DReW: a Reasoner for Datalog-rewritable Description Logics and DL-Programs*

Guohui Xiao, Stijn Heymans, and Thomas Eiter

Institute of Information Systems 184/3
Vienna University of Technology
Favoritenstraße 9–11, A–1040 Vienna, Austria
{xiao,heymans,eiter}@kr.tuwien.ac.at

Abstract. Nonmonotonic dl-programs provide a loose integration of Description Logic (DL) ontologies and Logic Programming (LP) rules with negation, where a rule engine can query an ontology with a native DL reasoner. However, even for tractable dl-programs, the overhead of an external DL reasoner might be considerable. Datalog-*rewritable* DL ontologies, such as $\mathcal{LDL}^+$, can be rewritten to Datalog programs, such that dl-programs can be reduced to Datalog$^\neg$, i.e, Datalog with negation, under well-founded semantics. We developed the reasoner DReW that uses the Datalog-rewriting technique. DReW can as such answer conjunctive queries over $\mathcal{LDL}^+$ ontologies, as well as reason on dl-programs over $\mathcal{LDL}^+$ ontologies under well-founded semantics. The preliminary but encouraging experimental results show that DReW can efficiently handle large knowledge bases.

1 Introduction

As the envisioned basis of future information systems, the Semantic Web is a fertile ground for deploying AI techniques, and in turn raises new research problems in AI. As a prominent example, the combination of rules with Description Logics (DLs), which is central to the Semantic Web architecture, has received high attention over the past years, with approaches like *Description Logic Programs* [10], *DL-safe rules* [21], *r-hybrid KBs* [24], $\mathcal{DL}$+*log* [25], *MKNF KBs* [20], *Description Logic Rules and ELP* [15, 16], and *dl-programs* [5].

Nonmonotonic dl-programs provide a loose integration of Description Logic (DL) ontologies and Logic Programming (LP) rules with negation, where a rule engine can query an ontology using a native DL reasoner. For dl-programs over tractable DL ontologies under well-founded semantics, the reasoning problem is tractable [6]. However, even for tractable dl-programs, the overhead of an external DL reasoner might be considerable.

In [13], Datalog-rewritability was proposed to remedy the overload of calling external DL reasoners. A Datalog-rewritable ontology can be polynomially rewritten to a Datalog program. Moreover, dl-programs over such Datalog-rewritable ontologies

* This work is partially supported by the Austrian Science Fund (FWF) projects P20305 and P20840, and by the EC FP7 project OntoRule (IST-2009-231875).

can then be reduced to Datalog$^\neg$— Datalog with negation — programs. A particular Datalog-rewritable DL, called $\mathcal{EDL}^+$, was also proposed in [13]. Reasoning in $\mathcal{EDL}^+$ is tractable, under both data and combined complexity. Despite its low complexity, $\mathcal{EDL}^+$ is still expressive enough in ontology applications sucha s role equvalences and transitive roles.

Based on the concept of Datalog-rewriting, we developed a new reasoner DReW (Datalog ReWriter)[1], which rewrites $\mathcal{EDL}^+$ontologies (dl-programs over $\mathcal{EDL}^+$ ontologies) to Datalog (Datalog$^\neg$) programs, and calls an underlying rule-based reasoner, currently DLV, to perform the actual reasoning. For $\mathcal{EDL}^+$ ontologies, DReW does instance checking as well as answering of conjunctive queries (CQs). For dl-programs over $\mathcal{EDL}^+$ ontologies, DReW computes the well-founded model.

The evaluation of the DReW reasoner goes along two axes: as a pure DL reasoner and as a reasoner for dl-programs. Furthermore, we show that several real-word ontologies fall to a large extent in the $\mathcal{EDL}^+$ fragment. We compare CQs over the LUBM [11] benchmark with Pellet, KAON2 and RacerPro. For dl-programs, we compare DReW with DLVHEX over LUBM ontologies with dl-rules. The preliminary but encouraging experimental results show that DReW can efficiently handle large knowledge bases.

The remainder of the paper is organized as follows: in Section 2, we recall Datalog, Datalog$^\neg$, Description Logics, and dl-programs under well-founded semantics. In Section 3, we show how Datalog-rewritability can be used to reason on DL ontologies and dl-programs, and we present our new reasoner DReW. Section 4 describes the evaluation of DReW. We compare our work with Horn-$\mathcal{SROIQ}$ [23], ELP [15, 16], and KAON2 [22] in Section 5. Finally, we conclude our work in Section 6.

2 Preliminaries

2.1 Datalog and Datalog$^\neg$

Constants, variables, terms, and atoms are defined as usual. We assume that a binary inequality predicate $\neq$ is available; atoms not using $\neq$ are *normal*. A Datalog$^\neg$ rule r has the form

$$h \leftarrow b_1, \ldots, b_k, not\ c_1, \ldots, not\ c_m \tag{1}$$

where the *body* $b_1, \ldots, b_k, c_1, \ldots, c_m$ are atoms and the *head* h is a normal atom. We call $B^-(r) = \{c_1, \ldots, c_m\}$ the *negative body* of r. If $B^-(r) = \emptyset$, then r is a Datalog rule. A finite set of Datalog$^\neg$ (Datalog) rules is a Datalog$^\neg$ (Datalog) *program*. *Ground* terms, atoms, and programs are defined as usual. A *fact* is a ground rule (1) with $k = m = 0$.

The *Herbrand Domain* $\mathcal{H}_P$ of a program P is the set of constants from P. The *Herbrand Base* $\mathcal{B}_P$ of P is the set of normal ground atoms with predicates and constants from P. An *interpretation* of P is any set $I \subseteq \mathcal{B}_P$. For a ground normal atom a, we write $I \models a$ if $a \in I$; for a ground atom $c_1 \neq c_2$, we write $I \models c_1 \neq c_2$ if c_1 and c_2 are different; for a ground *negation as failure atom* $l = not\ a$, we write $I \models l$ if $I \not\models a$. For a set of ground (negation as failure) atoms α, $I \models \alpha$ if $I \models l$ for all $l \in \alpha$. A ground rule $r : h \leftarrow \alpha$ is *satisfied* w.r.t. I, denoted $I \models r$, if $I \models h$ whenever $I \models \alpha$.

