Computing Datalog Rewritings for Disjunctive Datalog Programs and Description Logic Ontologies

Mark Kaminski, Yavor Nenov, and Bernardo Cuenca Grau

Department of Computer Science
University of Oxford, UK

Abstract. We study the closely related problems of rewriting disjunctive datalog programs and non-Horn DL ontologies into plain datalog programs that entail the same facts for every dataset. We first propose the class of *markable* disjunctive datalog programs, which is efficiently recognisable and admits polynomial rewritings into datalog. Markability naturally extends to $\mathcal{SHI}$ ontologies, and markable ontologies admit (possibly exponential) datalog rewritings. We then turn our attention to resolution-based rewriting techniques. We devise an enhanced rewriting procedure for disjunctive datalog, and propose a second class of $\mathcal{SHI}$ ontologies that admits exponential datalog rewritings via resolution. Finally, we focus on conjunctive query answering over disjunctive datalog programs. We identify classes of queries and programs that admit datalog rewritings and study the complexity of query answering in this setting. We evaluate the feasibility of our techniques over a large corpus of ontologies, with encouraging results.

1 Introduction

Answering conjunctive queries is a key reasoning problem for many applications of ontologies. Query answering can sometimes be implemented via rewriting into datalog, where a rewriting of a query q w.r.t. an ontology $\mathcal{O}$ is a datalog program $\mathcal{P}$ that preserves the answers to q for any dataset. Rewriting queries into datalog not only ensures tractability in data complexity—an important requirement in data-intensive applications—but also enables the reuse of scalable rule-based reasoners such as OWLIM [4], Oracle's Data Store [21], and RDFox [16].

Datalog rewriting techniques have been investigated in depth for Horn Description Logics (i.e., DLs whose ontologies can be normalised as first-order Horn clauses), and optimised algorithms have been implemented in systems such as Requiem [18], Clipper [6], and Rapid [20]. Techniques for non-Horn DLs, however, have been studied to a lesser extent, and only for atomic queries.

If we restrict ourselves to atomic queries, rewritability for non-Horn DL ontologies is strongly related to the rewritability of disjunctive datalog programs into datalog: every $\mathcal{SHIQ}$ ontology can be transformed into a (positive) disjunctive datalog program that entails the same facts for every dataset (and hence preserves answers to all atomic queries) [8].[1] It is well-known that disjunctive datalog programs cannot be generally rewritten into plain datalog. In particular, datalog rewritings may not exist even for disjunctive programs that correspond to ontologies expressed in the basic DL $\mathcal{ELU}$ [11, 5], and sufficient conditions that ensure rewritability were identified in [9]. Deciding datalog rewritability of atomic queries w.r.t. $\mathcal{SHI}$ ontologies was proved NExpTime-complete in [3].

In our previous work [10], we proved a characterisation of datalog rewritability for disjunctive programs based on linearity: a restriction that requires each rule to contain at most one IDB atom in the body. It was shown that every linear disjunctive program can be polynomially rewritten into plain datalog; conversely, every datalog program can be polynomially translated into an equivalent linear disjunctive datalog program. We then proposed *weakly linear disjunctive datalog*, which extends both datalog and linear disjunctive datalog, and which admits polynomial datalog rewritings. In a weakly linear program, the linearity requirement is relaxed: instead of applying to all IDB predicates, it applies only to those that "depend" on a disjunctive rule.

A different approach to rewriting disjunctive programs into datalog by means of a resolution-based procedure was proposed in [5]. The procedure works by saturating the input disjunctive program $\mathcal{P}$ such that in each resolution step at least one of the premises is a non-Horn rule; if this process terminates, the procedure outputs the subset of datalog rules in the saturation, which is guaranteed to be a rewriting of $\mathcal{P}$. The procedure was shown to terminate for so-called *simple* disjunctive programs; furthermore, it was shown that ontologies expressed in certain logics of the DL-Lite$_{\mathsf{bool}}$ family [1] can be transformed into disjunctive programs that satisfy the simplicity condition.

If we wish to go beyond atomic queries and consider general conjunctive queries, it is no longer possible to obtain query-independent datalog rewritings. Lutz and Wolter [12] showed that for *any* non-Horn ontology (or disjunctive program) $\mathcal{O}$ there exists a conjunctive query q such that answering the (fixed) q w.r.t. (fixed) $\mathcal{O}$ and an input dataset is co-NP-hard; thus, under standard complexity-theoretic assumptions no datalog rewriting for such q and $\mathcal{O}$ exists. To the best of our knowledge, no rewriting techniques for arbitrary CQs w.r.t. non-Horn ontologies and programs have been developed.

In this paper, we propose significant enhancements over existing techniques for rewriting atomic queries [10, 5], which we then extend to the setting of arbitrary conjunctive queries. Furthermore, we evaluate the practical feasibility of our techniques over a large corpus of non-Horn ontologies. Specifically, our contributions are as follows.

In Section 3, we propose the class of *markable* disjunctive datalog programs, in which the weak linearity condition from [10] is further relaxed. We show that our extended class of programs is efficiently recognisable and that each markable program admits a polynomial datalog rewriting. These results can be readily applied to ontology reasoning. We first consider the "intersection"

[1] Disjunctive datalog typically allows for negation-as-failure, which we don't consider since we focus on monotonic reasoning.

between OWL 2 and disjunctive datalog (which we call $\mathrm{RL}^{\sqcup}$), and show that fact entailment over $\mathrm{RL}^{\sqcup}$ ontologies corresponding to a markable program is tractable in combined complexity (and hence no harder than in OWL 2 RL [15]). We then lift the markability condition to ontologies, and show that markable $\mathcal{SHI}$-ontologies admit a (possibly exponential) datalog rewriting.

In Section 4, we refine the resolution-based rewriting procedure from [5] by further requiring that only atoms involving disjunctive predicates can participate in resolution inferences. This refinement can significantly reduce the number of inferences drawn during saturation, without affecting correctness. We then focus on ontologies, and propose an extension of the logics in the DL-Lite$_{\mathsf{bool}}$ family that admits (possibly exponential) datalog rewritings.

In Section 5, we shift our attention to conjunctive queries and propose classes of queries and disjunctive datalog programs that admit datalog rewritings. Furthermore, we discuss the implications of these results to ontology reasoning.

We have implemented and evaluated our techniques on a large ontology repository. Our results show that many realistic non-Horn ontologies can be rewritten into datalog. Furthermore, we have tested the scalability of query answering over the programs obtained using our techniques, with promising results.

The proofs of our technical results can be found in an extended version of the paper available online: `https://krr-nas.cs.ox.ac.uk/RR2014/report.pdf`

2 Preliminaries

We consider standard notions of terms, atoms, literals, formulae, sentences, and entailment. A *fact* is a ground atom and a *dataset* is a finite set of facts. We assume that equality $\approx$ is an ordinary predicate and that each set of formulae contains the axiomatisation of $\approx$ as a congruence relation for its signature. Clauses, substitutions, most general unifiers (MGUs), clause subsumption, tautologies, binary resolution, and factoring are as usual [2]. Clause C *θ-subsumes* D if C subsumes D and C has no more literals than D. Clause C is *redundant* in a set of clauses if C is tautological or if C is θ-subsumed by another clause in the set. A *condensation* of a clause C is a minimal subset that is subsumed by C.

