Towards Practical Query Answering for Horn-$\mathcal{SHIQ}^\star$

Thomas Eiter[1], Magdalena Ortiz[1], Mantas Šimkus[1]
Trung-Kien Tran[2], and Guohui Xiao[1]

[1] Institute of Information Systems, Vienna University of Technology
{eiter|ortiz|xiao}@kr.tuwien.ac.at
simkus@dbai.tuwien.ac.at
[2] STARLab, Vrije Universiteit Brussel
truntran@vub.ac.be

1 Introduction

Query answering has become a prominent reasoning task in Description Logics. This is witnessed not only by the high number of publications on the topic in the last decade, but also by the increasing number of query answering engines. A number of systems provide full conjunctive query (CQ) answering capabilities, including [1, 23, 25, 4, 11]. A common feature of these approaches is that they rely on existing technologies for relational or deductive databases. They focus on lightweight DLs like $\mathcal{DL}$-*Lite* and $\mathcal{EL}$, and they use *query rewriting* to reduce the problem of answering a query over a DL ontology to a database query evaluation problem. For more expressive DLs that are not tractable in combined complexity, however, CQs (with the complete first-order semantics) are not yet supported by current reasoning engines. A range of algorithms has been designed, but they serve for theoretical purposes such as showing complexity bounds and are not amenable to practical implementation. The only exception is the rewriting algorithm implemented in the REQUIEM system, which covers $\mathcal{ELHI}$ [23], an expressive extension of $\mathcal{EL}$ for which standard reasoning is EXPTIME-hard.

In this paper, we contribute to the development of practical query answering systems beyond $\mathcal{DL}$-*Lite* and $\mathcal{EL}$. We consider Horn-$\mathcal{SHIQ}$, the Horn fragment of the popular DL $\mathcal{SHIQ}$ that underlies OWL DL. It combines all the expressive features of $\mathcal{DL}$-*Lite* and $\mathcal{EL}$, and simultaneously extends them with transitive roles, qualified number restrictions and some universal quantification. Standard reasoning tasks in Horn-$\mathcal{SHIQ}$ are already EXPTIME-hard in combined complexity but, due to the absence of disjunction, they are polynomial in data complexity. Since in Horn-$\mathcal{SHIQ}$ models are significantly more complex than in $\mathcal{DL}$-*Lite* and (most dialects of) $\mathcal{EL}$, extending existing query rewriting techniques is not straightforward.

The main contribution of this paper is a query answering method for Horn-$\mathcal{SHIQ}$ that appears to be promising for practicable systems, as confirmed by the experimental evaluation of a prototype implementation.

– The core of the method is a novel query rewriting technique which transforms an input query q into a union Q of CQs (a UCQ) such that the answers of q over an ontology $\mathcal{O} = \langle \mathcal{T}, \mathcal{A} \rangle$ with TBox $\mathcal{T}$ and ABox $\mathcal{A}$ coincide with the answers over $\mathcal{A}$

⋆ This work was supported by the Austrian Science Fund (FWF) grants P20840 and T515.

of a Datalog program comprising Q and some rules to complete $\mathcal{A}$, showing Datalog-rewritability of CQ answering in Horn-$\mathcal{SHIQ}$.

– Naturally, the set Q may be exponential in the size of q in the worst case, but in practice we obtain rewritings Q that are of manageable size for real world ontologies. This is mostly due to the fact that new queries are only generated taking into account the anonymous domain elements implied by the terminology. Notably our algorithm is worst-case optimal in both data and combined complexity.

– We describe a prototype implementation of the approach that uses off-the-shelf Datalog reasoners. Despite being preliminary and lacking sophisticated optimizations, it shows that the approach is promising. It can answer queries efficiently over Horn-$\mathcal{SHIQ}$ ontologies and scales down nicely to $\mathcal{DL}$*-Lite*, where it is competitive with state of the art query rewriting systems.

– The technique works for full Horn-$\mathcal{SHIQ}$ and arbitrary CQs, but the implemented version does not support transitive (or more generally, non-simple) roles in the query. To keep presentation simple, we present here only the case without transitive roles, and refer to [6, 9] for a more general version with transitive roles and richer queries formulated in weakly DL-safe Datalog.

2 Preliminaries

Horn-$\mathcal{SHIQ}$ The syntax and semantics of Horn-$\mathcal{SHIQ}$ is defined in the usual way. A *role* is a role name p or its inverse p^-. A Horn-$\mathcal{SHIQ}$ *TBox* $\mathcal{T}$ in normal form is a set of *role inclusion axioms* $r \sqsubseteq s$, *transitivity axioms* $trans(r)$, and *general concept inclusion axioms (GCIs)* of the forms (F1) $A_1 \sqcap \ldots \sqcap A_n \sqsubseteq B$, (F2) $A_1 \sqsubseteq \forall r.B$, (F3) $A_1 \sqsubseteq \exists r.B$, and (F4) $A_1 \sqsubseteq \leqslant 1\, r.B$,where $A_1, \ldots, A_n, B$ are concept names and r, s are *roles*. Axioms (F3) are called *existential*. W.l.o.g. we consider only Horn-$\mathcal{SHIQ}$ TBoxes in normal form [16, 17]. We call s *transitive* in $\mathcal{T}$, if $trans(s) \in \mathcal{T}$ or $trans(s^-) \in \mathcal{T}$, and we call s *simple* in $\mathcal{T}$, if there is no transitive r in $\mathcal{T}$ s.t. $r \sqsubseteq^*_{\mathcal{T}} s$, where $\sqsubseteq^*$ is the reflexive transitive closure of $\{(r, s) \mid r \sqsubseteq s \in \mathcal{T} \text{ or } \mathsf{inv}(r) \sqsubseteq \mathsf{inv}(s) \in \mathcal{T}\}$. Only simple roles are allowed in axioms of the form (F4).

An *ABox* $\mathcal{A}$ is a set of *assertions* $A(a)$ and $r(a, b)$, where A is a concept name, r a role, and a, b are individuals; the set of all individuals is denoted by $\mathsf{N_I}$. An *ontology* is a pair $(\mathcal{T}, \mathcal{A})$ of a TBox $\mathcal{T}$ and an ABox $\mathcal{A}$. The semantics is given by *interpretations* $\mathcal{I} = \langle \Delta^{\mathcal{I}}, \cdot^{\mathcal{I}} \rangle$ in the usual way.

