# Conjunctive Query Entailment for $\mathcal{SHOQ}$

Birte Glimm*, Ian Horrocks, and Ulrike Sattler

[glimm,horrocks,sattler]@cs.man.ac.uk
The University of Manchester, UK

**Abstract.** An important reasoning task, in addition to the standard DL reasoning services, is conjunctive query answering. In this paper, we present a decision procedure for conjunctive query entailment in the expressive Description Logic $\mathcal{SHOQ}$. This is, to the best of our knowledge, the first decision procedure for conjunctive query entailment in a logic that allows for nominals. We achieve this by combining the techniques used in the conjunctive query entailment procedure for $\mathcal{SHIQ}$ with the techniques proposed for a restricted class of conjunctive queries in $\mathcal{SHOQ}$.

## 1 Introduction

Existing Description Logic (DL) reasoners[1] provide automated reasoning support for checking concepts for satisfiability and subsumption, and also for answering queries that retrieve known instances of concepts and roles. There are, however, still many open questions regarding the development of algorithms that decide conjunctive query (CQ) entailment in expressive Description Logics. For example, proposed techniques for deciding CQ entailment in expressive DLs mostly require that all roles that occur in the query are simple, i.e., neither transitive nor have transitive subroles. Furthermore, none of the existing conjunctive query answering techniques [10, 8, 2, 7, 9] is able to handle nominals. In this paper, we address both these issues and present a decision procedure for entailment of arbitrary CQs in the very expressive DL $\mathcal{SHOQ}$, i.e., we allow for both non-simple roles in the query and nominals. We also do not impose any restrictions on the structure of the queries as it is the case for a previously proposed query entailment technique for $\mathcal{SHOQ}$ [5].

Our algorithm combines ideas from the CQ entailment decision procedure for $\mathcal{SHIQ}$ [3, 4] with the technique for deciding entailment of a restricted class of CQs in $\mathcal{SHOQ}$ [5] (i.e., the algorithm does not accept arbitrary CQs as input, but only queries of a particular shape). We first rewrite a query into a set of queries that have a kind of forest shape. By applying the rolling-up or tuple-graph technique [2, 10], we build concepts that capture the rewritten queries. We then show that we can use the obtained concepts to reduce the task of deciding query entailment to the task of testing the consistency of extended knowledge bases.

* This work was supported by an EPSRC studentship.

[1] For example, FaCT++ `http://owl.man.ac.uk/factplusplus`, KAON2 `http://kaon2.semanticweb.org`, Pellet `http://pellet.owldl.com`, or Racer Pro `http://www.racer-systems.com`

## 2 Preliminaries

We assume readers to be familiar with the syntax and semantics of the DL $\mathcal{SHOQ}$ (for details see [1]). Since, in the presence of nominals, the ABox can be internalised, we assume that a $\mathcal{SHOQ}$ knowledge base $\mathcal{K}$ is a pair $(\mathcal{T}, \mathcal{R})$ over a signature $\mathcal{S} = (N_C, N_R)$, where $\mathcal{T}$ is a TBox, $\mathcal{R}$ is a role hierarchy, and $N_C$ and $N_R$ are countable, infinite, and pairwise disjoint sets of *concept names* and *role names* respectively. We assume that the set $N_C$ contains a subset $N_I$ of *nominal names* and the set $N_R$ contains a subset $N_{tR}$ of *transitive role names*. We say that a role name $r$ is simple if there is no $s \in N_{tR}$ such that $s \sqsubseteq^*_{\mathcal{R}} r$, where $\sqsubseteq^*_{\mathcal{R}}$ is the reflexive transitive closure of $\sqsubseteq$ over $\mathcal{R}$.

**Definition 1.** *Let $\mathcal{S} = (N_C, N_R)$ be a signature and $N_V$ a countably infinite set of variable names disjoint from $N_C$ and $N_R$. Let $C$ be a $\mathcal{SHOQ}$-concept over $\mathcal{S}$, $r \in N_R$ a role name, and $x, y \in N_V$. An atom is an expression $C(x)$ or $r(x, y)$, and we refer to these types of atoms as concept atoms and role atoms respectively. A* Boolean conjunctive query *$q$ is a non-empty set of atoms. We use $\mathsf{Vars}(q)$ to denote the set of all variables occurring in $q$ and $\sharp(q)$ for the cardinality of $q$. A* sub-query *of $q$ is simply a subset of $q$ (including $q$ itself).*

*Let $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ be an interpretation. For a total function $\pi\colon \mathsf{Vars}(q) \to \Delta^{\mathcal{I}}$, we write*

- $\mathcal{I} \models^{\pi} C(x)$ *if* $\pi(x) \in C^{\mathcal{I}}$;
- $\mathcal{I} \models^{\pi} r(x, y)$ *if* $(\pi(x), \pi(y)) \in r^{\mathcal{I}}$.

