base, q a conjunctive query, and $q_1, \ldots, q_n$ a partitioning of q. Then $\mathcal{K} \models q$ iff $\mathcal{K} \models q_i$ for each i with $1 \leq i \leq n$.*

A proof is given by Tessaris (2001, 7.3.2) and, with this lemma, it is clear that the restriction to connected queries is indeed w.l.o.g. since entailment of q can be decided by checking entailment of each q_i at a time. In what follows, we therefore assume queries to be connected without further notice.

Definition 4. A *union of Boolean conjunctive queries* is a formula $q_1 \vee \ldots \vee q_n$, where each disjunct q_i is a Boolean conjunctive query.

A knowledge base $\mathcal{K}$ *entails* a union of Boolean conjunctive queries $q_1 \vee \ldots \vee q_n$, written as $\mathcal{K} \models q_1 \vee \ldots \vee q_n$, if, for each interpretation $\mathcal{I}$ such that $\mathcal{I} \models \mathcal{K}$, there is some i such that $\mathcal{I} \models q_i$ and $1 \leq i \leq n$. $\triangle$

W.l.o.g. we assume that the variable names in each disjunct are different from the variable names in the other disjuncts. This can always be achieved by naming variables apart. We further assume that each disjunct is a connected conjunctive query. This is w.l.o.g. since a UCQ which contains unconnected disjuncts can always be transformed into conjunctive normal form; we can then decide entailment for each resulting conjunct separately and each conjunct is a union of connected conjunctive queries. We describe this transformation now in more detail and, for a more convenient notation, we write a conjunctive query $\{at_1, \ldots, at_k\}$ as $at_1 \wedge \ldots \wedge at_k$ in the following proof, instead of the usual set notation.

Lemma 5. *Let $\mathcal{K}$ be a knowledge base, $q = q_1 \vee \ldots \vee q_n$ a union of conjunctive queries such that, for $1 \leq i \leq n$, $q_i^1, \ldots, q_i^{k_i}$ is a partitioning of the conjunctive query q_i. Then $\mathcal{K} \models q$ iff*

$$\mathcal{K} \models \bigwedge_{(i_1,\ldots,i_n) \in \{1,\ldots,k_1\} \times \ldots \times \{1,\ldots,k_n\}} (q_1^{i_1} \vee \ldots \vee q_n^{i_n}).$$

Again, a detailed proof is given by Tessaris (2001, 7.3.3). Please note that, due to the transformation into conjunctive normal form, the resulting number of unions of connected conjunctive queries for which we have to test entailment can be exponential in the size of the original query. When analysing the complexity of the decision procedures presented in Section 6, we show that the assumption that each CQ in a UCQ is connected does not increase the complexity.

We now make the connection between query entailment and query answering clearer. For query answering, let the variables of a conjunctive query be typed: each variable can either be existentially quantified (also called *non-distinguished*) or free (also called *distinguished* or *answer variables*). Let q be a query in n variables (i.e., $\sharp(\mathsf{Vars}(q)) = n$), of which $v_1, \ldots, v_m$ ($m \leq n$) are answer variables. The *answers* of $\mathcal{K} = (\mathcal{T}, \mathcal{R}, \mathcal{A})$ to q are those m-tuples $(a_1, \ldots, a_m) \in \mathsf{Inds}(\mathcal{A})^m$ such that, for all models $\mathcal{I}$ of $\mathcal{K}$, $\mathcal{I} \models^\pi q$ for some π that satisfies $\pi(v_i) = a_i{}^\mathcal{I}$ for all i with $1 \leq i \leq m$. It is not hard to see that the answers of $\mathcal{K}$ to q can be computed by testing, for each $(a_1, \ldots, a_m) \in \mathsf{Inds}(\mathcal{A})^m$, whether the query $q_{[v_1,\ldots,v_m/a_1,\ldots,a_m]}$ obtained from q by replacing each occurrence of v_i with a_i for $1 \leq i \leq m$ is entailed by $\mathcal{K}$. The answer to q is then the set of all m-tuples $(a_1, \ldots, a_m)$ for which $\mathcal{K} \models q_{[v_1,\ldots,v_m/a_1,\ldots,a_m]}$. Let $k = \sharp(\mathsf{Inds}(\mathcal{A}))$ be the number of individual names used in the ABox $\mathcal{A}$. Since $\mathcal{A}$ is finite, clearly k is finite. Hence, deciding which tuples belong to the set of answers can be checked with at most k^m entailment tests. This is clearly not very efficient, but optimizations can be used, e.g., to identify a (hopefully small) set of candidate tuples.

The algorithm that we present in Section 6 decides query entailment. The reasons for devising a decision procedure for query entailment instead of query answering are two-fold: first, query answering can be reduced to query entailment as shown above; second, in contrast to query answering, query entailment is a decision problem and can be studied in terms of complexity theory.

In the remainder of this paper, if not stated otherwise, we use q (possibly with subscripts) for a connected Boolean conjunctive query, $\mathcal{K}$ for a $\mathcal{SHIQ}$ knowledge base ($\mathcal{T}$, $\mathcal{R}$, $\mathcal{A}$), $\mathcal{I}$ for an interpretation $(\Delta^\mathcal{I}, \cdot^\mathcal{I})$, and π for an evaluation.

3. Related Work

Very recently, an automata-based decision procedure for positive existential path queries over $\mathcal{ALCQIb}_{reg}$ knowledge bases has been presented (Calvanese, Eiter, & Ortiz, 2007). Positive existential path queries generalize unions of conjunctive queries and since a $\mathcal{SHIQ}$ knowledge base can be polynomially reduced to an $\mathcal{ALCQIb}_{reg}$ knowledge base, the presented algorithm is a decision procedure for (union of) conjunctive query entailment in $\mathcal{SHIQ}$ as well. The automata-based technique can be considered more elegant than our rewriting algorithm, but it does not give an NP upper bound for the data complexity as our technique.

Most existing algorithms for conjunctive query answering in expressive DLs assume, however, that role atoms in conjunctive queries use only roles that are not transitive. As a consequence, the example query from the introductory section cannot be answered. Under this restriction, decision procedures for various DLs around $\mathcal{SHIQ}$ are known (Horrocks & Tessaris, 2000; Ortiz, Calvanese, & Eiter, 2006b), and it is known that answering conjunctive queries in this setting is data complete for co-NP (Ortiz et al., 2006b). Another common restriction is that only individuals named in the ABox are considered for the assignments of variables. In this setting, the semantics of queries is no longer the standard First-Order one. With this restriction, the answer to the example query from the introduction would be false since Mary is the only named individual. It is not hard to see that conjunctive query answering with this restriction can be reduced to standard instance retrieval by replacing the variables with individual names from the ABox and then testing the entailment of each conjunct separately. Most of the implemented DL reasoners, e.g., KAON2, Pellet, and RacerPro, provide an interface for conjunctive query answering in this setting and employ several optimizations to improve the performance (Sirin & Parsia, 2006; Motik, Sattler, & Studer, 2004; Wessel & Möller, 2005). Pellet appears to be the only reasoner that also supports the standard First-Order semantics for $\mathcal{SHIQ}$ conjunctive queries under the restriction that the queries are acyclic.

To the best of our knowledge, it is still an open problem whether conjunctive query entailment is decidable in $\mathcal{SHOIQ}$. Regarding undecidability results, it is known that conjunctive query entailment in the two variable fragment of First-Order Logic $\mathcal{L}_2$ is undecidable (Rosati, 2007a) and Rosati identifies a relatively small set of constructors that causes the undecidability.

Query entailment and answering have also been studied in the context of databases with incomplete information (Rosati, 2006b; van der Meyden, 1998; Grahne, 1991). In this setting, DLs can be used as schema languages, but the expressivity of the considered DLs is much lower than the expressivity of $\mathcal{SHIQ}$. For example, the constructors provided by logics of the DL-Lite family (Calvanese, De Giacomo, Lembo, Lenzerini, & Rosati, 2007) are chosen such that the standard reasoning tasks are in PTIME and query entailment is in LOGSPACE with respect to data complexity. Furthermore, TBox reasoning can be done independently of the ABox and the ABox can be stored and accessed using a standard database SQL engine. Since the considered DLs are considerable less expressive than $\mathcal{SHIQ}$, the techniques used in databases with incomplete information cannot be applied in our setting.

Regarding the query language, it is well known that an extension of conjunctive queries with inequalities is undecidable (Calvanese et al., 1998a). Recently, it has further been shown that even for DLs with low expressivity, an extension of conjunctive queries with inequalities or safe role negation leads to undecidability (Rosati, 2007a).

A related reasoning problem is *query containment*. Given a schema (or TBox) $\mathcal{S}$ and two queries q and q', we have that q is contained in q' w.r.t. $\mathcal{S}$ iff every interpretation $\mathcal{I}$ that satisfies $\mathcal{S}$ and q also satisfies q'. It is well known that query containment w.r.t. a TBox can be reduced to deciding query entailment for (unions of) conjunctive queries w.r.t. a knowledge base (Calvanese et al., 1998a). Hence a decision procedure for (unions of) conjunctive queries in $\mathcal{SHIQ}$ can also be used for deciding query containment w.r.t. to a $\mathcal{SHIQ}$ TBox.

Entailment of unions of conjunctive queries is also closely related to the problem of adding rules to a DL knowledge base, e.g., in the form of Datalog rules. Augmenting a DL KB with an arbitrary Datalog program easily leads to undecidability (Levy & Rousset, 1998). In order to ensure decidability, the interaction between the Datalog rules and the DL knowledge base is usually restricted by imposing a safeness condition. The $\mathcal{DL}$+*log* framework (Rosati, 2006a) provides the least restrictive integration proposed so far. Rosati presents an algorithm that decides the consistency of a $\mathcal{DL}$+*log* knowledge base by reducing the problem to entailment of unions of conjunctive queries, and he proves that decidability of UCQs in $\mathcal{SHIQ}$ implies the decidability of consistency for $\mathcal{SHIQ}$+*log* knowledge bases.

4. Query Rewriting by Example

In this section, we motivate the ideas behind our query rewriting technique by means of examples. In the following section, we give precise definitions for all rewriting steps.

4.1 Forest Bases and Canonical Interpretations

The main idea is that we can focus on models of the knowledge base that have a kind of tree or forest shape. It is well known that one reason for Description and Modal Logics being so robustly decidable is that they enjoy some form of tree model property, i.e., every satisfiable concept has a model that is tree-shaped (Vardi, 1997; Grädel, 2001). When going from concept satisfiability to knowledge base consistency, we need to replace the tree model property with a form of forest model property, i.e., every consistent KB has a model that consists of a set of "trees", where each root corresponds to a named individual in the ABox. The roots can be connected via arbitrary relational structures, induced by the role assertions given in the ABox. A forest model is, therefore, not a forest in the graph theoretic sense. Furthermore, transitive roles can introduce "short-cut" edges between elements within a tree or even between elements of different trees. Hence we talk of "a form of" forest model property.

We now define forest models and show that, for deciding query entailment, we can restrict our attention to forest models. The rewriting steps are then used to transform cyclic subparts of the query into tree-shaped ones such that there is a "forest-shaped match" for the rewritten query into the forest models.

In order to make the forest model property even clearer, we also introduce *forest bases*, which are interpretations that interpret transitive roles in an unrestricted way, i.e., not necessarily in a transitive way. For a forest base, we require in particular that *all* relationships between elements of the domain that can be inferred by transitively closing a role are omitted. In the following, we assume that the ABox contains at least one individual name, i.e., $\mathsf{Inds}(\mathcal{A})$ is non-empty. This is w.l.o.g. since we can always add an assertion $\top(a)$ to the ABox for a fresh individual name $a \in N_I$. For readers familiar with tableau algorithms, it is worth noting that forest bases can also be thought of as those tableaux generated from a complete and clash-free completion tree (Horrocks et al., 2000).

Definition 6. Let $\mathbb{N}$ denote the non-negative integers and $\mathbb{N}^*$ 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 vice versa. The empty word ε is called the *root*.

A *forest base for* $\mathcal{K}$ is an interpretation $\mathcal{J} = (\Delta^\mathcal{J}, \cdot^\mathcal{J})$ that interprets transitive roles in an unrestricted (i.e., not necessarily transitive) way and, additionally, satisfies the following conditions:

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

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

T3 for each $a \in \mathsf{Inds}(\mathcal{A})$, $a^{\mathcal{J}} = (a, \varepsilon)$;

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

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

In this case, we say that $\mathcal{J}$ is a forest base *for* $\mathcal{I}$ and if $\mathcal{I} \models \mathcal{K}$ we say that $\mathcal{I}$ is a *canonical model for* $\mathcal{K}$. △

For convenience, we extend the notion of successors and neighbors to elements in canonical models. Let $\mathcal{I}$ be a canonical model with $(a, w), (a', w') \in \Delta^{\mathcal{I}}$. We call (a', w') a *successor* of (a, w) if either $a = a'$ and $w' = w \cdot c$ for some $c \in \mathbb{N}$ or $w = w' = \varepsilon$. We call (a', w') a *neighbor* of (a, w) if (a', w') is a successor of (a, w) or vice versa.

Please note that the above definition implicitly relies on the unique name assumption (UNA) (cf. T3). This is w.l.o.g. as we can guess an appropriate partition among the individual names and replace the individual names in each partition with one representative individual name from that partition. In Section 6, we show how the partitioning of individual names can be used to simulate the UNA, hence, our decision procedure does not rely on the UNA. We also show that this does not affect the complexity.

Lemma 7. *Let $\mathcal{K}$ be a $\mathcal{SHIQ}$ knowledge base and $q = q_1 \vee \ldots \vee q_n$ a union of conjunctive queries. Then $\mathcal{K} \not\models q$ iff there exists a canonical model $\mathcal{I}$ of $\mathcal{K}$ such that $\mathcal{I} \not\models q$.*

A detailed proof is given in the appendix. Informally, for the only if direction, we can take an arbitrary counter-model for the query, which exists by assumption, and "unravel" all non-tree structures. Since, during the unraveling process, we only replace cycles in the model by infinite paths and leave the interpretation of concepts unchanged, the query is still not satisfied in the unravelled canonical model. The if direction of the proof is trivial.

4.2 The Running Example

We use the following Boolean query and knowledge base as a running example:

Example 8. *Let $\mathcal{K} = (\mathcal{T}, \mathcal{R}, \mathcal{A})$ be a $\mathcal{SHIQ}$ knowledge base with $r, t \in N_{tR}$, $k \in \mathbb{N}$*

$$\begin{array}{rll}
\mathcal{T} = & \{ & C_k \sqsubseteq \geqslant k\ p.\top, \\
 & & C_3 \sqsubseteq \geqslant 3\ p.\top, \\
 & & D_2 \sqsubseteq \exists s^-.\top \sqcap \exists t.\top \\
 & \} & \\
\mathcal{R} = & \{ & t \sqsubseteq t^-, \\
 & & s^- \sqsubseteq r \\
 & \} & \\
\mathcal{A} = & \{ & r(a, b), \\
 & & (\exists p.C_k \sqcap \exists p.C \sqcap \exists r^-.C_3)(a), \\
 & & (\exists p.D_1 \sqcap \exists r.D_2)(b) \\
 & \} &
\end{array}$$

and $q = \{r(u, x), r(x, y), t(y, y), s(z, y), r(u, z)\}$ with $\mathsf{Inds}(q) = \emptyset$ and $\mathsf{Vars}(q) = \{u, x, y, z\}$.

For simplicity, we choose to use a CQ instead of a UCQ. In case of a UCQ, the rewriting steps are applied to each disjunct separately.

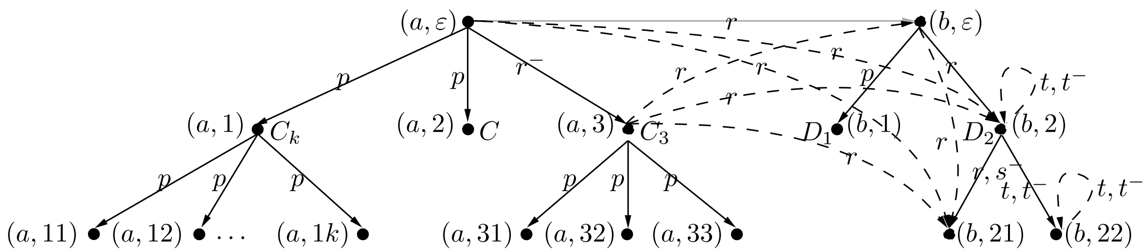

Figure 1: A representation of a canonical interpretation $\mathcal{I}$ for $\mathcal{K}$.

Figure 1 shows a representation of a canonical model $\mathcal{I}$ for the knowledge base $\mathcal{K}$ from Example 8. Each labeled node represents an element in the domain, e.g., the individual name a is represented by the node labeled (a, ε). The edges represent relationships between individuals. For example, we can read the r-labeled edge from (a, ε) to (b, ε) in both directions, i.e., $(a^\mathcal{I}, b^\mathcal{I}) = ((a, \varepsilon), (b, \varepsilon)) \in r^\mathcal{I}$ and $(b^\mathcal{I}, a^\mathcal{I}) = ((b, \varepsilon), (a, \varepsilon)) \in r^{-\mathcal{I}}$. The "short-cuts" due to transitive roles are shown as dashed lines, while the relationship between the nodes that represent ABox individuals is shown in grey. Please note that we did not indicate the interpretations of all concepts in the figure.

Since $\mathcal{I}$ is a canonical model for $\mathcal{K}$, the elements of the domain are pairs (a, w), where a indicates the individual name that corresponds to the root of the tree, i.e., $a^\mathcal{I} = (a, \varepsilon)$ and the elements in the second place form a tree according to our definition of trees. For each individual name a in our ABox, we can, therefore, easily define the tree rooted in a as $\{w \mid (a, w) \in \Delta^\mathcal{I}\}$.

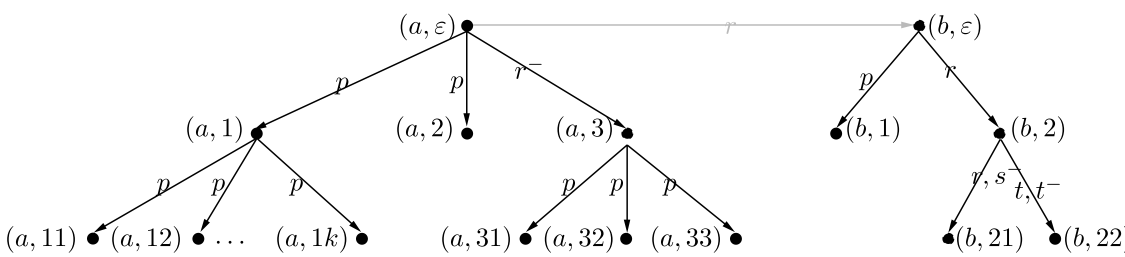

Figure 2: A forest base for the interpretation represented by Figure 1.

Figure 2 shows a representation of a forest base for the interpretation from Figure 1 above. For simplicity, the interpretation of concepts is no longer shown. The two trees, rooted in (a, ε) and (b, ε) respectively, are now clear.

A graphical representation of the query q from Example 8 is shown in Figure 3, where the meaning of the nodes and edges is analogous to the ones given for interpretations. We call this query a cyclic query since its underlying undirected graph is cyclic (cf. Definition 2).

Figure 4 shows a match π for q and $\mathcal{I}$ and, although we consider only one canonical model here, it is not hard to see that the query is true in each model of the knowledge base, i.e., $\mathcal{K} \models q$.

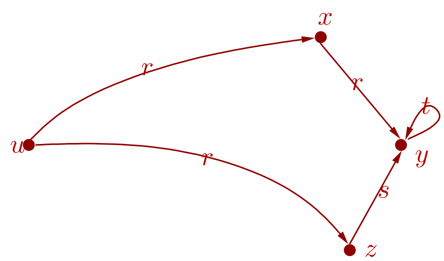

Figure 3: A graph representation of the query from Example 8.

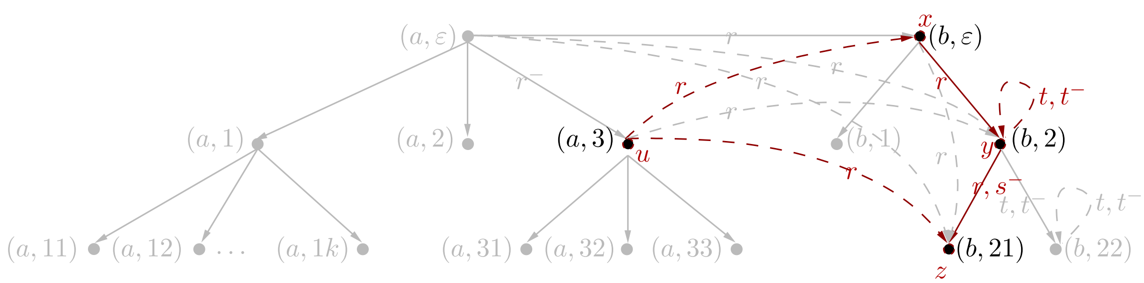

Figure 4: A match π for the query q from Example 8 onto the model $\mathcal{I}$ from Figure 1.

The forest model property is also exploited in the query rewriting process. We want to rewrite q into a set of queries $q_1, \ldots, q_n$ of ground or tree-shaped queries such that $\mathcal{K} \models q$ iff $\mathcal{K} \models q_1 \vee \ldots \vee q_n$. Since the resulting queries are ground or tree-shaped queries, we can explore the known techniques for deciding entailment of these queries. As a first step, we transform q into a set of forest-shaped queries. Intuitively, forest-shaped queries consist of a set of tree-shaped sub-queries, where the roots of these trees might be arbitrarily interconnected (by atoms of the form $r(t, t')$). A tree-shaped query is a special case of a forest-shaped query. We will call the arbitrarily interconnected terms of a forest-shaped query the root choice (or, for short, just roots). At the end of the rewriting process, we replace the roots with individual names from $\mathsf{Inds}(\mathcal{A})$ and transform the tree parts into a concept by applying the so called rolling-up or tuple graph technique (Tessaris, 2001; Calvanese et al., 1998a).