A Horn-$\mathcal{ALCHIQ}$ TBox is a Horn-$\mathcal{SHIQ}$ TBox with no transitivity axioms. Horn-$\mathcal{ALCHIQ}^{\sqcap}$ TBoxes are obtained by allowing the *conjunction* $r_1 \sqcap r_2$ of roles r_1 and r_2, interpreted $(r_1 \sqcap r_2)^{\mathcal{I}} = r_1^{\mathcal{I}} \cap r_2^{\mathcal{I}}$. We let $\mathsf{inv}(r_1 \sqcap r_2) = \mathsf{inv}(r_1) \sqcap \mathsf{inv}(r_2)$ and assume w.l.o.g. that for each role inclusion $r \sqsubseteq s$ of a Horn-$\mathcal{ALCHIQ}^{\sqcap}$ TBox $\mathcal{T}$, (i) $\mathsf{inv}(r) \sqsubseteq \mathsf{inv}(s) \in \mathcal{T}$, and (ii) $s \in \{p, p^-\}$ for a role name p. For a set W and a concept or role conjunction $\Gamma = \gamma_1 \sqcap \ldots \sqcap \gamma_m$, we write $\Gamma \subseteq W$ for $\{\gamma_1, \ldots, \gamma_m\} \subseteq W$.

Conjunctive Queries A *conjunctive query* (CQ) is an expression of the form

$$q(\boldsymbol{u}) \leftarrow p_1(\boldsymbol{v_1}), \ldots, p_m(\boldsymbol{v_m})$$

where each $p_i(\boldsymbol{v_i})$ is an *atom* of the form $A(x)$ or $r(x, y)$, where A is a concept name and r is a role, x, y are variables, and q is a special *query predicate* not occurring elsewhere. For convenience, we may identify such a CQ with the set of its atoms, and

Table 1: Inference rules. $M^{(\prime)}$, $N^{(\prime)}$, (resp., $S^{(\prime)}$) are conjunctions of atomic concepts (roles); A, B are atomic concepts

$$\frac{M \sqsubseteq \exists S.(N \sqcap N') \quad N \sqsubseteq A}{M \sqsubseteq \exists S.(N \sqcap N' \sqcap A)}\ \mathbf{R}^{c}_{\sqsubseteq} \qquad \frac{M \sqsubseteq \exists (S \sqcap S').N \quad S \sqsubseteq r}{M \sqsubseteq \exists (S \sqcap S' \sqcap r).N}\ \mathbf{R}^{r}_{\sqsubseteq} \qquad \frac{M \sqsubseteq \exists S.(N \sqcap \bot)}{M \sqsubseteq \bot}\ \mathbf{R}_{\bot}$$

$$\frac{M \sqsubseteq \exists (S \sqcap r).N \quad A \sqsubseteq \forall r.B}{M \sqcap A \sqsubseteq \exists (S \sqcap r).(N \sqcap B)}\ \mathbf{R}_{\forall} \qquad \frac{M \sqsubseteq \exists (S \sqcap \mathsf{inv}(r)).(N \sqcap A) \quad A \sqsubseteq \forall r.B}{M \sqsubseteq B}\ \mathbf{R}^{-}_{\forall}$$

$$\frac{M \sqsubseteq \exists (S \sqcap r).(N \sqcap B) \quad A \sqsubseteq \leqslant 1\, r.B \quad M' \sqsubseteq \exists (S' \sqcap r).(N' \sqcap B)}{M \sqcap M' \sqcap A \sqsubseteq \exists (S \sqcap S' \sqcap r).(N \sqcap N' \sqcap B)}\ \mathbf{R}_{\leq}$$

$$\frac{M \sqsubseteq \exists (S \sqcap \mathsf{inv}(r)).(N_1 \sqcap N_2 \sqcap A) \quad A \sqsubseteq \leqslant 1\, r.B \quad N_1 \sqcap A \sqsubseteq \exists (S' \sqcap r).(N' \sqcap B \sqcap C)}{M \sqcap B \sqsubseteq C \quad M \sqcap B \sqsubseteq \exists (S \sqcap \mathsf{inv}(S' \sqcap r)).(N_1 \sqcap N_2 \sqcap A)}\ \mathbf{R}^{-}_{\leq}$$

use $q(\boldsymbol{u})$ (or simply q) to refer to it. We call $\boldsymbol{u} \subseteq \bigcup_{1 \leq i \leq m} \boldsymbol{v_i}$ the *distinguished* variables of q. A *match* for q in $\mathcal{I}$ is a mapping from variables in q to elements in $\Delta^{\mathcal{I}}$ such that $\pi(\boldsymbol{t}) \in p^{\mathcal{I}}$ for each atom $p(\boldsymbol{t})$ of q. The *answer* to q over $\mathcal{O}$ is the set of all $\boldsymbol{c} \in \mathsf{N}_{\mathsf{I}}^{|\boldsymbol{u}|}$ such that in every model $\mathcal{I}$ of $\mathcal{O}$ some match π for q exists with $\pi(\boldsymbol{u}) = (\boldsymbol{c})^{\mathcal{I}}$.

Elimination of Transitivity. As usual, transitivity axioms roles can be eliminated from Horn-$\mathcal{SHIQ}$ TBoxes. To obtain a Horn-$\mathcal{ALCHIQ}$ TBox $\mathcal{T}^*$ that is also in normal form, we can use the transformation from [16]. This transformation preserves satisfiability and, provided that queries contain only simple roles, also query answers (answers are not preserved for arbitrary queries unless the notion of match is suitably relaxed). In the rest of the paper we describe a procedure for answering CQs in Horn-$\mathcal{ALCHIQ}^{\sqcap}$.

3 Canonical Models

For answering CQs in Horn DLs usually the *canonical model property* is employed [7, 21, 2]. In particular, for a consistent Horn-$\mathcal{ALCHIQ}^{\sqcap}$ ontology $\mathcal{O} = (\mathcal{T}, \mathcal{A})$, there exists a model $\mathcal{I}$ of $\mathcal{O}$ that can be homomorphically mapped into any other model $\mathcal{I}'$ of $\mathcal{O}$. We show that such an $\mathcal{I}$ can be built in three steps:

(1) close $\mathcal{T}$ under specially tailored inferences rules,
(2) close $\mathcal{A}$ under all but existential axioms of $\mathcal{T}$, and
(3) extend $\mathcal{A}$ by "applying" the existential axioms of $\mathcal{T}$.