*If $\mathcal{I} \models^{\pi} at$ 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 a mapping $\pi$ such that $\mathcal{I} \models^{\pi} q$. We call such a $\pi$ a* match *for $q$ in $\mathcal{I}$. For a $\mathcal{SHOQ}$ knowledge base $\mathcal{K}$, we say that $\mathcal{K}$* entails *$q$ and write $\mathcal{K} \models q$ if $\mathcal{I} \models \mathcal{K}$ implies $\mathcal{I} \models q$.*

The *query entailment problem* is defined as follows: given a knowledge base $\mathcal{K}$ and a query $q$, decide whether $\mathcal{K} \models q$. Please note that we do not allow for constants (individual names) in the query. In the presence of nominals this is clearly without loss of generality. Query entailment is the decision problem corresponding to query answering, which is a computation problem. 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 $v_1, \ldots, v_m$ $(m \leqslant n)$ are answer variables, $\mathcal{K}$ a $\mathcal{SHOQ}$ knowledge base, and $\mathsf{nom}(\mathcal{K})$ the set of nominals that occur in $\mathcal{K}$. The *answers* of $\mathcal{K}$ to $q$ are those $m$-tuples $(o_1, \ldots, o_m) \in \mathsf{nom}(\mathcal{K})$ such that, for all models $\mathcal{I}$ of $\mathcal{K}$, $\mathcal{I} \models^{\pi} q$ for some $\pi$ that satisfies $\pi(v_i) \in \{o_i{}^{\mathcal{I}}\}$ for all $i$ with $1 \leqslant i \leqslant m$. It is not hard to see that the answers of $\mathcal{K}$ to $q$ can be computed by testing, for each $(o_1, \ldots, o_m) \in \mathsf{nom}(\mathcal{K})^m$, whether the query $q'$ obtained from $q$ by adding, for each $v_i$ with $1 \leq i \leq m$, an atom $(\{o_i\})(v_i)$, is entailed by $\mathcal{K}$.[2] The answer

[2] Please note that, in the presence of constants, it is more common to replace the distinguished variables $v_1, \ldots, v_m$ with the constants $o_1, \ldots, o_m$ instead of adding constant atoms.

to $q$ is then the set of all $m$-tuples $(o_1, \ldots, o_m)$ for which $\mathcal{K} \models q_{[v_1,\ldots,v_m/o_1,\ldots,o_m]}$. Let $k = \sharp(\mathsf{nom}(\mathcal{K}))$ be the number of nominals used in the $\mathcal{K}$. Since $\mathcal{K}$ 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 optimisations can be used, e.g., to identify a (hopefully small) set of candidate tuples.

In the remainder of this paper, we concentrate on query entailment. In the following, we use $\mathcal{K}$ for a $\mathcal{SHOQ}$ knowledge base and $q$ for a Boolean CQ over a common signature $\mathcal{S}$. We use $\mathsf{nom}(\mathcal{K})$ for the set of nominals that occur in $\mathcal{K}$ and we assume that $\mathsf{nom}(\mathcal{K})$ is non-empty without further notice. This is w.l.o.g. since otherwise we can always add an axiom $\{o\} \sqsubseteq \top$ to the TBox for a new nominal $o \in N_I$.

As for the CQ entailment algorithm for $\mathcal{SHIQ}$, we first show that we can restrict our attention to the canonical models of $\mathcal{K}$. Canonical models have a kind of forest shape, i.e., the elements in the model can be seen as a collection of trees such that each nominal builds the root of a tree. Additionally, there can be arbitrary relational structures between the nominals and relations between an element from within a tree back to some nominal. In order to emphasise the forest shape of the canonical models, we also define forest bases, where we omit the shortcuts induced by transitive roles. The role of canonical models in our decision procedure is roughly speaking the following: for deciding query entailment, we first rewrite a given query into a set of forest-shaped queries such that the rewritten queries can be expressed as concepts. We use the forest structure of the canonical models in order to show that the disjunction of the obtained concepts is indeed enough for deciding query entailment.