[1] http://www.kr.tuwien.ac.at/research/systems/drew

An interpretation I of a ground program P is a *model* of P, if $I \models r$ for every $r \in P$; in addition, I is *minimal*, if P has no model $J \subset I$. For a non-ground P, I is a (minimal) model of P iff it is a (minimal) model of $gr(P)$, the *grounding* of P with the constants of P defined as usual. Each Datalog program P has some minimal model, which in fact is unique; we denote it with $MM(P)$. We write $P \models a$ if $MM(P) \models a$.

We recall the *well-founded semantics* [8] for Datalog$^\neg$. Let I be an interpretation for a Datalog$^\neg$ program P. The *GL-reduct* [9] P^I of a program P is the set of Datalog rules $h \leftarrow b_1, \ldots, b_k$ such that $r : h \leftarrow b_1, \ldots, b_k, not\ c_1, \ldots, not\ c_m \in gr(P)$ and $I \not\models c_i$, for all $i, 1 \leq i \leq m$.

Using the γ operator [3], one can define the well-founded semantics as follows. Let $\gamma_P(I) = MM(P^I)$ and $\gamma_P^2(I) = \gamma_P(\gamma_P(I))$, i.e., applying the γ operator twice; as γ_P is anti-monotone, γ_P^2 is monotone. The set of *well-founded* atoms of P, denoted $WFS(P)$, is exactly the least fixed point of γ_P^2. We denote with $P \models^{wf} a$ that $a \in WFS(P)$.

For a Datalog (Datalog$^\neg$) program P and an atom a, deciding $P \models a$ ($P \models^{wf} a$) is *data complete* (P is fixed except for facts) for PTIME and *(combined) complete* (P is arbitrary) for EXPTIME [4].

2.2 Description Logics

For space constraints, we assume the reader is familiar with DLs and adopt the usual conventions, see [2]. We highlight some points below.

A *DL knowledge base (KB)* $\Sigma = \langle \mathcal{T}, \mathcal{A} \rangle$ consists of a finite set $\mathcal{T}$ (called *TBox*) of *terminological* and *role axioms* $\alpha \sqsubseteq \beta$, where α and β are concept (respectively role) expressions, and a finite set $\mathcal{A}$ (called *ABox*) of assertions $A(o_1)$ and $R(o_1, o_2)$ where A is a concept name, R is a role name, and o_1, o_2 are individuals (i.e., constants). We also view Σ as the set $\mathcal{T} \cup \mathcal{A}$.

For particular classes of DL KBs Σ, we assume that (1) Σ is defined over a (finite) set $\mathcal{P}_o$ of concept and role names; we call the constants appearing in Σ the *Herbrand domain of* Σ, denoted with $\Delta_{\mathcal{H}(\Sigma)}$; (2) Σ can be extended with arbitrary assertions, i.e., for any ABox $\mathcal{A}'$ (over $\mathcal{P}_o$), $\Sigma \cup \mathcal{A}'$ is an admissible DL KB, and (3) Σ defines a ground entailment relation $\models$ such that $\Sigma \models Q(\mathbf{e})$ is defined for *dl-queries* $Q(\mathbf{e})$, $\mathbf{e}$ ground terms, which indicates that all models of Σ satisfy $Q(\mathbf{e})$. Here, a *dl-query* $Q(\mathbf{t})$ is either of the form (a) $C(t)$, where C is a concept and t is a term; or (b) $R(t_1, t_2)$, where R is a role and t_1, t_2 are terms.

The relation $\Sigma \models Q(\mathbf{e})$ is defined relative to the models of Σ, which are the interpretations $\mathcal{I} = (\Delta^\mathcal{I}, \cdot^\mathcal{I})$ that satisfy all axioms and assertions of Σ, where $\Delta^\mathcal{I} \neq \emptyset$ is the domain and $\cdot^\mathcal{I}$ is an interpretation function for concept- and role names as well as individuals. Note that we do not allow for subsumption checking in this paper. We are mainly interested in query answering.

We will assume that the *unique names assumption* (UNA) holds in interpretations $\mathcal{I}$, i.e., $o_1^\mathcal{I} \neq o_2^\mathcal{I}$ for distinct o_1 and o_2, and moreover for simplicity that $o^\mathcal{I} = o$ for individuals o (in particular $\{o\}^\mathcal{I} = \{o^\mathcal{I}\}$ for *nominals*) appearing in the KB. UNA is required due to the translation to datalog later where we have UNA.

Example 1. Take the DL KB Σ:

$$\begin{aligned}(\geq 2\ PapToRev.\top) &\sqsubseteq Over\\ Over &\sqsubseteq \forall Super^{+}.Over\\ \{(a,b)\}\sqcup\{(b,c)\} &\sqsubseteq Super\end{aligned}$$

where $Super^{+}$ is the transitive closure of the role $Super$. The first two axioms indicate that someone who has more than two papers to review is overloaded, and that an overloaded person causes all the supervised persons to be overloaded as well (otherwise the manager delegates badly). The final axiom — equivalent to the assertions $Super(a,b)$ and $Super(b,c)$ — defines the supervision hierarchy.

The particular Description Logic that the reasoner DReW is able to handle is $\mathcal{LDL}^{+}$, as introduced in [13].

$\mathcal{LDL}^{+}$ is designed by syntactic restrictions on the expressions that occur in axioms, distinguishing between occurrence in the "body" α and the "head" β of an axiom $\alpha \sqsubseteq \beta$. We define

- *b-roles* (b for *body*) E, F to be *role names* P, *role inverses* E^{-}, *role conjunctions* $E \sqcap F$, *role disjunctions* $E \sqcup F$, *role sequences* $E \circ F$, *transitive closures* E^{+}, *role nominals* $\{(o_1, o_2)\}$, and *role top* $\top^2$, where o_1, o_2 are individuals, and $\top^2$ is the universal role;
- *h-roles* (h for *head*) E, F to be *role names* P, *role inverses* E^{-}, *role conjunctions* $E \sqcap F$, and *role top* $\top^2$.

