ECAI 2010
H. Coelho et al. (Eds.)
IOS Press, 2010

doi:10.3233/978-1-60750-606-5-35

Tractable Reasoning with DL-Programs over Datalog-rewritable Description Logics

Stijn Heymans and **Thomas Eiter** and **Guohui Xiao**[1]

Abstract. The deployment of KR formalisms to the Web has created the need for formalisms that combine heterogeneous knowledge bases. 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. To remedy this, we consider Datalog-*rewritable* DL ontologies, i.e., ones that 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. To illustrate this framework, we consider several Datalog-rewritable DLs. Besides fragments of the tractable OWL 2 Profiles, we also present $\mathcal{LDL}^+$ as an interesting DL that is tractable while it has some expressive constructs. Our results enable the usage of DBLP technology to reason efficiently with dl-programs in presence of negation and recursion, as a basis for advanced applications.

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* [12], *DL-safe rules* [20], *r-hybrid KBs* [21], $\mathcal{DL}$+*log* [22], *MKNF KBs* [19], *Description Logic Rules and ELP* [15, 16], and *dl-programs* [8].

In particular, dl-programs support a loosely-coupled integration of rules and ontologies, and provide an expressive combination framework based on the interaction of rules with a DL knowledge base (KB) via so-called *dl-atoms*. Such dl-atoms query the DL KB by checking for entailment of ground atoms or axioms w.r.t. the KB; as knowledge deduced by the rules can be streamed up to the DL KB in turn, a bi-directional flow of information is possible.

The *answer set semantics* of dl-programs in [8], based on [11], is highly expressive, but on the other hand already intractable on the rule side; hence, towards scalable reasoning with negation, [9] presents a *well-founded semantics* for dl-programs, based on [10]. Given that the queries in dl-atoms are tractable, such programs can be evaluated in polynomial time (as usual, under data complexity).

Tractability of queries in dl-atoms is in line with recent tractable DLs such as the $DL\text{-}Lite$ families [6], $\mathcal{EL}^{++}$ [1, 2], and *Description Logic Programs* [12], that strive for scalability. In fact, they gave rise to three families of languages that resulted in the OWL 2 Profiles of the emerging Web Ontology Language OWL 2 [18].

However, even when loosely coupling such a tractable DL with rules via dl-programs under well-founded semantics, one still needs a dedicated algorithm that uses native DL reasoners to perform the external queries, thus causing a significant overhead. This paper tries to overcome this, by identifying a class of Description Logics, so-called Datalog-*rewritable DLs*, for which reasoning with dl-programs can be reduced to pure Logic Programming, i.e., to Datalog$^\neg$ (Datalog with negation under well-founded semantics). This class is defined non-constructively: a transformation of DL KBs to Datalog programs must exist, such that ground entailment from a DL KB carries over to the Datalog program. Besides this non-constructive class, we do present several (syntactically defined) DLs which have this property, including the novel DL $\mathcal{LDL}^+$.

The main contributions of this paper are as follows.

- We define a class of Datalog-rewritable DLs (Section 3), and show how reasoning with dl-programs over such DLs under well-founded semantics can be reduced to Datalog$^\neg$ by means of an efficient transformation. Noticeably, for dl-programs without negation, the result is a standard Datalog program; moreover, the transformation preserves stratified negation.
- We introduce $\mathcal{LDL}^+$ as a particular Datalog-rewritable DL (Section 4). This DL has no negation (hence the $+$) and distinguishes between expressions on the left- and right-hand side of axioms. $\mathcal{LDL}^+$ offers expressive concept- and role expressions on the left-hand side of axioms (hence the $\mathcal{L}$ in $\mathcal{LDL}^+$), e.g., qualified number restrictions and transitive closure of roles. The Datalog-rewritability of $\mathcal{LDL}^+$ (Section 5) is interesting in itself, showing how to do reasoning in DLs with expressive constructs efficiently via Logic Programming. As a side result, we obtain that reasoning in $\mathcal{LDL}^+$ is tractable, considering both data and combined complexity; more precisely, we show that it is PTIME-complete in both settings. Despite its low complexity, $\mathcal{LDL}^+$ is still expressive enough to represent many constructs useful in ontology applications [2] such as role equivalences and transitive roles.
- We review the different OWL 2 Profiles and relate them to $\mathcal{LDL}^+$ (Section 6). While $\mathcal{LDL}^+$ misses some constructs, e.g., the *exists restriction* on axiom right-hand sides as in $\mathcal{EL}^{++}$ and $DL\text{-}Lite$, or negation as in the $DL\text{-}Lite$ families, it adds others, e.g., expressive role constructs and *transitive closure* (which is not expressible in first-order logic). Furthermore, we show that $\mathcal{LDL}^+$ encompasses Description Logic Programs without a complexity increase.

Our results enable the use of mature LP technology, e.g., systems like XSB or Datalog engines like DLV, and emerging implementations of recursive SQL, to reason efficiently with dl-programs involving recursion and negation, as a basis for advanced applications.