**Definition 2.** *A* tree $T$ *is a 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. The empty word* $\varepsilon$ *is the* root *of the tree. Given a set of roots* $R = \{r_1, \ldots, r_n\}$*, a* forest $F$ w.r.t. $R$ *is a subset of* $R \times \mathbb{N}^*$ *such that, for each* $r_i \in R$*,* $f(r_i) = (r_i, \varepsilon)$ *and the set* $\{w \mid (r_i, w) \in F\}$ *is a tree.*

*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 $\Delta^{\mathcal{J}}$ *is a forest w.r.t.* $\mathsf{nom}(\mathcal{K})$*, and*
T2 *if* $((o, w), (o', w')) \in r^{\mathcal{J}}$*, then either* $w' = \varepsilon$ *or* $o = o'$ *and* $w'$ *is a successor of* $w$*.*

*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*

$$r^{\mathcal{I}} = r^{\mathcal{J}} \cup \bigcup_{s \sqsubseteq_{\mathcal{R}} r,\ s \in N_{tR}} (s^{\mathcal{J}})^+$$

*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 $\mathcal{K}$*.*

**Lemma 1.** *$\mathcal{K} \not\models q$ iff there is some canonical model $\mathcal{I}$ of $\mathcal{K}$ such that $\mathcal{I} \not\models q$.*

*Proof Sketch*: The if direction is trivial. For the only if direction, the proof is similar to the one for $\mathcal{SHIQ}$. Let $\mathcal{I}$ be such that $\mathcal{I} \models \mathcal{K}$ and $\mathcal{I} \not\models q$. Intuitively, we first unravel $\mathcal{I}$ into a model $\mathcal{I}'$ of $\mathcal{K}$ and then construct a forest base from the unravelled model. Finally, we obtain a canonical model from the forest base by transitively closing all roles $r \in N_{tR}$. Since in the unravelling process we only "break" cycles, the query is still not satisfied in the constructed canonical model.

## 3 Reducing Query Entailment to Concept Unsatisfiability

In this section, we introduce the basic ideas that have been used in the development of algorithms for CQ entailment. In the following section, we show more formally how the techniques presented here can be combined in order to obtain a decision procedure for $\mathcal{SHOQ}$.

The initial ideas used in this paper were first introduced by Calvanese et al. [2] for deciding CQ containment and hence CQ entailment for $\mathcal{DLR}_{reg}$. The authors show how a query $q$ can be expressed as a concept $C_q$, such that $q$ is entailed by a knowledge base iff adding $\top \sqsubseteq \neg C_q$ makes the KB inconsistent. In order to obtain the concept $C_q$, the query $q$ is represented as a directed, labelled graph. This graph, called a tuple graph or a query graph, is traversed in a depth-first manner and, during the traversal, nodes and edges are replaced with appropriate concept expressions, leading to the concept $C_q$ after completing the traversal.

The nodes in a query graph correspond to the variables in the query and are labelled with the concepts that occur in the corresponding concept atoms. The edges correspond to the role atoms in $q$ and are labelled accordingly. E.g., let $q_1 = \{C(x), s(x,y), D(y)\}$ and $q_2 = \{C(x), r(x,y), r(x,y'), s(y,z), s(y',z), D(z)\}$. The query graphs for $q_1$ and $q_2$ are depicted in Figure 1 and Figure 2 respectively. We call $q_2$ a cyclic query since its underlying undirected query graph is cyclic. For acyclic queries such as $q_1$, we can build the concept that represents $q_1$ as follows: start at $x$ and traverse the graph to $y$. Since $y$ is a leaf node, remove $y$ and its incoming edge and conjoin $\exists s.D$ to the label $C$ of $x$, resulting in $C \sqcap \exists s.D$ for $C_{q_1}$. A given KB $\mathcal{K}$ entails $q_1$ iff $\mathcal{K} \cup \{\top \sqsubseteq \neg C_{q_1}\}$ is inconsistent. Please note that in the absence of inverse roles in the logic, this process requires that the query graph has the form of a directed tree.

This reduction is not directly extendable to cyclic queries since, due to the tree model property of most DLs, a concept cannot capture cyclic relationships. For simpler logics, only ABox assertions can enforce cyclic relational structures in every model. One could argue, therefore, that we can replace variables in a cycle with individual names from the ABox. By identifying variables with each other, however, some cyclic queries become acyclic. For example, identifying $y$ and $y'$ in $q_2$ leads to an acyclic query that can be expressed as $C \sqcap \exists r.(\exists s.D)$. Hence, $\mathcal{K} \models q_2$ if $\mathcal{K} \cup \{\top \sqsubseteq \neg(C \sqcap \exists r.(\exists s.D))\}$ is inconsistent.

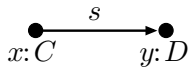


Fig. 1: The (acyclic) query graph for $q_1$.

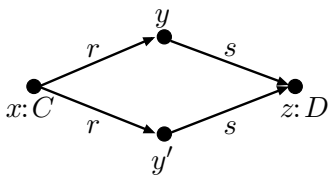


Fig. 2: A query graph for the cyclic query $q_2$.