Furthermore, let *basic concepts* C, D be concept names A, the top symbol $\top$, and *conjunctions* $C \sqcap D$; then we define

- *b-concepts* C, D as *concept names* A, *conjunctions* $C \sqcap D$, *disjunctions* $C \sqcup D$, *exists restrictions* $\exists E.C$, *atleast restrictions* $\geq n\, E.C$, *nominals* $\{o\}$, and the *top* symbol $\top$, where E is a b-role as above, and o is an individual.
- *h-concepts* (h for *head*) as *basic concepts* B or *value restrictions* $\forall E.B$ where B is a basic concept and E a b-role.

Now an $\mathcal{LDL}^{+}$ *KB* is a pair $\Sigma = \langle \mathcal{T}, \mathcal{A} \rangle$ of a finite TBox $\mathcal{T}$ and a finite ABox $\mathcal{A}$, where

- $\mathcal{T}$ is a set of *terminological axioms* $B \sqsubseteq H$, where B is a b-concept and H is an h-concept, and *role axioms* $S \sqsubseteq T$, where S is a b-role and T is an h-role, and
- $\mathcal{A}$ is a set of assertions of the form $C(o)$ and $E(o_1, o_2)$ where C is an h-concept and E an h-role.

Example 2. Reconsider the DL KB Σ from Example 1. It is easily checked that Σ amounts to an $\mathcal{LDL}^{+}$ KB.

As established in [13], one can check entailment of an atom w.r.t an $\mathcal{LDL}^{+}$ KB by calculating a minimal model w.r.t. to the Herbrand domain of the KB (the individuals appearing in the KB) and checking membership of the atom in that minimal model. This core result allows us to reduce entailment of atoms w.r.t. an $\mathcal{LDL}^{+}$ KB to checking entailment w.r.t. a Datalog program.

2.3 DL-Programs under Well-Founded Semantics

We introduce dl-programs under well-founded semantics (WFS) [6].

Informally, a dl-program consists of a DL KB Σ over $\mathcal{P}_o$ and a $\mathsf{Datalog}^\neg$ program P over a set of predicates $\mathcal{P}_p$ distinct from $\mathcal{P}_o$, which may contain queries to Σ. Roughly, such queries ask whether a certain ground atom logically follows from Σ. Note that the Herbrand domains of Σ and P are not necessarily distinct.

Syntax. A *dl-atom* $a(\mathbf{t})$ has the form

$$\mathrm{DL}[S_1 \uplus p_1, \ldots, S_m \uplus p_m; Q](\mathbf{t}) \;\; m \geq 0, \tag{2}$$

where each S_i is either a concept or a role name from $\mathcal{P}_o$, p_i is a unary, resp. binary, predicate symbol from $\mathcal{P}_p$, and $Q(\mathbf{t})$ is a dl-query. We call the list $S_1 \uplus p_1, \ldots, S_m \uplus p_m$ the *input signature* and $p_1, \ldots, p_m$ the *input predicate symbols*. Intuitively, $\uplus$ increases S_i by the extension of p_i prior to the evaluation of query $Q(\mathbf{t})$.[2]

A *dl-rule* r has the form (1), where any atom b_i, c_j may be a dl-atom. A *dl-program* $\mathcal{KB} = (\Sigma, P)$ consists of a DL KB Σ and a finite set of dl-rules P — $\mathcal{KB}$ *is a dl-program over* $\mathcal{DL}$, if Σ is a $\mathcal{DL}$ KB.

Semantics. We define the *Herbrand base* $\mathcal{B}_{\mathcal{KB}}$ of a dl-program $\mathcal{KB} = (\Sigma, P)$ as the set of ground atoms with predicate symbols from P (i.e., from $\mathcal{P}_p$) and constants from the Herbrand domains of Σ and P. An *interpretation* of $\mathcal{KB}$ is any subset $I \subseteq \mathcal{B}_{\mathcal{KB}}$. It satisfies a ground atom a under Σ, denoted $I \models_\Sigma a$,

– in case a is a non-dl-atom, iff $I \models a$, and
– in case a is a dl-atom of form (2), iff $\Sigma \cup \tau^I(a) \models Q(\mathbf{c})$,

where $\tau^I(a)$, the *extension of* a *under* I, is $\tau^I(a) = \bigcup_{i=1}^m A_i(I)$ with $A_i(I) = \{S_i(\mathbf{e}) \mid p_i(\mathbf{e}) \in I\}$. Satisfaction of ground dl-rules r under Σ is then as usual (see $\mathsf{Datalog}^\neg$) and denoted with $I \models_\Sigma r$. I is a model of $\mathcal{KB}$, denoted $I \models \mathcal{KB}$, iff $I \models_\Sigma r$ for all $r \in gr(P)$.

We define the well-founded semantics for dl-programs as in [6] using the γ^2 operator. For I and $\mathcal{KB} = (\Sigma, P)$, let $\mathcal{KB}^I = (\Sigma, sP^I_\Sigma)$, the *reduct of* $\mathcal{KB}$ *wrt.* I, be the dl-program where sP^I_Σ results from $gr(P)$ by deleting (1) every dl-rule r where $I \models_\Sigma a$ for some $a \in B^-(r)$, and (2) from the remaining dl-rules r the negative body $B^-(r)$. Note that sP^I_Σ may still contain positive dl-atoms. As shown in [6], $\mathcal{KB}^I$ has a single minimal model, denoted $MM(\mathcal{KB}^I)$.