[1] Knowledge Based Systems Group, Institute of Information Systems, Vienna University of Technology, Favoritenstraße 9-11, A-1040 Austria, email: {heymans,eiter,xiao}@kr.tuwien.ac.at. This work has been partially supported by the Austrian Science Fund (FWF) projects P20305 and P20840, and by the EC project OntoRule (IST-2009-231875).

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 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 $I \models \alpha$.

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* [10] for Datalog$^\neg$. Let I be an interpretation for a Datalog$^\neg$ program P. The *GL-reduct* [11] 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 [5], 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 [7].

2.2 Description Logics

For space constraints, we assume the reader is familiar with DLs and adopt the usual conventions, see [3]. 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.

We will assume that the *unique names assumption* 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.

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.*

2.3 DL-Programs under Well-Founded Semantics

We introduce dl-programs under well-founded semantics [9].

Informally, a dl-program consists of a DL KB Σ over $\mathcal{P}_o$ and a 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 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 [9] using the γ^2 operator. For I and $\mathcal{KB} = (\Sigma, P)$, let $\mathcal{KB}^I = (\Sigma, sP_\Sigma^I)$,

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

the *reduct of $\mathcal{KB}$ wrt. I*, be the dl-program where sP_Σ^I results from $gr(P)$ by deleting (1) every dl-rule r where $I \not\models_\Sigma a$ for some $a \in B^-(r)$, and (2) from the remaining dl-rules r the negative body $B^-(r)$. Note that sP_Σ^I may still contain positive dl-atoms. As shown in [9], $\mathcal{KB}^I$ 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 $a \in WFS(\mathcal{KB})$.

Example 2. *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 ($\text{PTIME}^{\text{NEXP}}$) for Σ in $\mathcal{SHIF}(\mathbf{D})$ ($\mathcal{SHOIN}(\mathbf{D})$) and data complete for PTIME^{NP} for Σ in $\mathcal{SHIF}(\mathbf{D})$ and $\mathcal{SHOIN}(\mathbf{D})$ [9]; here data complete means that only the constants in Σ and P, the ABox $\mathcal{A}$, and the facts in P may vary.

3 Reducing DL-Programs to Datalog$^\neg$

Let $\mathcal{KB} = (\Sigma, P)$ be a dl-program and let a be a ground atom from $\mathcal{B}_{\mathcal{KB}}$. We define a class of DLs, so-called Datalog-*rewritable* DLs, such that reasoning w.r.t. dl-programs over such DLs becomes reducible to Datalog$^\neg$. In particular, we show that for such Datalog-rewritable DLs, we can reduce a dl-program $\mathcal{KB} = (\Sigma, P)$ to a Datalog$^\neg$ program $\Psi(\mathcal{KB})$ such that $\mathcal{KB} \models^{wf} a$ iff $\Psi(\mathcal{KB}) \models^{wf} a$.

We abstractly define which DLs we consider Datalog-rewritable.

Definition 1. *A DL $\mathcal{DL}$ is* Datalog-rewritable *if there exists a transformation $\Phi_{\mathcal{DL}}$ from $\mathcal{DL}$ KBs to* Datalog *programs such that, for any $\mathcal{DL}$ KB Σ,*

(i) $\Sigma \models Q(\mathbf{o})$ iff $\Phi_{\mathcal{DL}}(\Sigma) \models Q(\mathbf{o})$ for any concept or role name Q from Σ, and individuals $\mathbf{o}$ from Σ;
(ii) $\Phi_{\mathcal{DL}}$ is modular*, i.e., for $\Sigma = \langle \mathcal{T}, \mathcal{A} \rangle$ where $\mathcal{T}$ is a TBox and $\mathcal{A}$ an ABox, $\Phi_{\mathcal{DL}}(\Sigma) = \Phi_{\mathcal{DL}}(\mathcal{T}) \cup \mathcal{A}$;*

In other words, a ground atom a is entailed by the $\mathcal{DL}$ KB Σ iff $a \in MM(\Phi_{\mathcal{DL}}(\Sigma))$, the unique minimal model of the Datalog program $\Phi_{\mathcal{DL}}(\Sigma)$. Furthermore, we refer to a *polynomial* Datalog-rewritable DL $\mathcal{DL}$, if $\Phi_{\mathcal{DL}}(\Sigma)$ for a $\mathcal{DL}$ KB Σ is computable in polynomial time.

We assume w.l.o.g. that both P and $\Phi_{\mathcal{DL}}(\Sigma)$ are *safe*—each variable appears in a positive normal atom in the body—for $\mathcal{KB} = (\Sigma, P)$.