Last year, we presented a decision procedure for entailment of a restricted class of CQs in $\mathcal{SHOQ}$ [5]. Due to the absence of inverse roles in $\mathcal{SHOQ}$, the algorithm handles only queries where the tree parts of the query form a directed tree and all cyclic subgraphs build a directed cycle. For example, the query $\{r(x, y), r(z, y)\}$ is not accepted as an input for the algorithm. The technique introduces, however, ideas for dealing with cycles that can arise when nominals and non-simple roles are present in the knowledge base. For example, Figure 3 represents a model for the knowledge base $\mathcal{K}$ containing the axiom $\{o\} \sqsubseteq \neg C \sqcap \neg D \sqcap \exists s.(C \sqcap \exists r.(D \sqcap \exists s.\{o\}))$ with $s \in N_{tR}$. The query $q_3 = \{C(x), D(y), r(x, y), s(y, x)\}$ (see Figure 4) would clearly be satisfied in each model of $\mathcal{K}$, although in the relevant matches neither $x$ nor $y$ can be mapped to the nominal $o^{\mathcal{I}}$ and without using the nominal $o$ in the query concept, we cannot enforce the coreference for closing the cycle.

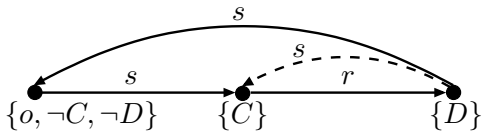


Fig. 3: The dashed line indicates the relationship added due to $s$ being transitive. Therefore, there is a cycle not directly containing the nominal $o$.

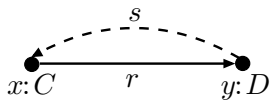


Fig. 4: The query graph for $q_3$.

In canonical models for a $\mathcal{SHOQ}$ knowledge base, such a directed cycle among non-nominals can only occur due to a transitive role that provides a shortcut for "skipping" the nominal. Hence, a nominal is always at least indirectly involved, e.g., $o$ in the current example. The proposed algorithm allows, therefore, the replacement of the role atom $s(y, x)$ with two role atoms $s(y, v), s(v, x)$ for a new variable $v$. We can then guess that $v$ corresponds to the nominal $o$ and express the query as a concept in which we use $\{o\}$ to close the cycle.

In the decision procedure for CQ entailment in $\mathcal{SHIQ}$ [3, 4], the rewriting steps also allow for eliminating shortcuts induced by transitive roles that do not involve nominals (ABox individuals in the case of $\mathcal{SHIQ}$). Let $\mathcal{K}_4 = (\mathcal{T}, \mathcal{R}, \mathcal{A})$

be a $\mathcal{SHIQ}$ knowledge base with $\mathcal{T} = \emptyset, \mathcal{R} = \{r \sqsubseteq t\}, \mathcal{A} = \{(\exists s.\exists r.\exists r.\exists r.\top)(a)\}$, and $t \in N_{tR}$ (see Figure 5). The cyclic query $q_4 = \{r(x_1, x_2), r(x_2, x_3), r(x_3, x_4), t(x_1, x_4)\}$ (see Figure 6) is clearly entailed by $\mathcal{K}_4$.

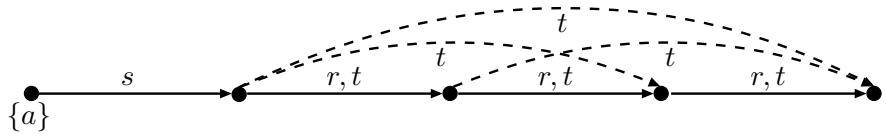


Fig. 5: A representation of a canonical model $\mathcal{I}$ for $\mathcal{K}_4$. Transitive "shortcuts" are again indicated by dashed lines.

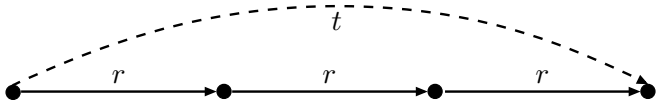


Fig. 6: The query graph for the query $q_4$.