Now the operator $\gamma_{\mathcal{KB}}$ on interpretations I of $\mathcal{KB}$ is defined by $\gamma_{\mathcal{KB}}(I) = MM(\mathcal{KB}^I)$. As $\gamma_{\mathcal{KB}}$ is anti-monotone, $\gamma^2_{\mathcal{KB}}(I) = \gamma_{\mathcal{KB}}(\gamma_{\mathcal{KB}}(I))$ is monotone and has a least fixpoint. This fixpoint is the set of *well-founded* atoms of $\mathcal{KB}$, denoted $WFS(\mathcal{KB})$; we denote with $\mathcal{KB} \models^{wf} a$ that $a \in WFS(\mathcal{KB})$.

[2] Modifiers that were included in the original dl-program, $\cup, \cap$, may be expressed by $\uplus$ in strong enough DLs and similarly for subsumption expressions $C \sqsubseteq D$. However, Datalog-rewritability precludes such constructs.

Example 3. Take $\mathcal{KB} = (\Sigma, P)$ where Σ as in Example 1 and P:

$$\begin{array}{lll} r_1: & good(X) & \leftarrow \mathrm{DL}[;\, Super](X,\, Y), \\ & & \quad not\ \mathrm{DL}[PapToRev \uplus paper;\, Over](Y); \\ r_2: & over(X) & \leftarrow not\ good(X); \\ r_3: & paper(b, p_1) & \leftarrow\ ; \\ r_4: & paper(b, p_2) & \leftarrow\ . \end{array}$$

Note that the first dl-atom has no input signature. Intuitively, r_1 indicates that if X is supervising Y and Y is not overloaded, then X is a good manager and r_2 indicates that if X is a not a good manager then X is overloaded. Then, $KB \models^{wf} over(a)$.

Deciding $(\Sigma, P) \models^{wf} a$ is combined complete for ExpTime (PTime$^{\mathrm{NExp}}$) for Σ in $\mathcal{SHIF}(\mathbf{D})$ ($\mathcal{SHOIN}(\mathbf{D})$) and data complete for PTime$^{\mathrm{NP}}$ for Σ in $\mathcal{SHIF}(\mathbf{D})$ and $\mathcal{SHOIN}(\mathbf{D})$ [6]; here data complete means that only the constants in Σ and P, the ABox $\mathcal{A}$, and the facts in P may vary.

3 DReW: Reasoning with Description Logics and DL-Programs using Datalog

As seen in the previous section, one can check entailment of an atom w.r.t. an $\mathcal{LDL}^+$ KB by calculating a minimal model w.r.t. to the Herbrand domain of the KB [13]. Thus, if we manage to write a Datalog program that has a minimal model corresponding to that minimal model of the KB, we obtain a procedure to check entailment of an atom w.r.t. an $\mathcal{LDL}^+$ using standard Datalog engines.

This is exactly the approach DReW takes for conjunctive query answering w.r.t. $\mathcal{LDL}^+$ knowledge bases, and by extension, for reasoning with DL-Programs that have as an underlying DL $\mathcal{LDL}^+$.

We illustrate the approach by means of Example 3. Take the DL part Σ of the particular dl-program. How to rewrite Σ to a Datalog program P such that CQs against Σ can be solved by posing the query to P?

Take $(\geq 2\ PapToRev.\top) \sqsubseteq Over$ from Σ. This corresponds trivially to a rule

$$Over(X) \leftarrow (\geq 2\ PapToRev.\top)(X)$$

where $(\geq 2\ PapToRev.\top)$ is treated as a predicate. In order to make the DL semantics of this number restriction explicit in the program, we add its definition:

$$\begin{array}{rl} (\geq 2\ PapToRev.\top)(X) \leftarrow & PapToRev(X,\, Y_1), PapToRev(X,\, Y_2), \\ & \top(Y_1), \top(Y_2),\, Y_1 \neq Y_2 \end{array}$$

where again $\top$ is treated as a predicate. Note that if there are two different domain elements y_1 and y_2 such that $PapToRev(x, y_1)$ and $PapToRev(x, y_2)$ are true in the model of the program, then the literal $(\geq 2\ PapToRev.\top)(x)$ will be true as well. Vice versa, due to minimality of models of a program, the latter literal will only be true in a minimal model, if there are such two successors y_1 and y_2.

For the axiom $Over \sqsubseteq \forall Super^+.Over$, we have a rule

$$Over(Y) \leftarrow Super^+(X, Y), Over(X)$$

The transitive closure of $Super$ is defined by the traditional Datalog recursive rules to define transitive closure of a predicate:

$$\begin{aligned} Super^+(X, Y) &\leftarrow Super(X, Y) \\ Super^+(X, Y) &\leftarrow Super(X, Z), Super^+(Z, Y) \end{aligned}$$

Note that in contrast with the Horn fragment of Description Logic Programs [10], the translation uses recursive rules and thus full capabilities of Datalog reasoning.

The axiom $\{(a, b)\} \sqcup \{(b, c)\} \sqsubseteq Super$ results in rules

$$\begin{aligned} Super(X, Y) &\leftarrow \{(a, b)\}(X, Y) \\ Super(X, Y) &\leftarrow \{(b, c)\}(X, Y) \end{aligned}$$

Together with the rules for nominals,

$$\begin{aligned} \{(a, b)\}(a, b) &\leftarrow \\ \{(b, c)\}(b, c) &\leftarrow \end{aligned}$$

this ensures that both (a, b) and (b, c) are in the extension of $Super$ as intended by the axiom. Finally, we have rules to ensure the proper handling of inverses,

$$\begin{aligned} Super(X, Y) &\leftarrow Super^-(Y, X) \\ PapToRev(X, Y) &\leftarrow PapToRev^-(Y, X) \end{aligned}$$

and the rules that introduce the Herbrand domain of the DL KB via $\top$ and $\top^2$ predicates:

$$\begin{aligned} \top(a) &\leftarrow \\ \top(b) &\leftarrow \\ \top(c) &\leftarrow \\ \top^2(X, Y) &\leftarrow \top(X), \top(Y) \end{aligned}$$

The formal translation of any $\mathcal{LDL}^+$ knowledge base Σ to a Datalog program $\Phi_{\mathcal{LDL}^+}(\Sigma)$ can be found in [13]. In particular, we can reduce ground entailment of a ground atom w.r.t. $\mathcal{LDL}^+$ knowledge bases to checking whether the atom is present in the minimal model of the translation in Datalog [13, Proposition 10].