A *rule* r is a function-free sentence $\forall \boldsymbol{x} \forall \boldsymbol{z}.[\varphi(\boldsymbol{x}, \boldsymbol{z}) \rightarrow \psi(\boldsymbol{x})]$ where tuples of variables $\boldsymbol{x}$ and $\boldsymbol{z}$ are disjoint, $\varphi(\boldsymbol{x}, \boldsymbol{z})$ is a conjunction of distinct equality-free atoms, and $\psi(\boldsymbol{x})$ is a disjunction of distinct atoms. Formula φ is the *body* of r, and ψ is the *head*. Quantifiers in rules are omitted. We assume that rules are safe. A rule is *datalog* if $\psi(\boldsymbol{x})$ has at most one atom, and it is *disjunctive* otherwise. A *program* $\mathcal{P}$ is a finite set of rules; it is *datalog* if it consists only of datalog rules, and *disjunctive* otherwise. We assume that rules in $\mathcal{P}$ do not share variables. For convenience, we treat $\top$ and $\bot$ in a non-standard way as a unary and a nullary predicate, respectively. Given a program $\mathcal{P}$, $\mathcal{P}_{\top}$ is the program with a rule $P(x_1, \dots, x_n) \rightarrow \top(x_i)$ for each predicate P in $\mathcal{P}$ and each $1 \leq i \leq n$, and a rule $\rightarrow \top(a)$ for each constant a in $\mathcal{P}$. We assume that $\mathcal{P}_{\top} \subseteq \mathcal{P}$ and $\top$ does not occur in head position in $\mathcal{P} \setminus \mathcal{P}_{\top}$. We define $\mathcal{P}_{\bot}$ as consisting of a rule with $\bot$ as body and empty head. We assume $\mathcal{P}_{\bot} \subseteq \mathcal{P}$ and no rule in $\mathcal{P} \setminus \mathcal{P}_{\bot}$ has an

1.	$\bigsqcap_{i=1}^{n} A_i \sqsubseteq \bigsqcup_{j=1}^{m} C_j$	$\bigwedge_{i=1}^{n} A_i(x) \to \bigvee_{j=1}^{m} C_j(x)$
2.	$\exists R.A \sqsubseteq B$	$R(x,y) \wedge A(y) \to B(x)$
3.	$A \sqsubseteq \mathsf{Self}(R)$	$A(x) \to R(x,x)$
4.	$\mathsf{Self}(R) \sqsubseteq A$	$R(x,x) \to A(x)$
5.	$R \sqsubseteq S$	$R(x,y) \to S(x,y)$
6.	$R \sqsubseteq S^-$	$R(x,y) \to S(y,x)$
7.	$R \circ S \sqsubseteq T$	$R(x,z) \wedge S(z,y) \to T(x,y)$
8.	$A \sqsubseteq \geq m\, R.B$	$A(x) \to \exists^{\geq m} y.(R(x,y) \wedge B(y))$
9.	$A \sqsubseteq \leq m\, R.B$	$A(z) \wedge \bigwedge_{i=0}^{m} R(z,x_i) \wedge B(x_i) \to \bigvee_{0 \leq i < j \leq m} x_i \approx x_j$

Table 1. Normalised axioms. A, B are atomic or $\top$, C atomic or $\bot$, and R, S, T atomic.

empty head or $\bot$ in the body. Thus, $\mathcal{P} \cup \mathcal{D} \models \top(a)$ for every a in $\mathcal{P} \cup \mathcal{D}$, and $\mathcal{P} \cup \mathcal{D}$ is unsatisfiable iff $\mathcal{P} \cup \mathcal{D} \models \bot$. Head predicates in $\mathcal{P} \setminus \mathcal{P}_\top$ are *intensional* (or *IDB*) in $\mathcal{P}$. All other predicates (including $\top$) are *extensional* (*EDB*). An atom is intensional (extensional) if so is its predicate. A rule is *linear* if it has at most one IDB body atom. A program $\mathcal{P}$ is linear if all its rules are.

We assume familiarity with DLs. W.l.o.g. we consider normalised axioms as in Table 1. An ontology $\mathcal{O}$ is a finite set of axioms. An ontology $\mathcal{O}$ is $\mathcal{SHIQ}$ if each axiom of type 7 satisfies $R = S = T$;[2] it is $\mathcal{SHI}$ if it is $\mathcal{SHIQ}$, it does not contain axioms of type 9, and each axiom of type 8 satisfies $m = 1$; it is $\mathcal{ALCHI}$ if it is $\mathcal{SHI}$ and it has no axiom of type 7; it is $\mathrm{RL}^{\sqcup}$ if it does not contain axioms of type 8, and it is RL if it is $\mathrm{RL}^{\sqcup}$ and $m = 1$ for each axiom of type 1 and 9. Programs obtained from $\mathrm{RL}^{\sqcup}$ ontologies have rules with bounded number of variables: fact entailment is PTime-complete for RL and co-NP-complete for $\mathrm{RL}^{\sqcup}$ (in combined complexity).[3]

A *conjunctive query* (*CQ*) q is a datalog rule of the form $\varphi(\boldsymbol{x}, \boldsymbol{y}) \to A_q(\boldsymbol{x})$, with A_q a distinguished query predicate uniquely associated with q. A CQ is Boolean if A_q is propositional, and it is *atomic* if $\varphi(\boldsymbol{x}, \boldsymbol{y})$ consists of a single atom. A (disjunctive) program $\mathcal{P}$ is a *rewriting* of q w.r.t. a set of sentences $\mathcal{F}$ if for each dataset $\mathcal{D}$ over the signature of $\mathcal{F}$ and each tuple of constants $\boldsymbol{a}$ we have $\mathcal{F} \cup \mathcal{D} \cup \{q\} \models A_q(\boldsymbol{a})$ iff $\mathcal{P} \cup \mathcal{D} \models A_q(\boldsymbol{a})$. Program $\mathcal{P}$ is a rewriting of $\mathcal{F}$ if for each dataset $\mathcal{D}$ and each fact α over the signature of $\mathcal{F}$ we have $\mathcal{F} \cup \mathcal{D} \models \alpha$ iff $\mathcal{P} \cup \mathcal{D} \models \alpha$. Clearly, $\mathcal{P}$ is a rewriting of $\mathcal{F}$ if and only if $\mathcal{P}$ is a rewriting of every atomic query over the signature of $\mathcal{F}$. Hudstadt et al. [8] developed an algorithm for transforming a $\mathcal{SHIQ}$ ontology into a disjunctive program that preserves entailment of facts over non-transitive relations. This technique was extended in [5] to preserve all facts. Thus, every $\mathcal{SHIQ}$ ontology $\mathcal{O}$ admits a disjunctive datalog rewriting $\mathsf{DD}(\mathcal{O})$, which can be of exponential size.

[2] $\mathcal{SHIQ}$ enforces additional restrictions to ensure decidability, which we omit here.
[3] $\mathrm{RL}^{\sqcup}$ and RL allow for nominals, which we omit. All our results immediately extend.

$$\begin{aligned} \mathcal{P}_0 = \{ & C(x) \rightarrow B(x) \vee G(x) && (1) \\ & G(y) \wedge E(x,y) \rightarrow B(x) && (2) \\ & B(y) \wedge E(x,y) \rightarrow G(x) && (3) \\ & E(y,x) \rightarrow E(x,y) \, \} && (4) \end{aligned}$$

$$\begin{tikzcd}
& B \arrow[dd, bend left, "(3)"] & \\
C \arrow[ur, "(1)"] \arrow[dr, "(1)"'] & & E \arrow[ul, "(2)"'] \arrow[dl, "(3)"] \arrow[loop right, "(4)"] \\
& G \arrow[uu, bend left, "(2)"] &
\end{tikzcd}$$

Fig. 1. A weakly linear disjunctive datalog program

3 Datalog Rewritings Based on Linearity

In [10], we proposed the class of *weakly linear programs* (WL), which extends both datalog and linear disjunctive datalog. In a WL program predicates are partitioned into disjunctive (i.e., those whose extension may depend on a disjunctive rule) and datalog (those that depend solely on datalog rules). A program is WL if all rules have at most one occurrence of a disjunctive predicate in the body.