For Step (1) we use the calculus in Table 1, which is similar to [16, 20]. Given a Horn-$\mathcal{ALCHIQ}^{\sqcap}$ TBox $\mathcal{T}$, we denote by $\Xi(\mathcal{T})$ the TBox obtained from $\mathcal{T}$ by exhaustively applying the inference rules in Table 1. In Step (2) we simply 'apply' in the ABox all but existential axioms in $\mathcal{T}$. For convenience, this is done using the set $\mathsf{cr}(\mathcal{T})$ of Datalog rules in Table 2. Since every ABox $\mathcal{A}$ can be seen as a set of Datalog facts, $\mathcal{A} \cup \mathsf{cr}(\mathcal{T})$ is a Datalog program (with constraints) which has a unique minimal Herbrand model $\mathcal{J} = MM(\mathcal{A} \cup \mathsf{cr}(\mathcal{T}))$ if $\mathcal{O}$ is consistent. This model is almost a canonical model of $(\mathcal{T}, \mathcal{A})$; however, existential axioms may be violated. To deal with this, in Step (3) we extend $\mathcal{J}$ with new domain elements as required by axioms $M \sqsubseteq \exists r.N$ in $\Xi(\mathcal{T})$, using a procedure similar to the well known *chase* in databases.

Table 2: (Completion rules) Datalog program $\mathsf{cr}(\mathcal{T})$.

$B(y) \leftarrow A(x), r(x,y)$ for each $A \sqsubseteq \forall r.B \in \mathcal{T}$

$B(x) \leftarrow A_1(x), \ldots, A_n(x)$ for all $A_1 \sqcap \ldots \sqcap A_n \sqsubseteq B \in \Xi(\mathcal{T})$

$r(x,y) \leftarrow r_1(x,y), \ldots, r_n(x,y)$ for all $r_1 \sqcap \ldots \sqcap r_n \sqsubseteq r \in \mathcal{T}$

$\bot(x) \leftarrow A(x), r(x,y_1), r(x,y_2), B(y_1), B(y_2), y_1 \neq y_2$ for each $A \sqsubseteq \leqslant 1\, r.B \in \mathcal{T}$

$\Gamma \leftarrow A(x), A_1(x), \ldots, A_n(x), r(x,y), B(y)$
for all $A_1 \sqcap \ldots \sqcap A_n \sqsubseteq \exists (r_1 \sqcap \ldots \sqcap r_m).(B_1 \sqcap \ldots \sqcap B_k)$
and $A \sqsubseteq \leqslant 1\, r.B$ of $\Xi(\mathcal{T})$ such that $r{=}r_i$ and $B{=}B_j$ for some i, j
with $\Gamma \in \{B_1(y), \ldots, B_k(y), r_1(x,y), \ldots, r_k(x,y)\}$

Definition 1. *Let $\mathcal{T}$ be a Horn-$\mathcal{ALCHIQ}^{\sqcap}$ TBox and let $\mathcal{I}$ be an interpretation. A GCI $M \sqsubseteq \exists S.N$ is* applicable at $e \in \Delta^{\mathcal{I}}$ *if (a) $e \in M^{\mathcal{I}}$, (b) there is no $e' \in \Delta^{\mathcal{I}}$ with $(e,e') \in S^{\mathcal{I}}$ and $e' \in N^{\mathcal{I}}$, (c) there is no axiom $M' \sqsubseteq \exists S'.N' \in \mathcal{T}$ such that $e \in (M')^{\mathcal{I}}$, $S \subseteq S'$, $N \subseteq N'$, and $S \subset S'$ or $N \subset N'$. An interpretation $\mathcal{J}$ obtained from $\mathcal{I}$ by an* application *of an applicable axiom $M \sqsubseteq \exists S.N$ at $e \in \Delta^{\mathcal{I}}$ is defined as:*

- *$\Delta^{\mathcal{J}} = \Delta^{\mathcal{I}} \cup \{d\}$ with d a new element not present in $\Delta^{\mathcal{I}}$ (we call d a* successor *of e),*
- *For each concept name A and each $o \in \Delta^{\mathcal{J}}$, we have $o \in A^{\mathcal{J}}$ if (a) $o \in \Delta^{\mathcal{I}}$ and $o \in A^{\mathcal{I}}$; or (b) $o = d$ and $A \in N$.*
- *For each role name r and $o, o' \in \Delta^{\mathcal{J}}$, we have $(o,o') \in r^{\mathcal{J}}$ if (a) $o, o' \in \Delta^{\mathcal{I}}$ and $(o,o') \in r^{\mathcal{I}}$; or (b) $(o,o') = (e,d)$ and $r \in S$; or (c) $(o,o') = (d,e)$ and $\mathsf{inv}(r) \in S$.*

We denote by chase$(\mathcal{I}, \mathcal{T})$ a possibly infinite interpretation obtained from $\mathcal{I}$ by applying the existential axioms in $\mathcal{T}$. We require fairness*: the application of an applicable axiom can not be infinitely postponed.*

We note that *chase*$(\mathcal{I}, \mathcal{T})$ is unique up to renaming of domain elements. As usual in DLs, it can be seen as a 'forest': the application of existential axioms simply attaches 'trees' to an arbitrarily shaped model $\mathcal{I}$.