We can use this translation of $\mathcal{LDL}^+$ knowledge bases to Datalog to translate reasoning with dl-programs over $\mathcal{LDL}^+$ under well-founded semantics to Datalog$^\neg$ under well-founded semantics. In other words, for a dl-program $\mathcal{KB} = (\Sigma, P)$ where Σ is an $\mathcal{LDL}^+$ knowledge base and P is a Datalog$^\neg$ program, there is a Datalog$^\neg$ program $\Psi(\mathcal{KB})$ that is equivalent w.r.t. checking ground entailment.

Take the dl-program $\mathcal{KB} = (\Sigma, P)$ where $\Sigma = \{\ C \sqsubseteq D\ \}$ and

$$P \triangleq \{\ p(a) \leftarrow;\quad s(a) \leftarrow;\quad s(b) \leftarrow;$$
$$q \leftarrow DL[C \uplus s; D](a), not\ DL[C \uplus p; D](b)\ \}.$$

Intuitively, the dl-atom $DL[C \uplus s; D](a)$ extends C in Σ with the extension of s (i.e., with a and b) and queries Σ then for $D(a)$ which is indeed entailed from Σ once $C(a)$ holds; similarly $DL[C \uplus p; D](b)$ extends C with the extension of p (i.e., a), and queries Σ for $D(b)$ which is not entailed from $C \sqsubseteq D$ if only $C(a)$ is assumed to hold in Σ.

Note that each dl-atom sends up a different input/hypothesis to Σ and that entailments for each different input might be different. To this purpose, we copy Σ to new disjoint equivalent versions for each dl-atom, i.e., for each distinct dl-atom λ, we define a new knowledge base Σ_λ that results from replacing all concept and role names by a λ-subscripted version. Thus, for the set $\Lambda_P = \{\lambda_1 \triangleq C \uplus s, \lambda_2 \triangleq C \uplus p\}$ of dl-atoms, we have $\Sigma_{\lambda_1} = \{\ C_{\lambda_1} \sqsubseteq D_{\lambda_1}\ \}$ and $\Sigma_{\lambda_2} = \{\ C_{\lambda_2} \sqsubseteq D_{\lambda_2}\ \}$.

We translate these disjoint knowledge bases to a Datalog program using $\Phi_{\mathcal{LDL}^+}$ as described above, resulting in the rules $D_{\lambda_1}(X) \leftarrow C_{\lambda_1}(X)$ and $D_{\lambda_2}(X) \leftarrow C_{\lambda_2}(X)$.

The inputs in the dl-atoms Λ_P can then be encoded as rules $\rho(\Lambda_P)$:

$$C_{\lambda_1}(X) \ \leftarrow\ s(X)$$
$$C_{\lambda_2}(X) \ \leftarrow\ p(X)$$

Remains to replace the original dl-rules with rules not containing dl-atoms: P^{ord} results from replacing each dl-atom $DL[\lambda; Q](\mathbf{t})$ in P with a new atom $Q_\lambda(\mathbf{t})$, such that P^{ord} is the Datalog$^\neg$ program

$$P^{ord} \triangleq \{\ p(a) \leftarrow;\quad s(a) \leftarrow;\quad s(b) \leftarrow;$$
$$q \leftarrow D_{\lambda_1}(a), not\ D_{\lambda_2}(b) \qquad \}.$$

One sees that indeed $\mathcal{KB} \models q$ and $\Phi_{\mathcal{LDL}^+}(\Sigma_{\lambda_1}) \cup \Phi_{\mathcal{LDL}^+}(\Sigma_{\lambda_2}) \cup P^{ord} \cup \rho(\Lambda_P) \models q$, effectively reducing reasoning w.r.t. the dl-program to a Datalog$^\neg$ program.

Such a translation of the loose coupling via dl-programs is not limited to the use of $\mathcal{LDL}^+$. In [13], we showed that such a translation works for so-called Datalog-rewriteable DLs, roughly, DLs for which entailment can be reduced to Datalog reasoning. An example of such DLs are the (individual-free) Horn-DL fragments of OWL 1 and 2 [23].

Figure 1 shows a schematic overview of the component of DReW responsible for reducing entailment from DLs to Datalog. The extension for dl-programs is a straightforward elaboration of this. Taking as input a conjunctive query and an ontology in OWL 2 syntax extended for the complex role expressions of $\mathcal{LDL}^+$, DReW checks whether the ontology is in the $\mathcal{LDL}^+$ fragment. If it is, we translate the ontology according to the format suitable for the specified Datalog reasoner (DLV in our case).

DReW is written in Java using an extension of the OWL API 3.0.0[3] for parsing $\mathcal{LDL}^+$ ontologies. The underlying Datalog engine we used is the latest version of DLV