Let $\Lambda_P \triangleq \{\lambda \mid DL[\lambda; Q]$ occurs in $P\}$, i.e., the input signatures appearing in P. The translation of $\mathcal{KB} = (\Sigma, P)$ to a Datalog$^\neg$ program is then built up of the following four components:

- $\Sigma_{\Lambda_P} \triangleq \cup_{\lambda \in \Lambda_P} \Sigma_\lambda$ where Σ_λ is Σ with all concept and role names subscripted with λ. Intuitively, each input signature of a dl-atom in P will influence Σ differently. As we want to cater for these influences in one program, we have to differentiate between the KBs with different inputs.
- A Datalog program $\rho(\Lambda_P)$ containing for each $\lambda = S_1 \uplus p_1, \ldots, S_m \uplus p_m \in \Lambda_P$ the rules $S_{i\lambda}(\mathbf{X_i}) \leftarrow p_i(\mathbf{X_i}), 1 \leq i \leq m$, where the arity of $\mathbf{X_i}$ matches the one of S_i. Intuitively, we add the extension of p_i to the appropriate concept or role.
- A set T_P of Datalog rules $\top(a) \leftarrow$ and $\top^2(a, b) \leftarrow$ for all a, b in the Herbrand domain of P to ensure their introduction in Σ.
- Finally, P^{ord} results from replacing each dl-atom $DL[\lambda; Q](\mathbf{t})$ in P with a new atom $Q_\lambda(\mathbf{t})$.

The transformation of the dl-program $\mathcal{KB}$ is then defined as

$$\Psi(\mathcal{KB}) \triangleq \Phi_{\mathcal{DL}}(\Sigma_{\Lambda_P}) \cup P^{ord} \cup \rho(\Lambda_P) \cup T_P. \tag{3}$$

Example 3. *Let $\mathcal{KB} = (\Sigma, P)$ where $\Sigma = \{\ C \sqsubseteq D\ \}$ and*

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

Then $\Lambda_P = \{\lambda_1 \triangleq C \uplus s, \lambda_2 \triangleq C \uplus p\}$, such that $\rho(\Lambda_P)$ consists of $C_{\lambda_1}(X) \leftarrow s(X)$ and $C_{\lambda_2}(X) \leftarrow p(X)$. The component P^{ord} consists of $q \leftarrow D_{\lambda_1}(a), not\ D_{\lambda_2}(b)$ and the original facts.

Note that $\Psi(\mathcal{KB})$ is a Datalog program, if $\mathcal{KB}$ is negation-free, and a stratified Datalog$^\neg$ program, if $\mathcal{KB}$ is stratified (cf. [8]); thus, beneficial for evaluation, acyclic negation is fully preserved.

Proposition 1. *Let $\mathcal{KB}$ be a dl-program over a polynomial* Datalog*-rewritable DL. Then, $\Psi(\mathcal{KB})$ is constructible in polynomial time.*

The following result allows us to reduce reasoning with dl-programs to Datalog$^\neg$ under well-founded semantics.

Theorem 2. *Let $\mathcal{KB}$ be a dl-program over a* Datalog*-rewritable DL and a from $\mathcal{B}_{\mathcal{KB}}$. Then, $\mathcal{KB} \models^{wf} a$ iff $\Psi(\mathcal{KB}) \models^{wf} a$.*

From Theorem 2 and the fact that any Datalog$^\neg$ program P amounts to a dl-program $(\emptyset, P)$ [9], we obtain the following result.

Corollary 3. *For any dl-program $\mathcal{KB}$ over a DL $\mathcal{DL}$ and ground atom a from $\mathcal{B}_{\mathcal{KB}}$, deciding $\mathcal{KB} \models^{wf} a$ is (i) data complete for* PTIME*, if $\mathcal{DL}$ is* Datalog*-rewritable and (ii) combined complete for* EXPTIME*, if $\mathcal{DL}$ is polynomial* Datalog*-rewritable.*

Thus, over Datalog-rewritable DLs, the data complexity of dl-programs decreases from PTIME^{NP} to PTIME compared to $\mathcal{SHIF}(\mathbf{D})$ and $\mathcal{SHOIN}(\mathbf{D})$, and the combined complexity from $\text{PTIME}^{\text{NEXP}}$ to EXPTIME compared to $\mathcal{SHOIN}(\mathbf{D})$ (and is the same as for $\mathcal{SHIF}(\mathbf{D})$) over polynomial Datalog-rewritable DLs.

4 The Description Logic $\mathcal{LDL}^+$

In this section, we introduce the Description Logic $\mathcal{LDL}^+$ and derive some basic model-theoretic properties.

4.1 Basic Definitions

We design $\mathcal{LDL}^+$ 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.

Note that all h-roles are also b-roles, but an analog relation does not hold for concepts: $\forall E.C$ is an h-concept but not a b-concept. When immaterial, we will refer to both b-concepts and h-concepts as ($\mathcal{LDL}^+$) concepts; we use an analog convention for roles.