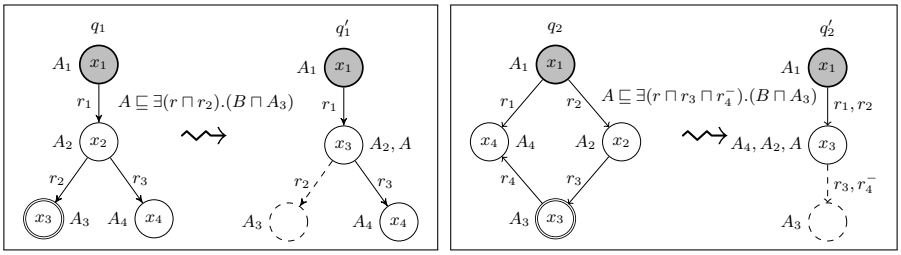
Theorem 1. *Let $\mathcal{O} = (\mathcal{T}, \mathcal{A})$ be a Horn-$\mathcal{ALCHIQ}^{\sqcap}$ ontology. Then $\mathcal{O}$ is consistent iff $\mathcal{A} \cup \mathsf{cr}(\mathcal{T})$ consistent. Moreover, if $\mathcal{O}$ is consistent, then (a) chase$(MM(\mathcal{A} \cup \mathsf{cr}(\mathcal{T})), \Xi(\mathcal{T}))$ is a model of $\mathcal{O}$, and (b) chase$(MM(\mathcal{A} \cup \mathsf{cr}(\mathcal{T})), \Xi(\mathcal{T}))$ can be homomorphically mapped into any model of $\mathcal{O}$.*

The proof of Theorem 1 can be found in [9]; see [21] for a proof of a similar result.

Observe that checking consistency of $\mathcal{O} = (\mathcal{T}, \mathcal{A})$ reduces to evaluating the Datalog program $\mathcal{A} \cup \mathsf{cr}(\mathcal{T})$. We note that $\Xi(\mathcal{T})$ can be computed in exponential time in the size of $\mathcal{T}$: the calculus only infers axioms of the form $M \sqsubseteq B$ and $M \sqsubseteq \exists S.N$, where M, N are conjunctions of atomic concepts, B is atomic and S is a conjunction of roles, and there are exponentially many such axioms.

4 Query Rewriting

The following theorem, which is immediate from Theorem 1, allows us to concentrate on models obtained by the chase procedure.



(a) Example 1 (b) Example 2

Fig. 1: Examples of query rewriting

Theorem 2. *Let $\mathcal{O} = (\mathcal{T}, \mathcal{A})$ be a Horn-$\mathcal{ALCHIQ}^\sqcap$ ontology. Then $\mathcal{A} \cup \mathsf{cr}(\mathcal{T})$ is consistent iff $\mathcal{O}$ is consistent. Moreover, if $\mathcal{O}$ is consistent, then $ans(\mathcal{O}, q) = ans(\mathcal{I}_\mathcal{O}, q)$, where $\mathcal{I}_\mathcal{O} = chase(MM(\mathcal{A} \cup \mathsf{cr}(\mathcal{T})), \Xi(\mathcal{T}))$.*

Computing $ans(\mathcal{I}_\mathcal{O}, q)$ is still not trivial, as $\mathcal{I}_\mathcal{O}$ can be infinite. Hence, we rewrite q into a set Q of CQs such that $ans(\mathcal{I}_\mathcal{O}, q) = \bigcup_{q' \in Q} ans(MM(\mathcal{A} \cup \mathsf{cr}(\mathcal{T})), q')$, that is, we can evaluate them over the finite $MM(\mathcal{A} \cup \mathsf{cr}(\mathcal{T}))$.

The intuition is the following. Suppose q has a non-distinguished variable x, and that there is some match π in $\mathcal{I}_\mathcal{O}$ such that $\pi(x)$ is an object in the 'tree part' introduced by the chase procedure, and it has no descendant in the image of π. Then for all atoms $r(y, x)$ of q, the "neighbor" variable y must mapped to the parent of $\pi(x)$. A rewrite step makes a choice of such an x, and employs an existential axiom from $\Xi(\mathcal{T})$ to 'clip off' x, eliminating all query atoms involving it. By repeating this procedure, we can clip off all variables matched in the tree part and obtain a query with a match in $MM(\mathcal{A} \cup \mathsf{cr}(\mathcal{T}))$.

Definition 2 (Query rewriting). *For a CQ q and a Horn-$\mathcal{ALCHIQ}^\sqcap$ TBox $\mathcal{T}$, we write $q \to_\mathcal{T} q'$ if q' can be obtained from q in the following steps:*

(S1) Select in q an arbitrary non-distinguished variable x such that there are no atoms of the form $r(x, x)$ in q.
(S2) Replace each role atom $r(x, y)$ in q, where y is arbitrary, by the atom $\mathsf{inv}(r)(y, x)$.
(S3) Let $V_p = \{y \mid \exists r : r(y, x) \in q\}$, and select some $M \sqsubseteq \exists S.N \in \Xi(\mathcal{T})$ such that
  (a) $\{r \mid r(y, x) \in q \wedge y \in V_p\} \subseteq S$, and
  (b) $\{A \mid A(x) \in q\} \subseteq N$.
(S4) Drop from q each atom containing x.
(S5) Rename each $y \in V_p$ of q by x.
(S6) Add the atoms $\{A(x) \mid A \in M\}$ to q.

We write $q \to_\mathcal{T}^* q'$ if $q = q_0$ and $q' = q_n$ for some finite rewrite sequence $q_0 \to_\mathcal{T} q_1 \cdots \to_\mathcal{T} q_n$, $n \geq 0$. Furthermore, we let $\mathsf{rew}_\mathcal{T}(q) = \{q' \mid q \to_\mathcal{T}^* q'\}$.

Example 1. The query $q_1(x_1) \leftarrow A_1(x_1), r_1(x_1, x_2), A_2(x_2), r_2(x_2, x_3), A_3(x_3), r_3(x_2, x_4), A_4(x_4)$ is depicted on the left hand side of Figure 1a. The node in bold corresponds to the answer variable x_1. Assume that $A \sqsubseteq \exists(r \sqcap r_2).(B \sqcap A_3)$ and $A \sqsubseteq \exists(r \sqcap r_3 \sqcap r_4^-).(B$