In the proof of the correctness of our procedure, we use the structure of the forest bases in order to explicate the transitive "short-cuts" used in the query match. By explicating we mean that we replace each role atom that is mapped to such a short-cut with a sequence of role atoms such that an extended match for the modified query uses only paths that are in the forest base.

4.3 The Rewriting Steps

The rewriting process for a query q is a six stage process. At the end of this process, the rewritten query may or may not be in a forest shape. As we show later, this "don't know" non-determinism does not compromise the correctness of the algorithm. In the first stage, we derive a *collapsing* q_{co} of q by adding (possibly several) equality atoms to q. Consider, for example, the cyclic query $q = \{r(x, y), r(x, y'), s(y, z), s(y', z)\}$ (see Figure 5), which can be transformed into a tree-shaped one by adding the equality atom $y \approx y'$.

Figure 5: A representation of a cyclic query and of the tree-shaped query obtained by adding the atom $y \approx y'$ to the query depicted on the left hand side.

A common property of the next three rewriting steps is that they allow for substituting the implicit short-cut edges with explicit paths that induce the short-cut. The three steps aim at different cases in which these short-cuts can occur and we describe their goals and application now in more detail:

The second stage is called *split rewriting*. In a split rewriting we take care of all role atoms that are matched to transitive "short-cuts" connecting elements of two different trees and by-passing one or both of their roots. We substitute these short-cuts with either one or two role atoms such that the roots are included. In our running example, π maps u to $(a, 3)$ and x to (b, ε). Hence $\mathcal{I} \models^\pi r(u, x)$, but the used r-edge is a transitive short-cut connecting the tree rooted in a with the tree rooted in b, and by-passing (a, ε). Similar arguments hold for the atom $r(u, z)$, where the path that implies this short-cut relationship goes via the two roots (a, ε) and (b, ε). It is clear that r must be a non-simple role since, in the forest base $\mathcal{J}$ for $\mathcal{I}$, there is no "direct" connection between different trees other than between the roots of the trees. Hence, $(\pi(u), \pi(x)) \in r^\mathcal{I}$ holds only because there is a role $s \in \mathsf{Trans}_\mathcal{R}$ such that $s \mathrel{\underline{\circledast}}_\mathcal{R} r$. In case of our example, r itself is transitive. A split rewriting eliminates transitive short-cuts between different trees of a canonical model and adds the "missing" variables and role atoms matching the sequence of edges that induce the short-cut.

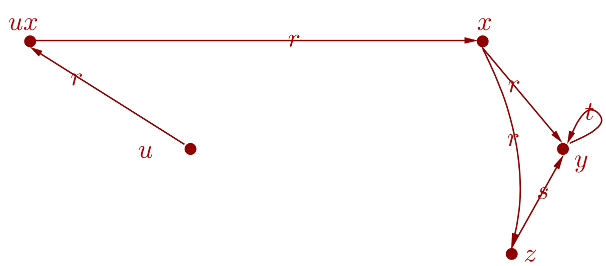

Figure 6: A split rewriting q_{sr} for the query shown in Figure 3.

Figure 6 depicts the split rewriting

$$\begin{aligned} q_{sr} = \{ & r(u, ux), r(ux, x), r(x, y), t(y, y), s(z, y), \\ & r(u, ux), r(ux, x), r(x, z)\} \end{aligned}$$

of q that is obtained from q by replacing (i) $r(u, x)$ with $r(u, ux)$ and $r(ux, x)$ and (ii) $r(u, z)$ with $r(u, ux), r(ux, x)$, and $r(x, z)$. Please note that we both introduced a new variable (ux) and re-used an existing variable (x). Figure 7 shows a match for q_{sr} and the canonical model $\mathcal{I}$ of $\mathcal{K}$ in which the two trees are only connected via the roots. For the rewritten query, we also guess a set of roots, which contains the variables that are mapped to the roots in the canonical model. For our running example, we guess that the set of roots is $\{ux, x\}$.

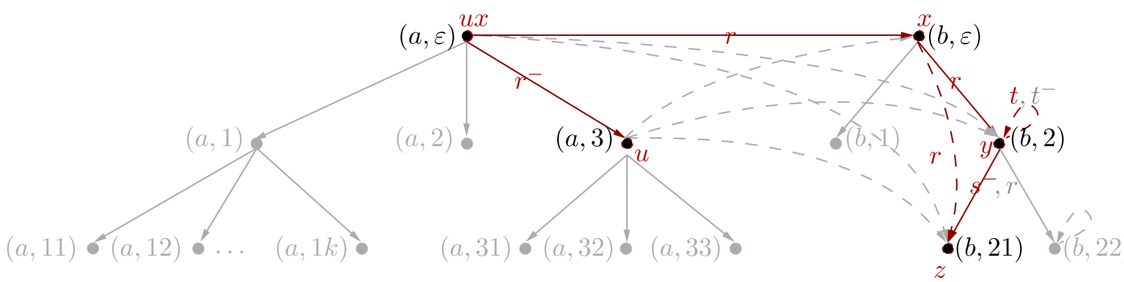

Figure 7: A split match π_{sr} for the query q_{sr} from Figure 6 onto the canonical interpretation from Figure 1.

In the third step, called *loop rewriting*, we eliminate "loops" for variables v that do not correspond to roots by replacing atoms $r(v, v)$ with two atom $r(v, v')$ and $r(v', v)$, where v' can either be a new or an existing variable in q. In our running example, we eliminate the loop $t(y, y)$ as follows:

$$\begin{aligned} q_{\ell r} = \{\, & r(u, ux), r(ux, x), r(x, y), t(y, y'), t(y', y), s(z, y), \\ & r(u, ux), r(ux, x), r(x, z)\} \end{aligned}$$

is the query obtained from q_{sr} (see Figure 6) by replacing $t(y, y)$ with $t(y, y')$ and $t(y', y)$ for a new variable y'. Please note that, since t is defined as transitive and symmetric, $t(y, y)$ is still implied, i.e., the loop is also a transitive short-cut. Figure 8 shows the canonical interpretation $\mathcal{I}$ from Figure 1 with a match $\pi_{\ell r}$ for $q_{\ell r}$. The introduction of the new variable y' is needed in this case since there is no variable that could be re-used and the individual $(b, 22)$ is not in the range of the match π_{sr}.

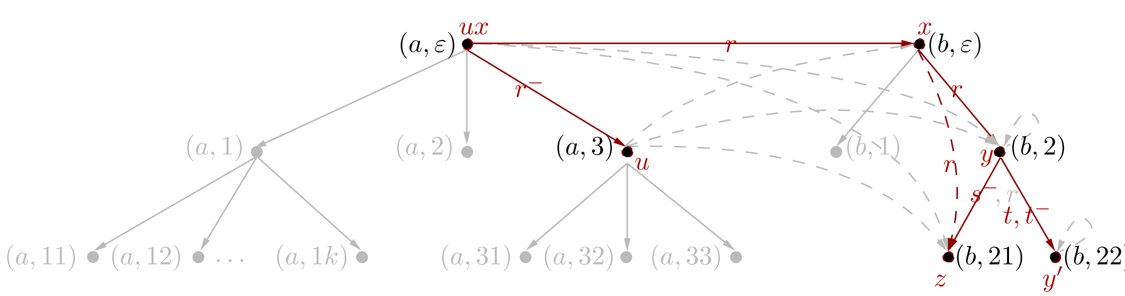

Figure 8: A loop rewriting $q_{\ell r}$ and a match for the canonical interpretation from Figure 1.

The forth rewriting step, called *forest rewriting*, allows again the replacement of role atoms with sets of role atoms. This allows the elimination of cycles that are within a single tree. A forest rewriting q_{fr} for our example can be obtained from $q_{\ell r}$ by replacing the role atom $r(x, z)$ with $r(x, y)$ and $r(y, z)$, resulting in the query

$$\begin{aligned} q_{fr} = \{ & r(u, ux), r(ux, x), r(x, y), t(y, y'), t(y', y), s(z, y), \\ & r(u, ux), r(ux, x), r(x, y), r(y, z)\}. \end{aligned}$$

Clearly, this results in tree-shaped sub-queries, one rooted in ux and one rooted in x. Hence q_{fr} is forest-shaped w.r.t. the root terms ux and x. Figure 9 shows the canonical interpretation $\mathcal{I}$ from Figure 1 with a match π_{fr} for q_{fr}.

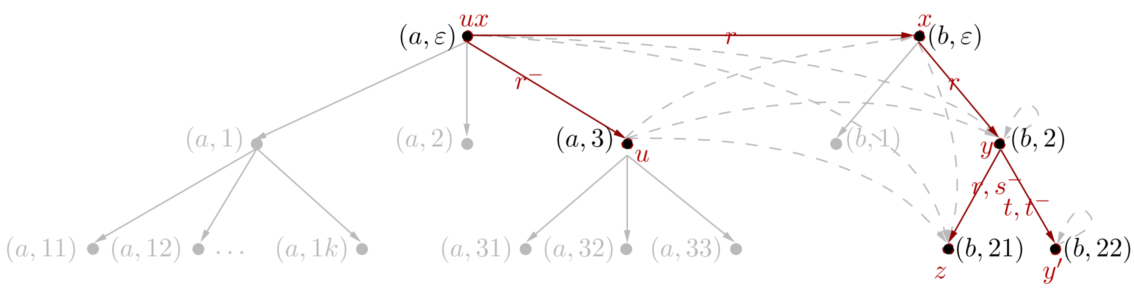

Figure 9: A forest rewriting q_{fr} and a forest match π_{fr} for the canonical interpretation from Figure 1.

In the fifth step, we use the standard rolling-up technique (Horrocks & Tessaris, 2000; Calvanese et al., 1998a) and express the tree-shaped sub-queries as concepts. In order to do this, we traverse each tree in a bottom-up fashion and replace each leaf (labeled with a concept C, say) and its incoming edge (labeled with a role r, say) with the concept $\exists r.C$ added to its predecessor. For example, the tree rooted in ux (i.e., the role atom $r(u, ux)$) can be replaced with the atom $(\exists r^-.\top)(ux)$. Similarly, the tree rooted in x (i.e., the role atoms $r(x, y), r(y, z), s(z, y), t(y, y')$, and $t(y', y)$) can be replaced with the atom

$$(\exists r.((\exists(r \sqcap \mathsf{Inv}(s)).\top) \sqcap (\exists(t \sqcap \mathsf{Inv}(t)).\top))(x).$$

Please note that we have to use role conjunctions in the resulting query in order to capture the semantics of multiple role atoms relating the same pair of variables.

Recall that, in the split rewriting, we have guessed that x and ux correspond to roots and, therefore, correspond to individual names in $\mathsf{Inds}(\mathcal{A})$. In the sixth and last rewriting step, we guess which variable corresponds to which individual name and replace the variables with the guessed names. A possible guess for our running example would be that ux corresponds to a and x to b. This results in the (ground) query

$$\{(\exists r^-.\top)(a), r(a, b), (\exists r.((\exists(r \sqcap \mathsf{Inv}(s)).\top) \sqcap (\exists(t \sqcap \mathsf{Inv}(t)).\top)))(b)\},$$

which is entailed by $\mathcal{K}$.

Please note that we focused in the running example on the most reasonable rewriting. There are several other possible rewritings, e.g., we obtain another rewriting from q_{fr} by replacing ux with b and x with a in the last step. For a UCQ, we apply the rewriting steps to each of the disjuncts separately.

At the end of the rewriting process, we have, for each disjunct, a set of ground queries and/or queries that were rolled-up into a single concept atom. The latter queries result from forest rewritings that are tree-shaped and have an empty set of roots. Such tree-shaped rewritings can match anywhere in a tree and can, thus, not be grounded. Finally, we check if our knowledge base entails the disjunction of all the rewritten queries. We show that there is a bound on the number of (forest-shaped) rewritings and hence on the number of queries produced in the rewriting process.

Summing up, the rewriting process for a connected conjunctive query q involves the following steps:

1. Build all collapsings of q.
2. Build all split rewritings of each collapsing w.r.t. a subset R of roots.
3. Build all loop rewritings of the split rewritings.
4. Build all (forest-shaped) forest rewritings of the loop rewritings.
5. Roll up each tree-shaped sub-query in a forest-rewriting into a concept atom and
6. replace the roots in R with individual names from the ABox in all possible ways.

Let $q_1, \dots, q_n$ be the queries resulting from the rewriting process. In the next section, we define each rewriting step and prove that $\mathcal{K} \models q$ iff $\mathcal{K} \models q_1 \vee \dots \vee q_n$. Checking entailment for the rewritten queries can easily be reduced to KB consistency and any decision procedure for $\mathcal{SHIQ}^\sqcap$ KB consistency could be used in order to decide if $\mathcal{K} \models q$. We present one such decision procedure in Section 6.

5. Query Rewriting

In the previous section, we have used several terms, e.g., tree- or forest-shaped query, rather informally. In the following, we give definitions for the terms used in the query rewriting process. Once this is done, we formalize the query rewriting steps and prove the correctness of the procedure, i.e., we show that the forest-shaped queries obtained in the rewriting process can indeed be used for deciding whether a knowledge base entails the original query. We do not give the detailed proofs here, but rather some intuitions behind the proofs. Proofs in full detail are given in the appendix.

5.1 Tree- and Forest-Shaped Queries

In order to define tree- or forest-shaped queries more precisely, we use mappings between queries and trees or forests. Instead of mapping equivalence classes of terms by $\stackrel{*}{\approx}$ to nodes in a tree, we extend some well-known properties of functions as follows:

Definition 9. For a mapping $f\colon A \to B$, we use $\mathsf{dom}(f)$ and $\mathsf{ran}(f)$ to denote f's *domain* A and *range* B, respectively. Given an equivalence relation $\stackrel{*}{\approx}$ on $\mathsf{dom}(f)$, we say that f is *injective modulo* $\stackrel{*}{\approx}$ if, for all $a, a' \in \mathsf{dom}(f)$, $f(a) = f(a')$ implies $a \stackrel{*}{\approx} a'$ and we say that f is *bijective modulo* $\stackrel{*}{\approx}$ if f is injective modulo $\stackrel{*}{\approx}$ and surjective. Let q be a query. A *tree mapping* for q is a total function f from terms in q to a tree such that

1. f is bijective modulo $\stackrel{*}{\approx}$,
2. if $r(t, t') \mathrel{\bar{\in}} q$, then $f(t)$ is a neighbor of $f(t')$, and,
3. if $a \in \mathsf{Inds}(q)$, then $f(a) = \varepsilon$.

The query q is *tree-shaped* if $\sharp(\mathsf{Inds}(q)) \leq 1$ and there is a tree mapping for q.

A *root choice* R for q is a subset of $\mathsf{Terms}(q)$ such that $\mathsf{Inds}(q) \subseteq R$ and, if $t \in R$ and $t \stackrel{*}{\approx} t'$, then $t' \in R$. For $t \in R$, we use $\mathsf{Reach}(t)$ to denote the set of terms $t' \in \mathsf{Terms}(q)$ for which there exists a sequence of terms $t_1, \ldots, t_n \in \mathsf{Terms}(q)$ such that

1. $t_1 = t$ and $t_n = t'$,
2. for all $1 \leq i < n$, there is a role r such that $r(t_i, t_{i+1}) \mathrel{\bar{\in}} q$, and,
3. for $1 < i \leq n$, if $t_i \in R$, then $t_i \stackrel{*}{\approx} t$.

We call R a *root splitting w.r.t.* q if either $R = \emptyset$ or if, for $t_i, t_j \in R$, $t_i \stackrel{*}{\not\approx} t_j$ implies that $\mathsf{Reach}(t_i) \cap \mathsf{Reach}(t_j) = \emptyset$. Each term $t \in R$ induces a sub-query

$$\begin{aligned} \mathsf{subq}(q, t) := \quad & \{at \mathrel{\bar{\in}} q \mid \text{ the terms in } at \text{ occur in } \mathsf{Reach}(t)\} \setminus \\ & \{r(t, t) \mid r(t, t) \mathrel{\bar{\in}} q\}. \end{aligned}$$

A query q is *forest-shaped w.r.t. a root splitting* R if either $R = \emptyset$ and q is tree-shaped or each sub-query $\mathsf{subq}(q, t)$ for $t \in R$ is tree-shaped. △

For each term $t \in R$, we collect the terms that are reachable from t in the set $\mathsf{Reach}(t)$. By Condition 3, we make sure that R and $\stackrel{*}{\approx}$ are such that each $t' \in \mathsf{Reach}(t)$ is either not in R or $t \stackrel{*}{\approx} t'$. Since queries are connected by assumption, we would otherwise collect all terms in $\mathsf{Reach}(t)$ and not just those $t' \notin R$. For a root splitting, we require that the resulting sets are mutually disjoint for all terms $t, t' \in R$ that are not equivalent. This guarantees that all paths between the sub-queries go via the root nodes of their respective trees. Intuitively, a forest-shaped query is one that can potentially be mapped onto a canonical interpretation $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ such that the terms in the root splitting R correspond to roots $(a, \varepsilon) \in \Delta^{\mathcal{I}}$. In the definition of $\mathsf{subq}(q, t)$, we exclude loops of the form $r(t, t) \mathrel{\bar{\in}} q$, as these parts of the query are grounded later in the query rewriting process and between ground terms, we allow arbitrary relationships.

Consider, for example, the query q_{sr} of our running example from the previous section (cf. Figure 6). Let us again make the root choice $R := \{ux, x\}$ for q. The sets $\mathsf{Reach}(ux)$ and $\mathsf{Reach}(x)$ w.r.t. q_{sr} and R are $\{ux, u\}$ and $\{x, y, z\}$ respectively. Since both sets are disjoint, R is a root splitting w.r.t. q_{sr}. If we choose, however, $R := \{x, y\}$, the set R is not a root splitting w.r.t. q_{sr} since $\mathsf{Reach}(x) = \{ux, u, z\}$ and $\mathsf{Reach}(y) = \{z\}$ are not disjoint.

5.2 From Graphs to Forests

We are now ready to define the query rewriting steps. Given an arbitrary query, we exhaustively apply the rewriting steps and show that we can use the resulting queries that are forest-shaped for deciding entailment of the original query. Please note that the following definitions are for conjunctive queries and not for unions of conjunctive queries since we apply the rewriting steps for each disjunct separately.

Definition 10. Let q be a Boolean conjunctive query. A *collapsing* q_{co} of q is obtained by adding zero or more equality atoms of the form $t \approx t'$ for $t, t' \in \mathsf{Terms}(q)$ to q. We use $\mathsf{co}(q)$ to denote the set of all queries that are a collapsing of q.

Let $\mathcal{K}$ be a $\mathcal{SHIQ}$ knowledge base. A query q_{sr} is called a *split rewriting* of q w.r.t. $\mathcal{K}$ if it is obtained from q by choosing, for each atom $r(t, t') \bar{\in} q$, to either:

1. do nothing,
2. choose a role $s \in \mathsf{Trans}_\mathcal{R}$ such that $s \,\underline{\circledast}_\mathcal{R}\, r$ and replace $r(t, t')$ with $s(t, u), s(u, t')$, or
3. choose a role $s \in \mathsf{Trans}_\mathcal{R}$ such that $s \,\underline{\circledast}_\mathcal{R}\, r$ and replace $r(t, t')$ with $s(t, u)$, $s(u, u')$, $s(u', t')$,

where $u, u' \in N_V$ are possibly fresh variables. We use $\mathsf{sr}_\mathcal{K}(q)$ to denote the set of all pairs (q_{sr}, R) for which there is a query $q_{co} \in \mathsf{co}(q)$ such that q_{sr} is a split rewriting of q_{co} and R is a root splitting w.r.t. q_{sr}.

A query $q_{\ell r}$ is called a *loop rewriting* of q w.r.t. a root splitting R and $\mathcal{K}$ if it is obtained from q by choosing, for all atoms of the form $r(t, t) \bar{\in} q$ with $t \notin R$, a role $s \in \mathsf{Trans}_\mathcal{R}$ such that $s \,\underline{\circledast}_\mathcal{R}\, r$ and by replacing $r(t, t)$ with two atoms $s(t, t')$ and $s(t', t)$ for $t' \in N_V$ a possibly fresh variable. We use $\mathsf{lr}_\mathcal{K}(q)$ to denote the set of all pairs $(q_{\ell r}, R)$ for which there is a tuple $(q_{sr}, R) \in \mathsf{sr}_\mathcal{K}(q)$ such that $q_{\ell r}$ is a loop rewriting of q_{sr} w.r.t. R and $\mathcal{K}$.

For a forest rewriting, fix a set $V \subseteq N_V$ of variables not occurring in q such that $\sharp(V) \leq \sharp(\mathsf{Vars}(q))$. A *forest rewriting* q_{fr} w.r.t. a root splitting R of q and $\mathcal{K}$ is obtained from q by choosing, for each role atom $r(t, t')$ such that either $R = \emptyset$ and $r(t, t') \bar{\in} q$ or there is some $t_r \in R$ and $r(t, t') \bar{\in} \mathsf{subq}(q, t_r)$ to either

1. do nothing, or
2. choose a role $s \in \mathsf{Trans}_\mathcal{R}$ such that $s \,\underline{\circledast}_\mathcal{R}\, r$ and replace $r(t, t')$ with $\ell \leq \sharp(\mathsf{Vars}(q))$ role atoms $s(t_1, t_2), \ldots, s(t_\ell, t_{\ell+1})$, where $t_1 = t$, $t_{\ell+1} = t'$, and $t_2, \ldots, t_\ell \in \mathsf{Vars}(q) \cup V$.