In order to obtain a tree-shaped query that matches in such a canonical model, the rewriting steps for $\mathcal{SHIQ}$ allow, for each role atom $s(x, x')$ in a query such that there is a role $s' \in N_{tR}$ with $s' \sqsubseteq_{\mathcal{R}} s$, to replace $s(x, x')$ with up to $\sharp(q)$ role atoms $s'(x_1, x_2), \ldots, s'(x_{\ell-1}, x_\ell)$ such that $x_1 = x$ and $x_\ell = x'$. In the above example, we can replace $t(x_1, x_4)$ with $t(x_1, x_2), t(x_2, x_3), t(x_3, x_4)$ and obtain the query $q_4' = \{r(x_1, x_2), r(x_2, x_3), r(x_3, x_4), t(x_1, x_2), t(x_2, x_3), t(x_3, x_4)\}$. Using role conjunctions, we can now build the query concept $C_{q_4'} = \exists(r \sqcap t).(\exists(r \sqcap t).(\exists(r \sqcap t).\top))$. We can now use the concept $C_{q_4'}$ as described above and reduce the query entailment problem for a $\mathcal{SHIQ}$ knowledge base to a knowledge base consistency problem for $\mathcal{SHIQ}^{\sqcap}$, i.e., $\mathcal{SHIQ}$ with role conjunctions.

Usually, there is not just one query concept for a given query. In the rewriting process, we build query concepts for *all* tree-shaped queries obtained by identifying variables and by replacing role atoms as described above. We then check whether $\mathcal{K}$ entails the disjunction of the obtained concepts, which can be reduced to checking the consistency of $\mathcal{K}$ extended with all axioms $\top \sqsubseteq \neg C_q$ such that $C_q$ is one of the obtained query concepts.

We now define the different rewriting steps more formally and show how the different rewriting steps can be combined into a decision procedure for general CQs in $\mathcal{SHOQ}$.

## 4 Conjunctive Query Entailment for $\mathcal{SHOQ}$

For deciding whether a given Boolean CQ is entailed by a $\mathcal{SHOQ}$ knowledge base, we transform the query in a four stage process into a set of $\mathcal{SHOQ}^{\sqcap}$

concepts, i.e., $\mathcal{SHOQ}$ with role conjunctions. We can then reduce the task of deciding CQ entailment to the task of deciding $\mathcal{SHOQ}^{\sqcap}$ knowledge base consistency.

In the first step, called collapsing, we can identify variables. In the second step, we can replace role atoms of the form $r(x, x')$ for which $r$ is non-simple with up to $\sharp(q)$ role atoms. This allows for explicating all shortcuts due to transitive roles in the query. In the third step, we "guess" which variables correspond to nominals and filter out those queries that can still not be expressed as a $\mathcal{SHOQ}^{\sqcap}$ concept. Those queries are trivially false since the structure specified by the query cannot be enforced by a $\mathcal{SHOQ}$ concept and hence cannot be mapped to the canonical models of the knowledge base. Finally, we express the resulting queries as concepts and use these concepts for deciding entailment of the original query.

**Definition 3.** *A* collapsing *of $q$ is obtained as follows:*

1. *Build a partition $\mathcal{P}$ of $\mathsf{Vars}(q)$,*
2. *choose, for each $P \in \mathcal{P}$, one variable name $x \in P$, and*
3. *replace each occurrence of $x' \in P$ with $x$.*

*We use $\mathsf{co}(q)$ to denote the set of all queries that are a collapsing of $q$.*

*A* transitivity rewriting *of $q$ is obtained by fixing a set $V \subseteq N_V$ of variables not occurring in $q$ such that $\sharp(V) \leqslant \sharp(\mathsf{Vars}(q))$ and by choosing, for each role atom $r(x, x') \in q$ such that there is an $s \in N_{tR}$ with $s \sqsubseteq^{*}_{\mathcal{R}} r$, to replace $r(x, x')$ with $1 \leq \ell \leq \sharp(q)$ role atoms $s(x_1, x_2), \ldots, s(x_{\ell-1}, x_\ell)$, where $x_1 = x$, $x_\ell = x'$, and $x_2, \ldots, x_\ell \in \mathsf{Vars}(q) \cup V$. We use $\mathsf{tr}_{\mathcal{K}}(q)$ to denote the set of all queries that are a transitivity rewriting of a query $q_{co} \in \mathsf{co}(q)$.*

We assume that $\mathsf{tr}_{\mathcal{K}}(q)$ contains no isomorphic queries, i.e., differences in (newly introduced) variable names only are neglected.

We now show how we can filter out those queries that are trivially false since they have a structure that cannot occur in canonical models. For this, we use forest structures that are similar to canonical models. We first guess which variables of the query correspond to nominals. Between those variables, the role atoms of the query can induce arbitrary relational structures. All other variables are mapped to trees such that for a role atom $r(x, y)$ either the image of $y$ is a successor of the image of $x$ in the tree or $y$ corresponds to a nominal and $r(x, y)$ corresponds to an edge back to some nominal.