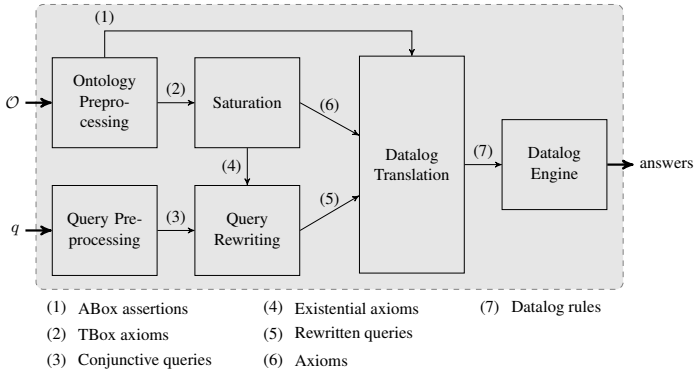


Fig. 2: CLIPPER system architecture

$\sqcap A_3$) are in $\Xi(\mathcal{T})$. If we pick for (S1) the variable x_3, we get $V_p = \{x_2\}$ and we can select $A \sqsubseteq \exists(r \sqcap r_2).(B \sqcap A_3) \in \Xi(\mathcal{T})$, as it satisfies (S3.a) and (S3.b). After performing (S4), (S5) and (S6) we obtain the rewritten query $q_1'(x_1) \leftarrow A_1(x_1), r_1(x_1, x_3), A_2(x_3), A(x_3), r_3(x_3, x_4), A_4(x_4)$. Intuitively, we can safely remove from q_1 all atoms containing x_3 because the added atom $A(x_3)$ ensures that whenever q_1' has a match so does q_1.

Example 2 (ctd). Now we consider the query $q_2(x_1) \leftarrow A_1(x_1), r_2(x_1, x_2), A_2(x_2), r_3(x_2, x_3), A_3(x_3), r_1(x_1, x_4), A_4(x_4), r_4(x_3, x_4)$ in Figure 1b. We choose the variable x_3, replace $r_4(x_3, x_4)$ by $r_4^-(x_4, x_3)$ in step (S2), and get $V_p = \{x_2, x_4\}$. Intuitively, if $\pi(x_3)$ is a leaf in a tree-shaped match π, then x_2 and x_4 must both be mapped to the parent of $\pi(x_3)$. Since the GCI $A \sqsubseteq \exists(r \sqcap r_3 \sqcap r_4^-).(B \sqcap A_3)$ in $\Xi(\mathcal{T})$ satisfies (S3.a,b), we can drop the atoms containing x_3 from q_2, and perform (S5) and (S6) to obtain the rewritten query $q_2'(x_1) \leftarrow A_1(x_1), r_1(x_1, x_3), r_2(x_1, x_3), A_4(x_3), A_2(x_3), A(x_3)$.

Now we can state our main result (see [9] for the proof of a more general result):

Theorem 3. *Suppose $\mathcal{O} = (\mathcal{T}, \mathcal{A})$ is a consistent Horn-$\mathcal{ALCHIQ}^\sqcap$ ontology and let q be a CQ. Then $ans(\mathcal{O}, q) = \bigcup_{q' \in \mathsf{rew}_\mathcal{T}(q)} ans(MM(\mathcal{A} \cup \mathsf{cr}(\mathcal{T})), q')$.*

By the above reduction, we can answer q over $\mathcal{O} = (\mathcal{T}, \mathcal{A}))$ by evaluating $\mathsf{rew}_\mathcal{T}(q)$ over $MM(\mathcal{A} \cup \mathsf{cr}(\mathcal{T})$ or, equivalently, by evaluating the Datalog program $\mathsf{rew}_\mathcal{T}(q) \cup \mathcal{A} \cup \mathsf{cr}(\mathcal{T})$ and collecting the tuples $\boldsymbol{u}$ with $q(\boldsymbol{u})$ in the minimal model. We note that $\mathsf{rew}_\mathcal{T}(q)$ is finite and computable in time exponential in the size of $\mathcal{T}$ and q: rules in $\mathsf{rew}_\mathcal{T}(q)$ use only relation names and variables that occur in q and $\mathcal{T}$. Furthermore, the *grounding* of $\mathsf{rew}_\mathcal{T}(q) \cup \mathcal{A} \cup \mathsf{cr}(\mathcal{T})$ is exponential in the size of $\mathcal{O}$, but polynomial for fixed $\mathcal{T}$ and q. By the complexity of Datalog, it follows that the resulting algorithm is exponential in combined but polynomial in data complexity; this is worst-case optimal [7].

5 Implementation

To evaluate the feasibility of the new rewriting, we have implemented a prototype system CLIPPER,[3] which supports CQ answering over Horn-$\mathcal{SHIQ}$ ontologies (non-simple

[3] `http://www.kr.tuwien.ac.at/research/systems/clipper/`

Algorithm 1: Answering CQs via Query Rewriting

Input: Horn-$\mathcal{SHIQ}$ ontology $\mathcal{O} = (\mathcal{T}, \mathcal{A})$, Conjunctive Query q
Output: query results
$\mathcal{T} \leftarrow$ `Normalize`$(\mathcal{T})$; ▷ Normalization
$\mathcal{T}^* \leftarrow$ `ElimTrans`$(\mathcal{T})$; ▷ Eliminate Transitive Roles
$\Xi(\mathcal{T}^*) \leftarrow$ `Saturate`$(\mathcal{T}^*)$; ▷ TBox Saturation
$Q \leftarrow$ `Rewrite`$(q, \Xi(\mathcal{T}^*))$; ▷ Query Rewriting
$\mathsf{cr}(\mathcal{T}) \leftarrow$ `CompletionRules`$(\mathcal{T})$; ▷ Completion Rules
$\mathcal{P} = \mathcal{A} \cup \mathsf{cr}(\mathcal{T}) \cup Q$; ▷ Datalog Translation
$ans \leftarrow \{\boldsymbol{u} \mid q(\boldsymbol{u}) \in$ `MinimalModel`$(\mathcal{P})\}$; ▷ Call Datalog Reasoner
return ans ;

roles are disallowed in queries). To the best of our knowledge, it is the first such system for Horn-$\mathcal{SHIQ}$ (under the standard semantics of first-order logic), and in expressiveness subsumes similar $\mathcal{DL}$-*Lite* and $\mathcal{EL}$ reasoning engines (see below).

We describe the architecture of CLIPPER in Figure 2, and the main steps in Algorithm 1. CLIPPER is implemented in Java and uses OWLAPI 3.2.2 [13] for parsing ontologies. It accepts an ontology $\mathcal{O} = (\mathcal{T}, \mathcal{A})$ and a CQ q in the SPARQL syntax as input. For efficiency reasons we implemented a lightweight ontology representation: all concepts, roles and individuals are encoded as integers; the conjunction of concepts and roles are stored in hash sets. Since we often need to manipulate large tables of axioms, we built inverted indexes over such axioms to support fast lookup and matching.