We use $\mathsf{fr}_\mathcal{K}(q)$ to denote the set of all pairs (q_{fr}, R) for which there is a tuple $(q_{\ell r}, R) \in \mathsf{lr}_\mathcal{K}(q)$ such that q_{fr} is a forest-shaped forest rewriting of $q_{\ell r}$ w.r.t. R and $\mathcal{K}$. △

If $\mathcal{K}$ is clear from the context, we say that q' is a split, loop, or forest rewriting of q instead of saying that q' is a split, loop, or forest rewriting of q w.r.t. $\mathcal{K}$. We assume that $\mathsf{sr}_\mathcal{K}(q), \mathsf{lr}_\mathcal{K}(q)$, and $\mathsf{fr}_\mathcal{K}(q)$ contain no isomorphic queries, i.e., differences in (newly introduced) variable names only are neglected.

In the next section, we show how we can build a disjunction of conjunctive queries $q_1 \vee \cdots \vee q_\ell$ from the queries in $\mathsf{fr}_\mathcal{K}(q)$ such that each q_i for $1 \leq i \leq \ell$ is either of the form $C(v)$ for a single variable $v \in \mathsf{Vars}(q_i)$ or q_i is ground, i.e., q_i contains only constants and no variables. It then remains to show that $\mathcal{K} \models q$ iff $\mathcal{K} \models q_1 \vee \cdots \vee q_\ell$.

5.3 From Trees to Concepts

In order to transform a tree-shaped query into a single concept atom and a forest-shaped query into a ground query, we define a mapping f from the terms in each tree-shaped subquery to a tree. We then incrementally build a concept that corresponds to the tree-shaped query by traversing the tree in a bottom-up fashion, i.e., from the leaves upwards to the root.

Definition 11. Let q be a tree-shaped query with at most one individual name. If $a \in \mathsf{Inds}(q)$, then let $t_r = a$ otherwise let $t_r = v$ for some variable $v \in \mathsf{Vars}(q)$. Let f be a tree mapping such that $f(t_r) = \varepsilon$. We now inductively assign, to each term $t \in \mathsf{Terms}(q)$, a concept $\mathsf{con}(q, t)$ as follows:

- if $f(t)$ is a leaf of $\mathsf{ran}(f)$, then $\mathsf{con}(q,t) := \bigsqcap_{C(t) \bar{\in} q} C$,
- if $f(t)$ has successors $f(t_1), \ldots, f(t_k)$, then

$$\begin{aligned}\mathsf{con}(q,t) := \;& \textstyle\bigsqcap_{C(t)\bar{\in} q} C \sqcap \\ & \textstyle\bigsqcap_{1 \leq i \leq k} \exists \big(\bigsqcap_{r(t,t_i) \bar{\in} q} r \big).\mathsf{con}(q, t_i).\end{aligned}$$

Finally, the *query concept of* q *w.r.t.* t_r is $\mathsf{con}(q, t_r)$. △

Please note that the above definition takes equality atoms into account. This is because the function f is bijective modulo $\stackrel{*}{\approx}$ and, in case there are concept atoms $C(t)$ and $C(t')$ for $t \stackrel{*}{\approx} t'$, both concepts are conjoined in the query concept due to the use of the relation $\bar{\in}$. Similar arguments can be applied to the role atoms.

The following lemma shows that query concepts indeed capture the semantics of q.

Lemma 12. *Let q be a tree-shaped query with $t_r \in \mathsf{Terms}(q)$ as defined above, $C_q = \mathsf{con}(q, t_r)$, and $\mathcal{I}$ an interpretation. Then $\mathcal{I} \models q$ iff there is a match π and an element $d \in C_q{}^{\mathcal{I}}$ such that $\pi(t_r) = d$.*

The proof given by Horrocks, Sattler, Tessaris, and Tobies (1999) easily transfers from $\mathcal{DLR}$ to $\mathcal{SHIQ}$. By applying the result from the above lemma, we can now transform a forest-shaped query into a ground query as follows:

Definition 13. Let $(q_{fr}, R) \in \mathsf{fr}_{\mathcal{K}}(q)$ for $R \neq \emptyset$, and $\tau \colon R \to \mathsf{Inds}(\mathcal{A})$ a total function such that, for each $a \in \mathsf{Inds}(q)$, $\tau(a) = a$ and, for $t, t' \in R$, $\tau(t) = \tau(t')$ iff $t \stackrel{*}{\approx} t'$. We call such a mapping τ a *ground mapping for R w.r.t. $\mathcal{A}$*. We obtain a ground query $\mathsf{ground}(q_{fr}, R, \tau)$ of q_{fr} w.r.t. the root splitting R and ground mapping τ as follows:

- replace each $t \in R$ with $\tau(t)$, and,
- for each $a \in \mathsf{ran}(\tau)$, replace the sub-query $q_a = \mathsf{subq}(q_{fr}, a)$ with $\mathsf{con}(q_a, a)$.

We define the set $\mathsf{ground}_{\mathcal{K}}(q)$ of *ground queries for q* w.r.t. $\mathcal{K}$ as follows:

$$\begin{aligned}\mathsf{ground}_{\mathcal{K}}(q) := \{q' \mid\; & \text{there exists some } (q_{fr}, R) \in \mathsf{fr}_{\mathcal{K}}(q) \text{ with } R \neq \emptyset \\ & \text{and some ground mapping } \tau \text{ w.r.t. } \mathcal{A} \text{ and } R \\ & \text{such that } q' = \mathsf{ground}(q_{fr}, R, \tau)\}\end{aligned}$$

We define the set of $\mathsf{trees}_{\mathcal{K}}(q)$ of *tree queries for q* as follows:

$$\begin{aligned}\mathsf{trees}_{\mathcal{K}}(q) := \{q' \mid\; & \text{there exists some } (q_{fr}, \emptyset) \in \mathsf{fr}_{\mathcal{K}}(q) \text{ and} \\ & v \in \mathsf{Vars}(q_{fr}) \text{ such that } q' = (\mathsf{con}(q_{fr}, v))(v)\}\end{aligned}$$

△

Going back to our running example, we have already seen that $(q_{fr}, \{ux, x\})$ belongs to the set $\mathsf{fr}_{\mathcal{K}}(q)$ for

$$q_{fr} = \{r(u, ux), r(ux, x), r(x, y), t(y, y'), t(y', y), s(z, y), r(y, z)\}.$$

There are also several other queries in the set $\mathsf{fr}_{\mathcal{K}}(q)$, e.g., $(q, \{u, x, y, z\})$, where q is the original query and the root splitting R is such that $R = \mathsf{Terms}(q)$, i.e., all terms are in the root choice for q. In order to build the set $\mathsf{ground}_{\mathcal{K}}(q)$, we now build all possible ground mappings τ for the set $\mathsf{Inds}(\mathcal{A})$ of individual names in our ABox and the root splittings for the queries in $\mathsf{fr}_{\mathcal{K}}(q)$. The tuple $(q_{fr}, \{ux, x\}) \in \mathsf{fr}_{\mathcal{K}}(q)$ contributes two ground queries for the set $\mathsf{ground}_{\mathcal{K}}(q)$:

$$\begin{array}{l}\mathsf{ground}(q_{fr}, \{ux, x\}, \{ux \mapsto a, x \mapsto b\}) = \\ \{r(a, b), (\exists \mathsf{Inv}(r).\top)(a), (\exists r.((\exists (r \sqcap \mathsf{Inv}(s)).\top) \sqcap (\exists (t \sqcap \mathsf{Inv}(t)).\top)))(b)\},\end{array}$$

where $\exists \mathsf{Inv}(r).\top$ is the query concept for the (tree-shaped) sub-query $\mathsf{subq}(q_{fr}, ux)$ and $\exists r.((\exists (r \sqcap \mathsf{Inv}(s)).\top) \sqcap (\exists (t \sqcap \mathsf{Inv}(t)).\top)$ is the query concept for $\mathsf{subq}(q_{fr}, x)$ and

$$\begin{array}{l}\mathsf{ground}(q_{fr}, \{ux, x\}, \{ux \mapsto b, x \mapsto a\}) = \\ \{r(b, a), (\exists \mathsf{Inv}(r).\top)(b), (\exists r.((\exists (r \sqcap \mathsf{Inv}(s)).\top) \sqcap (\exists (t \sqcap \mathsf{Inv}(t)).\top)))(a)\}.\end{array}$$

The tuple $(q, \{u, x, y, z\}) \in \mathsf{fr}_{\mathcal{K}}(q)$, however, does not contribute a ground query since, for a ground mapping, we require that $\tau(t) = \tau(t')$ iff $t \overset{*}{\approx} t'$ and there are only two individual names in $\mathsf{Inds}(\mathcal{A})$ compared to four terms q that need a distinct value. Intuitively, this is not a restriction, since in the first rewriting step (collapsing) we produce all those queries in which the terms of q have been identified with each other in all possible ways. In our example, $\mathcal{K} \models q$ and $\mathcal{K} \models q_1 \vee \cdots \vee q_\ell$, where $q_1 \vee \cdots \vee q_\ell$ are the queries from $\mathsf{trees}_{\mathcal{K}}(q)$ and $\mathsf{ground}_{\mathcal{K}}(q)$ since each model $\mathcal{I}$ of $\mathcal{K}$ satisfies $q_i = \mathsf{ground}(q_{fr}, \{ux, x\}, \{ux \mapsto a, x \mapsto b\})$.

5.4 Query Matches

Even if a query is true in a canonical model, it does not necessarily mean that the query is tree- or forest-shaped. However, a match π for a canonical interpretation can guide the process of rewriting a query. Similarly to the definition of tree- or forest-shaped queries, we define the shape of matches for a query. In particular, we introduce three different kinds of matches: *split matches, forest matches*, and *tree matches* such that every tree match is a forest match, and every forest match is a split match. The correspondence to the query shapes is as follows: given a split match π, the set of all root nodes (a, ε) in the range of the match define a root splitting for the query, if π is additionally a forest match, the query is forest-shaped w.r.t. the root splitting induced by π, and if π is additionally a tree match, then the whole query can be mapped to a single tree (i.e., the query is tree-shaped or forest-shaped w.r.t. an empty root splitting). Given an arbitrary query match into a canonical model, we can first obtain a split match and then a tree or forest match, by using the structure of the canonical model for guiding the application of the rewriting steps.

Definition 14. Let $\mathcal{K}$ be a $\mathcal{SHIQ}$ knowledge base, q a query, $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ a canonical model of $\mathcal{K}$, and $\pi\colon \mathsf{Terms}(q) \to \Delta^{\mathcal{I}}$ an evaluation such that $\mathcal{I} \models^{\pi} q$. We call π a *split match* if, for all $r(t, t') \mathrel{\bar{\in}} q$, one of the following holds:

1. $\pi(t) = (a, \varepsilon)$ and $\pi(t') = (b, \varepsilon)$ for some $a, b \in \mathsf{Inds}(\mathcal{A})$; or

2. $\pi(t) = (a, w)$ and $\pi(t') = (a, w')$ for some $a \in \mathsf{Inds}(\mathcal{A})$ and $w, w' \in \mathbb{N}^*$.

We call π a *forest match* if, additionally, for each term $t_r \in \mathsf{Terms}(q)$ with $\pi(t_r) = (a, \varepsilon)$ and $a \in \mathsf{Inds}(\mathcal{A})$, there is a total and bijective mapping f from $\{(a, w) \mid (a, w) \in \mathsf{ran}(\pi)\}$ to a tree T such that $r(t, t') \bar{\in} \mathsf{subq}(q, t_r)$ implies that $f(\pi(t))$ is a neighbor of $f(\pi(t'))$. We call π a *tree match* if, additionally, there is an $a \in \mathsf{Inds}(\mathcal{A})$ such that each element in $\mathsf{ran}(\pi)$ is of the form (a, w).

A split match π for a canonical interpretation induces a (possibly empty) root splitting R such that $t \in R$ iff $\pi(t) = (a, \varepsilon)$ for some $a \in \mathsf{Inds}(\mathcal{A})$. We call R the *root splitting induced by* π. △

For two elements (a, w) and (a, w') in a canonical model, the path from (a, w) to (a, w') is the sequence $(a, w_1), \ldots, (a, w_n)$ where $w = w_1, w' = w_n$, and, for $1 \leq i < n, w_{i+1}$ is a successor of w_i. The length of the path is n. Please note that, for a forest match, we do not require that w is a neighbor of w' or vice versa. This still allows to map role atoms to paths in the canonical model of length greater than two, but such paths must be between ancestors and not between elements in different branches of the tree. The mapping f to a tree also makes sure that if R is the induced root splitting, then each sub-query $\mathsf{subq}(q, t)$ for $t \in R$ is tree-shaped. For a tree match, the root splitting is either empty or $t \stackrel{*}{\approx} t'$ for each $t, t' \in R$, i.e., there is a single root modulo $\stackrel{*}{\approx}$, and the whole query is tree-shaped.

5.5 Correctness of the Query Rewriting

The following lemmas state the correctness of the rewriting step by step for each of the rewriting stages. Full proofs are given in the appendix. As motivated in the previous section, we can use a given canonical model to guide the rewriting process such that we obtain a forest-shaped query that also has a match into the model.

Lemma 15. *Let $\mathcal{I}$ be a model for $\mathcal{K}$.*

1. *If $\mathcal{I} \models q$, then there is a collapsing q_{co} of q such that $\mathcal{I} \models^{\pi_{co}} q_{co}$ for π_{co} an injection modulo $\stackrel{*}{\approx}$.*

2. *If $\mathcal{I} \models^{\pi_{co}} q_{co}$ for a collapsing q_{co} of q, then $\mathcal{I} \models q$.*

Given a model $\mathcal{I}$ that satisfies q, we can simply add equality atoms for all pairs of terms that are mapped to the same element in $\mathcal{I}$. It is not hard to see that this results in a mapping that is injective modulo $\stackrel{*}{\approx}$. For the second part, it is easy to see that a model that satisfies a collapsing also satisfies the original query.

Lemma 16. *Let $\mathcal{I}$ be a model for $\mathcal{K}$.*

1. *If $\mathcal{I}$ is canonical and $\mathcal{I} \models^{\pi} q$, then there is a pair $(q_{sr}, R) \in \mathsf{sr}_{\mathcal{K}}(q)$ and a split match π_{sr} such that $\mathcal{I} \models^{\pi_{sr}} q_{sr}$, R is the induced root splitting of π_{sr}, and π_{sr} is an injection modulo $\stackrel{*}{\approx}$.*

2. *If $(q_{sr}, R) \in \mathsf{sr}_{\mathcal{K}}(q)$ and $\mathcal{I} \models^{\pi_{sr}} q_{sr}$ for some match π_{sr}, then $\mathcal{I} \models q$.*

For the first part of the lemma, we proceed exactly as illustrated in the example section and use the canonical model $\mathcal{I}$ and the match π to guide the rewriting steps. We first build a collapsing $q_{co} \in \mathsf{co}(q)$ as described in the proof of Lemma 15 such that $\mathcal{I} \models^{\pi_{co}} q_{co}$ for π_{co} an injection modulo $\stackrel{*}{\approx}$. Since $\mathcal{I}$ is canonical, paths between different trees can only occur due to non-simple roles, and thus we can replace each role atom that uses such a short-cut with two or three role atoms such that these roots are explicitly included in the query (cf. the query and match in Figure 4 and the obtained split rewriting and with a split match in Figure 7). The second part of the lemma follows immediately from the fact that we use only transitive sub-roles in the replacement.

Lemma 17. *Let $\mathcal{I}$ be a model of $\mathcal{K}$.*

1. *If $\mathcal{I}$ is canonical and $\mathcal{I} \models q$, then there is a pair $(q_{\ell r}, R) \in \mathsf{lr}_{\mathcal{K}}(q)$ and a mapping $\pi_{\ell r}$ such that $\mathcal{I} \models^{\pi_{\ell r}} q_{\ell r}$, $\pi_{\ell r}$ is an injection modulo $\stackrel{*}{\approx}$, R is the root splitting induced by $\pi_{\ell r}$ and, for each $r(t,t) \mathrel{\bar{\in}} q_{\ell r}$, $t \in R$.*

2. *If $(q_{\ell r}, R) \in \mathsf{lr}_{\mathcal{K}}(q)$ and $\mathcal{I} \models^{\pi_{\ell r}} q_{\ell r}$ for some match $\pi_{\ell r}$, then $\mathcal{I} \models q$.*

The second part is again straightforward, given that we can only use transitive sub-roles in the loop rewriting. For the first part, we proceed again as described in the examples section and use the canonical model $\mathcal{I}$ and the match π to guide the rewriting process. We first build a split rewriting q_{sr} and its root splitting R as described in the proof of Lemma 16 such that $(q_{sr}, R) \in \mathsf{sr}_{\mathcal{K}}(q)$ and $\mathcal{I} \models^{\pi_{sr}} q_{sr}$ for a split match π_{sr}. Since $\mathcal{I}$ is a canonical model, it has a forest base $\mathcal{J}$. In a forest base, non-root nodes cannot be successors of themselves, so each such loop is a short-cut due to some transitive role. An element that is, say, r-related to itself has, therefore, a neighbor that is both an r- and $\mathsf{Inv}(r)$-successor. Depending on whether this neighbor is already in the range of the match, we can either re-use an existing variable or introduce a new one, when making this path explicit (cf. the loop rewriting depicted in Figure 8 obtained from the split rewriting shown in Figure 7).

Lemma 18. *Let $\mathcal{I}$ be a model of $\mathcal{K}$.*

1. *If $\mathcal{I}$ is canonical and $\mathcal{I} \models q$, then there is a pair $(q_{fr}, R) \in \mathsf{fr}_{\mathcal{K}}(q)$ such that $\mathcal{I} \models^{\pi_{fr}} q_{fr}$ for a forest match π_{fr}, R is the induced root splitting of π_{fr}, and π_{fr} is an injection modulo $\stackrel{*}{\approx}$.*

2. *If $(q_{fr}, R) \in \mathsf{fr}_{\mathcal{K}}(q)$ and $\mathcal{I} \models^{\pi_{fr}} q_{fr}$ for some match π_{fr}, then $\mathcal{I} \models q$.*

The main challenge is again the proof of (1) and we just give a short idea of it here. At this point, we know from Lemma 17 that we can use a query $q_{\ell r}$ for which there is a root splitting R and a split match $\pi_{\ell r}$. Since $\pi_{\ell r}$ is a split match, the match for each such sub-query is restricted to a tree and thus we can transform each sub-query of $q_{\ell r}$ induced by a term t in the root choice separately. The following example is meant to illustrate why the given bound of $\sharp(\mathsf{Vars}(q))$ on the number of new variables and role atoms that can be introduced in a forest rewriting suffices. Figure 10 depicts the representation of a tree from a canonical model, where we use only the second part of the names for the elements, e.g., we use just ε instead of (a, ε). For simplicity, we also do not indicate the concepts and roles that label the nodes and edges, respectively. We use black color to indicate the nodes and edges that are used in the match for a query and dashed lines for short-cuts due to transitive roles. In the example, the grey edges are also those that belong to the forest base and the query match uses only short-cuts.

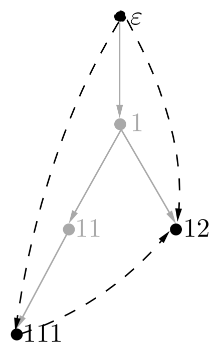

Figure 10: A part of a representation of a canonical model, where the black nodes and edges are used in a match for a query and dashed edges indicate short-cuts due to transitive roles.

The forest rewriting aims at making the short-cuts more explicit by replacing them with as few edges as necessary to obtain a tree match. In order to do this, we need to include the "common ancestors" in the forest base between each two nodes used in the match. For $w, w' \in \mathbb{N}^*$, we therefore define the *longest common prefix* (LCP) of w and w' as the longest $\hat{w} \in \mathbb{N}^*$ such that $\hat{w}$ is a prefix of both w and w'. For a forest rewriting, we now determine the LCPs of any two nodes in the range of the match and add a variable for those LCPs that are not yet in the range of the match to the set V of new variables used in the forest rewriting. In the example from Figure 10 the set V contains a single variable v_1 for the node 1.

We now explicate the short-cuts as follows: for any edge used in the match, e.g., the edge from ε to 111 in the example, we define its *path* as the sequence of elements on the path in the forest base, e.g., the path for the edge from ε to 111 is $\varepsilon, 1, 11, 111$. The *relevant path* is obtained by dropping all elements from the path that are not in the range of the mapping or correspond to a variable in the set V, resulting in a relevant path of $\varepsilon, 1, 111$ for the example. We now replace the role atom that was matched to the edge from ε to 111 with two role atoms such that the match uses the edge from ε to 1 and from 1 to 111. An appropriate transitive sub-role exists since otherwise there could not be a short-cut. Similar arguments can be used to replace the role atom mapped to the edge from 111 to 12 and for the one that is mapped to the edge from ε to 12, resulting in a match as represented by Figure 11. The given restriction on the cardinality of the set V is no limitation since the number of LCPs in the set V is maximal if there is no pair of nodes such that one is an ancestor of the other. We can see these nodes as n leaf nodes of a tree that is at least binarily branching. Since such a tree can have at most n inner nodes, we need at most n new variables for a query in n variables.

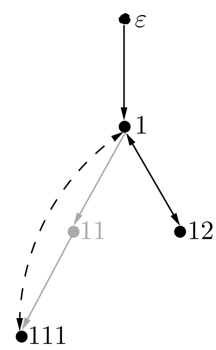

Figure 11: The match for a forest rewriting obtained from the example given in Figure 10.