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

Normal Form. To simplify matters, we restrict to an expressive normal form of $\mathcal{LDL}^+$ knowledge bases Σ. First, an assertion $C(o)$ is equivalent to the axiom $\{o\} \sqsubseteq C$, and and similarly $E(o_1, o_2)$ is equivalent to $\{(o_1, o_2)\} \sqsubseteq E$; hence, we assume that the ABox is empty and identify Σ with its TBox. Second, every axiom $B \sqsubseteq H$ as above can be equivalently rewritten such that H is either a concept name A, the $\top$ symbol, or $\forall E.A$, where A is a concept name and E is a b-role. We can similarly remove conjunction from the head T of role axioms $S \sqsubseteq T$, and restrict the h-role T to role names, inverse role names, and $\top^2$.

Proposition 4. *Every $\mathcal{LDL}^+$ KB Σ can be transformed into the form described in polynomial (in fact, in linear) time.*

In the sequel, we tacitly deal with such *normalized* $\mathcal{LDL}^+$ KBs.

4.2 Immediate Consequence Operator

In this section, we define an immediate consequence operator for $\mathcal{LDL}^+$ that allows us to calculate the ground entailment of atoms. Moreover, we show that ground entailment for $\mathcal{LDL}^+$ is domain independent, and thus can be confined to the constants in the KB.

We first show that b-concepts satisfy a *monotonicity* property. For a given KB Σ and interpretations $\mathcal{I} = (\Delta, \cdot^{\mathcal{I}})$ and $\mathcal{J} = (\Delta, \cdot^{\mathcal{J}})$ over the same domain Δ, we write $\mathcal{I} \subseteq \mathcal{J}$ if $A^{\mathcal{I}} \subseteq A^{\mathcal{J}}$ for concept names A in Σ and $P^{\mathcal{I}} \subseteq P^{\mathcal{J}}$ for role names P in Σ; note that $o^{\mathcal{I}} = o^{\mathcal{J}}$ for any individual o due to the unique names assumption. Then $\mathcal{I} \subset \mathcal{J}$ if $\mathcal{I} \subseteq \mathcal{J}$ but $\mathcal{I} \neq \mathcal{J}$. We say that an interpretation (resp. model) $\mathcal{I} = (\Delta, \cdot^{\mathcal{I}})$ of Σ is *minimal*, if there is no interpretation (resp. model) $\mathcal{J} = (\Delta, \cdot^{\mathcal{J}})$ of Σ such that $\mathcal{J} \subset \mathcal{I}$.

Definition 2. *An $\mathcal{LDL}^+$ concept (role) C (E) is* monotonic*, if for each pair of interpretations $\mathcal{I} = (\Delta, \cdot^{\mathcal{I}})$ and $\mathcal{J} = (\Delta, \cdot^{\mathcal{J}})$ of Σ, $\mathcal{I} \subseteq \mathcal{J}$ implies $C^{\mathcal{I}} \subseteq C^{\mathcal{J}}$ ($E^{\mathcal{I}} \subseteq E^{\mathcal{J}}$).*

Proposition 5. *All b-concepts and all $\mathcal{LDL}^+$ roles are monotonic.*

Note that an h-concept $\forall E.B$ is not monotonic.

We can write interpretations $\mathcal{I} = (\Delta, \cdot^{\mathcal{I}})$ as sets, called *set-interpretations*, consisting of $A(x)$ if $x \in A^{\mathcal{I}}$, $P(x, y)$ if $(x, y) \in P^{\mathcal{I}}$ for concept (role) names A (P), and $\{o\}(o)$ for individuals o. Instead of $x \in C^{\mathcal{I}}$ ($(x, y) \in E^{\mathcal{I}}$), we write $\mathcal{I} \models C(x)$ ($\mathcal{I} \models E(x, y)$) for concepts (roles) C (E). We furthermore assume that each such $\mathcal{I}$ contains $\top(x)$ for every $x \in \Delta$ as well as $\top^2(x, y)$ for all $x, y \in \Delta$.

One can see that for a fixed Δ, the set $\mathbf{I}_{\Delta}$ of all set-interpretations over Δ is under the usual subset relation $\subseteq$ a complete lattice as in [24]. For an $\mathcal{LDL}^+$ KB Σ and a domain Δ, we then define an *immediate consequence operator* T_{Δ} on $\mathbf{I}_{\Delta}$ as follows, where A ranges over the concept names, P over the role names, and x, y over Δ:

$$\begin{aligned} T_{\Delta}(\mathcal{I}) = \mathcal{I} &\cup \{A(x) \mid B \sqsubseteq A \in \Sigma, \mathcal{I} \models B(x)\} \\ &\cup \{A(x) \mid B \sqsubseteq \forall E.A \in \Sigma, \mathcal{I} \models B(y), \mathcal{I} \models E(y, x)\} \\ &\cup \{P(x, y) \mid S \sqsubseteq P \in \Sigma, \mathcal{I} \models S(x, y)\} \\ &\cup \{P(y, x) \mid S \sqsubseteq P^- \in \Sigma, \mathcal{I} \models S(x, y)\} \ . \end{aligned}$$

For a set-interpretation $\mathcal{I}$ of Σ over Δ, $T_{\Delta}(\mathcal{I})$ is still a set-interpretation of Σ over Δ, such that T_{Δ} is well-defined.