Ontology Preprocessing. This component is responsible for (1) ontology parsing (using OWLAPI 3.2.2), (2) profile checking and ontology normalization [16], and (3) converting the ontology into the internal format.

Query Preprocessing. This component simply parses CQs in SPARQL syntax and converts them into the internal format.

Saturation. This component exhaustively applies the saturation rules in Table 1 on TBox. We use the index structure to find which rules can be applied and the new axioms are generated incrementally.

Query Rewriting. This component uses Algorithm 2 to rewrite the input q. It implements the rewriting step from Definition 2, exhaustively traversing all existential axioms in $\Xi(\mathcal{T})$ for all the non-distinguished variables. The index structure helps the system efficiently search through the set of existential axioms while rewriting.

Datalog Translation. This component generates a Datalog program with the rewritten set of queries Q, the completion rules $\mathsf{cr}(\mathcal{T})$ in Table 2, and the facts in $\mathcal{A}$.

Datalog Engine. The resulting program is evaluated using the Datalog engine DLV-20101014 [18] or Clingo 3.0.3 [10]. If the program (and hence the ontology) is consistent, its minimal model is returned and the answer tuples are filtered from it.

6 Experiments

We tested CLIPPER on a Pentium Core2 Duo 2.00GHZ with 2GB RAM under Ubuntu 10.04 and 512MB heap for the Java VM. We conducted the following experiments.

Algorithm 2: Rewrite(q, $\mathcal{T}$)

Input: CQ q with only simple roles; TBox $\mathcal{T}$
Output: Rewritten queries of q w.r.t. $\mathcal{T}$

$\mathsf{rew}_{\mathcal{T}}(q) \leftarrow \emptyset$; ▷ will be updated from the sub procedure
`rewrite`(q) ; ▷ Call sub procedure
return $\mathsf{rew}_{\mathcal{T}}(q)$;

Sub Procedure `rewrite` (q)
$\mathsf{rew}_{\mathcal{T}}(q) \leftarrow \mathsf{rew}_{\mathcal{T}}(q) \cup \{q\}$;
foreach *non-distinguished variables* x *of* q **do**
  if $r(x, x) \notin q$ **then**
    Replace each $r(x, y)$ in q by $r^{-}(y, x)$;
    $S \leftarrow \{r \mid r(y, x) \in q\}$; $P \leftarrow \{y \mid r(y, x) \in q\}$; $N \leftarrow \{A \mid A(x) \in q\}$;
    foreach $M \sqsubseteq \exists S' N' \in \mathcal{T}$ **do**
      if $S \subseteq S'$ ***and*** $N \subseteq N'$ **then**
        Obtain q' from q by:
        begin
          (1) Drop from q each atom containing the variable x;
          (2) Rename each $y \in P$ by x;
          (3) Add $\{A(x) \mid A \in M\}$ to q;
        if $q' \notin \mathsf{rew}_{\mathcal{T}}(q)$ **then**
          `rewrite`(q') ; ▷ Recursion

1. Downscaling test. We compared CLIPPER with other query rewriting systems for $\mathcal{DL}$*-Lite*, viz. REQUIEM (Perez-Urbina et al. [23]) and PRESTO [25], and found that it is competitive and scales down well on $\mathcal{DL}$*-Lite* ontologies. We used the ontologies and queries (Q1–Q5) from the REQUIEM test suite, which have been widely used for system tests; in addition we considered the queries in Table 3a.

Table 3b shows the number of rewritten queries and the rewriting time for the ontologies ADOLENA (A), STOCK-EXCHANGE (S), VICODI (V), and UNIVERSITY (U); the rewriting time excludes loading and preprocessing. CLIPPER and PRESTO generated in most cases rule sets of comparable size, and in short time. In a few cases PRESTO generated significantly less rules than CLIPPER, and only for V PRESTO was notably faster. REQUIEM generated in several cases significantly more rules, despite considering the G-version which generates optimized rules (and hence uses considerably more time). The difference seems to be caused by rule unfolding required in their rewriting.

For UNIVERSITY, the only ontology in the suite having an ABox, we evaluated the rewritten queries over four different ABoxes (67k to 320k assertions) using DLV. Interestingly, in all cases the execution times for the three rewritings were very similar; the average runtime of each query on the four ABoxes is shown in brackets.

2. Full Horn-$\mathcal{SHIQ}$. To test CLIPPER on a full Horn-$\mathcal{SHIQ}$ ontology, we modified the UOBM ontology [19], which is in $\mathcal{SHOIN}(\mathrm{D})$, by dropping or strengthening (in case of disjunctions) non-Horn-$\mathcal{SHIQ}$ TBox axioms; the final ontology has 196 TBox axioms. We used ABoxes $\mathcal{A}_i$, $1 \leq i \leq 4$, with 20k, 80k, 140k and 200k assertions. The test queries in Table 4a were tailored to require reasoning with Horn-$\mathcal{SHIQ}$ constructs

Table 3: Experiments with $\mathcal{DL}$-*Lite* ontology

(a) Additional Queries

A	Q6(x) ←Device(x), assistsWith(x,y), ReadingDevice(y) Q7(x) ←Device(x), assistsWith(x,y), ReadingDevice(y), assistsWith(y,z), SpeechAbility(z)
S	Q6(x,z) ←Investor(x), hasStock(x,y), Stock(y), Company(z), hasStock(z,y) Q7(x,z,w) ← Investor(x), hasStock(x,y), Stock(y), isListedIn(y,z), StockExchangeList(z), Company(w), hasStock(w,y)
U	Q6(x,y) ←Professor(x), teacherOf(x,y), GraduateCourse(y) Q7(x,z) ←Faculty(y), Professor(z), Student(x), memberOf(x,y),worksFor(z,y)
V	Q6(x,y,z) ←Person(x), hasRole(x,y), Leader(y), exists(y,z) Q7(x,y,z,w) ← Person(x), hasRole(x,y), Leader(y), exists(y,z), TemporalInterval(z), related(x,w), Country(w)