For the bound on the number of role atoms that can be used in the replacement of a single role atom, consider, for example, the cyclic query

$$q = \{r(x_1, x_2), r(x_2, x_3), r(x_3, x_4), t(x_1, x_4)\},$$

for the knowledge base $\mathcal{K} = (\mathcal{T}, \mathcal{R}, \mathcal{A})$ with $\mathcal{T} = \emptyset, \mathcal{R} = \{r \sqsubseteq t\}$ with $t \in \mathsf{Trans}_{\mathcal{R}}$ and $\mathcal{A} = \{(\exists r.(\exists r.(\exists r.\top)))(a)\}$. It is not hard to check that $\mathcal{K} \models q$. Similarly to our running example from the previous section, there is also a single rewriting that is true in each canonical model of the KB, which is obtained by building only a forest rewriting and doing nothing in the other rewriting steps, except for choosing the empty set as root splitting in the split rewriting step. In the forest rewriting, we can explicate the short-cut used in the mapping for $t(x_1, x_4)$ by replacing $t(x_1, x_4)$ with $t(x_1, x_2), t(x_2, x_3), t(x_3, x_4)$.

By using Lemmas 15 to 18, we get the following theorem, which shows that we can use the ground queries in $\mathsf{ground}_{\mathcal{K}}(q)$ and the queries in $\mathsf{trees}_{\mathcal{K}}(q)$ in order to check whether $\mathcal{K}$ entails q, which is a well understood problem.

Theorem 19. *Let $\mathcal{K}$ be a $\mathcal{SHIQ}$ knowledge base, q a Boolean conjunctive query, and $\{q_1, \ldots, q_\ell\} = \mathsf{trees}_{\mathcal{K}}(q) \cup \mathsf{ground}_{\mathcal{K}}(q)$. Then $\mathcal{K} \models q$ iff $\mathcal{K} \models q_1 \vee \ldots \vee q_\ell$.*

We now give upper bounds on the size and number of queries in $\mathsf{trees}_{\mathcal{K}}(q)$ and $\mathsf{ground}_{\mathcal{K}}(q)$. As before, we use $\sharp(S)$ to denote the *cardinality* of a set S. The *size* $|\mathcal{K}|$ ($|q|$) of a knowledge base $\mathcal{K}$ (a query q) is simply the number of symbols needed to write it over the alphabet of constructors, concept names, and role names that occur in $\mathcal{K}$ (q), where numbers are encoded in binary. Obviously, the number of atoms in a query is bounded by its size, hence $\sharp(q) \leq |q|$ and, for simplicity, we use n as the size and the cardinality of q in what follows.

Lemma 20. *Let q be a Boolean conjunctive query, $\mathcal{K} = (\mathcal{T}, \mathcal{R}, \mathcal{A})$ a $\mathcal{SHIQ}$ knowledge base, $|q| := n$ and $|\mathcal{K}| := m$. Then there is a polynomial p such that*

1. *$\sharp(\mathsf{co}(q)) \leq 2^{p(n)}$ and, for each $q' \in \mathsf{co}(q)$, $|q'| \leq p(n)$,*
2. *$\sharp(\mathsf{sr}_{\mathcal{K}}(q)) \leq 2^{p(n) \cdot \log p(m)}$, and, for each $q' \in \mathsf{sr}_{\mathcal{K}}(q)$, $|q'| \leq p(n)$,*
3. *$\sharp(\mathsf{lr}_{\mathcal{K}}(q)) \leq 2^{p(n) \cdot \log p(m)}$, and, for each $q' \in \mathsf{lr}_{\mathcal{K}}(q)$, $|q'| \leq p(n)$,*

4. $\sharp(\mathsf{fr}_{\mathcal{K}}(q)) \leq 2^{p(n) \cdot \log p(m)}$, *and, for each* $q' \in \mathsf{fr}_{\mathcal{K}}(q)$, $|q'| \leq p(n)$,

5. $\sharp(\mathsf{trees}_{\mathcal{K}}(q)) \leq 2^{p(n) \cdot \log p(m)}$, *and, for each* $q' \in \mathsf{trees}_{\mathcal{K}}(q)$, $|q'| \leq p(n)$, *and*

6. $\sharp(\mathsf{ground}_{\mathcal{K}}(q)) \leq 2^{p(n) \cdot \log p(m)}$, *and, for each* $q' \in \mathsf{ground}_{\mathcal{K}}(q)$, $|q'| \leq p(n)$.

As a consequence of the above lemma, there is a bound on the number of queries in $\mathsf{ground}_{\mathcal{K}}(q)$ and $\mathsf{trees}_{\mathcal{K}}(q)$ and it is not hard to see that the two sets can be computed in time polynomial in m and exponential in n.

In the next section, we present an algorithm that decides entailment of unions of conjunctive queries, where each of the queries is either a ground query or consists of a single concept atom $C(x)$ for an existentially quantified variable x. By Theorem 19 and Lemma 20, such an algorithm is a decision procedure for arbitrary unions of conjunctive queries.

5.6 Summary and Discussion

In this section, we have presented the main technical foundations for answering (unions of) conjunctive queries. It is known that queries that contain non-simple roles in cycles among existentially quantified variables are difficult to handle. By applying the rewriting steps from Definition 10, we can rewrite such cyclic conjunctive queries into a set of acyclic and/or ground queries. Both types of queries are easier to handle and algorithms for both types exist. At this point, any reasoning algorithm for $\mathcal{SHIQ}^{\sqcap}$ knowledge base consistency can be used for deciding query entailment. In order to obtain tight complexity results, we present in the following section a decision procedure that is based on an extension of the translation to looping tree automata given by Tobies (2001).

It is worth mentioning that, for queries with only simple roles, our algorithm behaves exactly as the existing rewriting algorithms (i.e., the rolling-up and tuple graph technique) since, in this case, only the collapsing step is applicable. The need for identifying variables was first pointed out in the work of Horrocks et al. (1999) and is also required (although not mentioned) for the algorithm proposed by Calvanese et al. (1998a).

The new rewriting steps (split, loop, and forest rewriting) are only required for and applicable to non-simple roles and, when replacing a role atom, only transitive sub-roles of the replaced role can be used. Hence the number of resulting queries is in fact not determined by the size of the whole knowledge base, but by the number of transitive sub-roles for the non-simple roles in the query. Therefore, the number of resulting queries really depends on the number of transitive roles and the depth of the role hierarchy for the non-simple roles in the query, which can, usually, expected to be small.

6. The Decision Procedure

We now devise a decision procedure for entailment of unions of Boolean conjunctive queries that uses, for each disjunct, the queries obtained in the rewriting process as defined in the previous section. Detailed proofs for the lemmas and theorems in this section can again be found in the appendix. For a knowledge base $\mathcal{K}$ and a union of Boolean conjunctive queries $q_1 \vee \ldots \vee q_\ell$, we show how we can use the queries in $\mathsf{trees}_{\mathcal{K}}(q_i)$ and $\mathsf{ground}_{\mathcal{K}}(q_i)$ for $1 \leq i \leq \ell$ in order to build a set of knowledge bases $\mathcal{K}_1, \ldots, \mathcal{K}_n$ such that $\mathcal{K} \models q_1 \vee \ldots \vee q_\ell$ 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 a tight upper bound for the combined complexity; and a non-deterministic one in which we guess a $\mathcal{K}_i$, and which yields a tight upper bound for the data complexity. Recall that, for combined complexity, the knowledge base $\mathcal{K}$ and the queries q_i both count as input, whereas for the data complexity only the ABox $\mathcal{A}$ counts as an input, and all other parts are assumed to be fixed.

6.1 A Deterministic Decision Procedure for Query Entailment in $\mathcal{SHIQ}$

We first define the deterministic version of the decision procedure and give an upper bound for its combined complexity. The given algorithm takes as input a union of connected conjunctive queries and works under the unique name assumption (UNA). We show afterwards how it can be extended to an algorithm that does not make the UNA and that takes arbitrary UCQs as input, and that the complexity results carry over.

We construct a set of knowledge bases that extend the original knowledge base $\mathcal{K}$ both w.r.t. the TBox and ABox. The extended knowledge bases are such that a given KB $\mathcal{K}$ entails a query q iff all the extended KBs are inconsistent. We handle the concepts obtained from the tree-shaped queries differently to the ground queries: the axioms we add to the TBox prevent matches for the tree-shaped queries, whereas the extended ABoxes contain assertions that prevent matches for the ground queries.

Definition 21. Let $\mathcal{K} = (\mathcal{T}, \mathcal{R}, \mathcal{A})$ be a $\mathcal{SHIQ}$ knowledge base and $q = q_1 \vee \ldots \vee q_\ell$ a union of Boolean conjunctive queries. We set

1. $T := \mathsf{trees}_{\mathcal{K}}(q_1) \cup \ldots \cup \mathsf{trees}_{\mathcal{K}}(q_\ell)$,
2. $G := \mathsf{ground}_{\mathcal{K}}(q_1) \cup \ldots \cup \mathsf{ground}_{\mathcal{K}}(q_\ell)$, and
3. $\mathcal{T}_q := \{\top \sqsubseteq \neg C \mid C(v) \in T\}$.

An *extended knowledge base* $\mathcal{K}_q$ w.r.t. $\mathcal{K}$ and q is a tuple $(\mathcal{T} \cup \mathcal{T}_q, \mathcal{R}, \mathcal{A} \cup \mathcal{A}_q)$ such that $\mathcal{A}_q$ contains, for each $q' \in G$, at least one assertion $\neg at$ with $at \in q'$. △

Informally, the extended TBox $\mathcal{T} \cup \mathcal{T}_q$ ensures that there are no tree matches. Each extended ABox $\mathcal{A} \cup \mathcal{A}_q$ contains, for each ground query q' obtained in the rewriting process, at least one assertion $\neg at$ with $at \in q'$ that "spoils" a match for q'. A model for such an extended ABox can, therefore, not satisfy any of the ground queries. If there is a model for any of the extended knowledge bases, we know that this is a counter-model for the original query.

We can now use the extended knowledge bases in order to define the deterministic version of our algorithm for deciding entailment of unions of Boolean conjunctive queries in $\mathcal{SHIQ}$.

Definition 22. Given a $\mathcal{SHIQ}$ knowledge base $\mathcal{K} = (\mathcal{T}, \mathcal{R}, \mathcal{A})$ and a union of connected Boolean conjunctive queries q as input, the algorithm answers "$\mathcal{K}$ entails q" if each extended knowledge base w.r.t. $\mathcal{K}$ and q is inconsistent and it answers "$\mathcal{K}$ does not entail q" otherwise. △

The following lemma shows that the above described algorithm is indeed correct.

Lemma 23. *Let $\mathcal{K}$ be a $\mathcal{SHIQ}$ knowledge base and q a union of connected Boolean conjunctive queries. Given $\mathcal{K}$ and q as input, the algorithm from Definition 22 answers "$\mathcal{K}$ entails q" iff $\mathcal{K} \models q$ under the unique name assumption.*

In the proof of the if direction for the above lemma, we can use a canonical model $\mathcal{I}$ of $\mathcal{K}$ in order to guide the rewriting process. For the only if direction, we assume to the contrary of what is to be shown that there is no consistent extended knowledge base, but $\mathcal{K} \not\models q$. We then use a model $\mathcal{I}$ of $\mathcal{K}$ such that $\mathcal{I} \not\models q$, which exists by assumption, and show that $\mathcal{I}$ is also a model of some extended knowledge base.

6.1.1 Combined Complexity of Query Entailment in $\mathcal{SHIQ}$

According to the above lemma, the algorithm given in Definition 22 is correct. We now analyse its combined complexity and thereby prove that it is also terminating.

For the complexity analysis, we assume, as usual (Hustadt et al., 2005; Calvanese, De Giacomo, Lembo, Lenzerini, & Rosati, 2006; Ortiz et al., 2006b), that all concepts in concept atoms and ABox assertions are literals, i.e., concept names or negated concept names. If the input query or ABox contains non-literal atoms or assertions, we can easily transform these into literal ones in a truth preserving way: for each concept atom $C(t)$ in the query where C is a non-literal concept, we introduce a new atomic concept $A_C \in N_C$, add the axiom $C \sqsubseteq A_C$ to the TBox, and replace $C(t)$ with $A_C(t)$; for each non-literal concept assertion $C(a)$ in the ABox, we introduce a new atomic concept $A_C \in N_C$, add an axiom $A_C \sqsubseteq C$ to the TBox, and replace $C(a)$ with $A_C(a)$. Such a transformation is obviously polynomial, so without loss of generality, it is safe to assume that the ABox and query contain only literal concepts. This has the advantage that the size of each atom and ABox assertion is constant.

Since our algorithm involves checking the consistency of a $\mathcal{SHIQ}^{\sqcap}$ knowledge base, we analyse the complexity of this reasoning service. Tobies (2001) shows an ExpTime upper bound for deciding the consistency of $\mathcal{SHIQ}$ knowledge bases (even with binary coding of numbers) by translating a $\mathcal{SHIQ}$ KB to an equisatisfiable $\mathcal{ALCQI}b$ knowledge base. The b stands for *safe Boolean role expressions* built from $\mathcal{ALCQI}b$ roles using the operator $\sqcap$ (role intersection), $\sqcup$ (role union), and $\neg$ (role negation/complement) such that, when transformed into disjunctive normal form, every disjunct contains at least one non-negated conjunct. Given a query q and a $\mathcal{SHIQ}$ knowledge base $\mathcal{K} = (\mathcal{T}, \mathcal{R}, \mathcal{A})$, we reduce query entailment to deciding knowledge base consistency of an extended $\mathcal{SHIQ}^{\sqcap}$ knowledge base $\mathcal{K}_q = (\mathcal{T} \cup \mathcal{T}_q, \mathcal{R}, \mathcal{A} \cup \mathcal{A}_q)$. Recall that $\mathcal{T}_q$ and $\mathcal{A}_q$ are the only parts that contain role conjunctions and that we use role negation only in ABox assertions. We extend the translation given for $\mathcal{SHIQ}$ so that it can be used for deciding the consistency of $\mathcal{SHIQ}^{\sqcap}$ KBs. Although the translation works for all $\mathcal{SHIQ}^{\sqcap}$ KBs, we assume the input KB to be of exactly the form of extended knowledge bases as described above. This is so because the translation for unrestricted $\mathcal{SHIQ}^{\sqcap}$ is no longer polynomial, as in the case of $\mathcal{SHIQ}$, but exponential in the size of the longest role conjunction under a universal quantifier. Since role conjunctions occur only in the extended ABox and TBox, and since the size of each role conjunction is, by Lemma 20, polynomial in the size of q, the translation is only exponential in the size of the query in the case of extended knowledge bases.

We assume here, as usual, that all concepts are in *negation normal form (NNF)*; any concept can be transformed in linear time into an equivalent one in NNF by pushing negation inwards, making use of de Morgan's laws and the duality between existential and universal restrictions, and between atmost and atleast number restrictions ($\leqslant n\ r.C$ and $\geqslant n\ r.C$ respectively) (Horrocks et al., 2000). For a concept C, we use $\dot\neg C$ to denote the NNF of $\neg C$.

We define the *closure* $\mathsf{cl}(C, \mathcal{R})$ of a concept C w.r.t. a role hierarchy $\mathcal{R}$ as the smallest set satisfying the following conditions:

- if D is a sub-concept of C, then $D \in \mathsf{cl}(C, \mathcal{R})$,
- if $D \in \mathsf{cl}(C, \mathcal{R})$, then $\dot\neg D \in \mathsf{cl}(C, \mathcal{R})$,
- if $\forall r.D \in \mathsf{cl}(C, \mathcal{R})$, $s \sqsubseteq^*_{\mathcal{R}} r$, and $s \in \mathsf{Trans}_{\mathcal{R}}$, then $\forall s.D \in \mathsf{cl}(C, \mathcal{R})$.

We now show how we can extend the translation from $\mathcal{SHIQ}$ to $\mathcal{ALCQI}b$ given by Tobies. We first consider $\mathcal{SHIQ}^{\sqcap}$-concepts and then extend the translation to KBs.

Definition 24. For a role hierarchy $\mathcal{R}$ and roles $r, r_1, \ldots, r_n$, let

$$\uparrow(r, \mathcal{R}) = \bigsqcap_{r \sqsubseteq^*_{\mathcal{R}} s} s \quad \text{and} \quad \uparrow(r_1 \sqcap \ldots \sqcap r_n, \mathcal{R}) = \uparrow(r_1, \mathcal{R}) \sqcap \ldots \sqcap \uparrow(r_n, \mathcal{R}).$$

△

Please note that, since $r \sqsubseteq^*_{\mathcal{R}} r$, r occurs in $\uparrow(r, \mathcal{R})$.

Lemma 25. *Let $\mathcal{R}$ be a role hierarchy, and $r_1, \ldots, r_n$ roles. For every interpretation $\mathcal{I}$ such that $\mathcal{I} \models \mathcal{R}$, it holds that $(\uparrow(r_1 \sqcap \ldots \sqcap r_n, \mathcal{R}))^{\mathcal{I}} = (r_1 \sqcap \ldots \sqcap r_n)^{\mathcal{I}}$.*

With the extended definition of $\uparrow$ on role conjunctions, we can now adapt the definition (Def. 6.22) that Tobies provides for translating $\mathcal{SHIQ}$-concepts into $\mathcal{ALCQI}b$-concepts.

Definition 26. Let C be a $\mathcal{SHIQ}^{\sqcap}$-concept in NNF and $\mathcal{R}$ a role hierarchy. For every concept $\forall(r_1 \sqcap \ldots \sqcap r_n).D \in \mathsf{cl}(C, \mathcal{R})$, let $X_{r_1 \sqcap \ldots \sqcap r_n, D} \in N_C$ be a unique concept name that does not occur in $\mathsf{cl}(C, \mathcal{R})$. Given a role hierarchy $\mathcal{R}$, we define the function tr inductively on the structure of concepts by setting

$$\begin{array}{rcl}
\mathsf{tr}(A, \mathcal{R}) & = & A \text{ for all A } \in N_C \\
\mathsf{tr}(\neg A, \mathcal{R}) & = & \neg A \text{ for all A } \in N_C \\
\mathsf{tr}(C_1 \sqcap C_2, \mathcal{R}) & = & \mathsf{tr}(C_1, \mathcal{R}) \sqcap \mathsf{tr}(C_2, \mathcal{R}) \\
\mathsf{tr}(C_1 \sqcup C_2, \mathcal{R}) & = & \mathsf{tr}(C_1, \mathcal{R}) \sqcup \mathsf{tr}(C_2, \mathcal{R}) \\
\mathsf{tr}(\bowtie n (r_1 \sqcap \ldots \sqcap r_n).D, \mathcal{R}) & = & (\bowtie n \uparrow(r_1 \sqcap \ldots \sqcap r_n, \mathcal{R}).\mathsf{tr}(D, \mathcal{R})) \\
\mathsf{tr}(\forall(r_1 \sqcap \ldots \sqcap r_n).D, \mathcal{R}) & = & X_{r_1 \sqcap \ldots \sqcap r_n, D} \\
\mathsf{tr}(\exists(r_1 \sqcap \ldots \sqcap r_n).D, \mathcal{R}) & = & \neg(X_{r_1 \sqcap \ldots \sqcap r_n, \dot\neg D})
\end{array}$$

where $\bowtie$ stands for $\leqslant$ or $\geqslant$. Set $\mathsf{tc}((r_1 \sqcap \ldots \sqcap r_n), \mathcal{R}) := \{(t_1 \sqcap \ldots \sqcap t_n) \mid t_i \sqsubseteq^*_{\mathcal{R}} r_i$ and $t_i \in \mathsf{Trans}_{\mathcal{R}}$ for each i such that $1 \leq i \leq n\}$ and define an extended TBox $\mathcal{T}_{C,\mathcal{R}}$ as

$$\begin{array}{rll}
\mathcal{T}_{C,\mathcal{R}} = & \{X_{r_1 \sqcap \ldots \sqcap r_n, D} \equiv \forall \uparrow(r_1 \sqcap \ldots \sqcap r_n, \mathcal{R}).\mathsf{tr}(D, \mathcal{R}) & \mid \forall(r_1 \sqcap \ldots \sqcap r_n).D \in \mathsf{cl}(C, \mathcal{R})\} \cup \\
& \{X_{r_1 \sqcap \ldots \sqcap r_n, D} \sqsubseteq \forall \uparrow(T, \mathcal{R}).X_{T,D} & \mid T \in \mathsf{tc}(r_1 \sqcap \ldots \sqcap r_n, \mathcal{R})\}
\end{array}$$

△

Lemma 27. *Let C be a $\mathcal{SHIQ}^{\sqcap}$-concept in NNF, $\mathcal{R}$ a role hierarchy, and tr and $\mathcal{T}_{C,\mathcal{R}}$ as defined in Definition 26. The concept C is satisfiable w.r.t. $\mathcal{R}$ iff the $\mathcal{ALCQI}b$-concept $\mathsf{tr}(C, \mathcal{R})$ is satisfiable w.r.t. $\mathcal{T}_{C,\mathcal{R}}$.*

Given Lemma 25, the proof of Lemma 27 is a long, but straightforward extension of the proof given by Tobies (2001, Lemma 6.23).

We now analyse the complexity of the above described problem. Let $m := |\mathcal{R}|$ and $r_1 \sqcap \ldots \sqcap r_n$ the longest role conjunction occurring in C, i.e., the maximal number of roles that occur in a role conjunction in C is n. The TBox $\mathcal{T}_{C,\mathcal{R}}$ can contain exponentially many axioms in n since the cardinality of the set $\mathsf{tc}((r_1 \sqcap \ldots \sqcap r_n), \mathcal{R})$ for the longest role conjunction can only be bounded by m^n because each r_i can have more than one transitive sub-role. It is not hard to check that the size of each axiom is polynomial in $|C|$. Since deciding whether an $\mathcal{ALCQIb}$ concept C is satisfiable w.r.t. an $\mathcal{ALCQIb}$ TBox $\mathcal{T}$ is an ExpTime-complete problem (even with binary coding of numbers) (Tobies, 2001, Thm. 4.42), the satisfiability of a $\mathcal{SHIQ}^{\sqcap}$-concept C can be checked in time $2^{p(m)2^{p(n)}}$.