As easily seen, T_{Δ} is *monotone*, i.e., $\mathcal{J} \subseteq \mathcal{I}$ implies $T_{\Delta}(\mathcal{J}) \subseteq T_{\Delta}(\mathcal{I})$, and thus has a least fixpoint $\mathrm{LFP}(T_{\Delta})$, i.e., a unique minimal $\mathcal{I}$ such that $T_{\Delta}(\mathcal{I}) = \mathcal{I}$ [24]. This fixpoint corresponds to a model of Σ, which in fact is the singe minimal model of Σ over Δ.

Proposition 6. *Let Σ be an $\mathcal{LDL}^+$ KB and let Δ be a domain for Σ. Then, (i) Σ has a unique minimal model $\mathcal{I} = (\Delta, \cdot^{\mathcal{I}})$, denoted $MM(\Delta, \Sigma)$, and (ii) $\mathrm{LFP}(T_{\Delta}) = MM(\Delta, \Sigma)$.*

Entailment checking of b-concepts can then in each domain be restricted to the unique minimal model for that domain.

Proposition 7. *Let Σ be an $\mathcal{LDL}^+$ KB, C a b-concept, and $o \in \Delta_{\mathcal{H}(\Sigma)}$. Then, $\Sigma \models C(o)$ iff for all Δ, $MM(\Delta, \Sigma) \models C(o)$.*

Note that the proposition does not necessarily hold if C is an h-concept. For example, consider $\Sigma = \{\{a\} \sqsubseteq A\}$ and the h-concept $C = \forall R.A$, where A is a concept name and R is a role name. Clearly, $\Sigma \not\models \forall R.A(a)$. However, when we consider the domain $\Delta_{\mathcal{H}(\Sigma)} = \{a\}$, $MM(\Delta_{\mathcal{H}(\Sigma)}, \Sigma) = \{A(a)\}$ and $MM(\Delta_{\mathcal{H}(\Sigma)}, \Sigma) \models \forall R.A(a)$.

Importantly, the only relevant interpretation domain is the Herbrand domain $\Delta_{\mathcal{H}(\Sigma)}$ of the KB Σ.

Proposition 8. *Let Σ be an $\mathcal{LDL}^+$ KB, C a b-concept, and $o \in \Delta_{\mathcal{H}(\Sigma)}$. Then, $\Sigma \models C(o)$ iff $MM(\Delta_{\mathcal{H}(\Sigma)}, \Sigma) \models C(o)$.*

Note that $MM(\Delta_{\mathcal{H}(\Sigma)}, \Sigma) = \mathrm{LFP}(T_{\Delta_{\mathcal{H}(\Sigma)}})$ is effectively constructable by fixpoint iteration (for a finite KB in finite time).

Proposition 8 is at the core of the argument that $\mathcal{LDL}^+$ is a Datalog-rewritable DL, which we show in the next section.

5 $\mathcal{LDL}^+$ is Datalog-rewritable

To show that a (normalized) $\mathcal{LDL}^+$ KB Σ is Datalog-rewritable, we construct a suitable Datalog program $\Phi_{\mathcal{LDL}^+}(\Sigma)$ such that $\Sigma \models Q(\mathbf{o})$ iff $\Phi_{\mathcal{LDL}^+}(\Sigma) \models Q(\mathbf{o})$, whenever A is a concept- or role name appearing in Σ and $\mathbf{o} \subseteq \Delta_{\mathcal{H}(\Sigma)}$.

Define the *closure* of Σ, $clos(\Sigma)$, as the smallest set containing *(i)* all subexpressions that occur in Σ (both roles and concepts) except value restrictions, and *(ii)* for each role name occurring in Σ, its inverse. Formally, $\Phi_{\mathcal{LDL}^+}(\Sigma)$ is the following program:

- For each axiom $B \sqsubseteq H \in \Sigma$ where H is a concept name, add the rule $H(X) \leftarrow B(X)$.
- For each axiom $B \sqsubseteq \forall E.A \in \Sigma$ where A is a concept name, add the rule $A(Y) \leftarrow B(X), E(X, Y)$.
- For each role axiom $S \sqsubseteq T \in \Sigma$, add $T(X, Y) \leftarrow S(X, Y)$. (Here $T = P^-$ may be an inverse for a role name P.)
- For each role name P that occurs in Σ, add the rule $P(X, Y) \leftarrow P^-(Y, X)$.
- For each concept (role) name or (role) nominal Q (Q') in $clos(\Sigma)$, add the rules $\top(X) \leftarrow Q(X)$, $\top(X) \leftarrow Q'(X, Y)$, and $\top(Y) \leftarrow Q'(X, Y)$. This ensures that newly introduced constants, e.g., in the context of dl-programs, are also assigned to $\top$ — a relevant property for modularity.
- To deduce the top role, add $\top^2(X, Y) \leftarrow \top(X), \top(Y)$.
- Next, we distinguish between the types of concepts D in $clos(\Sigma)$:
  - if $D = \{o\}$, add $D(o) \leftarrow$.
  - if $D = D_1 \sqcap D_2$, add $D(X) \leftarrow D_1(X), D_2(X)$.
  - if $D = D_1 \sqcup D_2$, add $D(X) \leftarrow D_1(X)$ and $D(X) \leftarrow D_2(X)$.
  - if $D = \exists E.D_1$, add the rule $D(X) \leftarrow E(X, Y), D_1(Y)$.
  - if $D = \geq n\, E.D_1$, add