Definition 3.1. *The* dependency graph $G_{\mathcal{P}} = (V, E, \mu)$ *of a program* $\mathcal{P}$ *is the smallest edge-labeled digraph such that:*

1. V *contains every predicate occurring in* $\mathcal{P}$*;*
2. $r \in \mu(P,Q)$ *whenever* $P, Q \in V$*,* $r \in \mathcal{P} \setminus \mathcal{P}_{\top}$*,* P *occurs in the body of* r*, and* Q *occurs in the head of* r*; and*
3. $(P,Q) \in E$ *whenever* $\mu(P,Q)$ *is nonempty.*

A predicate Q depends on a rule $r \in \mathcal{P}$ *if* $G_{\mathcal{P}}$ *has a path that ends in* Q *and involves an* r*-labeled edge. Predicate* Q *is* datalog *if it only depends on datalog rules; otherwise,* Q *is* disjunctive*. Program* $\mathcal{P}$ *is* weakly linear *(WL for short) if each rule body in* $\mathcal{P}$ *has at most one occurrence of a disjunctive predicate.*

Consider the disjunctive program $\mathcal{P}_0$ and its dependency graph depicted in Fig. 1. Predicate C is EDB, predicates B and G depend on Rule (1) and hence are disjunctive, whereas E depends only on Rule (4) and hence it is datalog. Each rule has at most one disjunctive body atom and the program is WL.

WL programs admit a polynomial rewriting [10]. Roughly speaking, they are translated into datalog by "moving" all disjunctive body atoms to the head and all disjunctive head atoms to the body while replacing their predicates with fresh ones of higher arity; the new predicates are "initialised" using additional rules.

Markable Programs We next propose the class of *markable* disjunctive datalog programs, which extends WL programs. A key feature of a markable program is that one can identify a subset of disjunctive predicates, called *marked predicates*, such that the program can be translated into datalog by "moving" only those disjunctive atoms in a rule whose predicates are marked.

Definition 3.2. *Let* $\mathcal{P}$ *be a disjunctive program. A* marking *of* $\mathcal{P}$ *is a set* M *of disjunctive predicates in* $\mathcal{P}$ *such that:*

1. *Every rule in $\mathcal{P}$ has at most one body atom $Q(\boldsymbol{t})$ with $Q \in M$.*
2. *Every rule in $\mathcal{P}$ has at most one head atom $Q(\boldsymbol{t})$ with $Q \notin M$.*
3. *If $Q \in M$ and P is reachable from Q in $G_{\mathcal{P}}$, then $P \in M$.*

A predicate Q is marked *by M if $Q \in M$. An atom is marked if so is its predicate. A disjunctive program is markable if it has a marking.*

Markability generalises weak linearity in the following sense.

Proposition 3.3. *A disjunctive program $\mathcal{P}$ is WL if and only if the set of all disjunctive predicates in $\mathcal{P}$ constitutes a marking of $\mathcal{P}$.*

Let $\mathcal{P}_1$ extend $\mathcal{P}_0$ with the following rules:

$$V(x) \rightarrow C(x) \vee U(x) \qquad (5) \qquad\qquad C(x) \wedge U(x) \rightarrow \bot \qquad (6)$$

The dependency graph is given next. Note that C, U, B, and G are disjunctive as they depend on Rule (5). Thus, (6) has two disjunctive body atoms and $\mathcal{P}_1$ is not WL. The program, however, has markings $\{C, B, G\}$ and $\{U, B, G\}$.

$$\begin{tikzcd}
 & V \arrow[dl, "(5)"'] \arrow[dr, "(5)"] & & & B \arrow[d, bend left, "(3)"] & \\
U \arrow[dr, "(6)"'] & & C \arrow[dl, "(6)"] \arrow[urr, "(1)"] \arrow[drr, "(1)"'] & & & E \arrow[ul, "(2)"'] \arrow[dl, "(3)"] \arrow[loop right, "(4)"] \\
 & \bot & & & G \arrow[u, bend left, "(2)"] &
\end{tikzcd}$$

Checking markability of a disjunctive program $\mathcal{P}$ is amenable to efficient implementation via reduction to 2-SAT. To this end, we first associate with every predicate Q in $\mathcal{P}$ a distinct propositional variable X_Q. Then, for each rule $\varphi \wedge P_1(\boldsymbol{s}_1) \wedge \cdots \wedge P_n(\boldsymbol{s}_n) \rightarrow Q_1(\boldsymbol{t}_1) \vee \cdots \vee Q_m(\boldsymbol{t}_m) \in \mathcal{P}$, where φ is the conjunction of all datalog atoms in the rule, we associate the following binary clauses:

1. $\neg X_{P_i} \vee \neg X_{P_j}$ for all $1 \leq i < j \leq n$;
2. $\neg X_{P_i} \vee X_{Q_j}$ for all $1 \leq i \leq n$ and $1 \leq j \leq m$;
3. $X_{Q_i} \vee X_{Q_j}$ for all $1 \leq i < j \leq m$.

Clauses of the form (1) indicate that at most one body atom in the rule may be marked. By (2), if a body atom is marked, then so must be all head atoms. Finally, (3) ensures that at most one head atom may be unmarked. The resulting set $\mathcal{N}$ of propositional clauses is quadratic in the size of $\mathcal{P}$. Moreover, $\mathcal{N}$ is satisfiable if and only if $\mathcal{P}$ has a marking, and every model I of $\mathcal{N}$ yields a marking $M_I = \{ Q \mid Q$ occurs in $\mathcal{P}$ and $X_Q \in I \}$. Since 2-SAT is solvable in linear time, we obtain the following.

Proposition 3.4. *Markability can be checked in time quadratic in the size of the input program.*

Datalog Rewritability of Markable Programs We now show that markable programs are rewritable into datalog by means of a quadratic translation Ξ_M, which extends the translation for weakly linear programs given in [10].

Consider $\mathcal{P}_1$ and the marking $M = \{B, G, U\}$. We introduce fresh binary predicates $\overline{B}^Y$, $\overline{G}^Y$, and $\overline{U}^Y$ for every disjunctive predicate Y. Intuitively, if a fact $\overline{B}^G(c,d)$ holds in $\Xi_M(\mathcal{P}_1) \cup \mathcal{D}$ then proving $B(c)$ suffices for proving $G(d)$ in $\mathcal{P}_1 \cup \mathcal{D}$ (or, in other words, we have $\mathcal{P}_1 \cup \mathcal{D} \models B(c) \rightarrow G(d)$). Analogously, for the unmarked disjunctive predicate C we introduce fresh binary predicates C^Y for each disjunctive predicate Y; these predicates have a different intuitive interpretation: if a fact $C^U(c,d)$ holds in $\Xi_M(\mathcal{P}_1) \cup \mathcal{D}$ then $\mathcal{P}_1 \cup \mathcal{D}$ entails $C(c) \vee U(d)$. To "initialise" the extension of the fresh predicates we need the following rules for every $X \in M$ and every disjunctive predicate Y.