[3] http://owlapi.sourceforge.net/

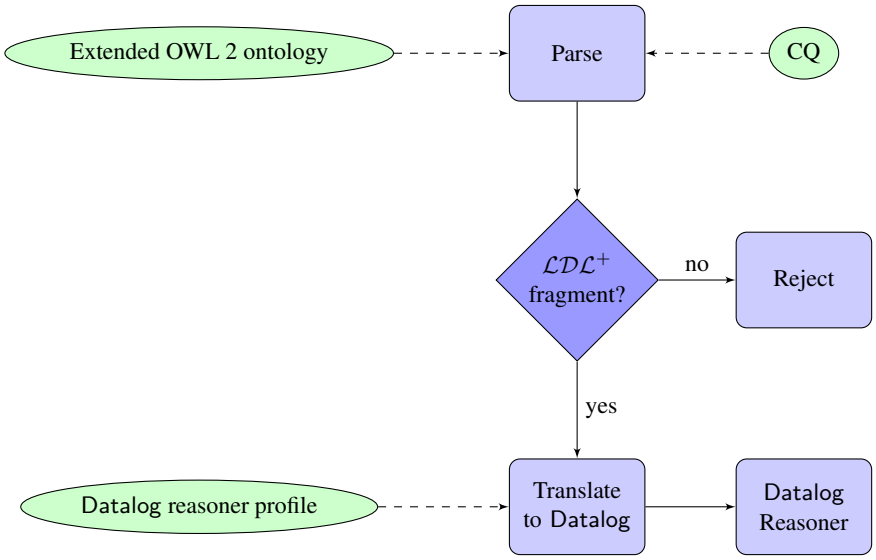

Fig. 1. DReW Control Flow — DL Component

(*dl-magic-snapshot-2009-11-26*) [4] which supports magic sets and well-founded semantics.

4 Evaluation

In this section, we evaluate the DReW reasoner. We do so along two axes: as a pure Description Logic reasoner and as a reasoner for dl-programs.

All experiments were performed on a laptop running Ubuntu 10.04 with a 1.83G CPU and 2G of memory; the memory of the Java Virtual Machine was set to 1G.

4.1 Reasoning with Description Logics

We first analyze to what extent common ontologies fall in the $\mathcal{LDL}^+$ fragment. Next, we analyze the performance of conjunctive query answering with DReW compared to standard DL reasoners on those ontologies.

Expressiveness of $\mathcal{LDL}^+$ To assess the expressiveness of $\mathcal{LDL}^+$, we select several ontologies and show that they fall to a large extent in the $\mathcal{LDL}^+$profile. We picked the ontologies that are used in Motik's thesis for testing the DL reasoner KAON2 [22]; they can be downloaded from the KAON 2 site[5].

The results of this experiment are listed in Table 1. Note that for Galen over 40% of the axioms are not in the $\mathcal{LDL}^+$ profile, but that for Dolce and Wine over 80% of

[4] http://www.dbai.tuwien.ac.at/proj/dlv/magic/

[5] http://kaon2.semanticweb.org/download/test_ontologies.zip

Ontology	Axioms	Inds	Concepts	Object Props	$\mathcal{DL}^+$?	Violated Axioms	Violated %
Galen	4,356	0	2,747	261	no	1,881	0.43
Dolce	1,185	2	125	251	no	162	0.14
Wine	773	162	142	13	no	137	0.18
Vicodi	53,876	16,942	194	10	yes	0	0
Semintec	65,459	17,941	60	16	no	113	$1.73 \cdot 10^{-3}$
LUBM	8,612	1,555	43	25	no	8	$9.29 \cdot 10^{-4}$

Table 1. $\mathcal{DL}^+$profile checking

axioms are in $\mathcal{DL}^+$. Only Vicodi is fully in $\mathcal{DL}^+$and over 99% of Semintec and LUBM axioms are in $\mathcal{DL}^+$. Most of the violations are due to existential quantifiers occurring on the right side of axioms.

Conjunctive Query Evaluation To evaluate the performance of CQs over ontologies using DReW, we compare it with 3 state-of-the-art DL reasoners: KAON2, RacerPro, and Pellet. We did not consider other DL reasoners, such as HermiT or Fact++ as they cannot handle CQs; we did not consider REQUIEM and QuOnto as they can not handle $\mathcal{DL}^+$ ontologies.

Reasoning in KAON2[6] [22] is implemented using novel algorithms that reduce a $\mathcal{SHIQ}(\mathbf{D})$ knowledge base to a disjunctive Datalog program based on resolution techniques. Pellet[7] [26] fully supports OWL 2[19]. In contrast with KAON2, it is a reasoner based on tableaux algorithms. RacerPro[8] [12] is a tableaux-based reasoner as well and implements the Description Logic $\mathcal{SHIQ}$. All 3 reasoners support conjunctive query answering.

We specifically tested CQs on The Lehigh University Benchmark (LUBM) [11]. LUBM is developed to facilitate the evaluation of Semantic Web repositories in a standard and systematic way. The benchmark is intended to evaluate the performance of those repositories with respect to extensional queries over a large data set that commits to a single realistic ontology. It consists of a university domain ontology, customizable and repeatable synthetic data, a set of test queries, and several performance metrics. The queries we evaluated are as in
http://swat.cse.lehigh.edu/projects/lubm/query.htm, referring to numbers 1-14.

As we indicated in Table 1, LUBM is not fully in $\mathcal{DL}^+$: there are 8 violated axioms, e.g.,

$$Person \sqcap \exists headOf.Department \equiv Chair \ .$$

For our experiments and to have an $\mathcal{DL}^+$ conformant fragment of LUBM, we replace such equivalence axioms by subsumption axioms, e.g., by

$$Person \sqcap \exists headOf.Department \sqsubseteq Chair \ .$$

In general, such a conversion changes the semantics of the ontology. However, in our considered test of the benchmark queries, the query results are exactly the same as

[6] http://kaon2.semanticweb.org/
[7] http://clarkparsia.com/pellet/
[8] http://www.racer-systems.com

Query	DReW	KAON2	RacerPro	Pellet
1	3.13	2.84	3.78	4.55
2	2.23	2.39	4.24	4.54
3	2.29	2.35	3.68	4.54
4	2.25	2.61	26.05	4.63
5	2.29	2.60	5.12	4.52
6	2.24	2.56	5.05	4.51
7	2.21	2.63	3.39	4.44
8	2.28	2.65	27.13	4.62
9	2.22	2.67	4.80	4.54
10	2.22	2.42	3.85	4.53
11	2.23	2.31	4.39	4.49
12	2.27	2.55	4.08	4.63
13	2.31	2.58	4.44	4.42
14	2.26	2.35	5.30	4.52

Table 2. Conjunctive Queries on LUBM (in secs.)