$$\begin{aligned} D(X) \;\leftarrow\; & E(X, Y_1), D(Y_1), \ldots, E(X, Y_n), D(Y_n), \\ & Y_1 \neq Y_2, \ldots, Y_i \neq Y_j, \ldots, Y_{n-1} \neq Y_n \end{aligned} \tag{4}$$

    (where $1 \leq i < j \leq n$).
- Finally, for each role $E \in clos(\Sigma)$:
  - if $E = \{(o_1, o_2)\}$, add $E(o_1, o_2) \leftarrow$.
  - if $E = F^-$, add $E(X, Y) \leftarrow F(Y, X)$.
  - if $E = E_1 \sqcap E_2$, add $E(X, Y) \leftarrow E_1(X, Y), E_2(X, Y)$.
  - if $E = E_1 \sqcup E_2$, add $E(X, Y) \leftarrow E_1(X, Y)$ and $E(X, Y) \leftarrow E_2(X, Y)$.
  - if $E = E_1 \circ E_2$, add $E(X, Y) \leftarrow E_1(X, Z), E_2(Z, Y)$.
  - if $E = F^+$, add

$$\begin{aligned} E(X, Y) \;&\leftarrow\; F(X, Y) \\ E(X, Y) \;&\leftarrow\; F(X, Z), E(Z, Y) \end{aligned} \tag{5}$$

Proposition 9. *$\Phi_{\mathcal{DL}^+}$ is polynomial rewritable. Furthermore, $\Phi_{\mathcal{DL}^+}$ is modular.*

The next result shows that $\Phi_{\mathcal{DL}^+}(\Sigma)$ works as desired.

Proposition 10. *For every (normalized) $\mathcal{DL}^+$ KB Σ, $Q \in clos(\Sigma)$, and $\mathbf{o} \subseteq \Delta_{\mathcal{H}(\Sigma)}$, it holds that $\Sigma \models Q(\mathbf{o})$ iff $\Phi_{\mathcal{DL}^+}(\Sigma) \models Q(\mathbf{o})$.*

Corollary 11. *$\mathcal{DL}^+$ is (polynomial)* Datalog*-rewritable.*

Thus, using Theorem 2, reasoning with dl-programs over $\mathcal{DL}^+$ reduces to reasoning with Datalog$^\neg$ under well-founded semantics.

Example 5. *Take the $\mathcal{DL}^+$ KB Σ from Example 1. Then, the reduction yields the* Datalog *program $\Phi_{\mathcal{DL}^+}(\Sigma)$:*

$$\begin{aligned} Over(X) \;&\leftarrow\; (\geq 2\ PapToRev.\top)(X) \\ (\geq 2\ PapToRev.\top)(X) \;&\leftarrow\; PapToRev(X, Y_1), \top(Y_1) \\ & \qquad PapToRev(X, Y_2), \top(Y_2), Y_1 \neq Y_2 \\ Over(Y) \;&\leftarrow\; Super^+(X, Y), Over(X) \\ Super^+(X, Y) \;&\leftarrow\; Super(X, Y) \\ Super^+(X, Y) \;&\leftarrow\; Super(X, Z), Super^+(Z, Y) \\ Super(X, Y) \;&\leftarrow\; \{(a, b)\}(X, Y) \\ Super(X, Y) \;&\leftarrow\; \{(b, c)\}(X, Y) \\ \{(a, b)\}(a, b) \;&\leftarrow \\ \{(b, c)\}(b, c) \;&\leftarrow \\ Super(X, Y) \;&\leftarrow\; Super^-(Y, X) \\ PapToRev(X, Y) \;&\leftarrow\; PapToRev^-(Y, X) \\ \top^2(X, Y) \;&\leftarrow\; \top(X), \top(Y) \end{aligned}$$

and in addition the rules for $\top$.For $\mathcal{KB}$ in Example 2, we then can easily construct $\Psi(\mathcal{KB})$.

Reductions of DLs to LP have been considered before, e.g., in [14, 23]. Swift [23] reduces reasoning in the DL $\mathcal{ALCQI}$ (in fact, consistency checking of concept expressions) to Datalog$^\neg$ under answer set semantics (employing a guess and check methodology), while Hustadt et al. [14] reduce reasoning in the DL $\mathcal{SHIQ}^-$ to disjunctive Datalog in a non-modular way, i.e., the translation as such is not usable in the context of dl-programs; both DLs considered in [14] and [23] do not feature transitive closure.