(b) Downscaling evaluation

		# Rules/CQs			Time (ms)		
		RG	Presto	CLIPPER	RG	Presto	CLIPPER
A	Q1	27	53	42	281	45	50
	Q2	50	32	31	184	46	62
	Q3	104	32	31	292	27	65
	Q4	224	43	36	523	32	71
	Q5	624	37	36	1177	25	70
	Q6	364	35	30	523	31	65
	Q7	2548	43	32	7741	61	64
S	Q1	6	7	10	14	7	19
	Q2	2	3	22	263	9	22
	Q3	4	4	9	1717	10	21
	Q4	4	4	24	1611	9	23
	Q5	8	5	10	18941	10	22
	Q6	4	8	5	204	11	21
	Q7	8	6	7	1733	11	17

		# Rules/CQs			Time (ms)		
		RG	Presto	CLIPPER	RG	Presto	CLIPPER
V	Q1	15	16	15	13	8	73
	Q2	10	3	10	16	10	58
	Q3	72	28	26	77	12	63
	Q4	185	44	41	261	17	71
	Q5	30	16	8	99	15	44
	Q6	18	22	18	27	11	69
	Q7	180	34	27	359	12	105
U	Q1	2	4	2	14 (1247)	12 (1252)	27 (1255)
	Q2	1	2	45	201 (1247)	23 (1262)	36 (1637)
	Q3	4	8	17	477 (2055)	26 (2172)	29 (1890)
	Q4	2	56	63	2431 (1260)	20 (1235)	28 (1735)
	Q5	10	8	16	7216 (1267)	26 (1305)	36 (1372)
	Q6	10	13	10	13 (1272)	14 (1260)	27 (1262)
	Q7	960	24	19	1890 (1730)	15 (1310)	35 (1322)

unavailable in $\mathcal{DL}$-*Lite* and $\mathcal{EL}$. Table 4b shows the number of rewritten queries, rewriting time and DLV running time. We see that CLIPPER answered all queries in reasonable time and scaled well (time printed $\mathcal{A}_1$/ $\mathcal{A}_2$ / $\mathcal{A}_3$ /$\mathcal{A}_4$). The rewriting times for all the queries are small and at most within a factor of 3. The high number of rules generated for Q3 is due to many different possibilities for deriving some atoms in the query, like Person(x). However, the evaluation still performs well (it stays within a small factor).

7 Related Work

Since Calvanese et al. introduced query rewriting in their seminal work on $\mathcal{DL}$-*Lite* [3], many query rewriting techniques have been developed and implemented, e.g. (Perez-Urbina et al. [23], Rosati and Almatelli [25], Chortaras et al. [4], Gottlob et al. [11]), usually aiming at an optimized rewriting size. Some of them also go beyond $\mathcal{DL}$-*Lite*; e.g. Perez-Urbina et al. cover $\mathcal{ELHI}$, while Gottlob et al. consider $Datalog^{\pm}$. Most approaches rewrite a query into a (union of) CQs; in [25] a non-recursive Datalog program is generated, while Perez-Urbina et al. produce a CQ for $\mathcal{DL}$-*Lite* and a (recursive) Datalog program for DLs of the $\mathcal{EL}$ family. Our approach rewrites a CQ into a union of CQs, but generates possibly recursive Datalog rules to capture the TBox. The closest DL to Horn-$\mathcal{SHIQ}$ for which a rewriting technique has been implemented is $\mathcal{ELHI}$ [23]. Unlike $\mathcal{ELHI}$, Horn-$\mathcal{SHIQ}$ can express functionality, a feature supported by $\mathcal{DL}$-*Lite*

Table 4: Experiments with UOBM Horn-$\mathcal{SHIQ}$ ontology

(a) Queries

Q1(x) ←worksFor(x,y), isAffiliatedOrganizationOf(y,z), College(z)
Q2(x) ←Postdoc(x), worksFor(x,y), University(y), hasAlumnus(y,x)
Q3(x) ←Person(x), like(x,y), Chair(z), isHeadOf(z,w), like(z,y)
Q4(x) ←takeCourse(x,y), GraduateCourse(y), isTaughtBy(y,z), Professor(z)
Q5(x,z) ←LeisureStudent(x), takesCourse(x,y), CSCourse(y), isStudentOf(x, z), University(z)
Q6(x,y) ←enrollIn(x,y), hasDegreeFrom(x,y), University(y)
Q7(x,y) ←PeopleWithManyHobbies(x), isMemberOf(x,z), like(x,w), TennisClass(w), hasMember(z,y), like(y,w)
Q8(x,z) ←TennisFan(x), like(x,y), Sport(y), isHeadOf(x,z), ReserachGroup(z)
Q9(x,y,z) ←Student(x), hasDegreeFrom(x,y), Professor(z), worksFor(z,y), isAdvisorOf(z,x)
Q10(x,y,w) ←Professor(x), Dean(y), isMemberOf(y,w), worksFor(x,w), hasPublication(x,z), isPublicationOf(z,y)

(b) Running time

	# of Rules	Rew (ms)	Datalog time (DLV) (ms)
Q1	3	87	120/ 370/ 620/ 870
Q2	16	98	130/ 380/ 630/ 880
Q3	180	212	370/ 890/ 1420/ 1960
Q4	45	156	180/ 480/ 780/ 1070
Q5	37	152	160/ 440/ 720/ 1010
Q6	16	112	130/ 370/ 630/ 880
Q7	55	143	180/ 470/ 750/ 1050
Q8	17	105	130/ 380/ 630/ 870
Q9	33	126	150/ 430/ 720/ 1010
Q10	23	137	150/ 390/ 640/ 880

considered relevant for applications. A comparison of both systems on ontologies beyond $\mathcal{DL}$*-Lite* remains for future work. Our technique resembles Rosati's for CQs in $\mathcal{EL}$ [24], which incorporates the CQ into the TBox *before* saturation and then (after saturation) translates it into Datalog, resulting in a best-case exponential algorithm. We avoid this by doing a rewrite step only if the TBox has an applicable existential axiom.