Ontology	Inds	DReW	KAON2	RacerPro	Pellet
LUBM0	904	1.61	2.27	4.51	3.53
LUBM1	1,555	2.27	2.54	7.52	4.53
LUBM2	2,753	5.07	3.72	9.38	7.57

Table 3. Conjunctive Queries on LUBM with Different Number of Individuals

on the original LUBM. It is part of future research to investigate how DReW can deal with partial $\mathcal{LDL}^+$ ontologies in answering queries as faithfully as possible.

The results of evaluating the 14 CQs on LUBM are shown in Table 2. From the table, we see that DReW outperforms RacerPro and Pellet in all the queries and that it is slightly better than KAON2 for most of the queries. Note the out-of-the-normal times for RacerPro on query 4 and query 8; we assume they are caused by the use of data properties.

As DReW and KAON2 have evaluation times close to each other, we also evaluate CQs on LUBM ontologies with a different numbers of individuals. The result is summarized in Table 3.

LUBM1 is the original LUBM ontology. By removing and adding individuals, we get LUBM0 and LUBM2. The number under each reasoner is the average time for answering the 14 queries. In all the LUBMs, DReW is better than RacerPro and Pellet. However, compared with KAON2, we also see that DReW is not so good at dealing with large number of individuals. We assume that the reason is the use of DLV as the underlying Datalog engine. Since there is no public API for DLV, we have to use it as a standalone process. When the translated ontology is big, the communication of processes costs significant time.

Query	DReW	DLVHEX+DL-Plugin	dl-atoms	Factor
0	2.81	4.31	1	1.53
1	2.63	3.04	1	1.16
2	2.60	3.88	1	1.49
3	2.59	4.04	1	1.56
4	2.75	3.51	1	1.27
5	3.00	5.10	1	1.70
6	4.69	19.59	6	4.17
7	3.20	8.38	2	2.62

Table 4. Reasoning on dl-programs

4.2 Reasoning with DL-Programs

DReW is designed for reasoning over dl-programs under well-founded semantics. The only reasoner available for comparison is DLVHEX [9] [7]. DLVHEX is a prototype implementation for computing the stable models of so-called HEX-programs – an extension of dl-programs for reasoning with external sources (not necessarily DL knowledge bases) under the answer set semantics. By using the *Description Logic Plugin* [10], which interfaces to OWL ontologies via a Description Logic reasoner (currently RacerPro), DLVHEX can reason on dl-programs under the answer set semantics.

Note that for Datalog programs (i.e., without negation), the well-founded semantics coincides with the answer set semantics. We thus evaluate both reasoners on LUBM, which is negation free. We manually generate serveral dl-program over LUBM to evaluate reasoning over Datalog-rewriteable ontologies.

All the test results are shown in Table 4. We see that DReW outperforms DLVHEX for all the tests. As the number of dl-atoms increases, the advantage of DReW becomes more clear, confirming our hypothesis that translating dl-programs to Datalog programs reduces the overload of calling external DL reasoners as is the case in DLVHEX.

5 Related Work

Horn Fragments of Description Logics Horn-$\mathcal{SHOIQ}$ and Horn-$\mathcal{SROIQ}$ are Horn fragments of OWL 1 and OWL 2 [23] respectively. Reasoning in Horn-$\mathcal{SHOIQ}$ is EXPTIME-complete, and reasoning in Horn-$\mathcal{SROIQ}$ is 2-EXPTIME-complete. Despite their high expressiveness, both Horn-$\mathcal{SHOIQ}$ and Horn-$\mathcal{SROIQ}$ have polynomial data complexity and as shown in [23] can be translated to Datalog.

However, $\mathcal{SROIQ}$ ($\mathcal{SHOIQ}$) is not Datalog-rewritable (in the sense of [13]) as the modularity property does not hold in general. In particular, $\Phi(\langle\emptyset, \mathcal{A}\rangle) \neq \mathcal{A}$. We do have that it is Datalog-rewritable if we restrict to individual-free knowledge bases. In essence, this means that we can use DReW to reason on dl-programs over Horn-$\mathcal{SHOIQ}$ and Horn-$\mathcal{SROIQ}$ KBs provided the latter do not contain individuals.

[9] http://www.kr.tuwien.ac.at/research/systems/dlvhex

[10] http://www.kr.tuwien.ac.at/research/systems/dlvhex/dlplugin.html

ELP ELP [15, 16] is a decidable fragment of the Semantic Web Rule Language (SWRL) that admits reasoning in polynomial time. ELP is based on the tractable DL $\mathcal{EL}^{++}$ and encompasses an extended notion of the DL rules[15]. Also ELP extends $\mathcal{EL}^{++}$ with a number of features introduced by OWL 2, such as disjoint roles, local reflexivity, certain range restrictions, and the universal role. A reasoning algorithm is based on a translation of ELP to Datalog in a in a tractable fashion.

There are several differences between ELP and dl-programs over $\mathcal{DL}^{+}$: (1) ELP is a tightly-coupled combination of ontologies and rules, while dl-programs are loosely coupled; (2) in the rule part of a dl-program one can use default negation well-founded semantics, which can not be expressed in ELP. Indeed, ELPs have a first-order semantics compared to the minimal model semantics of dl-programs.

KAON2 KAON2 does not implement the tableaux calculus. Rather, reasoning in KAON2 is implemented by novel algorithms which reduce a $\mathcal{SHIQ}$ knowledge base to a disjunctive Datalog program of exponential size. The translation is not modular in the above-mentioned sense if the ABox is non-empty. However, with an empty ABox, also the KAON2 rewriting can be used in the context of our dl-programs to Datalog$^{\neg}$ reduction.

Hybrid MKNF KBs under WFS Well-founded Semantics is also used in other combinations. In [14, 1], WFS for the tightly-coupled Hybrid MKNF KBs was proposed. When the underlying DL of MKNF KBs is tractable, the data complexity of MKNF under WFS is in PTIME.