From the complexity of Datalog, we obtain by Datalog-rewritability of $\mathcal{DL}^+$ immediately that it is tractable under data complexity. Moreover, due to the structure of $\Phi_{\mathcal{DL}^+}(\Sigma)$, the same holds under combined complexity.

Corollary 12. *For every $\mathcal{DL}^+$ KB Σ, concept name A, and $o \in \Delta_{\mathcal{H}(\Sigma)}$, deciding $\Sigma \models A(o)$ is in* PTIME *under both data and combined complexity.*

Indeed, all rules in $\Phi_{\mathcal{DL}^+}(\Sigma)$ except (4) can be grounded in polynomial time (they use only constantly many variables). The rule (4) can be partially grounded for all values of X; whether the body of such a partially grounded rule can be satisfied in a given set of ground atoms is easily decided in polynomial time; hence, we can compute $MM(\Phi_{\mathcal{DL}^+}(\Sigma))$ by simple fixpoint iteration in polynomial time.

We will establish matching lower complexity bounds below.

6 The OWL 2 Profiles

In this section, we review the OWL 2 Profiles [17], which are fragments of OWL 2 [18] that can be more efficiently evaluated than OWL 2, and discuss their relation with $\mathcal{DL}^+$.

OWL 2 EL. The OWL 2 EL Profile corresponds to the DL $\mathcal{EL}^{++}$ [1, 2]. We consider the definition of $\mathcal{EL}^{++}$ in [2], which extends [1], in particular its *normal form for TBoxes*. One can verify that the only constructs preventing $\mathcal{EL}^{++}$ axioms from being equivalent $\mathcal{DL}^+$ axioms are $\bot$, concrete domains, and exists restrictions in axiom right-hand sides. Let $\mathcal{EL}^{++}_-$ denote $\mathcal{EL}^{++}$ without $\bot$, concrete domains, and such exists restrictions.

Proposition 13. *$\mathcal{EL}^{++}_-$ is a fragment of $\mathcal{DL}^+$, i.e., each $\mathcal{EL}^{++}_-$ KB is an $\mathcal{DL}^+$ KB, and thus polynomially* Datalog*-rewritable.*

Even though $\mathcal{EL}^{++}$ is not a fragment of $\mathcal{DL}^+$, in turn $\mathcal{DL}^+$ contains many constructs that $\mathcal{EL}^{++}$ does not allow, e.g., qualified number restrictions, inverses, general sequences of roles, role conjunction, role disjunction, concept disjunction in axiom bodies.

OWL 2 QL. The OWL 2 QL Profile corresponds to the *DL-Lite* family $DL\text{-}Lite_{core}$, $DL\text{-}Lite_{\mathcal{R}}$, and $DL\text{-}Lite_{\mathcal{F}}$ [6]. Denote by $DL\text{-}Lite^-_X$ the DL $DL\text{-}Lite_X$ without negation and exists restrictions in axiom right-hand sides, $X \in \{core, \mathcal{R}, \mathcal{F}\}$. Then, both terminological and role axioms in $DL\text{-}Lite^-_{\mathcal{R}}$ are $\mathcal{DL}^+$ axioms; and, any $DL\text{-}Lite^-_{\mathcal{R}}$ ABox can be rewritten using the nominals of $\mathcal{DL}^+$ as usual.

Proposition 14. *The DLs $DL\text{-}Lite^{-}_{core}$ and $DL\text{-}Lite^{-}_{\mathcal{R}}$ are fragments of $\mathcal{LDL}^{+}$, and thus polynomially* Datalog*-rewritable.*

Similar to $\mathcal{EL}^{++}$, full $DL\text{-}Lite_{core}$ and $DL\text{-}Lite_{\mathcal{R}}$ are not fragments of $\mathcal{LDL}^{+}$, but in turn $\mathcal{LDL}^{+}$ has constructs which none of the DLs $DL\text{-}Lite_{X}$ allows, e.g., role sequences.

The DL $DL\text{-}Lite^{-}_{\mathcal{F}}$, however, is not a fragment of $\mathcal{LDL}^{+}$. Indeed, like $DL\text{-}Lite_{\mathcal{F}}$ it allows for functional restrictions on roles, something that is not expressible in Datalog as such.

OWL 2 RL. The OWL 2 RL Profile extends so-called *Description Logic Programs* [12]. The latter have a classical model semantics and correspond to the restriction of $\mathcal{LDL}^{+}$ to conjunction and disjunction of concepts, exists restrictions, and value restrictions. Thus, Description Logic Programs are a strict subset of $\mathcal{LDL}^{+}$, missing, e.g., nominals, qualified number restrictions, and role constructors.