$$\top(x) \rightarrow \overline{X}^X(x,x) \qquad (7) \qquad\qquad \top(y) \wedge C(x) \rightarrow C^Y(x,y) \qquad (9)$$

$$\overline{X}^Y(x,y) \wedge X(x) \rightarrow Y(y) \qquad (8) \qquad\qquad C^C(x,x) \rightarrow C(x) \qquad (10)$$

These rules encode the intended meaning of the auxiliary predicates. For example, Rule (8) states that if $X(c)$ holds for some constant c and this is sufficient to prove $Y(d)$ for some d, then $Y(d)$ holds. The key step is to "flip" the direction of all rules in $\mathcal{P}_1$ involving the marked predicates B, G and U by moving all marked atoms from the head to the body and vice versa while at the same time replacing their predicates with the relevant auxiliary predicates. Thus, Rule (2) leads to the following rules in $\Xi_M(\mathcal{P}_1)$ for each disjunctive predicate Y:

$$\overline{B}^Y(x,z) \wedge E(x,y) \rightarrow \overline{G}^Y(y,z)$$

These rules are natural consequences of Rule (2) under the intended meaning of the auxiliary predicates: if we can prove a goal $Y(z)$ by proving first $B(x)$, and $E(x,y)$ holds, then by Rule (2) we deduce that proving $G(y)$ suffices to prove $Y(z)$. In contrast to (2), Rule (1) contains no disjunctive body atoms. We "flip" this rule as follows, for each disjunctive predicate Y:

$$C(x) \wedge \overline{B}^Y(x,y) \wedge \overline{G}^Y(x,y) \rightarrow Y(y)$$

Similarly to the previous case, this rule follows from Rule (1): if $C(x)$ holds and we can establish that $Y(y)$ can be proved from $B(x)$ and also from $G(x)$, then $Y(y)$ must hold. In contrast to marked atoms, unmarked atoms are not moved. So, Rules (5) and (6) yield the following rules for each disjunctive predicate Y:

$$V(x) \wedge \overline{U}^Y(x,y) \rightarrow C^Y(x,y) \qquad\qquad C^Y(x,y) \rightarrow \overline{U}^Y(x,y)$$

And indeed, these rules are consequences of Rule (5) and (6), respectively, under the intended meaning of the auxiliary predicates: $V(x)$ and $U(x) \rightarrow Y(y)$ imply $C(x) \vee Y(y)$ by Rule (5), while $C(x) \vee Y(y)$ and $U(x)$ imply $Y(y)$ by Rule (6).

Definition 3.5. *Let $\mathcal{P}$ be a disjunctive program, Σ the set of disjunctive predicates in $\mathcal{P} \setminus \mathcal{P}_\top$, and $M \subseteq \Sigma$ a marking of $\mathcal{P}$. For each $(P,Q) \in \Sigma^2$, let P^Q*

and $\overline{P}^Q$ be fresh predicates of arity $\mathsf{arity}(P) + \mathsf{arity}(Q)$*. Then, $\Xi_M(\mathcal{P})$ is the datalog program with the rules given next, where φ is the conjunction of all datalog atoms in a rule, $\varphi_\top$ is the least conjunction of $\top$-atoms that makes a rule safe, all predicates P_i, Q_j are in Σ, and $\boldsymbol{y}, \boldsymbol{z}$ are disjoint vectors of fresh variables:*

1. *every rule in $\mathcal{P}$ that contains no disjunctive predicates;*
2. *a rule $\varphi_\top \wedge \varphi \wedge \bigwedge_{j=1}^m Q_j^R(\boldsymbol{t}_j, \boldsymbol{y}) \wedge \bigwedge_{i=1}^n \overline{P}_i^R(\boldsymbol{s}_i, \boldsymbol{y}) \rightarrow \overline{Q}^R(\boldsymbol{t}, \boldsymbol{y})$ for every rule $r = \varphi \wedge Q(\boldsymbol{t}) \wedge \bigwedge_{j=1}^m Q_j(\boldsymbol{t}_j) \rightarrow \bigvee_{i=1}^n P_i(\boldsymbol{s}_i) \in \mathcal{P} \setminus \mathcal{P}_\top$ and every $R \in \Sigma$, where $Q(\boldsymbol{t})$ is the unique marked body atom of r;*
3. *a rule $\varphi_\top \wedge \varphi \wedge \bigwedge_{j=1}^m Q_j^R(\boldsymbol{t}_j, \boldsymbol{y}) \wedge \bigwedge_{i=1}^n \overline{P}_i^R(\boldsymbol{s}_i, \boldsymbol{y}) \rightarrow R(\boldsymbol{y})$ for every rule $r = \varphi \wedge \bigwedge_{j=1}^m Q_j(\boldsymbol{t}_j) \rightarrow \bigvee_{i=1}^n P_i(\boldsymbol{s}_i) \in \mathcal{P} \setminus \mathcal{P}_\top$ and each $R \in \Sigma$, where r has no marked body atoms and no unmarked head atoms;*
4. *a rule $\varphi_\top \wedge \varphi \wedge \bigwedge_{j=1}^m Q_j^R(\boldsymbol{t}_j, \boldsymbol{y}) \wedge \bigwedge_{i=1}^n \overline{P}_i^R(\boldsymbol{s}_i, \boldsymbol{y}) \rightarrow P^R(\boldsymbol{s}, \boldsymbol{y})$ for every rule $r = \varphi \wedge \bigwedge_{j=1}^m Q_j(\boldsymbol{t}_j) \rightarrow P(\boldsymbol{s}) \vee \bigvee_{i=1}^n P_i(\boldsymbol{s}_i) \in \mathcal{P} \setminus \mathcal{P}_\top$ and each $R \in \Sigma$, where r has no marked body atoms, and $P(\boldsymbol{s})$ is the unique unmarked head atom;*
5. *a rule $\varphi_\top \rightarrow \overline{R}^R(\boldsymbol{y}, \boldsymbol{y})$ for every $R \in M$;*
6. *a rule $Q(\boldsymbol{z}) \wedge \overline{Q}^R(\boldsymbol{z}, \boldsymbol{y}) \rightarrow R(\boldsymbol{y})$ for every pair $(Q, R) \in M \times \Sigma$;*
7. *a rule $\varphi_\top \wedge Q(\boldsymbol{z}) \rightarrow Q^R(\boldsymbol{z}, \boldsymbol{y})$ for every pair $(Q, R) \in (\Sigma \setminus M) \times \Sigma$;*
8. *a rule $R^R(\boldsymbol{y}, \boldsymbol{y}) \rightarrow R(\boldsymbol{y})$ for every $R \in \Sigma \setminus M$.*

The transformation is quadratic and the arity of predicates is at most doubled. For $\mathcal{P}_1$ and the marking $M = \{B, G, U\}$, we obtain the datalog program $\Xi_M(\mathcal{P}_1)$ consisting of the following rules, where $X \in M$ and Y is disjunctive:

$$C(x) \wedge \overline{B}^Y(x, y) \wedge \overline{G}^Y(x, y) \rightarrow Y(y) \quad (1') \qquad E(y, x) \rightarrow E(x, y) \quad (4)$$

$$\overline{B}^Y(x, z) \wedge E(x, y) \rightarrow \overline{G}^Y(y, z) \quad (2') \qquad \top(x) \rightarrow \overline{X}^X(x, x) \quad (7)$$

$$\overline{G}^Y(x, z) \wedge E(x, y) \rightarrow \overline{B}^Y(y, z) \quad (3') \qquad X(x) \wedge \overline{X}^Y(x, y) \rightarrow Y(y) \quad (8)$$

$$V(x) \wedge \overline{U}^Y(x, y) \rightarrow C^Y(x, y) \quad (5') \qquad \top(y) \wedge C(x) \rightarrow C^Y(x, y) \quad (9)$$

$$C^Y(x, y) \rightarrow \overline{U}^Y(x, y) \quad (6') \qquad C^C(x, x) \rightarrow C(x) \quad (10)$$

In total, this yields 41 rules. Additionally, $\Xi_M(\mathcal{P}_1)$ contains the rules in $\Xi_M(\mathcal{P}_1)_\bot$ and an axiomatisation of $\approx$ (which can be omitted since $\approx$ does not occur in the above rules). Correctness of Ξ_M is established by the following theorem.