Rewriting approaches for more expressive DLs are less common. The most notable exception is Hustadt et al.'s translation of $\mathcal{SHIQ}$ terminologies into disjunctive Datalog [15], which is implemented in the KAON2 reasoner. The latter can be used to answer queries over arbitrary ABoxes, but supports only instance queries. An extension to CQs (without transitive roles) is given in [14], but it is not implemented. To our knowledge, also the extension of the rewriting in [23] to nominals remains to be implemented [22]. In [20] a Datalog rewriting is used to establish complexity bounds of standard reasoning in the Horn fragments of $\mathcal{SHOIQ}$ and $\mathcal{SROIQ}$, but it does not cover CQs.

8 Conclusion

We presented a rewriting-based algorithm for answering CQs over Horn-$\mathcal{SHIQ}$ ontologies. Our prototype implementation shows potential for practical applications, and further optimizations will improve it. Future versions of CLIPPER will support transitive roles and queries formulated in weakly DL-safe Datalog, for which the theoretic foundations have been already developed here and in [9].

As an interesting application, we mention that our method allows to improve reasoning with *DL-programs*, which loosely couple rules and ontologies [5]. To avoid the overhead caused by the interaction of a rule reasoner and an ontology reasoner of traditional methods, the *inline evaluation* framework translates ontologies into rules [12, 8]. The techniques of this paper can be faithfully integrated into the inline evaluation framework to efficiently evaluate DL-programs involving Horn-$\mathcal{SHIQ}$ ontologies.

References

1. Acciarri, A., Calvanese, D., Giacomo, G.D., Lembo, D., Lenzerini, M., Palmieri, M., Rosati, R.: QuOnto: Querying ontologies. In: AAAI. pp. 1670–1671 (2005)

2. Calì, A., Gottlob, G., Lukasiewicz, T.: Datalog$^{\pm}$: a unified approach to ontologies and integrity constraints. In: ICDT'09. pp. 14–30. ACM (2009)
3. Calvanese, D., Giacomo, G.D., Lembo, D., Lenzerini, M., Rosati, R.: Tractable reasoning and efficient query answering in description logics: The DL-Lite family. J. Autom. Reasoning 39(3), 385–429 (2007)
4. Chortaras, A., Trivela, D., Stamou, G.: Optimized query rewriting for OWL 2 QL. In: CADE'11. pp. 192–206. Springer-Verlag (2011)
5. Eiter, T., Ianni, G., Lukasiewicz, T., Schindlauer, R., Tompits, H.: Combining answer set programming with description logics for the Semantic Web. Artificial Intelligence 172(12-13), 1495–1539 (2008)
6. Eiter, T., Ortiz, M., Šimkus, M., Tran, T., Xiao, G.: Query rewriting for Horn-SHIQ plus rules. In: AAAI'12 (2012), (To appear)
7. Eiter, T., Gottlob, G., Ortiz, M., Simkus, M.: Query answering in the description logic Horn-SHIQ. In: JELIA'08. pp. 166–179. Springer (2008)
8. Eiter, T., Krennwallner, T., Schneider, P., Xiao, G.: Uniform evaluation of nonmonotonic DL-programs. In: FoIKS'12. LNCS, vol. 7153, pp. 1–22. Springer (March 2012)
9. Eiter, T., Ortiz, M., Šimkus, M., Tran, T., Xiao, G.: Query rewriting for Horn-SHIQ plus rules. Tech. Rep. INFSYS RR-1843-12-04, TU Vienna (2012), http://www.kr.tuwien.ac.at/research/reports/rr1204.pdf
10. Gebser, M., Kaufmann, B., Kaminski, R., Ostrowski, M., Schaub, T., Schneider, M.T.: Potassco: The potsdam answer set solving collection. AI Commun. 24(2), 107–124 (2011)
11. Gottlob, G., Orsi, G., Pieris, A.: Ontological queries: Rewriting and optimization. In: ICDE'11. pp. 2 –13 (2011)
12. Heymans, S., Eiter, T., Xiao, G.: Tractable reasoning with DL-programs over Datalog-rewritable description logics. In: ECAI'10. pp. 35–40. IOS Press (2010)
13. Horridge, M., Bechhofer, S.: The OWL API: A java API for OWL ontologies. Semantic Web 2(1), 11–21 (2011)
14. Hustadt, U., Motik, B., Sattler, U.: A decomposition rule for decision procedures by resolution-based calculi. In: LPAR'04. pp. 21–35. Springer (2004)
15. Hustadt, U., Motik, B., Sattler, U.: Reasoning in description logics by a reduction to disjunctive Datalog. J. Autom. Reasoning 39(3), 351–384 (2007)
16. Kazakov, Y.: Consequence-driven reasoning for Horn SHIQ ontologies. In: IJCAI'09. pp. 2040–2045 (2009)
17. Krötzsch, M., Rudolph, S., Hitzler, P.: Complexity boundaries for Horn description logics. In: AAAI'07. pp. 452–457. AAAI Press (2007)
18. Leone, N., Pfeifer, G., Faber, W., Eiter, T., Gottlob, G., Perri, S., Scarcello, F.: The DLV system for knowledge representation and reasoning. ACM ToCL 7(3), 499–562 (2006)
19. Ma, L., Yang, Y., Qiu, Z., Xie, G.T., Pan, Y., Liu, S.: Towards a complete OWL ontology benchmark. In: ESWC'06. pp. 125–139. Springer (2006)
20. Ortiz, M., Rudolph, S., Simkus, M.: Worst-case optimal reasoning for the Horn-DL fragments of OWL 1 and 2. In: KR'10. AAAI Press (2010)
21. Ortiz, M., Rudolph, S., Simkus, M.: Query answering in the Horn fragments of the description logics SHOIQ and SROIQ. In: IJCAI'11. pp. 1039–1044. IJCAI/AAAI (2011)
22. Pérez-Urbina, H., Motik, B., Horrocks, I.: Tractable query answering and rewriting under description logic constraints. J. Applied Logic 8(2), 186–209 (2010)
23. Pérez-Urbina, H., Motik, B., Horrocks, I.: A comparison of query rewriting techniques for DL-Lite. In: DL'09. CEUR-WS.org (2009)
24. Rosati, R.: On conjunctive query answering in EL. In: DL'07. CEUR-WS.org (2007)
25. Rosati, R., Almatelli, A.: Improving query answering over DL-Lite ontologies. In: KR'10. AAAI (2010)