6 Conclusions and Outlook

We presented the class of Datalog-rewritable DLs and showed that reasoning with dl-programs over such DLs can be reduced to Datalog$^{\neg}$ under well-founded semantics. This reduction avoids the overhead that is normally associated with the calling of a native DL reasoner. The $\mathcal{DL}^{+}$ DL is such a particular Datalog-rewritable DL. We developed a new reasoner, DReW, which can efficiently reason over $\mathcal{DL}^{+}$ DL ontologies and dl-programs over $\mathcal{DL}^{+}$ ontologies.

We plan to extend $\mathcal{DL}^{+}$ with more DL constructors as in [17], while keeping the Datalog-rewritability. For example, disjoint classes in OWL 2 Profiles [18] can be added. Furthermore, currently, we only use DLV as the underlying rule-based reasoner. We plan to experiment with different rule engines, e.g., XSB. Finally, Datalog-rewritable DLs are a natural candidate for tightly-coupling approaches as well.

References

1. J. J. Alferes, M. Knorr, and T. Swift. Queries to hybrid mknf knowledge bases through oracular tabling. In *International Semantic Web Conference*, LNCS 5823:1-16, 2009.
2. F. Baader, D. Calvanese, D. L. McGuinness, D. Nardi, and P. F. Patel-Schneider, editors. *The Description Logic Handbook*. CUP, 2003.
3. C. Baral and V. S. Subrahmanian. Dualities between alternative semantics for logic programming and nonmonotonic reasoning. *JAR*, 10(3):399–420, 1993.

4. E. Dantsin, T. Eiter, G. Gottlob, and A. Voronkov. Complexity and expressive power of logic programming. *ACM Computing Surveys*, 33(3):374–425, 2001.
5. T. Eiter, G. Ianni, T. Lukasiewicz, R. Schindlauer, and H. Tompits. Combining answer set programming with description logics for the Semantic Web. *Artificial Intelligence*, 172(12-13):1495–1539, 2008.
6. T. Eiter, T. Lukasiewicz, R. Schindlauer, and H. Tompits. Well-founded semantics for description logic programs in the Semantic Web. In *Proc. RuleML*, pages 81–97, 2004. Full paper *ACM TOCL*, (to appear).
7. T. Eiter, G. Ianni, R. Schindlauer, and H. Tompits. Effective integration of declarative rules with external evaluations for semantic-web reasoning. In *ESWC*, LNCS 4011:273-287, 2006.
8. A. V. Gelder, K. Ross, and J. S. Schlipf. The well-founded semantics for general logic programs. *JACM*, 38(3):620–650, 1991.
9. M. Gelfond and V. Lifschitz. The stable model semantics for logic programming. In *Proc. ICLP*, pages 1070–1080. The MIT Press, 1988.
10. B. N. Grosof, I. Horrocks, R. Volz, and S. Decker. Description logic programs: Combining logic programs with description logic. In *Proc. WWW 2003*, pages 48–57. ACM.
11. Y. Guo, Z. Pan, and J. Heflin. An evaluation of knowledge base systems for large owl datasets. Technical report, CSE Department, Lehigh University, 2004.
12. V. Haarslev and R. Möller. Racer system description. In *IJCAR*, LNCS 2083:701-706, 2001.
13. S. Heymans, T. Eiter, and G. Xiao. Tractable reasoning with DL-programs over datalog-rewritable description logics. In *roc. of 19th European Conference on Artificial Intelligence (ECAI)*. IOS Press, 2010.
14. M. Knorr, J. J. Alferes, and P. Hitzler. A coherent well-founded model for hybrid mknf knowledge bases. In *ECAI*, volume 178 of *Frontiers in Artificial Intelligence and Applications*, pages 99–103. IOS Press, 2008.
15. M. Krötzsch, S. Rudolph, and P. Hitzler. Description logic rules. In *Proc. ECAI*, pages 80–84. IOS Press, 2008.
16. M. Krötzsch, S. Rudolph, and P. Hitzler. ELP: Tractable rules for OWL 2. In *Proc. ISWC 2008*, pages 649–664.
17. M. Krötzsch and S. Rudolph. A matter of principles: Towards the largest dlp possible. In *Description Logics*, volume 477 of *CEUR Workshop Proceedings*, 2009.
18. B. Motik, B. C. Grau, I. Horrocks, Z. Wu, A. Fokoue, and C. Lutz, editors. *OWL 2 Web Ontology Profiles*. 2008. W3C Rec. 27 Oct. 2009.
19. B. Motik, P. F. Patel-Schneider, and B. Parsia, editors. *OWL 2 Web Ontology Language: Structural Specification and Functional-Style Syntax*. 2008. W3C Working Draft April 2009.
20. B. Motik and R. Rosati. A faithful integration of description logics with logic programming. In *Proc. IJCAI*, pages 477–482, 2007.
21. B. Motik, U. Sattler, and R. Studer. Query answering for OWL-DL with rules. *Journal of Web Semantics*, 3(1):41–60, July 2005.
22. B. Motik. *Reasoning in Description Logics using Resolution and Deductive Databases*. PhD thesis, University of Karlsruhe, Karlsruhe, Germany, January 2006.
23. M. Ortiz, S. Rudolph, and M. Simkus. Worst-case optimal reasoning for the horn-dl fragments of owl 1 and 2. In *KR*, pages 269–279. AAAI Press, May 2010.
24. R. Rosati. On the decidability and complexity of integrating ontologies and rules. *Journal of Web Semantics*, 3(1):41–60, 2005.
25. R. Rosati. DL+log: Tight integration of description logics and disjunctive datalog. In *Proc. KR*, pages 68–78, 2006.
26. E. Sirin, B. Parsia, B. C. Grau, A. Kalyanpur, and Y. Katz. Pellet: A practical OWL-DL reasoner. *J. Web Sem.*, 5(2):51–53, 2007.