Theorem 3.6. *Let $\mathcal{P}$ be a disjunctive program and let M be a marking of $\mathcal{P}$. Then $\Xi_M(\mathcal{P})$ is a polynomial datalog rewriting of $\mathcal{P}$.*

$\Xi_M(\mathcal{P})$ preserves answers to all atomic queries over $\mathcal{P}$. If we only want to query a specific predicate Q, we can compute a smaller program, which is linear in the size of $\mathcal{P}$ and preserves the extension of Q. If Q is datalog, each proof in $\mathcal{P}$ of a fact about Q involves only datalog rules, and if Q is disjunctive each such proof involves only fresh predicates X^Q and $\overline{X}^Q$. Thus, in Ξ_M we can dispense with all rules involving auxiliary predicates X^R or $\overline{X}^R$ for $R \neq Q$ (if Q is datalog the rewriting has no auxiliary predicates).

Theorem 3.7. *Let $\mathcal{P}$ be a program, M a marking of $\mathcal{P}$, S a set of predicates, and $\mathcal{P}'$ obtained from $\Xi_M(\mathcal{P})$ by removing all rules with a predicate X^R or $\overline{X}^R$ for $R \notin S \cup \{\bot\}$. Then $\mathcal{P}'$ is a rewriting of $\mathcal{P}$ w.r.t. all atomic queries over S.*

Rewriting Ontologies Our results are directly applicable to $\mathrm{RL}^{\sqcup}$. In [10], we showed tractability of fact entailment for the class of $\mathrm{RL}^{\sqcup}$ ontologies corresponding to WL programs. The following theorem extends this result to the more general class of markable programs.

Theorem 3.8. *Checking $\mathcal{O} \cup \mathcal{D} \models \alpha$, for $\mathcal{O}$ an $RL^{\sqcup}$ ontology that corresponds to a markable program, is* PTime*-complete w.r.t. data and combined complexity.*

We next lift the markability condition from disjunctive programs to $\mathcal{SHI}$ ontologies. Observe that the notions of dependency graph and markability naturally extend to sets of first-order clauses (written as rules where function symbols are allowed). We define a predicate to be *disjunctive in* $\mathcal{O}$ if it is disjunctive in the set $\mathcal{F}_{\mathcal{O}}$ of clauses obtained by skolemisation; we call $\mathcal{O}$ *markable* if so is $\mathcal{F}_{\mathcal{O}}$; and we call a set of predicates a *marking of* $\mathcal{O}$ if it is a marking of $\mathcal{F}_{\mathcal{O}}$.

Example 3.9. Consider the ontology $\mathcal{O}_1$ and its corresponding clauses $\mathcal{F}_{\mathcal{O}_1}$:

$$\begin{aligned}
\mathcal{O}_1 = \{&\mathsf{Person} \sqsubseteq \mathsf{Man} \sqcup \mathsf{Woman}, \mathsf{Person} \sqsubseteq \exists \mathsf{parent}.\mathsf{Person},\\
&\exists \mathsf{married}.\mathsf{Person} \sqsubseteq \mathsf{Person}, \mathsf{Woman} \sqsubseteq \mathsf{Person}, \mathsf{Man} \sqsubseteq \mathsf{Person}\}\\
\mathcal{F}_{\mathcal{O}_1} = \{&\mathsf{Person}(x) \rightarrow \mathsf{Man}(x) \vee \mathsf{Woman}(x), \mathsf{Person}(x) \rightarrow \mathsf{parent}(x, f(x)),\\
&\mathsf{Person}(x) \rightarrow \mathsf{Person}(f(x)), \mathsf{Person}(y) \wedge \mathsf{married}(x,y) \rightarrow \mathsf{Person}(x),\\
&\mathsf{Woman}(x) \rightarrow \mathsf{Person}(x), \mathsf{Man}(x) \rightarrow \mathsf{Person}(x)\}
\end{aligned}$$

Ontology $\mathcal{O}_1$ is markable since the set $\{\mathsf{Person}, \mathsf{Man}, \mathsf{Woman}\}$ is a marking of $\mathcal{F}_{\mathcal{O}_1}$.

As already mentioned, every normalised $\mathcal{SHI}$ ontology can be rewritten into disjunctive datalog by means of a resolution-based calculus [8, 5]. The following lemma establishes that binary resolution and factoring preserve markability.

Lemma 3.10. *Let M be a marking of a set of clauses $\mathcal{F}$, and let $\mathcal{F}'$ be obtained from $\mathcal{F}$ using binary resolution and factoring. Then M is a marking of $\mathcal{F}'$.*

Thus, markable $\mathcal{SHI}$ ontologies admit a (possibly exponential) rewriting.

Theorem 3.11. *Let $\mathcal{O}$ be a $\mathcal{SHI}$ ontology and let M be a marking of $\mathcal{O}$. Then M is a marking of $\mathsf{DD}(\mathcal{O})$ and $\Xi_M(\mathsf{DD}(\mathcal{O}))$ is a datalog rewriting of $\mathcal{O}$ (where $\mathsf{DD}(\mathcal{O})$ is defined as in [5]).*

Corollary 3.12. *Checking $\mathcal{O} \cup \mathcal{D} \models \alpha$, for $\mathcal{O}$ a markable $\mathcal{SHI}$ ontology is* PTime*-complete w.r.t. data and in* ExpTime *w.r.t. combined complexity.*

Procedure 1 Compile-Horn

```
Input: $\mathcal{S}$: set of clauses
Output: $\mathcal{S}_H$: set of Horn clauses
 1: $\mathcal{S}_H := \{C \in \mathcal{S} \mid C$ is a Horn clause and not a tautology$\}$
 2: $\mathcal{S}_{\overline{H}} := \{C \in \mathcal{S} \mid C$ is a non-Horn clause and not a tautology$\}$
 3: repeat
 4:     $\mathcal{F}$ := factors of each $C_1 \in \mathcal{S}_{\overline{H}}$ non-redundant in $\mathcal{S}_H \cup \mathcal{S}_{\overline{H}}$
 5:     $\mathcal{R}$ := resolvents of each $C_1 \in \mathcal{S}_{\overline{H}}$ and $C_2 \in \mathcal{S}_{\overline{H}} \cup \mathcal{S}_H$ not redundant in $\mathcal{S}_H \cup \mathcal{S}_{\overline{H}}$
 6:     for each $C \in \mathcal{F} \cup \mathcal{R}$ do
 7:         $C'$ := the condensation of $C$
 8:         Delete from $\mathcal{S}_H$ and $\mathcal{S}_{\overline{H}}$ all clauses $\theta$-subsumed by $C'$
 9:         if $C'$ is Horn then $\mathcal{S}_H := \mathcal{S}_H \cup \{C'\}$
10:         else $\mathcal{S}_{\overline{H}} := \mathcal{S}_{\overline{H}} \cup \{C'\}$
11: until $\mathcal{F} \cup \mathcal{R} = \emptyset$
12: return $\mathcal{S}_H$
```

4 Resolution-Based Rewritings

Resolution provides an alternative technique for rewriting disjunctive programs into datalog [5]. Procedure 1 saturates the input program $\mathcal{P}$ under binary resolution and positive factoring, with the restriction that two Horn clauses are never resolved together. The procedure is compatible with redundancy elimination techniques such as tautology elimination, subsumption and condensation. If it terminates, the procedure returns the subset of Horn clauses (equivalently, datalog rules) in the saturation, which is guaranteed to be a rewriting of $\mathcal{P}$.