**Definition 4.** *A* tree mapping w.r.t. *$q$ is a total and bijective function $f$ from $\mathsf{Vars}(q)$ to a tree $T$ such that $r(x, x') \in q$ implies that $f(x')$ is a successor of $f(x)$. A query $q$ is* tree-shaped *if there exists a tree mapping w.r.t. $q$.*

*A* root choice *is a subset of $\mathsf{Vars}(q)$. A* forest mapping w.r.t. *$q$* and a root choice *$R$ is a total function $f$ from $\mathsf{Vars}(q)$ to a forest $F$ w.r.t. $R$ such that if $r(x, x') \in q$, then either $x' \in R$ or there is some $x_r \in R$ such that $f(x) = (x_r, w)$ and $f(x') = (x_r, w \cdot c)$. We say that $q$ is* forest-shaped w.r.t. *$R$ if either $R = \emptyset$ and $q$ is tree-shaped or $R \neq \emptyset$ and there exists a forest mapping w.r.t. $q$ and $R$.*

*We use $\mathsf{fr}_{\mathcal{K}}(q)$ to denote the set of all pairs $(q_{tr}, R)$ such that $q_{tr} \in \mathsf{tr}_{\mathcal{K}}(q)$, $R$ is a root choice w.r.t. $q_{tr}$, and $q_{tr}$ is forest-shaped w.r.t. $R$.*

Similarly to forest-shaped queries, we define tree- and forest-shaped matches on canonical models.

**Definition 5.** *Let $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ be a canonical model for $\mathcal{K}$ such that $\mathcal{I} \models^{\pi} q$. A match $\pi$ induces a root choice $R = \{x \mid \pi(x) = (o, \varepsilon)$ for some $o \in \mathsf{nom}(\mathcal{K})\}$. We call $\pi$ a* tree match *if $R = \emptyset$ and there exists a bijective function $f$ from $\mathsf{ran}(\pi)$ to a tree $T$ such that $r(x, x') \in q$ implies that $f(\pi(x'))$ is a successor of $f(\pi(x))$. We call $\pi$ a* forest match *if either $\pi$ is a tree match or there is a total mapping $f$ from $\mathsf{ran}(\pi)$ to a forest $F$ w.r.t. $R$ such that $r(x, x') \in q$ implies that either $f(\pi(x')) = (x', \varepsilon)$ or there is some $x_r \in R$ such that $f(\pi(x)) = (x_r, w), f(\pi(x')) = (x_r, w \cdot c)$.*

The following lemma shows that we can indeed omit queries that are not forest-shaped w.r.t. some subset of variables $R$.

**Lemma 2.** *Let $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ be a model for $\mathcal{K}$.*

*(1) If $\mathcal{I}$ is canonical and $\mathcal{I} \models q$, then there is a pair $(q_{tr}, R) \in \mathsf{fr}_{\mathcal{K}}(q)$ and a forest match $\pi$ such that $\mathcal{I} \models^{\pi} q_{tr}$ and $R$ is the root choice induced by $\pi$ .*
*(2) If $(q_{tr}, R) \in \mathsf{fr}_{\mathcal{K}}(q)$ and $\mathcal{I} \models q_{tr}$, then $\mathcal{I} \models q$.*

The proof is very similar to the proofs for $\mathcal{SHIQ}$ [3, 4]. Intuitively, we use the canonical model $\mathcal{I}$ to guide the rewriting process in the proof of part (1) of Lemma 2. Part (2) of Lemma 2 mainly follows from the fact that we only use non-simple roles in the transitivity rewritings.

We now build a query that consists of only concept atoms for each $(q_{tr}, R) \in \mathsf{fr}_{\mathcal{K}}(q)$ by replacing the variables from $R$ with nominals from $\mathsf{nom}(\mathcal{K})$ and applying the rolling-up technique.

**Definition 6.** *Let $(q_{tr}, R) \in \mathsf{fr}_{\mathcal{K}}(q)$. A* grounding for $q_{tr}$ w.r.t. $R$ *is a total function $\tau\colon R \to \mathsf{nom}(\mathcal{K})$. Let $f$ be a forest mapping w.r.t. $q$ and $R$. We build $\mathsf{con}(q_{tr}, R, \tau)$ as follows:*

1. *For each $r(x, x_r) \in q_{tr}$ with $x_r \in R$, replace $r(x, x_r)$ with $(\exists r.\{\tau(x_r)\})(x)$.*
2. *For each $x_r \in R$ add a concept atom $(\{\tau(x_r)\})(x_r)$ to $q_{tr}$.*
3. *We now inductively assign to each $x \in \mathsf{Vars}(q_{tr})$ a concept $\mathsf{con}(x)$ as follows:*
   - *if there is no role atom $r(x, x') \in q_{tr}$, then $\mathsf{con}(x) := \bigsqcap_{C(x) \in q_{tr}} C$,*
   - *if there are role atoms $r(x, x_1), \ldots, r(x, x_k) \in q_{tr}$, then*

$$\mathsf{con}(x) := \bigsqcap_{C(x) \in q_{tr}} C \sqcap \bigsqcap_{1 \leqslant i \leqslant k} \exists \big( \bigsqcap_{r(x, x_i) \in q_{tr}} r \big).\mathsf{con}(x_i).$$