We now extend the translation from concepts to knowledge bases. Tobies assumes that all role assertions in the ABox are of the form $r(a, b)$ with r a role name or the inverse of a role name. Extended ABoxes contain, however, also negated roles in role assertions, which require a different translation. A positive role assertion such as $r(a, b)$ is translated in the standard way by closing the role upwards. The only difference of using $\uparrow$ directly is that we additionally split the conjunction $(\uparrow(r, \mathcal{R}))(a, b) = (r_1 \sqcap \ldots \sqcap r_n)(a, b)$ into n different role assertions $r_1(a, b), \ldots, r_n(a, b)$, which is clearly justified by the semantics. For negated roles in a role assertion such as $\neg r(a, b)$, we close the role downwards instead of upwards and add a role atom $\neg s(a, b)$ for each sub-role s of r. This is again justified by the semantics. Let $\mathcal{K} = (\mathcal{T} \cup \mathcal{T}_q, \mathcal{R}, \mathcal{A} \cup \mathcal{A}_q)$ be an extended knowledge base. More precisely, we set

$$\mathsf{tr}(\mathcal{T} \cup \mathcal{T}_q, \mathcal{R}) := \{\mathsf{tr}(C, \mathcal{R}) \sqsubseteq \mathsf{tr}(D, \mathcal{R}) \mid C \sqsubseteq D \in \mathcal{T} \cup \mathcal{T}_q\},$$

$$\begin{aligned} \mathsf{tr}(\mathcal{A} \cup \mathcal{A}_q, \mathcal{R}) := \quad & \{(\mathsf{tr}(C, \mathcal{R}))(a) \mid C(a) \in \mathcal{A} \cup \mathcal{A}_q\} \cup \\ & \{s(a, b) \mid r(a, b) \in \mathcal{A} \cup \mathcal{A}_q \text{ and } r \,\underline{\circledast}_{\mathcal{R}} s\} \cup \\ & \{\neg s(a, b) \mid \neg r(a, b) \in \mathcal{A} \cup \mathcal{A}_q \text{ and } s \,\underline{\circledast}_{\mathcal{R}} r\}, \end{aligned}$$

and we use $\mathsf{tr}(\mathcal{K}, \mathcal{R})$ to denote the $\mathcal{ALCQIb}$ knowledge base ($\mathsf{tr}(\mathcal{T} \cup \mathcal{T}_q, \mathcal{R})$, $\mathsf{tr}(\mathcal{A} \cup \mathcal{A}_q, \mathcal{R})$).

For the complexity of deciding the consistency of a translated $\mathcal{SHIQ}^{\sqcap}$ knowledge base, we can apply the same arguments as above for concept satisfiability, which gives the following result:

Lemma 28. *Given a $\mathcal{SHIQ}^{\sqcap}$ knowledge base $\mathcal{K} = (\mathcal{T}, \mathcal{R}, \mathcal{A})$ where $m := |\mathcal{K}|$ and the size of the longest role conjunction is n, we can decide consistency of $\mathcal{K}$ in deterministic time $2^{p(m)2^{p(n)}}$ with p a polynomial.*

We are now ready to show that the algorithm given in Definition 22 runs in deterministic time single exponential in the size of the input KB and double exponential in the size of the input query.

Lemma 29. *Let $\mathcal{K} = (\mathcal{T}, \mathcal{R}, \mathcal{A})$ be a $\mathcal{SHIQ}$ knowledge base with $m = |\mathcal{K}|$ and q a union of connected Boolean conjunctive queries with $n = |q|$. Given $\mathcal{K}$ and q as input, the algorithm given in Definition 22 decides whether $\mathcal{K} \models q$ under the unique name assumption in deterministic time in $2^{p(m)2^{p(n)}}$.*

In the proof of the above lemma, we show that there is some polynomial p such that we have to check at most $2^{p(m)2^{p(n)}}$ extended knowledge bases for consistency and that each consistency check can be done in this time bound as well.

More precisely, let $q = q_1 \vee \ldots \vee q_\ell$, $T = \mathsf{trees}_\mathcal{K}(q_1) \cup \ldots \cup \mathsf{trees}_\mathcal{K}(q_\ell)$, and $G = \mathsf{ground}_\mathcal{K}(q_1) \cup \ldots \cup \mathsf{ground}_\mathcal{K}(q_\ell)$. Together with Lemma 20, we get that $\sharp(T)$ and $\sharp(G)$ are bounded by $2^{p(n) \cdot \log p(m)}$ for some polynomial p and that the size of each query in G and T is polynomial in n. Each of the $2^{p(n) \cdot \log p(m)}$ ground queries in G contributes at most $p(n)$ negated assertion to an extended ABox $\mathcal{A}_q$. Hence, there are at most $2^{p(m)2^{p(n)}}$ extended ABoxes $\mathcal{A}_q$ and, therefore, $2^{p(m)2^{p(n)}}$ extended knowledge bases that have to be tested for consistency.

Given the bounds on the cardinalities of T and G and the fact that the size of each query in T and G is polynomial in n, it is not hard to check that the size of each extended knowledge base $\mathcal{K}_q = (\mathcal{T} \cup \mathcal{T}_q, \mathcal{R}, \mathcal{A} \cup \mathcal{A}_q)$ is bounded by $2^{p(n) \cdot \log p(m)}$ and that each $\mathcal{K}_q$ can be computed in this time bound as well. Since only the extended parts contain role conjunctions and the number of roles in a role conjunction is polynomial in n, there is a polynomial p such that

1. $|\mathsf{tr}(\mathcal{T}, \mathcal{R})| \leq p(m)$,
2. $|\mathsf{tr}(\mathcal{T}_q, \mathcal{R})| \leq 2^{p(n) \cdot \log p(m)}$,
3. $|\mathsf{tr}(\mathcal{A}, \mathcal{R})| \leq p(m)$,
4. $|\mathsf{tr}(\mathcal{A}_q, \mathcal{R})| \leq 2^{p(n) \cdot \log p(m)}$, and, hence,
5. $|\mathsf{tr}(\mathcal{K}_q, \mathcal{R})| \leq 2^{p(n) \cdot \log p(m)}$.

By Lemma 28, each consistency check can be done in time $2^{p(m)2^{p(n)}}$ for some polynomial p. Since we have to check at most $2^{p(m)2^{p(n)}}$ extended knowledge bases for consistency, and each check can be done in time $2^{p(m)2^{p(n)}}$, we obtain the desired upper bound.

We now show that this result carries over even when we do not restrict interpretations to the unique name assumption.

Definition 30. Let $\mathcal{K} = (\mathcal{T}, \mathcal{R}, \mathcal{A})$ be a $\mathcal{SHIQ}$ knowledge base and q a $\mathcal{SHIQ}$ union of Boolean conjunctive queries. For a partition $\mathcal{P}$ of $\mathsf{Inds}(\mathcal{A})$, a knowledge base $\mathcal{K}^\mathcal{P} = (\mathcal{T}, \mathcal{R}, \mathcal{A}^\mathcal{P})$ and a query $q^\mathcal{P}$ are called an *$\mathcal{A}$-partition w.r.t. $\mathcal{K}$ and q* if $\mathcal{A}^\mathcal{P}$ and $q^\mathcal{P}$ are obtained from $\mathcal{A}$ and q as follows:
For each $P \in \mathcal{P}$

1. Choose one individual name $a \in P$.
2. For each $b \in P$, replace each occurrence of b in $\mathcal{A}$ and q with a.

$\triangle$

Please note that w.l.o.g. we assume that all constants that occur in the query occur in the ABox as well and that thus a partition of the individual names in the ABox also partitions the query.

Lemma 31. *Let $\mathcal{K} = (\mathcal{T}, \mathcal{R}, \mathcal{A})$ be a $\mathcal{SHIQ}$ knowledge base and q a union of Boolean conjunctive queries. $\mathcal{K} \not\models q$ without making the unique name assumption iff there is an $\mathcal{A}$-partition $\mathcal{K}^\mathcal{P} = (\mathcal{T}, \mathcal{R}, \mathcal{A}^\mathcal{P})$ and $q^\mathcal{P}$ w.r.t. $\mathcal{K}$ and q such that $\mathcal{K}^\mathcal{P} \not\models q^\mathcal{P}$ under the unique name assumption.*

Let $\mathcal{K} = (\mathcal{T}, \mathcal{R}, \mathcal{A})$ be a knowledge base in a Description Logic $\mathcal{DL}$, $\mathcal{C}$ be the complexity class such that deciding whether $\mathcal{K} \models q$ under the unique name assumption is in $\mathcal{C}$, and let $n = 2^{|\mathcal{A}|}$. Since the number of partitions for an ABox is at most exponential in the number of individual names that occur in the ABox, the following is a straightforward consequence of the above lemma: for a Boolean conjunctive $\mathcal{DL}$ query q, deciding whether $\mathcal{K} \models q$ without making the unique name assumption can be reduced to deciding n times a problem in $\mathcal{C}$.

In order to extend our algorithm to unions of possibly unconnected Boolean conjunctive queries, we first transform the input query q into conjunctive normal form (CNF). We then check entailment for each conjunct q_i, which is now a union of connected Boolean conjunctive queries. The algorithm returns "$\mathcal{K}$ entails q" if each entailment check succeeds and it answers "$\mathcal{K}$ does not entail q" otherwise. By Lemma 5 and Lemma 23, the algorithm is correct.

Let $\mathcal{K}$ be a knowledge base in a Description Logic $\mathcal{DL}$, q a union of connected Boolean conjunctive $\mathcal{DL}$ queries, and $\mathcal{C}$ the complexity class such that deciding whether $\mathcal{K} \models q$ is in $\mathcal{C}$. Let q' be a union of possibly unconnected Boolean conjunctive queries and $\mathsf{cnf}(q')$ the CNF of q'. Since the number of conjuncts in $\mathsf{cnf}(q')$ is at most exponential in $|q'|$, deciding whether $\mathcal{K} \models q'$ can be reduced to deciding n times a problem in $\mathcal{C}$, with $n = 2^{p(|q'|)}$ and p a polynomial.

The above observation together with the results from Lemma 29 gives the following general result:

Theorem 32. *Let $\mathcal{K} = (\mathcal{T}, \mathcal{R}, \mathcal{A})$ be a $\mathcal{SHIQ}$ knowledge base with $m = |\mathcal{K}|$ and q a union of Boolean conjunctive queries with $n = |q|$. Deciding whether $\mathcal{K} \models q$ can be done in deterministic time in $2^{p(m)2^{p(n)}}$.*

A corresponding lower bound follows from the work by Lutz (2007). Hence the above result is tight. The result improves the known co-3NExpTime upper bound for the setting where the roles in the query are restricted to simple ones (Ortiz, Calvanese, & Eiter, 2006a).

Corollary 33. *Let $\mathcal{K}$ be a $\mathcal{SHIQ}$ knowledge base with $m = |\mathcal{K}|$ and q a union of Boolean conjunctive queries with $n = |q|$. Deciding whether $\mathcal{K} \models q$ is a 2ExpTime-complete problem.*

Regarding query answering, we refer back to the end of Section 2.2, where we explain that deciding which tuples belong to the set of answers can be checked with at most $m_{\mathcal{A}}^k$ entailment tests, where k is the number of answer variables in the query and $m_{\mathcal{A}}$ is the number of individual names in $\mathsf{Inds}(\mathcal{A})$. Hence, at least theoretically, this is absorbed by the combined complexity of query entailment in $\mathcal{SHIQ}$.

6.2 A Non-Deterministic Decision Procedure for Query Entailment in $\mathcal{SHIQ}$

In order to study the data complexity of query entailment, we devise a non-deterministic decision procedure which provides a tight bound for the complexity of the problem. Actually, the devised algorithm decides non-entailment of queries: we guess an extended knowledge base $\mathcal{K}_q$, check whether it is consistent, and return "$\mathcal{K}$ does not entail q" if the check succeeds and "$\mathcal{K}$ entails q" otherwise.

Definition 34. Let $\mathcal{T}$ be a $\mathcal{SHIQ}$ TBox, $\mathcal{R}$ a $\mathcal{SHIQ}$ role hierarchy, and q a union of Boolean conjunctive queries. Given a $\mathcal{SHIQ}$ ABox $\mathcal{A}$ as input, the algorithm guesses an

$\mathcal{A}$-partition $\mathcal{K}^{\mathcal{P}} = (\mathcal{T}, \mathcal{R}, \mathcal{A}^{\mathcal{P}})$ and $q^{\mathcal{P}}$ w.r.t. $\mathcal{K} = (\mathcal{T}, \mathcal{R}, \mathcal{A})$ and q. The query $q^{\mathcal{P}}$ is then transformed into CNF and one of the resulting conjuncts, say $q_i^{\mathcal{P}}$, is chosen. The algorithm then guesses an extended knowledge base $\mathcal{K}_{q_i}^{\mathcal{P}} = (\mathcal{T} \cup \mathcal{T}_{q_i}, \mathcal{R}, \mathcal{A}^{\mathcal{P}} \cup \mathcal{A}_{q_i}^{\mathcal{P}})$ w.r.t. $\mathcal{K}^{\mathcal{P}}$ and $q_i^{\mathcal{P}}$ and returns "$\mathcal{K}$ does not entail q" if $\mathcal{K}_{q_i}^{\mathcal{P}}$ is consistent and it returns "$\mathcal{K}$ entails q" otherwise. △

Compared to the deterministic version of the algorithm given in Definition 22, we do not make the UNA but guess a partition of the individual names. We also non-deterministically choose one of the conjuncts that result from the transformation into CNF. For this conjunct, we guess an extended ABox and check whether the extended knowledge base for the guessed ABox is consistent and, therefore, a counter-model for the query entailment.

In its (equivalent) negated form, Lemma 23 says that $\mathcal{K} \not\models q$ iff there is an extended knowledge base $\mathcal{K}_q$ w.r.t. $\mathcal{K}$ and q such that $\mathcal{K}_q$ is consistent. Together with Lemma 31 it follows, therefore, that the algorithm from Definition 34 is correct.

6.2.1 Data Complexity of Query Entailment in $\mathcal{SHIQ}$

We now analyze the data complexity of the algorithm given in Definition 34 and show that deciding UCQ entailment in $\mathcal{SHIQ}$ is indeed in co-NP for data complexity.

Theorem 35. *Let $\mathcal{T}$ be a $\mathcal{SHIQ}$ TBox, $\mathcal{R}$ a $\mathcal{SHIQ}$ role hierarchy, and q a union of Boolean conjunctive queries. Given a $\mathcal{SHIQ}$ ABox $\mathcal{A}$ with $m_a = |\mathcal{A}|$, the algorithm from Definition 34 decides in non-deterministic polynomial time in m_a whether $\mathcal{K} \not\models q$ for $\mathcal{K} = (\mathcal{T}, \mathcal{R}, \mathcal{A})$.*

Clearly, the size of an ABox $\mathcal{A}^{\mathcal{P}}$ in an $\mathcal{A}$-partition is bounded by m_a. Since the query is no longer an input, its size is constant and the transformation to CNF can be done in constant time. We then non-deterministically choose one of the resulting conjuncts. Let this conjunct be $q_i = q_{(i,1)} \vee \ldots \vee q_{(i,\ell)}$. As established in Lemma 32, the maximal size of an extended ABox $\mathcal{A}_{q_i}^{\mathcal{P}}$ is polynomial in m_a. Hence, $|\mathcal{A}^{\mathcal{P}} \cup \mathcal{A}_{q_i}^{\mathcal{P}}| \leq p(m_a)$ for some polynomial p. Due to Lemma 20 and since the size of q, $\mathcal{T}$, and $\mathcal{R}$ is fixed by assumption, the sets $\mathsf{trees}_{\mathcal{K}^{\mathcal{P}}}(q_{(i,j)})$ and $\mathsf{ground}_{\mathcal{K}^{\mathcal{P}}}(q_{(i,j)})$ for each j such that $1 \leq j \leq \ell$ can be computed in time polynomial in m_a. From Lemma 29, we know that the translation of an extended knowledge base into an $\mathcal{ALCQI}b$ knowledge base is polynomial in m_a and a close inspection of the algorithm by Tobies (2001) for deciding consistency of an $\mathcal{ALCQI}b$ knowledge base shows that its runtime is also polynomial in m_a.

The bound given in Theorem 35 is tight since the data complexity of conjunctive query entailment is already co-NP-hard for the $\mathcal{ALE}$ fragment of $\mathcal{SHIQ}$ (Schaerf, 1993).

Corollary 36. *Conjunctive query entailment in $\mathcal{SHIQ}$ is data complete for co-*NP.

Due to the correspondence between query containment and query answering (Calvanese et al., 1998a), the algorithm can also be used to decide containment of two unions of conjunctive queries over a $\mathcal{SHIQ}$ knowledge base, which gives the following result:

Corollary 37. *Given a $\mathcal{SHIQ}$ knowledge base $\mathcal{K}$ and two unions of conjunctive queries q and q', the problem whether $\mathcal{K} \models q \subseteq q'$ is decidable.*

By using the result of Rosati (2006a, Thm. 11), we further show that the consistency of a $\mathcal{SHIQ}$ knowledge base extended with (weakly-safe) Datalog rules is decidable.

Corollary 38. *The consistency of $\mathcal{SHIQ}$+log-KBs (both under FOL semantics and under NM semantics) is decidable.*

7. Conclusions

With the decision procedure presented for entailment of unions of conjunctive queries in $\mathcal{SHIQ}$, we close a long standing open problem. The solution has immediate consequences on related areas, as it shows that several other open problems such as query answering, query containment and the extension of a knowledge base with weakly safe Datalog rules for $\mathcal{SHIQ}$ are decidable as well. Regarding combined complexity, we present a deterministic algorithm that needs time single exponential in the size of the KB and double exponential in the size of the query, which gives a tight upper bound for the problem. This result shows that deciding conjunctive query entailment is strictly harder than instance checking for $\mathcal{SHIQ}$. We further prove co-NP-completeness for data complexity. Interestingly, this shows that regarding data complexity deciding UCQ entailment is (at least theoretically) not harder than instance checking for $\mathcal{SHIQ}$, which was also a previously open question.

It will be part of our future work to extend this procedure to $\mathcal{SHOIQ}$, which is the DL underlying OWL DL. We will also attempt to find more implementable algorithms for query answering in $\mathcal{SHIQ}$. Carrying out the query rewriting steps in a more goal directed way will be crucial to achieving this.

Acknowledgments

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

Appendix A. Complete Proofs

Lemma (7)**.** *Let $\mathcal{K}$ be a $\mathcal{SHIQ}$ knowledge base and $q = q_1 \vee \ldots \vee q_n$ a union of conjunctive queries, then $\mathcal{K} \not\models q$ iff there exists a canonical model $\mathcal{I}$ of $\mathcal{K}$ such that $\mathcal{I} \not\models q$.*

Proof of Lemma 7. The "if" direction is trivial.

For the "only if" direction, since an inconsistent knowledge base entails every query, we can assume that $\mathcal{K}$ is consistent. Hence, there is an interpretation $\mathcal{I}' = (\Delta^{\mathcal{I}'}, \cdot^{\mathcal{I}'})$ such that $\mathcal{I}' \models \mathcal{K}$ and $\mathcal{I}' \not\models q$. From $\mathcal{I}'$, we construct a canonical model $\mathcal{I}$ for $\mathcal{K}$ and its forest base $\mathcal{J}$ as follows: we define the set $P \subseteq (\Delta^{\mathcal{I}'})^*$ of *paths* to be the smallest set such that

- for all $a \in \mathsf{Inds}(\mathcal{A})$, $a^{\mathcal{I}'}$ is a path;
- $d_1 \cdots d_n \cdot d$ is a path, if
  - $d_1 \cdots d_n$ is a path,
  - $(d_n, d) \in r^{\mathcal{I}'}$ for some role r,
  - if there is an $a \in \mathsf{Inds}(\mathcal{A})$ such that $d = a^{\mathcal{I}'}$, then $n > 2$.

For a path $p = d_1 \cdots d_n$, the *length* $\mathsf{len}(p)$ of p is n. Now fix a set $S \subseteq \mathsf{Inds}(\mathcal{A}) \times \mathbb{N}^*$ and a bijection $f \colon S \to P$ such that

(i) $\mathsf{Inds}(\mathcal{A}) \times \{\varepsilon\} \subseteq S$,

(ii) for each $a \in \mathsf{Inds}(\mathcal{A})$, $\{w \mid (a, w) \in S\}$ is a tree,

(iii) $f((a, \varepsilon)) = a^{\mathcal{I}'}$,

(iv) if $(a, w), (a, w') \in S$ with w' a successor of w, then $f((a, w')) = f((a, w)) \cdot d$ for some $d \in \Delta^{\mathcal{I}'}$.

For all $(a, w) \in S$, set $\mathsf{Tail}((a, w)) := d_n$ if $f((a, w)) = d_1 \cdots d_n$. Now, define a forest base $\mathcal{J} = (\Delta^{\mathcal{J}}, \cdot^{\mathcal{J}})$ for $\mathcal{K}$ as follows:

(a) $\Delta^{\mathcal{J}} := S$;

(b) for each $a \in \mathsf{Inds}(\mathcal{A})$, $a^{\mathcal{J}} := (a, \varepsilon) \in S$;

(c) for each $b \in N_I \setminus \mathsf{Inds}(\mathcal{A})$, $b^{\mathcal{J}} = a^{\mathcal{J}}$ for some fixed $a \in \mathsf{Inds}(\mathcal{A})$;

(d) for each $C \in N_C$, $(a, w) \in C^{\mathcal{J}}$ if $(a, w) \in S$ and $\mathsf{Tail}((a, w)) \in C^{\mathcal{I}'}$;

(e) For all roles r, $((a, w), (b, w')) \in r^{\mathcal{J}}$ if either

  (I) $w = w' = \varepsilon$ and $(a^{\mathcal{I}'}, b^{\mathcal{I}'}) \in r^{\mathcal{I}'}$ or

  (II) $a = b$, w' is a neighbor of w and $(\mathsf{Tail}((a, w)), \mathsf{Tail}((b, w'))) \in r^{\mathcal{I}'}$.