We show that the separation between disjunctive and datalog predicates (Definition 3.1) can be exploited to refine this procedure. The idea is to further refine resolution by ensuring that the resolved atoms involve a disjunctive predicate.

Definition 4.1. Compile-Horn-Restricted *is obtained from Procedure 1 by adding to the definition of $\mathcal{R}$ in step 5 the additional restriction that the predicate in the atoms being resolved must be disjunctive in $\mathcal{S}$.*

Correctness of Compile-Horn-Restricted relies on the observation that resolutions on datalog predicates can always be delegated to the datalog reasoner and hence do not have to be performed as part of the rewriting process.

Theorem 4.2. *If* Compile-Horn-Restricted *terminates on a disjunctive program $\mathcal{P}$ with a program $\mathcal{P}'$, then $\mathcal{P}'$ is a datalog rewriting of $\mathcal{P}$.*

The class of disjunctive programs over which Compile-Horn-Restricted terminates is incomparable with the class of markable programs. Moreover, the rewritings produced by both approaches are quite different. Markable programs lead to polynomial rewritings, in which the arity of predicates is increased; rewritings computed via resolution can be much larger, but since all the datalog rules in the rewriting are logically entailed by the original program, the arity of predicates stays the same. In Section 6 we will discuss practical implications.

Rewriting Ontologies The procedure Compile-Horn was shown to terminate for a class of programs called *simple* [5]; furthermore, DL-Lite$^{\mathcal{H},+}_{bool}$ ontologies are transformed into disjunctive programs that satisfy the simplicity condition using the algorithm by Hustadt, Motik and Sattler [8]. We now extend this result by devising a sufficient condition for datalog rewritability of $\mathcal{SHI}$ ontologies via Compile-Horn-Restricted. Since transitivity axioms can be eliminated from $\mathcal{SHI}$ ontologies by a polynomial transformation while preserving fact entailment (see [8, 5]), it suffices to formulate our condition for $\mathcal{ALCHI}$.[4] First, we adapt the notion of simple rules in [5] as follows.

Definition 4.3. *An axiom of the form $\exists R.A \sqsubseteq B$ is* simple *w.r.t. a set of predicates S (or S-*simple*) if $A \notin S$. An ontology $\mathcal{O}$ is S-*simple *if so is every axiom of the form $\exists R.A \sqsubseteq B$ in $\mathcal{O}$.*

Note that ontology $\mathcal{O}_1$ from Example 3.9 is not simple w.r.t. its disjunctive predicates due to axiom $\exists \mathsf{married}.\mathsf{Person} \sqsubseteq \mathsf{Person}$. If, however, we replace this axiom with $\mathsf{Man} \sqcap \mathsf{Woman} \to \bot$, we obtain a simple ontology, which in turn is no longer markable. The following theorem then generalises the result in [5] to a sufficient condition for datalog rewritability of $\mathcal{ALCHI}$ ontologies.

Theorem 4.4. *Let $\mathcal{O}$ be an $\mathcal{ALCHI}$ ontology that is simple w.r.t. its disjunctive predicates. Then* Compile-Horn-Restricted *terminates on* $\mathsf{DD}(\mathcal{O})$ *with a datalog rewriting of $\mathcal{O}$.*

5 Conjunctive Queries

By the results in [12], disjunctive programs cannot be rewritten to datalog in a query-independent way while preserving answers to CQs. Nonetheless, rewriting techniques for atomic queries can still be used to answer specific queries, which can be appended to the program as additional rules.

Rewriting CQs using markability This observation immediately suggests how our markability condition in Section 3 can be applied to rewriting CQs.

Proposition 5.1. *Let $\mathcal{P}$ be a disjunctive program, let M be a marking of $\mathcal{P}$, and let q be a CQ with at most one atom marked by M. Then, $\Xi_M(\mathcal{P} \cup \{q\})$ is a rewriting of q w.r.t. $\mathcal{P}$.*

Indeed, M constitutes a marking of $\mathcal{P} \cup \{q\}$ if and only if q contains at most one body atom marked by M. From this, we obtain the following result, which applies equally to disjunctive programs and RL$^{\sqcup}$ ontologies.

[4] Note that neither Compile-Horn nor Compile-Horn-Restricted are well-suited for dealing with (axiomatised) equality. Both will diverge on every disjunctive program with equality due to the congruence axioms $P(x) \wedge x \approx y \to P(y)$ with P disjunctive.

Proposition 5.2. *Let $\mathcal{F}$ be a disjunctive program (or an RL$^{\sqcup}$ ontology), let $\boldsymbol{M}$ be the set of all (minimal) markings of $\mathcal{F}$, and let q be a (Boolean) CQ. If there is some $M \in \boldsymbol{M}$ that marks at most one atom of q, then answering the (fixed) q w.r.t. (fixed) $\mathcal{F}$ and an arbitrary dataset is a tractable problem.*

Example 5.3. Consider the following RL$^{\sqcup}$ ontology $\mathcal{O}$ and query q:[5]

$$\begin{aligned} \mathcal{O} &= \{A \sqsubseteq B \sqcup C\} \\ q &= R(x,y) \wedge R(y,z_1) \wedge R(y,z_2) \wedge B(z_1) \wedge C(z_2) \rightarrow A_q(x) \end{aligned}$$

The empty ontology is a rewriting of $\mathcal{O}$, which can be determined using markability or resolution. Indeed, for every dataset $\mathcal{D}$ and fact α we have $\mathcal{O} \cup \mathcal{D} \models \alpha$ iff $\mathcal{D} \models \alpha$. The empty ontology, however, is not a rewriting of q, as witnessed by the following dataset $\mathcal{D}$, for which $\mathcal{O} \cup \mathcal{D} \cup \{q\} \models A_q(a)$ but $\mathcal{D} \cup \{q\} \not\models A_q(a)$:

$$\{R(a,b_1), R(a,b_2), R(b_1,c_1), R(b_1,c_2), R(b_2,c_2), R(b_2,c_3), B(c_1), A(c_2), C(c_3)\}$$

Clearly, $M = \{B\}$ is a marking of $\mathcal{O}$, and q contains one marked atom. Then $\mathcal{P} = \Xi_M(\mathcal{O} \cup \{q\})$ has the following rules, with $X \in \{B, A_q\}$ and $Y \in \{B, C, A_q\}$:

$$A(x) \wedge \overline{B}^{Y}(x,y) \rightarrow C^{Y}(x,y) \tag{11}$$

$$\overline{A}_q^{Y}(x,u) \wedge R(x,y) \wedge R(y,z_1) \wedge R(y,z_2) \wedge C^{Y}(z_2,u) \rightarrow \overline{B}^{Y}(z_1,u) \tag{12}$$

$$\top(x) \rightarrow \overline{X}^{X}(x,x) \tag{13}$$

$$X(x) \wedge \overline{X}^{Y}(x,y) \rightarrow Y(y) \tag{14}$$

$$\top(y) \wedge C(x) \rightarrow C^{Y}(x,y) \tag{15}$$

$$C^{C}(x,x) \rightarrow C(x) \tag{16}$$

Figure 2 shows a derivation of $A_q(a)$ from $\mathcal{P} \cup \mathcal{D}$.