4. *Finally, $\mathsf{con}(q_{tr}, R, \tau) = \{(\mathsf{con}(x))(x) \mid x \in \mathsf{Vars}(q_{tr})$ and there is no role atom $r(x', x) \in q_{tr}\}$.*

*We use $\mathsf{con}_{\mathcal{K}}(q)$ for the set $\{\mathsf{con}(q_{tr}, R, \tau) \mid (q_{tr}, R) \in \mathsf{fr}_{\mathcal{K}}(q)$ and $\tau$ is a grounding w.r.t. $R\}$.*

Please note that after the first step, the resulting query consists of a set of unconnected components such that each component is a tree-shaped query with a distinguished root variable that has no incoming edges. In step 4, we collect all query concepts for these root variables in the set $\mathsf{con}(q_{tr}, R, \tau)$. Hence $\mathsf{con}(q_{tr}, R, \tau)$ is a conjunctive query of the form $\{C_1(x_1), \ldots, C_n(x_n)\}$ with $x_i \neq x_j$ for $1 \leq i < j \leq n$ and each $C_i$ is a $\mathcal{SHOQ}^{\sqcap}$-concept.

**Lemma 3.** *Let $\mathcal{I}$ be a model of $\mathcal{K}$.*

*(1) If $\mathcal{I}$ is canonical and $\mathcal{I} \models q$, then there is some $\mathsf{con}(q_{tr}, R, \tau) \in \mathsf{con}_{\mathcal{K}}(q)$ such that $\mathcal{I} \models \mathsf{con}(q_{tr}, R, \tau)$.*
*(2) If $\mathcal{I} \models \mathsf{con}(q_{tr}, R, \tau)$ for some $\mathsf{con}(q_{tr}, R, \tau) \in \mathsf{con}_{\mathcal{K}}(q)$, then $\mathcal{I} \models q$.*

Informally, the use of nominals in the constructed concepts still enforces the same structures that are required by the query.

We now show that the union of the queries in $\mathsf{con}_{\mathcal{K}}(q)$ can be used to decide entailment of $q$. Since we now use unions of conjunctive queries, we introduce their semantics more formally:

**Definition 7.** *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 $i$ with $1 \leq i \leq n$ such that $\mathcal{I} \models q_i$.*

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.

Putting everything together, we get the following theorem, which shows that the queries in $\mathsf{con}_{\mathcal{K}}(q)$ are indeed enough to decide whether $\mathcal{K} \models q$.

**Theorem 1.** *Let $\{q_1, \ldots, q_\ell\} = \mathsf{con}_{\mathcal{K}}(q)$. Then $\mathcal{K} \models q$ iff $\mathcal{K} \models q_1 \vee \ldots \vee q_\ell$.*

Please note that each disjunct $q_i$ is a set of concept atoms of the form $\{C_1^i(x_1^i), \ldots, C_{n_i}^i(x_{n_i}^i)\}$, i.e., each $q_i$ contains $n_i$ unconnected components. In the following, we assume for convenience that conjunctive queries are written as a conjunction of atoms instead of in the set notation, e.g., we now write each of the disjuncts $\{C_1^i(x_1^i), \ldots, C_{n_i}^i(x_{n_i}^i)\}$ as $C_1^i(x_1^i) \wedge \ldots \wedge C_{n_i}^i(x_{n_i}^i)$. By transforming the disjunction of queries in $\mathsf{con}_{\mathcal{K}}(q)$ into conjunctive normal form (cf. [10, 7.3.2]), we can reduce the problem of deciding whether $\mathcal{K} \models q_1 \vee \ldots \vee q_\ell$ to deciding whether $\mathcal{K}$ entails each union of connected conjunctive queries $\{at_1\} \vee \ldots \vee \{at_\ell\}$ such that $at_i$ a concept atom from $q_i$. Let $\mathsf{con}_{\mathcal{K}}(q) = \{q_1, \ldots, q_\ell\}$. We use $\mathsf{cnf}(\mathsf{con}_{\mathcal{K}}(q))$ for the conjunctive normal form of $q_1 \vee \ldots \vee q_\ell$. We now show how we can decide entailment of unions of conjunctive queries, where each conjunct consists of one concept atom only. This suffices to decide conjunctive query entailment for $\mathcal{SHOQ}$.