It is clear that $\mathcal{J}$ is a forest base for $\mathcal{K}$ due to the definition of S and the construction of $\mathcal{J}$ from S.

Let $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ be an interpretation that is identical to $\mathcal{J}$ except that, for all non-simple roles r, we set

$$r^{\mathcal{I}} = r^{\mathcal{J}} \cup \bigcup_{s \mathrel{\underline{\ast}}_{\mathcal{R}} r,\ s \in \mathsf{Trans}_{\mathcal{R}}} (s^{\mathcal{J}})^{+}$$

It is tedious but not too hard to verify that $\mathcal{I} \models \mathcal{K}$ and that $\mathcal{J}$ is a forest base for $\mathcal{I}$. Hence $\mathcal{I}$ is a canonical model for $\mathcal{K}$.

Therefore, we only have to show that $\mathcal{I} \not\models q$. Assume to the contrary that $\mathcal{I} \models q$. Then there is some π and i with $1 \leq i \leq n$ such that $\mathcal{I} \models^{\pi} q_i$. We now define a mapping $\pi' \colon \mathsf{Terms}(q_i) \rightarrow \Delta^{\mathcal{I}'}$ by setting $\pi'(t) := \mathsf{Tail}(\pi(t))$ for all $t \in \mathsf{Terms}(q_i)$. It is not difficult to check that $\mathcal{I}' \models^{\pi'} q_i$ and hence $\mathcal{I}' \models^{\pi'} q$, which is a contradiction. □

Lemma (15). *Let $\mathcal{I}$ be a model for $\mathcal{K}$.*

1. *If $\mathcal{I} \models q$, then there is a collapsing q_{co} of q such that $\mathcal{I} \models^{\pi_{co}} q_{co}$ for π_{co} an injection modulo $\stackrel{*}{\approx}$.*

2. *If $\mathcal{I} \models^{\pi_{co}} q_{co}$ for a collapsing q_{co} of q, then $\mathcal{I} \models q$.*

Proof of Lemma 15. For (1), let π be such that $\mathcal{I} \models^{\pi} q$, let q_{co} be the collapsing of q that is obtained by adding an atom $t \approx t'$ for all terms $t, t' \in \mathsf{Terms}(q)$ for which $\pi(t) = \pi(t')$. By definition of the semantics, $\mathcal{I} \models^{\pi} q_{co}$ and π is an injection modulo $\stackrel{*}{\approx}$.

Condition (2) trivially holds since $q \subseteq q_{co}$ and hence $\mathcal{I} \models^{\pi_{co}} q$. □

Lemma (16). *Let $\mathcal{I}$ be a model for $\mathcal{K}$.*

1. *If $\mathcal{I}$ is canonical and $\mathcal{I} \models^{\pi} q$, then there is a pair $(q_{sr}, R) \in \mathsf{sr}_{\mathcal{K}}(q)$ and a split match π_{sr} such that $\mathcal{I} \models^{\pi_{sr}} q_{sr}$, R is the induced root splitting of π_{sr}, and π_{sr} is an injection modulo $\stackrel{*}{\approx}$.*

2. *If $(q_{sr}, R) \in \mathsf{sr}_{\mathcal{K}}(q)$ and $\mathcal{I} \models^{\pi_{sr}} q_{sr}$ for some match π_{sr}, then $\mathcal{I} \models q$.*

Proof of Lemma 16. The proof of the second claim is relatively straightforward: since $(q_{sr}, R) \in \mathsf{sr}_{\mathcal{K}}(q)$, there is a collapsing q_{co} of q such that q_{sr} is a split rewriting of q_{co}. Since all roles replaced in a split rewriting are non-simple and $\mathcal{I} \models q_{sr}$ by assumption, we have that $\mathcal{I} \models q_{co}$. By Lemma 15 (2), we then have that $\mathcal{I} \models q$ as required.

We go through the proof of the first claim in more detail: let q_{co} be in $\mathsf{co}(q)$ such that $\mathcal{I} \models^{\pi_{co}} q_{co}$ for a match π_{co} that is injective modulo $\stackrel{*}{\approx}$. Such a collapsing q_{co} and match π_{co} exist due to Lemma 15. If π_{co} is a split match w.r.t. q and $\mathcal{I}$ already, we are done, since a split match induces a root splitting R and (q_{co}, R) is trivially in $\mathsf{sr}_{\mathcal{K}}(q)$. If π_{co} is not a split match, there are at least two terms t, t' with $r(t, t') \mathrel{\bar{\in}} q_{co}$ such that $\pi_{co}(t) = (a, w), \pi_{co}(t') = (a', w'), a \neq a'$, and $w \neq \varepsilon$ or $w' \neq \varepsilon$. We distinguish two cases:

1. Both t and t' are not mapped to roots, i.e., $w \neq \varepsilon$ and $w' \neq \varepsilon$. Since $\mathcal{I} \models^{\pi_{co}} r(t, t')$, we have that $(\pi_{co}(t), \pi_{co}(t')) \in r^{\mathcal{I}}$. Since $\mathcal{I}$ is a canonical model for $\mathcal{K}$, there must be a role s with $s \,\underline{\circledast}_{\mathcal{R}} r$ and $s \in \mathsf{Trans}_{\mathcal{R}}$ such that

$$\{(\pi_{co}(t), (a, \varepsilon)),\ ((a, \varepsilon), (a', \varepsilon)),\ ((a', \varepsilon), \pi_{co}(t'))\} \subseteq s^{\mathcal{I}}.$$

If there is some $\hat{t} \in \mathsf{Terms}(q_{co})$ such that $\pi_{co}(\hat{t}) = (a, \varepsilon)$, then let $u = \hat{t}$, otherwise let u be a fresh variable. Similarly, if there is some $\hat{t'} \in \mathsf{Terms}(q_{co})$ such that $\pi_{co}(\hat{t'}) = (a', \varepsilon)$, then let $u' = \hat{t'}$, otherwise let u' be a fresh variable. Hence, we can define a split rewriting q_{sr} of q_{co} by replacing $r(t, t')$ with $s(t, u)$, $s(u, u')$, and $s(u', t')$. We then define a new mapping π_{sr} that agrees with π_{co} on all terms that occur in q_{co} and that maps u to (a, ε) and u' to (a', ε).

2. Either t or t' is mapped to a root. W.l.o.g., let this be t, i.e., $\pi(t) = (a, \varepsilon)$. We can use the same arguments as above: since $\mathcal{I} \models^{\pi_{co}} r(t, t')$, we have that $(\pi(t), \pi(t')) \in r^{\mathcal{I}}$ and, since $\mathcal{I}$ is a canonical model for $\mathcal{K}$, there must be a role s with $s \,\underline{\circledast}_{\mathcal{R}} r$ and $s \in \mathsf{Trans}_{\mathcal{R}}$ such that $\{(\pi(t), (a', \varepsilon)),\ ((a', \varepsilon), \pi(t'))\} \subseteq s^{\mathcal{I}}$. If there is some $\hat{t} \in \mathsf{Terms}(q_{co})$ such that $\pi_{co}(\hat{t}) = (a', \varepsilon)$, then let $u = \hat{t}$, otherwise let u be a fresh variable. We then define a split rewriting q_{sr} of q_{co} by replacing $r(t, t')$ with $s(t, u), s(u, t')$and a mapping π_{sr} that agrees with π_{co} on all terms that occur in q_{co} and that maps u to (a', ε).

It immediately follows that $\mathcal{I} \models^{\pi_{sr}} q_{sr}$. We can proceed as described above for each role atom $r(t, t')$ for which $\pi(t) = (a, w)$ and $\pi(t') = (a', w')$ with $a \neq a'$ and $w \neq \varepsilon$ or $w' \neq \varepsilon$. This will result in a split rewriting q_{sr} and a split match π_{sr} such that $\mathcal{I} \models^{\pi_{sr}} q_{sr}$. Furthermore, π_{sr} is injective modulo $\overset{*}{\approx}$ since we only introduce new variables, when the variable is mapped to an element that is not yet in the range of the match. Since π_{sr} is a split match, it induces a root splitting R and, hence, $(q_{sr}, R) \in \mathsf{sr}_{\mathcal{K}}(q)$ as required. □

Lemma (17)**.** *Let $\mathcal{I}$ be a model of $\mathcal{K}$.*

1. *If $\mathcal{I}$ is canonical and $\mathcal{I} \models q$, then there is a pair $(q_{\ell r}, R) \in \mathsf{lr}_{\mathcal{K}}(q)$ and a mapping $\pi_{\ell r}$ such that $\mathcal{I} \models^{\pi_{\ell r}} q_{\ell r}$, $\pi_{\ell r}$ is an injection modulo $\overset{*}{\approx}$, R is the root splitting induced by $\pi_{\ell r}$ and, for each $r(t, t) \,\bar{\in}\, q_{\ell r}$, $t \in R$.*

2. *If $(q_{\ell r}, R) \in \mathsf{lr}_{\mathcal{K}}(q)$ and $\mathcal{I} \models^{\pi_{\ell r}} q_{\ell r}$ for some match $\pi_{\ell r}$, then $\mathcal{I} \models q$.*

Proof of Lemma 17. The proof of (2) is analogous to the one given in Lemma 16 since, by definition of loop rewritings, all roles replaced in a loop rewriting are again non-simple.

For (1), let $(q_{sr}, R) \in \mathsf{sr}_{\mathcal{K}}(q)$ be such that $\mathcal{I} \models^{\pi_{sr}} q_{sr}$, π_{sr} is a split match, and R is the root splitting induced by π_{sr}. Such a split rewriting q_{sr} and match π_{sr} exist due to Lemma 16 and the canonicity of $\mathcal{I}$.

Let $r(t, t) \,\bar{\in}\, q_{sr}$ for $t \notin R$. Since R is the root splitting induced by π_{sr} and since $t \notin R$, $\pi_{sr}(t) = (a, w)$ for some $a \in \mathsf{Inds}(\mathcal{A})$ and $w \neq \varepsilon$. Now, let $\mathcal{J}$ be a forest base for $\mathcal{I}$. We show that there exists a neighbor d of $\pi_{sr}(t)$ and a role $s \in \mathsf{Trans}_{\mathcal{R}}$ such that $s \,\underline{\circledast}_{\mathcal{R}} r$ and $(\pi_{sr}(t), d) \in s^{\mathcal{I}} \cap \mathsf{Inv}(s)^{\mathcal{I}}$. Since $\mathcal{I} \models^{\pi_{sr}} q_{sr}$, we have $(\pi_{sr}(t), \pi_{sr}(t)) \in r^{\mathcal{I}}$. Since $\mathcal{J}$ is a forest base and since $w \neq \varepsilon$, we have $(\pi_{sr}(t), \pi_{sr}(t)) \notin r^{\mathcal{J}}$. It follows that there is a sequence $d_1, \ldots, d_n \in \Delta^{\mathcal{I}}$ and a role $s \in \mathsf{Trans}_{\mathcal{R}}$ such that $s \,\underline{\circledast}_{\mathcal{R}} r$, $d_1 = \pi_{sr}(t) = d_n$, and

$(d_i, d_{i+1}) \in s^{\mathcal{J}}$ for $1 \leq i < n$ and $d_i \neq d_1$ for each i with $1 < i < n$. Then it is not hard to see that, because $\{w' \mid (a, w') \in \Delta^{\mathcal{I}}\}$ is a tree and $w \neq \varepsilon$, we have $d_2 = d_{n-1}$. Since $(d_1, d_2) \in s^{\mathcal{J}}$ and $(d_{n-1}, d_n) \in s^{\mathcal{J}}$ with $d_{n-1} = d_2$ and $d_n = d_1$, the role s and the element $d = d_2$ is as required. For each $r(t, t) \bar{\in} q_{sr}$ with $t \notin R$, select an element $d_{r,t}$ and a role $s_{r,t}$ as described above. Now let $q_{\ell r}$ be obtained from q_{sr} by doing the following for each $r(t, t) \bar{\in} q_{sr}$ with $t \notin R$:

- if $d_{r,t} = \pi_{sr}(t')$ for some $t' \in \mathsf{Terms}(q_{sr})$, then replace $r(t, t)$ with $s_{r,t}(t, t')$ and $s_{r,t}(t', t)$;
- otherwise, introduce a new variable $v_{r,t} \in N_V$ and replace $r(t, t)$ with $s_{r,t}(t, v_{r,t})$ and $s_{r,t}(v_{r,t}, t)$.

Let $\pi_{\ell r}$ be obtained from π_{sr} by extending it with $\pi_{\ell r}(v_{r,t}) = d_{r,t}$ for each newly introduced variable $v_{r,t}$. By definition of $q_{\ell r}$ and $\pi_{\ell r}$, $q_{\ell r}$ is connected, $\pi_{\ell r}$ is injective modulo $\stackrel{*}{\approx}$, and $\mathcal{I} \models^{\pi_{\ell r}} q_{\ell r}$. $\square$

Lemma (18)**.** *Let $\mathcal{I}$ be a model of $\mathcal{K}$.*

1. *If $\mathcal{I}$ is canonical and $\mathcal{I} \models q$, then there is a pair $(q_{fr}, R) \in \mathsf{fr}_{\mathcal{K}}(q)$ such that $\mathcal{I} \models^{\pi_{fr}} q_{fr}$ for a forest match π_{fr}, R is the induced root splitting of π_{fr}, and π_{fr} is an injection modulo $\stackrel{*}{\approx}$.*
2. *If $(q_{fr}, R) \in \mathsf{fr}_{\mathcal{K}}(q)$ and $\mathcal{I} \models^{\pi_{fr}} q_{fr}$ for some match π_{fr}, then $\mathcal{I} \models q$.*

Proof of Lemma 18. The proof of (2) is again analogous to the one given in Lemma 16. For (1), let $(q_{\ell r}, R) \in \mathsf{lr}_{\mathcal{K}}(q)$ be such that $\mathcal{I} \models^{\pi_{\ell r}} q_{\ell r}$, R is the root splitting induced by $\pi_{\ell r}$, $\pi_{\ell r}$ is injective modulo $\stackrel{*}{\approx}$ and, for each $r(t, t) \bar{\in} q_{\ell r}$, $t \in R$. Such a loop rewriting and match $\pi_{\ell r}$ exist due to Lemma 17 and the canonicity of $\mathcal{I}$. By definition, R is a root splitting w.r.t. $q_{\ell r}$ and $\mathcal{K}$.

For $w, w' \in \mathbb{N}^*$, the *longest common prefix* (LCP) of w, w' is the longest $w^* \in \mathbb{N}^*$ such that w^* is prefix of both w and w'. For the match $\pi_{\ell r}$ we now define the set D as follows:

$$D := \mathsf{ran}(\pi_{\ell r}) \cup \{(a, w) \in \Delta^{\mathcal{I}} \mid \begin{array}{l} w \text{ is the LCP of some } w, w' \\ \text{with } (a, w'), (a, w'') \in \mathsf{ran}(\pi_{\ell r})\}. \end{array}$$

Let $V \subseteq N_V \setminus \mathsf{Vars}(q_{\ell r})$ be such that, for each $d \in D \setminus \mathsf{ran}(\pi_{\ell r})$, there is a unique $v_d \in V$. We now define a mapping π_{fr} as $\pi_{\ell r} \cup \{v_d \in V \mapsto d\}$. By definition of V and v_d, π_{fr} is a split match as well. The set $V \cup \mathsf{Vars}(q_{\ell r})$ will be the set of variables for the new query q_{fr}. Note that $\mathsf{ran}(\pi_{fr}) = D$.

Fact (a) if $(a, w), (a, w') \in \mathsf{ran}(\pi_{fr})$, then $(a, w'') \in \mathsf{ran}(\pi_{fr})$, where w'' is the LCP of w and w';

Fact (b) $\sharp(V) \leq \sharp(\mathsf{Vars}(q_{\ell r}))$ (Because, in the worst case, all (a, w) in $\mathsf{ran}(\pi_{\ell r})$ are "incomparable" and can thus be seen as leaves of a binarily branching tree. Now, a tree that has n leaves and is at least binarily branching at every non-leaf has at most n inner nodes, and thus $\sharp(V) \leq \sharp(\mathsf{Vars}(q_{\ell r}))$.

For a pair of individuals $d, d' \in \Delta^\mathcal{I}$, the *path* from d to d' is the (unique) shortest sequence of elements $d_1, \ldots, d_n \in \Delta^\mathcal{I}$ such that $d_1 = d$, $d_n = d'$, and d_{i+1} is a neighbor of d_i for all $1 \leq i < n$. The *length of a path* is the number of elements in it, i.e., the path $d_1, \ldots, d_n$ is of length n. The *relevant path* $d'_1, \ldots, d'_\ell$ from d to d' is the sub-sequence of $d_1, \ldots, d_n$ that is obtained by dropping all elements $d_i \notin D$.

Claim 1. Let $r(t, t') \bar{\in} \mathsf{subq}(q_{\ell r}, t_r)$ for some $t_r \in R$ and let $d'_1, \ldots, d'_\ell$ be the relevant path from $d = d'_1 = \pi_{\ell r}(t)$ to $d' = d'_\ell = \pi_{\ell r}(t')$. If $\ell > 2$, there is a role $s \in \mathsf{Trans}_\mathcal{R}$ such that $s \mathrel{\underline{\circledast}}_\mathcal{R} r$ and $(d'_i, d'_{i+1}) \in s^\mathcal{I}$ for all $1 \leq i < \ell$.

Proof. Let $d_1, \ldots, d_n$ be the path and $d'_1, \ldots, d'_\ell$ the relevant path from $\pi_{\ell r}(t)$ to $\pi_{\ell r}(t')$. Then $\ell > 2$ implies $n > 2$. We have to show that there is a role s as in the claim. Let $\mathcal{J}$ be a forest base for $\mathcal{I}$. Since $\mathcal{I} \models^{\pi_{\ell r}} q_{\ell r}$, $n > 2$ implies $(\pi_{\ell r}(t), \pi_{\ell r}(t')) \in r^\mathcal{I} \setminus r^\mathcal{J}$. Since $\mathcal{I}$ is based on $\mathcal{J}$, it follows that there is an $s \in \mathsf{Trans}_\mathcal{R}$ such that $s \mathrel{\underline{\circledast}}_\mathcal{R} r$, and $(d_i, d_{i+1}) \in s^\mathcal{J}$ for all $1 \leq i < n$. By construction of $\mathcal{I}$ from $\mathcal{J}$, it follows that $(d'_i, d'_{i+1}) \in s^\mathcal{I}$ for all $1 \leq i < \ell$, which finishes the proof of the claim.

Now let q_{fr} be obtained from $q_{\ell r}$ as follows: for each role atom $r(t, t) \bar{\in} \mathsf{subq}(q_{\ell r}, t_r)$ with $t_r \in R$, if the length of the relevant path $d'_1, \ldots, d'_\ell$ from $d = d'_1 = \pi_{\ell r}(t)$ to $d' = d'_\ell = \pi_{\ell r}(t')$ is greater than 2, then select a role s and variables $t_j \in D$ such that $\pi_{fr}(t_j) = d'_j$ as in Claim 1 and replace the atom $r(t, t')$ with $s(t_1, t_2), \ldots, s(t_{\ell-1}, t_\ell)$, where $t = t_1$, $t' = t_\ell$. Please note that these t_j can be chosen in a "don't care" non-deterministic way since π_{fr} is injective modulo $\stackrel{*}{\approx}$, i.e., if $\pi_{fr}(t_j) = d_j = \pi_{fr}(t'_j)$, then $t_j \stackrel{*}{\approx} t'_j$ and we can pick any of these.

We now have to show that

(i) $\mathcal{I} \models^{\pi_{fr}} q_{fr}$, and

(ii) π_{fr} is a forest match.

For (i), let $r(t, t') \bar{\in} q_{\ell r} \setminus q_{fr}$ and let $s(t_1, t_2), \ldots, s(t_{\ell-1}, t_\ell)$ be the atoms that replaced $r(t, t')$. Since $\mathcal{I} \models^{\pi_{\ell r}} q_{\ell r}$, $\mathcal{I} \models^{\pi_{\ell r}} r(t, t')$ and $(\pi_{\ell r}(t), \pi_{\ell r}(t')) \in r^\mathcal{I}$. Since $r(t, t')$ was replaced in q_{fr}, the length of the relevant path from $\pi_{\ell r}(t)$ to $\pi_{\ell r}(t')$ is greater than 2. Hence, it must be the case that $(\pi_{\ell r}(t), \pi_{\ell r}(t')) \in r^\mathcal{I} \setminus r^\mathcal{J}$. Let $d_1, \ldots, d_n$ with $d_1 = \pi_{\ell r}(t)$ and $d_n = \pi_{\ell r}(t')$ be the path from $\pi_{\ell r}(t)$ to $\pi_{\ell r}(t')$ and $d'_1, \ldots, d'_\ell$ the relevant path from $\pi_{\ell r}(t)$ to $\pi_{\ell r}(t')$. By construction of $\mathcal{I}$ from $\mathcal{J}$, this means that there is a role $s \in \mathsf{Trans}_\mathcal{R}$ such that $s \mathrel{\underline{\circledast}}_\mathcal{R} r$ and $(d_i, d_{i+1}) \in s^\mathcal{J}$ for all $1 \leq i < n$. Again by construction of $\mathcal{I}$, this means $(d'_i, d'_{i+1}) \in s^\mathcal{I}$ for $1 \leq i < \ell$ as required. Hence $\mathcal{I} \models^{\pi_{fr}} s(t_i, t_{i+1})$ for each i with $\ \leq i < \ell$ by definition of π_{fr}.