Although this approach is immediately applicable to disjunctive programs and hence to RL$^{\sqcup}$ ontologies, it only transfers to $\mathcal{SHI}(\mathcal{Q})$ ontologies if q corresponds to a normalised $\mathcal{SHI}(\mathcal{Q})$ axiom. The reduction in [8, 5] from $\mathcal{SHI}(Q)$ to disjunctive datalog is only complete for inputs equivalent to $\mathcal{SHIQ}$ ontologies.

Rewriting CQs via resolution The resolution-based approach naturally extends to a class of CQs satisfying certain conditions closely related to simplicity.

Definition 5.4. *Let S be a set of unary and binary predicates. A CQ q is S-*simple *if for some variable x in q all of the following conditions are satisfied:*

1. *if q is not Boolean, then $A_q(x)$ is the head atom of q;*
2. *Every S-atom (i.e., atom whose predicate is in S) in q is of the form $B(x)$, $R(x,x)$, $S(x,y)$, or $T(y,x)$; and*

[5] This example is based on a personal communication with Carsten Lutz.

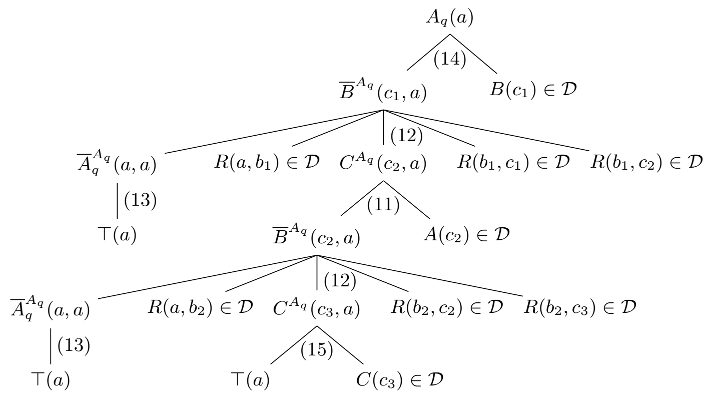
$A_q(a)$
(14)
$\overline{B}^{A_q}(c_1, a)$
$B(c_1) \in \mathcal{D}$
(12)
$\overline{A}_q^{A_q}(a, a)$
$R(a, b_1) \in \mathcal{D}$
$C^{A_q}(c_2, a)$
$R(b_1, c_1) \in \mathcal{D}$
$R(b_1, c_2) \in \mathcal{D}$
(13)
$\top(a)$
(11)
$\overline{B}^{A_q}(c_2, a)$
$A(c_2) \in \mathcal{D}$
(12)
$\overline{A}_q^{A_q}(a, a)$
$R(a, b_2) \in \mathcal{D}$
$C^{A_q}(c_3, a)$
$R(b_2, c_2) \in \mathcal{D}$
$R(b_2, c_3) \in \mathcal{D}$
(13)
$\top(a)$
(15)
$\top(a)$
$C(c_3) \in \mathcal{D}$

Fig. 2. Derivation of $A_q(a)$ from $\Xi_M(\mathcal{O} \cup \{q\}) \cup \mathcal{D}$ in Example 5.3

3. every variable $y \neq x$ occurs in at most one S-atom in q.

Example 5.5. Consider the following RL$^\sqcup$ ontology $\mathcal{O}$ and queries q_1, q_2:

$$\mathcal{O} = \{\mathsf{Person} \sqsubseteq \mathsf{Man} \sqcup \mathsf{Woman}, \exists \mathsf{married}.\mathsf{Person} \sqsubseteq \mathsf{Person}\}$$
$$q_1 = \mathsf{Man}(x) \wedge \mathsf{married}(x, y) \rightarrow A_{q_1}(x)$$
$$q_2 = \mathsf{Man}(x) \wedge \mathsf{married}(x, y) \wedge \mathsf{Woman}(y) \rightarrow A_{q_2}(x)$$

Ontology $\mathcal{O}$ is simple w.r.t. the set $S = \{\mathsf{Man}, \mathsf{Woman}\}$ of the disjunctive predicates in $\mathcal{O}$. Query q_1 is S-simple while q_2 is not. It is straightforward to verify that Compile-Horn-Restricted terminates on $\mathcal{O} \cup \{q_1\}$ but not on $\mathcal{O} \cup \{q_2\}$.

Theorem 5.6. *Let $\mathcal{O}$ be an RL$^\sqcup$ ontology that is simple w.r.t. the set S of the disjunctive predicates in $\mathcal{O}$. Then Procedure* Compile-Horn-Restricted *terminates on $\mathcal{O} \cup \{q\}$ with a datalog rewriting of q w.r.t. $\mathcal{O}$ for every S-simple CQ q.*

Consequently, answering any (fixed) CQ q over any (fixed) ontology $\mathcal{O}$ satisfying the conditions of Theorem 5.6 is a tractable problem.

6 Evaluation

Rewritability Experiments We have evaluated whether realistic ontologies can be rewritten into datalog using our approaches. We analysed 118 ontologies that use disjunctive constructs from BioPortal, the Protégé library, and the corpus in [7]. To transform ontologies into disjunctive datalog we used KAON2 [14], which succeeded to compute disjunctive programs for 103 ontologies.[6] Out of the

[6] We doctored the ontologies to remove constructs outside $\mathcal{SHIQ}$. The modified ontologies can be found on https://krr-nas.cs.ox.ac.uk/RR2014/ontologies.tar.bz2

	Linearity			Resolution		HermiT		Pellet	
	dlog	disj	err	all	err	all	err	all	err
U01	<1s	12s	1	<1s	1	47s	0	147s	5
U04	<1s	87s	1	1s	1	57s	1	—	—
U07	<1s	168s	2	2s	1	122s	1	—	—
U10	<1s	53s	5	3s	1	196s	1	—	—

Table 2. Average times for answering UOBM's 15 standard queries

103 disjunctive programs, 32 were WL, and 35 were markable. Furthermore, 26 programs could be rewritten using Compile-Horn, and 27 could be rewritten using Compile-Horn-Restricted.[7] In both cases, the average time for computing a rewriting was below 1s (where the average is taken over the successful runs). Despite the potentially exponential blowup, the increase in program size was modest in practice: 49% for Compile-Horn and 34% for Compile-Horn-Restricted on average w.r.t. the number of rules.

Many of the programs obtained by KAON2 contained equality, and hence could not be rewritten by means of resolution (see Section 4). Hence, we additionally considered simplified versions of the 103 programs where we removed all rules containing equality. Out of these, 33 turned out to be WL, and 36 were markable; as expected the effect of equality on linearity-based approaches is rather minor. In contrast, resolution-based approaches were significantly more effective than before: Compile-Horn succeeded in 39 cases, and Compile-Horn-Restricted in 41 cases. Again, computing a successful rewriting took less that 1s on average in both cases. The increase in program size was even smaller than before: 16% for Compile-Horn and 6% for Compile-Horn-Restricted on average.

Both Compile-Horn and Compile-Horn-Restricted succeeded on some ontologies that were not simple w.r.t. disjunctive predicates. At the same time, being worst-case exponential, both algorithms failed to rewrite (within 1h) one simple ontology and two that were simple w.r.t. disjunctive predicates.