**Definition 8.** *Let $\mathcal{K} = (\mathcal{T}, \mathcal{R})$ be a $\mathcal{SHOQ}$ knowledge base, $q$ a Boolean conjunctive query, and $\{C_1(x_1) \vee \ldots \vee C_\ell(x_\ell)\}$ a conjunct from $\mathsf{cnf}(\mathsf{con}_\mathcal{K}(q))$. An* extended knowledge base w.r.t. $\mathcal{K}$ and $q$ *is a pair $(\mathcal{T} \cup \mathcal{T}_q, \mathcal{R}))$ such that $\mathcal{T}_q$ contains an axiom $\top \sqsubseteq \neg C_i$ for each $i$ with $1 \leq i \leq \ell$.*

We can now use the extended knowledge bases in order to decide conjunctive query entailment as follows:

**Theorem 2.** *$\mathcal{K} \models q$ iff each extended knowledge base $\mathcal{K}_q$ w.r.t. $\mathcal{K}$ and $q$ is inconsistent.*

Please note that the extended knowledge bases are in $\mathcal{SHOQ}^\sqcap$. It is, however, not hard to see how the Tableaux algorithm for $\mathcal{SHOQ}$ [6] can be extended to handle such extended KBs, which then provides a decision procedure for CQ entailment in $\mathcal{SHOQ}$.

## 5 Conclusions

In the previous section, we have presented a decision procedure for CQ entailment in $\mathcal{SHOQ}$. This is, to the best of our knowledge, the first CQ entailment decision procedure that can handle nominals. In addition, we allow for non-simple roles in the query as well, which is a feature that is known to be tricky. Since the set of rewritten queries can potentially be large, the algorithm is more suitable for showing decidability of the problem rather than building the foundation of implementable algorithms. By analysing the role hierarchy one can, however, avoid several rewritings. Going back to the example knowledge base $\mathcal{K}_4$ and query $q_4$ (see Figure 5 and 6) in Section 3, it is not hard to see that with the given role hierarchy the atom $t(x_1, x_4)$ is redundant. Every match for the remaining role atoms implies the existence of a suitable $t$-edge. Since after removing this redundant role atom the query is acyclic, no rewriting is necessary in order to decide entailment. It will be part of our future work to investigate whether such an analysis of query and role hierarchy can be used instead of the transitivity rewriting step. Furthermore, we will try to find tight complexity bounds for the conjunctive query entailment problem in $\mathcal{SHOQ}$ and we will try to show decidability of conjunctive query entailment in $\mathcal{SHOIQ}$.

## References


1. Franz Baader, Diego Calvanese, Deborah L. McGuinness, Daniele Nardi, and Peter F. Patel-Schneider, editors. *The Description Logic Handbook*. Cambridge University Press, 2003.
2. Diego Calvanese, Giuseppe De Giacomo, and Maurizio Lenzerini. On the decidability of query containment under constraints. In *Proc. of PODS 1998*, 1998.
3. Birte Glimm, Ian Horrock, Carsten Lutz, and Uli Sattler. Conjunctive query answering in the description logic $\mathcal{SHIQ}$. In *Proc. of IJCAI 2007*, 2007.
4. Birte Glimm, Ian Horrocks, Carsten Lutz, and Ulrike Sattler. Conjunctive query answering for the description logic $\mathcal{SHIQ}$. Technical report, The University of Manchester, 2007. `http://www.cs.man.ac.uk/~glimmbx/download/GHLS07b.pdf`.
5. Birte Glimm, Ian Horrocks, and Ulrike Sattler. Conjunctive query answering for description logics with transitive roles. In *Proc. DL 2006*, 2006.
6. Ian Horrocks and Ulrike Sattler. Ontology reasoning in the $\mathcal{SHOQ}$ description logic. In *Proc. of IJCAI 2001*, 2001.
7. Ullrich Hustadt, Boris Motik, and Ulrike Sattler. A decomposition rule for decision procedures by resolution-based calculi. In *Proc. of LPAR 2004*, 2004.
8. Alon Y. Levy and Marie-Christine Rousset. CARIN: A representation language combining horn rules and description logics. In *Proc. of ECAI 1996*, 1996.
9. Maria M. Ortiz, Diego Calvanese, and Thomas Eiter. Characterizing data complexity for conjunctive query answering in expressive description logics. In *Proc. of AAAI 2006*, 2006.
10. Sergio Tessaris. *Questions and answers: reasoning and querying in Description Logic*. PhD thesis, University of Manchester, 2001.