For (ii): the mapping π_{fr} differs from $\pi_{\ell r}$ only for the newly introduced variables. Furthermore, we only introduced new role atoms within a sub-query $\mathsf{subq}(q_{\ell r}, t_r)$ and $\pi_{\ell r}$ is a split match by assumption. Hence, π_{fr} is trivially a split match and we only have to show that π_{fr} is a forest match. Since π_{fr} is a split match, we can do this "tree by tree".

For each $a \in \mathsf{Inds}(\mathcal{A})$, let $T_a := \{w \mid (a, w) \in \mathsf{ran}(\pi_{fr})\}$. We need to construct a mapping f as specified in Definition 14, and we start with its root t_r. If $T_a \neq \emptyset$, let $t_r \in \mathsf{Terms}(q)$ be the unique term such that $\pi_{fr}(t_r) = (a, w_r)$ and there is no $t \in \mathsf{Terms}(q)$ such that $\pi_{fr}(t) = (a, w)$ and w is a proper prefix of w_r. Such a term exists since π_{fr} is a split match and it is unique due to Fact (a) above. Define a *trace* to be a sequence $\bar{w} = w_1 \cdots w_n \in T_a^+$ such that

- $w_1 = w_r$;

- for all $1 \leq i < n$, w_i is the longest proper prefix of w_{i+1}.

Since $\mathcal{I}$ is canonical, each $w_i \in T_a$ is in $\mathbb{N}$. It is not hard to see that $T = \{\bar{w} \mid \bar{w} \text{ is a trace}\} \cup \{\varepsilon\}$ is a tree. For a trace $\bar{w} = w_1 \cdots w_n$, let $\mathsf{Tail}(\bar{w}) = w_n$. Define a mapping f that maps each term t with $\pi_{fr}(t) = (a, w) \in T_a$ to the unique trace $\bar{w}_t$ such that $w = \mathsf{Tail}(\bar{w}_t)$. Let $r(t, t') \in q_{fr}$ such that $\pi_{fr}(t), \pi_{fr}(t') \in T_a$. By construction of q_{fr}, this implies that the length of the relevant path from $\pi_{fr}(t)$ to $\pi_{fr}(t')$ is exactly 2. Thus, $f(t)$ and $f(t')$ are neighbors in T and, hence, π_{fr} is a forest match as required. □

Theorem (19)**.** *Let $\mathcal{K}$ be a $\mathcal{SHIQ}$ knowledge base, q a Boolean conjunctive query, and $\{q_1, \ldots, q_\ell\} = \mathsf{trees}_\mathcal{K}(q) \cup \mathsf{ground}_\mathcal{K}(q)$. Then $\mathcal{K} \models q$ iff $\mathcal{K} \models q_1 \vee \ldots \vee q_\ell$.*

Proof of Theorem 19. For the “if” direction: let us assume that $\mathcal{K} \models q_1 \vee \ldots \vee q_\ell$. Hence, for each model $\mathcal{I}$ of $\mathcal{K}$, there is a query q_i with $1 \leq i \leq \ell$ such that $\mathcal{I} \models q_i$. We distinguish two cases: (i) $q_i \in \mathsf{trees}_\mathcal{K}(q)$ and (ii) $q_i \in \mathsf{ground}_\mathcal{K}(q)$.

For (i): q_i is of the form $C(v)$ where C is the query concept for some query q_{fr} w.r.t. $v \in \mathsf{Vars}(q_{fr})$ and $(q_{fr}, \emptyset) \in \mathsf{fr}_\mathcal{K}(q)$. Hence $\mathcal{I} \models^\pi q_i$ for some match π, and thus $\mathcal{I} \models^\pi C(v)$. Let $d \in \Delta^\mathcal{I}$ with $d = \pi(v) \in C^\mathcal{I}$. By Lemma 12, we then have that $\mathcal{I} \models q_{fr}$ and, by Lemma 18, we then have that $\mathcal{I} \models q$ as required.

For (ii): since $q_i \in \mathsf{ground}_\mathcal{K}(q)$, there is some pair $(q_{fr}, R) \in \mathsf{fr}_\mathcal{K}(q)$ such that $q_i = \mathsf{ground}(q_{fr}, R, \tau)$. We show that $\mathcal{I} \models^{\pi_{fr}} q_{fr}$ for some match π_{fr}. Since $\mathcal{I} \models q_1$, there is a match π_i such that $\mathcal{I} \models^{\pi_i} q_i$. We now construct the match π_{fr}. For each $t \in R$, q_i contains a concept atom $C(\tau(t))$ where $C = \mathsf{con}(\mathsf{subq}(q_{fr}, t), t)$ is the query concept of $\mathsf{subq}(q_{fr}, t)$ w.r.t. t. Since $\mathcal{I} \models^{\pi_i} C(\tau(t))$ and by Lemma 12, there is a match π_t such that $\mathcal{I} \models^{\pi_t} \mathsf{subq}(q_{fr}, t)$. We now define π_{fr} as the union of π_t, for each $t \in R$. Please note that $\pi_{fr}(t) = \pi_i(\tau(t))$. Since $\mathsf{Inds}(q_{fr}) \subseteq R$ and τ is such that, for each $a \in \mathsf{Inds}(q_{fr})$, $\tau(a) = a$ and $\tau(t) = \tau(t')$ iff $t \stackrel{*}{\approx} t'$, it follows that $\mathcal{I} \models^{\pi_{fr}} at$ for each atom $at \mathrel{\bar{\in}} q_{fr}$ such that at contains only terms from the root choice R and hence $\mathcal{I} \models^{\pi_{fr}} q_{fr}$ as required.

For the “only if” direction we have to show that, if $\mathcal{K} \models q$, then $\mathcal{K} \models q_1 \vee \ldots \vee q_\ell$, so let us assume that $\mathcal{K} \models q$. By Lemma 7 in its negated form we have that $\mathcal{K} \models q$ iff all canonical models $\mathcal{I}$ of $\mathcal{K}$ are such that $\mathcal{I} \models q$. Hence, we can restrict our attention to the canonical models of $\mathcal{K}$. By Lemma 18, $\mathcal{I} \models \mathcal{K}$ and $\mathcal{I} \models q$ implies that there is a pair $(q_{fr}, R) \in \mathsf{fr}_\mathcal{K}(q)$ such that $\mathcal{I} \models^{\pi_{fr}} q_{fr}$ for a forest match π_{fr}, R is the induced root splitting of π_{fr}, and π_{fr} is an injection modulo $\stackrel{*}{\approx}$. We again distinguish two cases:

(i) $R = \emptyset$, i.e., the root splitting is empty and π_{fr} is a tree match, and

(ii) $R \neq \emptyset$, i.e., the root splitting is non-empty and π_{fr} is a forest match but not a tree match.

For (i): since $(q_{fr}, \emptyset) \in \mathsf{fr}_\mathcal{K}(q)$, there is some $v \in \mathsf{Terms}(q_{fr})$ such that $C = \mathsf{con}(q_{fr}, v)$ and $q_i = C(v)$. By Lemma 12 and, since $\mathcal{I} \models q_{fr}$, there is an element $d \in \Delta^\mathcal{I}$ such that $d \in C^\mathcal{I}$. Hence $\mathcal{I} \models^\pi C(v)$ with $\pi : v \mapsto d$ as required.

For (ii): since R is the root splitting induced by π_{fr}, for each $t \in R$ there is some $a_t \in \mathsf{Inds}(\mathcal{A})$ such that $\pi_{fr}(t) = (a_t, \varepsilon)$. We now define the mapping $\tau \colon R \to \mathsf{Inds}(\mathcal{A})$ as follows: for each $t \in R$, $\tau(t) = a_t$ iff $\pi_{fr}(t) = (a_t, \varepsilon)$. By definition of $\mathsf{ground}(q_{fr}, R, \tau)$, $q_i = \mathsf{ground}(q_{fr}, R, \tau) \in \mathsf{ground}_\mathcal{K}(q)$. Since $\mathcal{I} \models^{\pi_{fr}} q_{fr}$, $\mathcal{I} \models \mathsf{subq}(q_{fr}, t)$ for each $t \in R$.

Since q_{fr} is forest-shaped, each $\mathsf{subq}(q_{fr}, t)$ is tree-shaped. Then, by Lemma 12, $\mathcal{I} \models q'_i$, where q'_i is the query obtained from q_{fr} by replacing each sub-query $\mathsf{subq}(q_{fr}, t)$ with $C(t)$ for C the query concept of $\mathsf{subq}(q_{fr}, t)$ w.r.t. t. By definition of τ from the forest match π_{fr}, it is clear that $\mathcal{I} \models \mathsf{ground}(q_{fr}, R, \tau)$ as required. □

Lemma (20)**.** *Let q be a Boolean conjunctive query, $\mathcal{K} = (\mathcal{T}, \mathcal{R}, \mathcal{A})$ a $\mathcal{SHIQ}$ knowledge base, $|q| := n$ and $|\mathcal{K}| := m$. Then there is a polynomial p such that*

1. $\sharp(\mathsf{co}(q)) \leq 2^{p(n)}$ *and, for each* $q' \in \mathsf{co}(q)$, $|q'| \leq p(n)$,
2. $\sharp(\mathsf{sr}_\mathcal{K}(q)) \leq 2^{p(n)\cdot \log p(m)}$, *and, for each* $q' \in \mathsf{sr}_\mathcal{K}(q)$, $|q'| \leq p(n)$,
3. $\sharp(\mathsf{lr}_\mathcal{K}(q)) \leq 2^{p(n)\cdot \log p(m)}$, *and, for each* $q' \in \mathsf{lr}_\mathcal{K}(q)$, $|q'| \leq p(n)$,
4. $\sharp(\mathsf{fr}_\mathcal{K}(q)) \leq 2^{p(n)\cdot \log p(m)}$, *and, for each* $q' \in \mathsf{fr}_\mathcal{K}(q)$, $|q'| \leq p(n)$,
5. $\sharp(\mathsf{trees}_\mathcal{K}(q)) \leq 2^{p(n)\cdot \log p(m)}$, *and, for each* $q' \in \mathsf{trees}_\mathcal{K}(q)$, $|q'| \leq p(n)$, *and*
6. $\sharp(\mathsf{ground}_\mathcal{K}(q)) \leq 2^{p(n)\cdot \log p(m)}$, *and, for each* $q' \in \mathsf{ground}_\mathcal{K}(q)$, $|q'| \leq p(n)$.

Proof of Lemma 20.

1. The set $\mathsf{co}(q)$ contains those queries obtained from q by adding at most n equality atoms to q. The number of collapsings corresponds, therefore, to building all equivalence classes over the terms in q by $\stackrel{*}{\approx}$. Hence, the cardinality of the set $\mathsf{co}(q)$ is at most exponential in n. Since we add at most one equality atom for each pair of terms, the size of a query $q' \in \mathsf{co}(q)$ is at most $n + n^2$, and $|q'|$ is, therefore, polynomial in n.

2. For each of the at most n role atoms, we can choose to do nothing, replace the atom with two atoms, or with three atoms. For every replacement, we can choose to introduce a new variable or re-use one of the existing variables. If we introduce a new variable every time, the new query contains at most $3n$ terms. Since $\mathcal{K}$ can contain at most m non-simple roles that are a sub-role of a role used in role atoms of q, we have at most m roles to choose from when replacing a role atom. Overall, this gives us at most $1 + m(3n) + m(3n)(3n)$ choices for each of the at most n role atoms in a query and, therefore, the number of split rewritings for each query $q' \in \mathsf{co}(q)$ is polynomial in m and exponential in n. In combination with the results from (1), this also shows that the overall number of split rewritings is polynomial in m and exponential in n.

   Since we add at most two new role atoms for each of the existing role atoms, the size of a query $q' \in \mathsf{sr}_\mathcal{K}(q)$ is linear in n.

3. There are at most n role atoms of the form $r(t, t)$ in a query $q' \in \mathsf{sr}_\mathcal{K}(q)$ that could give rise to a loop rewriting, at most m non-simple sub-roles of r in $\mathcal{K}$ that can be used in the loop rewriting, and we can introduce at most one new variable for each role atom $r(t, t)$. Therefore, for each query in $\mathsf{sr}_\mathcal{K}(q)$, the number of loop rewritings is again polynomial in m and exponential in n. Combined with the results from (2), this bound also holds for the cardinality of $\mathsf{lr}_\mathcal{K}(q)$.

   In a loop rewriting, one role atom is replaced with two role atoms, hence, the size of a query $q' \in \mathsf{lr}_\mathcal{K}(q)$ at most doubles.

4. We can use similar arguments as above in order to derive a bound that is exponential in n and polynomial in m for the number of forest rewritings in $\mathsf{fr}_{\mathcal{K}}(q)$.

   Since the number of role atoms that we can introduce in a forest rewriting is polynomial in n, the size of each query $q' \in \mathsf{fr}_{\mathcal{K}}(q)$ is at most quadratic in n.

5. The cardinality of the set $\mathsf{trees}_{\mathcal{K}}(q)$ is clearly also polynomial in m and exponential in n since each query in $\mathsf{fr}_{\mathcal{K}}(q)$ can contribute at most one query to the set $\mathsf{trees}_{\mathcal{K}}(q)$. It is not hard to see that the size of a query $q' \in \mathsf{trees}_{\mathcal{K}}(q)$ is polynomial in n.

6. By (1)-(4) above, the number of terms in a root splitting is polynomial in n and there are at most m individual names occurring in $\mathcal{A}$ that can be used for the mapping τ from terms to individual names. Hence the number of different ground mappings τ is at most polynomial in m and exponential in n. The number of ground queries that a single tuple $(q_{fr}, R) \in \mathsf{fr}_{\mathcal{K}}(q)$ can contribute is, therefore, also at most polynomial in m and exponential in n. Together with the bound on the number of forest rewritings from (4), this shows that the cardinality of $\mathsf{ground}_{\mathcal{K}}(q)$ is polynomial in m and exponential in n. Again it is not hard to see that the size of each query $q' \in \mathsf{ground}_{\mathcal{K}}(q)$ is polynomial in n.

□

Lemma (23)**.** *Let $\mathcal{K}$ be a $\mathcal{SHIQ}$ knowledge base and q a union of connected Boolean conjunctive queries. The algorithm from Definition 22 answers "$\mathcal{K}$ entails q" iff $\mathcal{K} \models q$ under the unique name assumption.*

Proof of Lemma 23. For the "only if"-direction: let $q = q_1 \vee \ldots \vee q_\ell$. We show the contrapositive and assume that $\mathcal{K} \not\models q$. We can assume that $\mathcal{K}$ is consistent since an inconsistent knowledge base trivially entails every query. Let $\mathcal{I}$ be a model of $\mathcal{K}$ such that $\mathcal{I} \not\models q$. We show that $\mathcal{I}$ is also a model of some extended knowledge base $\mathcal{K}_q = (\mathcal{T} \cup \mathcal{T}_q, \mathcal{R}, \mathcal{A} \cup \mathcal{A}_q)$. We first show that $\mathcal{I}$ is a model of $\mathcal{T}_q$. To this end, let $\top \sqsubseteq \neg C$ in $\mathcal{T}_q$. Then $C(v) \in T$ and $C = \mathsf{con}(q_{fr}, v)$ for some pair $(q_{fr}, \emptyset) \in \mathsf{fr}_{\mathcal{K}}(q_1) \cup \ldots \cup \mathsf{fr}_{\mathcal{K}}(q_\ell)$ and $v \in \mathsf{Vars}(q_{fr})$. Let i be such that $(q_{fr}, \emptyset) \in \mathsf{fr}_{\mathcal{K}}(q_i)$. Now $C^{\mathcal{I}} \neq \emptyset$ implies, by Lemma 12, that $\mathcal{I} \models q_{fr}$ and, by Lemma 18, $\mathcal{I} \models q_i$ and, hence, $\mathcal{I} \models q$, contradicting our assumption. Thus $\mathcal{I} \models \top \sqsubseteq \neg C$ and, thus, $\mathcal{I} \models \mathcal{T}_q$.

Next, we define an extended ABox $\mathcal{A}_q$ such that, for each $q' \in G$,

- if $C(a) \in q'$ and $a^{\mathcal{I}} \in \neg C^{\mathcal{I}}$, then $\neg C(a) \in \mathcal{A}_q$;
- if $r(a, b) \in q'$ and $(a^{\mathcal{I}}, b^{\mathcal{I}}) \notin r^{\mathcal{I}}$, then $\neg r(a, b) \in \mathcal{A}_q$.

Now assume that we can have a query $q' = \mathsf{ground}(q_{fr}, R, \tau) \in \mathsf{ground}_{\mathcal{K}}(q_1) \cup \ldots \cup \mathsf{ground}_{\mathcal{K}}(q_\ell)$ such that there is no atom $at \in q'$ with $\neg at \in \mathcal{A}_q$. Then trivially $\mathcal{I} \models q'$. Let i be such that $(q_{fr}, R) \in \mathsf{fr}_{\mathcal{K}}(q_i)$. By Theorem 19, $\mathcal{I} \models q_i$ and thus $\mathcal{I} \models q$, which is a contradiction. Hence $\mathcal{K}_q$ is an extended knowledge base and $\mathcal{I} \models \mathcal{K}_q$ as required.

For the "if"-direction, we assume that $\mathcal{K} \models q$, but the algorithm answers "$\mathcal{K}$ does not entail q". Hence there is an extended knowledge base $\mathcal{K}_q = (\mathcal{T} \cup \mathcal{T}_q, \mathcal{R}, \mathcal{A} \cup \mathcal{A}_q)$ that is consistent, i.e., there is a model $\mathcal{I}$ such that $\mathcal{I} \models \mathcal{K}_q$. Since $\mathcal{K}_q$ is an extension of $\mathcal{K}$,

$\mathcal{I} \models \mathcal{K}$. Moreover, we have that $\mathcal{I} \models \mathcal{T}_q$ and hence, for each $d \in \Delta^\mathcal{I}$, $d \in \neg C^\mathcal{I}$ for each $C(v) \in \mathsf{trees}_\mathcal{K}(q_1) \cup \ldots \cup \mathsf{trees}_\mathcal{K}(q_\ell)$. By Lemma 12, we then have that $\mathcal{I} \not\models q'$ for each $q' \in \mathsf{trees}_\mathcal{K}(q_1) \cup \ldots \cup \mathsf{trees}_\mathcal{K}(q_\ell)$ and, by Lemma 18, $\mathcal{I} \not\models q_i$ for each i with $1 \leq i \leq \ell$.

By definition of extended knowledge bases, $\mathcal{A}_q$ contains an assertion $\neg at$ for at least one atom at in each query $q' = \mathsf{ground}(q_{fr}, R, \tau)$ from $\mathsf{ground}_\mathcal{K}(q_1) \cup \ldots \cup \mathsf{ground}_\mathcal{K}(q_\ell)$. Hence $\mathcal{I} \not\models q'$ for each $q' \in \mathsf{ground}_\mathcal{K}(q_1) \cup \ldots \cup \mathsf{ground}_\mathcal{K}(q_\ell)$. Then, by Theorem 19, $\mathcal{I} \not\models q$, which contradicts our assumption. □

Lemma (25)**.** *Let $\mathcal{R}$ be a role hierarchy, and $r_1, \ldots, r_n$ roles. For every interpretation $\mathcal{I}$ such that $\mathcal{I} \models \mathcal{R}$, it holds that $(\uparrow(r_1 \sqcap \ldots \sqcap r_n, \mathcal{R}))^\mathcal{I} = (r_1 \sqcap \ldots \sqcap r_n)^\mathcal{I}$.*

Proof of Lemma 25. The proof is a straightforward extension of Lemma 6.19 by Tobies (2001). By definition, $\uparrow(r_1 \sqcap \ldots \sqcap r_n, \mathcal{R}) = \uparrow(r_1, \mathcal{R}) \sqcap \ldots \sqcap \uparrow(r_n, \mathcal{R})$ and, by definition of the semantics of role conjunctions, we have that $(\uparrow(r_1, \mathcal{R}) \sqcap \ldots \sqcap \uparrow(r_n, \mathcal{R}))^\mathcal{I} = \uparrow(r_1, \mathcal{R})^\mathcal{I} \cap \ldots \cap \uparrow(r_n, \mathcal{R})^\mathcal{I}$. If $s \sqsubseteq^*_\mathcal{R} r$, then $\{s' \mid r \sqsubseteq^*_\mathcal{R} s'\} \subseteq \{s' \mid s \sqsubseteq^*_\mathcal{R} s'\}$ and hence $\uparrow(s, \mathcal{R})^\mathcal{I} \subseteq \uparrow(r, \mathcal{R})^\mathcal{I}$. If $\mathcal{I} \models \mathcal{R}$, then $r^\mathcal{I} \subseteq s^\mathcal{I}$ for every s with $r \sqsubseteq^*_\mathcal{R} s$. Hence, $\uparrow(r, \mathcal{R})^\mathcal{I} = r^\mathcal{I}$ and $(\uparrow(r_1 \sqcap \ldots \sqcap r_n, \mathcal{R}))^\mathcal{I} = (\uparrow(r_1, \mathcal{R}) \sqcap \ldots \sqcap \uparrow(r_n, \mathcal{R}))^\mathcal{I} = \uparrow(r_1, \mathcal{R})^\mathcal{I} \cap \ldots \cap \uparrow(r_n, \mathcal{R})^\mathcal{I} = {r_1}^\mathcal{I} \cap \ldots \cap {r_n}^\mathcal{I} = (r_1 \sqcap \ldots \sqcap r_n)^\mathcal{I}$ as required. □

Lemma (28)**.** *Given a $\mathcal{SHIQ}^\sqcap$ knowledge base $\mathcal{K} = (\mathcal{T}, \mathcal{R}, \mathcal{A})$ where $m := |\mathcal{K}|$ and the size of the longest role conjunction is n, we can decide consistency of $\mathcal{K}$ in deterministic time $2^{p(m)2^{p(n)}}$ with p a polynomial.*

Proof of Lemma 28. We first translate $\mathcal{K}$ into an $\mathcal{ALCQI}b$ knowledge base $\mathsf{tr}(\mathcal{K}, \mathcal{R}) = (\mathsf{tr}(\mathcal{T}, \mathcal{R}), \mathsf{tr}(\mathcal{A}, \mathcal{R}))$. Since the longest role conjunction is of size n, the cardinality of each set $\mathsf{tc}(R, \mathcal{R})$ for a role conjunction R is bounded by m^n. Hence, the TBox $\mathsf{tr}(\mathcal{T}, \mathcal{R})$ can contain exponentially many axioms in n. It is not hard to check that the size of each axiom is polynomial in m. Since deciding whether an $\mathcal{ALCQI}b$ KB is consistent is an ExpTime-complete problem (even with binary coding of numbers) (Tobies, 2001, Theorem 4.42), the consistency of $\mathsf{tr}(\mathcal{K}, \mathcal{R})$ can be checked in time $2^{p(m)2^{p(n)}}$. □

Lemma (29)**.** *Let $\mathcal{K} = (\mathcal{T}, \mathcal{R}, \mathcal{A})$ be a $\mathcal{SHIQ}$ knowledge base with $m := |\mathcal{K}|$ and q a union of connected Boolean conjunctive queries with $n := |q|$. The algorithm given in Definition 22 decides whether $\mathcal{K} \models q$ under the unique name assumption in deterministic time in $2^{p(m)2^{p(n)}}$.*

Proof of Lemma 29. We first show that there is some polynomial p such that we have to check at most $2^{p(m)2^{p(n)}}$ extended knowledge bases for consistency and then that each consistency check can be done in time $2^{p(m)2^{p(n)}}$, which gives an upper bound of $2^{p(m)2^{p(n)}}$ on the time needed for deciding whether $\mathcal{K} \models q$.