The intersection between the programs rewritable using markability and resolution turned out to be quite large: in the general case, there were 16 programs that could be rewritten by only one approach, and in the equality-free case only 5. Still, taken together, the two approaches succeeded to rewrite 39 programs (38%) in the general case and 41 programs (40%) in the equality-free case. Moreover, on average, 73% of the predicates were datalog, and so could be queried using a datalog engine even if the disjunctive program was not rewritable. Finally, we found that 20 out of the 103 ontologies were $\mathrm{RL}^{\sqcup}$, out of which 17 were markable. Of the remaining three, two could be rewritten via resolution.

CQ Answering We have also tested scalability of CQ answering over the UOBM benchmark [13]. We considered the $\mathrm{RL}^{\sqcup}$ subset of UOBM without equal-

[7] We ran the rewritability experiments on a laptop with a 2.5GHz Intel Core i5 processor and 8GB RAM, and set a timeout of 1h per ontology.

ity,[8] and generated datasets for 1 to 10 universities (denoted as U01-U10). Furthermore, we considered the 15 standard queries in the benchmark. While not markable, our test ontology can be converted to a markable (in fact, WL) program by the algorithm in [10]. Moreover, it is rewritable using Compile-Horn-Restricted (but not using Compile-Horn). We used RDFox as a datalog engine, and measured performance against HermiT [17] and Pellet [19]. We used a server with two Intel Xeon E5-2643 processors and 128GB RAM. Systems were compared on individual queries with a timeout of 10min per query. We ran RDFox on 16 threads. Table 2 shows average query answering times, and number of queries on which a system failed.[9] The time spent on computing the rewritings[10] is not included into the query answering times since query rewriting can be done in a data-independent way.

Pellet could only answer queries on U01. It timed out on 5 queries, and was much slower on the remaining queries than the other systems.

Using the linearity-based approach we could answer queries for all datasets. From the 15 test queries, 7 were disjunctive (i.e., contained at least one disjunctive atom), and 8 were datalog. One disjunctive query could not be rewritten. Datalog queries were answered instantaneously (<1s) for all datasets. Disjunctive queries were much harder, and performance on those was comparable to HermiT. Memory-outs were encountered for U07 (1 query) and U10 (4 queries). For all rewritable queries, computing the rewriting (including the conversion to a WL program) took less that 1s.

The resolution-based approach was clearly superior to the others. Only one query could not be rewritten, and all the remaining queries could be answered almost instantaneously even for the largest dataset (query rewriting itself took 26s on average). In contrast to the linearity-based approach, rewritings obtained by resolution introduce no predicates of higher arity, and thus lead to smaller materialisations. Also, there was no significant difference in query answering times for datalog and disjunctive queries. Once again, the increase in program size was modest (4.6 times on average). Notably, computing the rewritings took 26s on average—considerably longer than with the linearity-based approach.

7 Conclusion

We have proposed enhanced techniques for rewriting disjunctive datalog programs and DL ontologies into plain datalog programs. Our techniques enable the use of scalable datalog engines for data reasoning, and our experiments suggest practical feasibility of our approach. In the near future, we are planning to extend our results for CQ answering to capture larger classes of queries.

Acknowledgements This work was supported by the Royal Society, the EPSRC projects Score!, ExODA, and MaSI^3, and the FP7 project OPTIQUE.

[8] Equality makes the resolution-based approach non-applicable.
[9] Average times do not reflect queries on which a system failed.
[10] Query rewriting was performed with a 1h timeout per query.

References

1. Artale, A., Calvanese, D., Kontchakov, R., Zakharyaschev, M.: The DL-Lite family and relations. J. Artif. Intell. Res. 36, 1–69 (2009)
2. Bachmair, L., Ganzinger, H.: Resolution theorem proving. In: Handbook of Automated Reasoning, pp. 19–99 (2001)
3. Bienvenu, M., ten Cate, B., Lutz, C., Wolter, F.: Ontology-based data access: A study through disjunctive datalog, CSP, and MMSNP. In: PODS. pp. 213–224 (2013), arXiv:1301.6479
4. Bishop, B., Kiryakov, A., Ognyanoff, D., Peikov, I., Tashev, Z., Velkov, R.: OWLim: A family of scalable semantic repositories. Semantic Web J. 2(1), 33–42 (2011)
5. Cuenca Grau, B., Motik, B., Stoilos, G., Horrocks, I.: Computing datalog rewritings beyond Horn ontologies. In: IJCAI (2013), arXiv:1304.1402
6. Eiter, T., Ortiz, M., Šimkus, M., Tran, T.K., Xiao, G.: Query rewriting for Horn-SHIQ plus rules. In: AAAI. pp. 726–733 (2012)
7. Gardiner, T., Tsarkov, D., Horrocks, I.: Framework for an automated comparison of description logic reasoners. In: ISWC. pp. 654–667 (2006)
8. Hustadt, U., Motik, B., Sattler, U.: Reasoning in Description Logics by a Reduction to Disjunctive Datalog. J. Autom. Reasoning 39(3), 351–384 (2007)
9. Kaminski, M., Grau, B.C.: Sufficient conditions for first-order and datalog rewritability in elu. In: Description Logics. pp. 271–293 (2013)
10. Kaminski, M., Nenov, Y., Cuenca Grau, B.: Datalog rewritability of disjunctive datalog programs and its applications to ontology reasoning. In: AAAI (2014)
11. Krisnadhi, A., Lutz, C.: Data complexity in the $\mathcal{EL}$ family of description logics. In: LPAR (2007)
12. Lutz, C., Wolter, F.: Non-uniform data complexity of query answering in description logics. In: KR (2012)
13. Ma, L., Yang, Y., Qiu, Z., Xie, G.T., Pan, Y., Liu, S.: Towards a complete OWL ontology benchmark. In: ESWC. pp. 125–139 (2006)
14. Motik, B.: Reasoning in Description Logics using Resolution and Deductive Databases. Ph.D. thesis, Univesität Karlsruhe (TH), Karlsruhe, Germany (2006)
15. Motik, B., Cuenca Grau, B., Horrocks, I., Wu, Z., Fokoue, A., Lutz, C.: OWL 2 Web Ontology Language Profiles. W3C Recommendation (2009)
16. Motik, B., Nenov, Y., Piro, R., Horrocks, I, Olteanu, D.: Parallel materialisation of datalog programs in centralised, main-memory rdf systems. In: AAAI (2014)
17. Motik, B., Shearer, R., Horrocks, I.: Hypertableau Reasoning for Description Logics. J. Artif. Intell. Res. 36, 165–228 (2009)
18. Pérez-Urbina, H., Motik, B., Horrocks, I.: Tractable query answering and rewriting under description logic constraints. J. Appl. Log. 8(2), 186–209 (2010)
19. Sirin, E., Parsia, B., Cuenca Grau, B., Kalyanpur, A., Katz, Y.: Pellet: A practical OWL-DL reasoner. J. Web Sem. 5(2), 51–53 (2007)
20. Trivela, D., Stoilos, G., Chortaras, A., Stamou, G.B.: Optimising resolution-based rewriting algorithms for dl ontologies. In: DL. pp. 464–476 (2013)
21. Wu, Z., Eadon, G., Das, S., Chong, E.I., Kolovski, V., Annamalai, M., Srinivasan, J.: Implementing an inference engine for RDFS/OWL constructs and user-defined rules in Oracle. In: ICDE. pp. 1239–1248 (2008)