Let $q := q_1 \vee \ldots \vee q_\ell$. Clearly, we can use n as a bound for ℓ, i.e., $\ell \leq n$. Moreover, the size of each query q_i with $1 \leq i \leq \ell$ is bounded by n. Together with Lemma 20, we get that $\sharp(T)$ and $\sharp(G)$ are bounded by $2^{p(n) \cdot \log p(m)}$ for some polynomial p and it is clear that the sets can be computed in this time bound as well. The size of each query $q' \in G$ w.r.t. an ABox $\mathcal{A}$ is polynomial in n and, when constructing $\mathcal{A}_q$, we can add a subset of (negated) atoms from each $q' \in G$ to $\mathcal{A}_q$. Hence, there are at most $2^{p(m)2^{p(n)}}$ extended ABoxes $\mathcal{A}_q$ and, therefore, $2^{p(m)2^{p(n)}}$ extended knowledge bases that have to be tested for consistency.

Due to Lemma 20 (5), the size of each query $q' \in T$ is polynomial in n. Computing a query concept $C_{q'}$ of q' w.r.t. some variable $v \in \mathsf{Vars}(q')$ can be done in time polynomial in n. Thus the TBox $\mathcal{T}_q$ can be computed in time $2^{p(n)\cdot\log p(m)}$. The size of an extended ABox $\mathcal{A}_q$ is maximal if we add, for each of the $2^{p(n)\cdot\log p(m)}$ ground queries in G, all atoms in their negated form. Since, by Lemma 20 (6), the size of these queries is polynomial in n, the size of each extended ABox $\mathcal{A}_q$ is bounded by $2^{p(n)\cdot\log p(m)}$ and it is clear that we can compute an extended ABox in this time bound as well. Hence, the size of each extended KB $\mathcal{K}_q = (\mathcal{T} \cup \mathcal{T}_q, \mathcal{R}, \mathcal{A} \cup \mathcal{A}_q)$ is bounded by $2^{p(n)\cdot\log p(m)}$. Since role conjunctions occur only in $\mathcal{T}_q$ or $\mathcal{A}_q$, and the size of each concept in $\mathcal{T}_q$ and $\mathcal{A}_q$ is polynomial in n, the length of the longest role conjunction is also polynomial in n.

When translating an extended knowledge base into an $\mathcal{ALCQI}b$ knowledge base, the number of axioms resulting from each concept C that occurs in $\mathcal{T}_q$ or $\mathcal{A}_q$ can be exponential in n. Thus, the size of each extended knowledge base is bounded by $2^{p(n)\cdot\log p(m)}$.

Since deciding whether an $\mathcal{ALCQI}b$ knowledge base is consistent is an ExpTime-complete problem (even with binary coding of numbers) (Tobies, 2001, Theorem 4.42), it can be checked in time $2^{p(m)2^{p(n)}}$ if $\mathcal{K}$ is consistent or not.

Since we have to check at most $2^{p(m)2^{p(n)}}$ knowledge bases for consistency, and each check can be done in time $2^{p(m)2^{p(n)}}$, we obtain the desired upper bound of $2^{p(m)2^{p(n)}}$ for deciding whether $\mathcal{K} \models q$. □

Lemma (31). *Let $\mathcal{K} = (\mathcal{T}, \mathcal{R}, \mathcal{A})$ be a $\mathcal{SHIQ}$ knowledge base and q a union of Boolean conjunctive queries. $\mathcal{K} \not\models q$ without making the unique name assumption iff there is an $\mathcal{A}$-partition $\mathcal{K}^{\mathcal{P}} = (\mathcal{T}, \mathcal{R}, \mathcal{A}^{\mathcal{P}})$ and $q^{\mathcal{P}}$ w.r.t. $\mathcal{K}$ and q such that $\mathcal{K}^{\mathcal{P}} \not\models q^{\mathcal{P}}$ under the unique name assumption.*

Proof of Lemma 31. For the “only if”-direction: Since $\mathcal{K} \not\models q$, there is a model $\mathcal{I}$ of $\mathcal{K}$ such that $\mathcal{I} \not\models q$. Let $f\colon \mathsf{Inds}(\mathcal{A}) \to \mathsf{Inds}(\mathcal{A})$ be a total function such that, for each set of individual names $\{a_1, \ldots, a_n\}$ for which $a_1{}^{\mathcal{I}} = a_i{}^{\mathcal{I}}$ for $1 \leq i \leq n$, $f(a_i) = a_1$. Let $\mathcal{A}^{\mathcal{P}}$ and $q^{\mathcal{P}}$ be obtained from $\mathcal{A}$ and q by replacing each individual name a in $\mathcal{A}$ and q with $f(a)$. Clearly, $\mathcal{K}^{\mathcal{P}} = (\mathcal{T}, \mathcal{R}, \mathcal{A}^{\mathcal{P}})$ and $q^{\mathcal{P}}$ are an $\mathcal{A}$-partition w.r.t. $\mathcal{K}$ and q. Let $\mathcal{I}^{\mathcal{P}} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}^{\mathcal{P}}})$ be an interpretation that is obtained by restricting $\cdot^{\mathcal{I}}$ to individual names in $\mathsf{Inds}(\mathcal{A}^{\mathcal{P}})$. It is easy to see that $\mathcal{I}^{\mathcal{P}} \models \mathcal{K}^{\mathcal{P}}$ and that the unique name assumption holds in $\mathcal{I}^{\mathcal{P}}$. We now show that $\mathcal{I}^{\mathcal{P}} \not\models q^{\mathcal{P}}$. Assume, to the contrary of what is to be shown, that $\mathcal{I}^{\mathcal{P}} \models^{\pi'} q^{\mathcal{P}}$ for some match π'. We define a mapping $\pi\colon \mathsf{Terms}(q) \to \Delta^{\mathcal{I}}$ from π' such $\pi(a) = \pi'(f(a))$ for each individual name $a \in \mathsf{Inds}(q)$ and $\pi(v) = \pi'(v)$ for each variable $v \in \mathsf{Vars}(q)$. It is easy to see that $\mathcal{I} \models^{\pi} q$, which is a contradiction.

For the “if”-direction: Let $\mathcal{I}^{\mathcal{P}} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}^{\mathcal{P}}})$ be such that $\mathcal{I}^{\mathcal{P}} \models \mathcal{K}^{\mathcal{P}}$ under UNA and $\mathcal{I}^{\mathcal{P}} \not\models q^{\mathcal{P}}$ and let $f\colon \mathsf{Inds}(\mathcal{A}) \to \mathsf{Inds}(\mathcal{A}^{\mathcal{P}})$ be a total function such that $f(a)$ is the individual that replaced a in $\mathcal{A}^{\mathcal{P}}$ and $q^{\mathcal{P}}$. Let $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ be an interpretation that extends $\mathcal{I}^{\mathcal{P}}$ such that $a^{\mathcal{I}} = f(a)^{\mathcal{I}^{\mathcal{P}}}$. We show that $\mathcal{I} \models \mathcal{K}$ and that $\mathcal{I} \not\models q$. It is clear that $\mathcal{I} \models \mathcal{T}$. Let $C(a)$ be an assertion in $\mathcal{A}$ such that a was replaced with $a^{\mathcal{P}}$ in $\mathcal{A}^{\mathcal{P}}$. Since $\mathcal{I}^{\mathcal{P}} \models C(a^{\mathcal{P}})$ and $a^{\mathcal{I}} = f(a)^{\mathcal{I}^{\mathcal{P}}} = a^{\mathcal{P}\mathcal{I}^{\mathcal{P}}} \in C^{\mathcal{I}^{\mathcal{P}}}$, $\mathcal{I} \models C(a)$. We can use a similar argument for (possibly negated) role assertions. Let $a \not\approx b$ be an assertion in $\mathcal{A}$ such that a was replaced with $a^\mathcal{P}$ and b with $b^\mathcal{P}$ in $\mathcal{A}^\mathcal{P}$, i.e., $f(a) = a^\mathcal{P}$ and $f(b) = b^\mathcal{P}$. Since $\mathcal{I}^\mathcal{P} \models a^\mathcal{P} \not\approx b^\mathcal{P}$, $a^\mathcal{I} = f(a)^{\mathcal{I}^\mathcal{P}} = a^{\mathcal{P}\mathcal{I}^\mathcal{P}} \neq b^{\mathcal{P}\mathcal{I}^\mathcal{P}} = f(b)^{\mathcal{I}^\mathcal{P}} = b^\mathcal{I}$ and $\mathcal{I} \models a \not\approx b$ as required. Therefore, we have that $\mathcal{I} \models \mathcal{K}$ as required.

Assume that $\mathcal{I} \models^\pi q$ for a match π. Let $\pi^\mathcal{P} \colon \mathsf{Terms}(q^\mathcal{P}) \to \Delta^\mathcal{I}$ be a mapping such that $\pi^\mathcal{P}(v) = \pi(v)$ for $v \in \mathsf{Vars}(q^\mathcal{P})$ and $\pi^\mathcal{P}(a^\mathcal{P}) = \pi(a)$ for $a^\mathcal{P} \in \mathsf{Inds}(q^\mathcal{P})$ and some a such that $a^\mathcal{P} = f(a)$. Let $C(a^\mathcal{P}) \in q^\mathcal{P}$ be such that $C(a) \in q$ and a was replaced with $a^\mathcal{P}$, i.e., $f(a) = a^\mathcal{P}$. By assumption, $\pi(a) \in C^\mathcal{I}$, but then $\pi(a) = a^\mathcal{I} = f(a)^{\mathcal{I}^\mathcal{P}} = a^{\mathcal{P}\mathcal{I}^\mathcal{P}} = \pi^\mathcal{P}(a^\mathcal{P}) \in C^{\mathcal{I}^\mathcal{P}}$ and $\mathcal{I}^\mathcal{P} \models C(a^\mathcal{P})$. Similar arguments can be used to show entailment for role and equality atoms, which yields the desired contradiction. □

Theorem (35)**.** *Let $\mathcal{K} = (\mathcal{T}, \mathcal{R}, \mathcal{A})$ be a $\mathcal{SHIQ}$ knowledge base with $m := |\mathcal{K}|$ and $q := q_1 \vee \ldots \vee q_\ell$ a union of Boolean conjunctive queries with $n := |q|$. The algorithm given in Definition 34 decides in non-deterministic time $p(m_a)$ whether $\mathcal{K} \not\models q$ for $m_a := |\mathcal{A}|$ and p a polynomial.*

Proof of Theorem 35. Clearly, the size of an ABox $\mathcal{A}^\mathcal{P}$ in an $\mathcal{A}$-partition is bounded by m_a. As established in Lemma 32, the maximal size of an extended ABox $\mathcal{A}_q^\mathcal{P}$ is polynomial in m_a. Hence, $|\mathcal{A}^\mathcal{P} \cup \mathcal{A}_q^\mathcal{P}| \leq p(m_a)$ for some polynomial p. Due to Lemma 20 and since the size of q, $\mathcal{T}$, and $\mathcal{R}$ is fixed by assumption, the sets $\mathsf{trees}_{\mathcal{K}^\mathcal{P}}(q_i)$ and $\mathsf{ground}_{\mathcal{K}^\mathcal{P}}(q_i)$ for each i such that $1 \leq i \leq \ell$ can be computed in time polynomial in m_a. From Lemma 29, we know that the translation of an extended knowledge base into an $\mathcal{ALCQI}b$ knowledge base is polynomial in m_a and a close inspection of the algorithm by Tobies (2001) for deciding consistency of an $\mathcal{ALCQI}b$ knowledge base shows that its runtime is also polynomial in m_a. □

References

Baader, F., Calvanese, D., McGuinness, D. L., Nardi, D., & Patel-Schneider, P. F. (Eds.). (2003). *The Description Logic Handbook*. Cambridge University Press.

Bechhofer, S., van Harmelen, F., Hendler, J., Horrocks, I., McGuinness, D. L., Patel-Schneider, P. F., & Stein, L. A. (2004). OWL web ontology language reference. Tech. rep., World Wide Web Consortium. `http://www.w3.org/TR/2004/REC-owl-ref-20040210/`.

Calvanese, D., De Giacomo, G., Lembo, D., Lenzerini, M., & Rosati, R. (2006). Data complexity of query answering in description logics. In Doherty, P., Mylopoulos, J., & Welty, C. A. (Eds.), *Proceedings of the 10th International Conference on Principles of Knowledge Representation and Reasoning (KR 2006)*, pp. 260–270. AAAI Press/The MIT Press.

Calvanese, D., De Giacomo, G., Lembo, D., Lenzerini, M., & Rosati, R. (2007). Tractable reasoning and efficient query answering in description logics: The dl-lite family. *Journal of Automated Reasoning*, *39*(3), 385–429.

Calvanese, D., De Giacomo, G., & Lenzerini, M. (1998a). On the decidability of query containment under constraints. In *Proceedings of the 17th ACM SIGACT-SIGMOD-*

SIGART Symposium on Principles of Database Systems (PODS 1998), pp. 149–158. ACM Press and Addison Wesley.

Calvanese, D., De Giacomo, G., Lenzerini, M., Nardi, D., & Rosati, R. (1998b). Description logic framework for information integration. In *Proceedings of the 6th International Conference on Principles of Knowledge Representation and Reasoning (KR 1998)*.

Calvanese, D., Eiter, T., & Ortiz, M. (2007). Answering regular path queries in expressive description logics: An automata-theoretic approach. In *Proceedings of the 22th National Conference on Artificial Intelligence (AAAI 2007)*.

Chekuri, C., & Rajaraman, A. (1997). Conjunctive query containment revisited. In *Proceedings of the 6th International Conference on Database Theory (ICDT 1997)*, pp. 56–70, London, UK. Springer-Verlag.

Glimm, B., Horrocks, I., & Sattler, U. (2006). Conjunctive query answering for description logics with transitive roles. In *Proceedings of the 19th International Workshop on Description Logics (DL 2006)*. `http://www.cs.man.ac.uk/~glimmbx/download/GlHS06a.pdf`.

Grädel, E. (2001). Why are modal logics so robustly decidable?. In Paun, G., Rozenberg, G., & Salomaa, A. (Eds.), *Current Trends in Theoretical Computer Science, Entering the 21th Century*, Vol. 2, pp. 393–408. World Scientific.

Grahne, G. (1991). *Problem of Incomplete Information in Relational Databases*. Springer-Verlag.

Horrocks, I., Patel-Schneider, P. F., & van Harmelen, F. (2003). From SHIQ and RDF to OWL: The making of a web ontology language. *Journal of Web Semantics*, *1*(1), 7–26.

Horrocks, I., Sattler, U., Tessaris, S., & Tobies, S. (1999). Query containment using a DLR ABox. Ltcs-report LTCS-99-15, LuFG Theoretical Computer Science, RWTH Aachen, Germany. Available online at `http://www-lti.informatik.rwth-aachen.de/Forschung/Reports.html`.

Horrocks, I., Sattler, U., & Tobies, S. (2000). Reasoning with Individuals for the Description Logic $\mathcal{SHIQ}$. In McAllester, D. (Ed.), *Proceedings of the 17th International Conference on Automated Deduction (CADE 2000)*, No. 1831 in Lecture Notes in Artificial Intelligence, pp. 482–496. Springer-Verlag.

Horrocks, I., & Tessaris, S. (2000). A conjunctive query language for description logic aboxes. In *Proceedings of the 17th National Conference on Artificial Intelligence (AAAI 2000)*, pp. 399–404.

Hustadt, U., Motik, B., & Sattler, U. (2005). Data complexity of reasoning in very expressive description logics. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI 2005)*, pp. 466–471.

Levy, A. Y., & Rousset, M.-C. (1998). Combining horn rules and description logics in CARIN. *Artificial Intelligence*, *104*(1–2), 165–209.

Lutz, C. (2007). Inverse roles make conjunctive queries hard. In *Proceedings of the 20th International Workshop on Description Logics (DL 2007)*.

McGuinness, D. L., & Wright, J. R. (1998). An industrial strength description logic-based configuration platform. *IEEE Intelligent Systems*, *13*(4).

Motik, B., Sattler, U., & Studer, R. (2004). Query answering for OWL-DL with rules. In *Proceedings of the 3rd International Semantic Web Conference (ISWC 2004)*, Hiroshima, Japan.

Ortiz, M., Calvanese, D., & Eiter, T. (2006a). Data complexity of answering unions of conjunctive queries in $\mathcal{SHIQ}$. In *Proceedings of the 19th International Workshop on Description Logics (DL 2006)*.

Ortiz, M. M., Calvanese, D., & Eiter, T. (2006b). Characterizing data complexity for conjunctive query answering in expressive description logics. In *Proceedings of the 21th National Conference on Artificial Intelligence (AAAI 2006)*.

Rosati, R. (2006a). DL+log: Tight integration of description logics and disjunctive datalog. In *Proceedings of the Tenth International Conference on Principles of Knowledge Representation and Reasoning (KR 2006)*, pp. 68–78.

Rosati, R. (2006b). On the ddecidability and finite controllability of query processing in databases with incomplete information. In *Proceedings of the 25th ACM SIGACT SIGMOD Symposium on Principles of Database Systems (PODS-06)*, pp. 356–365. ACM Press and Addison Wesley.

Rosati, R. (2007a). The limits of querying ontologies. In *Proceedings of the Eleventh International Conference on Database Theory (ICDT 2007)*, Vol. 4353 of *Lecture Notes in Computer Science*, pp. 164–178. Springer-Verlag.

Rosati, R. (2007b). On conjunctive query answering in EL. In *Proceedings of the 2007 Description Logic Workshop (DL 2007)*. CEUR Workshop Proceedings.

Schaerf, A. (1993). On the complexity of the instance checking problem in concept languages with existential quantification. *Journal of Intelligent Information Systems*, *2*(3), 265–278.

Sirin, E., & Parsia, B. (2006). Optimizations for answering conjunctive abox queries. In *Proceedings of the 19th International Workshop on Description Logics (DL 2006)*.

Sirin, E., Parsia, B., Cuenca Grau, B., Kalyanpur, A., & Katz, Y. (2006). Pellet: A practical OWL-DL reasoner. Accepted for the Journal of Web Semantics, Available online at `http://www.mindswap.org/papers/PelletJWS.pdf`.

Tessaris, S. (2001). *Questions and answers: reasoning and querying in Description Logic*. PhD thesis, University of Manchester.

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

Tsarkov, D., & Horrocks, I. (2006). FaCT++ description logic reasoner: System description. In Furbach, U., & Shankar, N. (Eds.), *Proceedings of the Third International Joint Conference on Automated Reasoning (IJCAR 2006)*, Vol. 4130 of *Lecture Notes in Computer Science*, pp. 292 – 297. Springer-Verlag.

van der Meyden, R. (1998). Logical approaches to incomplete information: A survey. In *Logics for Databases and Information Systems*, pp. 307–356. Kluwer Academic Publishers.

Vardi, M. Y. (1997). Why is modal logic so robustly decidable?. In *Descriptive Complexity and Finite Models: Proceedings of a DIMACS Workshop*, Vol. 31 of *DIMACS: Series in Discrete Mathematics and Theoretical Computer Science*, pp. 149–184. American Mathematical Society.

Wessel, M., & Möller, R. (2005). A high performance semantic web query answering engine. In *Proceedings of the 18th International Workshop on Description Logics*.

Wolstencroft, K., Brass, A., Horrocks, I., Lord, P., Sattler, U., Turi, D., & Stevens, R. (2005). A Little Semantic Web Goes a Long Way in Biology. In *Proceedings of the 2005 International Semantic Web Conference (ISWC 2005).*