Proposition 15. *Description Logic Programs are a fragment of $\mathcal{LDL}^{+}$, and thus polynomially* Datalog*-rewritable.*

Note that the translation of the transitive closure of a role expression E^{+} results in the recursive rules (5) such that, in contrast with Description Logic Programs, the transformation $\Phi_{\mathcal{LDL}^{+}}$is not a first-order rewriting, justifying the term Datalog*-rewritable*. Although DLs with expressive role constructs such as role sequence, role disjunction and transitive closure tend to become undecidable (e.g., $\mathcal{ALC}^{+}\mathcal{N}(\circ,\sqcup)$ [4]), $\mathcal{LDL}^{+}$ remains decidable. Moreover, it has an Herbrand domain model property (a finite model property where the domain is the Herbrand domain). Indeed, from [4] one can see that the undecidability proofs for expressive DLs extensively use functional restrictions on roles, a feature $\mathcal{LDL}^{+}$ cannot express.

Checking ground entailment in OWL 2 RL and Description Logic Programs is data and combined complete for PTIME [17]. As the latter are a fragment of $\mathcal{LDL}^{+}$ without number restrictions, combined with Corollary 12 we obtain the following result.

Proposition 16. *For any $\mathcal{LDL}^{+}$ KB Σ, concept name A, and $o \in \Delta_{\mathcal{H}(\Sigma)}$, deciding $\Sigma \models A(o)$ is data and combined complete for* PTIME*. The hardness holds in absence of number restrictions.*

7 Conclusion

We have presented a transformation of nonmonotonic dl-programs, which are the major formalism for loosely-coupling DL KBs and nonmonotonic logic programming, to Datalog under well-founded semantics, which is a predominant nonmonotonic rule formalism in data and knowledge bases that allows for tractable reasoning. The transformation is applicable to a range of different DLs, including $\mathcal{LDL}^{+}$, a novel rich DL, as well as to large fragments of the OWL 2 Profiles that have been designed for tractable DL reasoning. In particular, the transformation of a negation-free (stratified) dl-program results in a Datalog (stratified Datalog$^{\neg}$) program. In this way, we obtain tractable reasoning with recursion and negation, which thanks to the availability of efficient engines for well-founded semantics (e.g., the XSB system) provides a basis for developing efficient and scalable applications that combine rules and ontologies.

Looking at the OWL 2 Profiles based on the $DL\text{-}Lite$ families and $\mathcal{EL}^{++}$, it appears that one of the missing features in $\mathcal{LDL}^{+}$ is the *exists restriction* on axiom right-hand sides. However, DLs allowing this are not straight Datalog-rewritable, as this feature can enforce new domain elements (beyond the Herbrand domain). One may handle this using function symbols in the logic program or *open domains* [13]. However, both extensions cause undecidability in general.

Compared to OWL 2 QL, $\mathcal{LDL}^{+}$ misses *negation*. Negation is not realizable in Datalog; it remains to be seen whether for Datalog$^{\neg}$ under well-founded semantics, transformations similar to the one we presented (with possibly restricted negation in DLs) are feasible.

Finally, Datalog-rewritability is not just useful for (1) DL reasoning via Datalog engines or (2) loosely-coupled reasoning via dl-programs, but also for tight-coupling approaches such as *r-hybrid KBs* [21]. Intuitively, while rules in r-hybrid KBs must be *DL-safe* to ensure that only the Herbrand domain is relevant, our approach hints that it is also interesting to look at DLs that have this property. Note that they are of particular interest for data management, where often just the Herbrand domain matters.

REFERENCES

[1] F. Baader, S. Brandt, and C. Lutz, 'Pushing the $\mathcal{EL}$ envelope', in *Proc. IJCAI*, pp. 364–369. Morgan-Kaufmann Publishers, (2005).

[2] F. Baader, S. Brandt, and C. Lutz, 'Pushing the $\mathcal{EL}$ envelope further', in *Proc. OWLED08DC*, (2008). `http://ceur-ws.org/Vol-496`.

[3] *The Description Logic Handbook*, eds., F. Baader, D. Calvanese, D. L. McGuinness, D. Nardi, and P. F. Patel-Schneider, CUP, 2003.

[4] F. Baader and U. Sattler, 'Number restrictions on complex roles in DLs: A preliminary report', in *Proc. KR*, pp. 328–339, (1996).

[5] C. Baral and V. S. Subrahmanian, 'Dualities between alternative semantics for logic programming and nonmonotonic reasoning', *JAR*, **10**(3), 399–420, (1993).

[6] D. Calvanese, G. de Giacomo, D. Lembo, M. Lenzerini, and Riccardo Rosati, 'Tractable reasoning and efficient query answering in description logics: The DL-Lite family', *JAR*, **39**(3), 385–429, (2007).

[7] E. Dantsin, T. Eiter, G. Gottlob, and A. Voronkov, 'Complexity and expressive power of logic programming', *ACM Computing Surveys*, **33**(3), 374–425, (2001).

[8] 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).

[9] T. Eiter, T. Lukasiewicz, R. Schindlauer, and H. Tompits, 'Well-founded semantics for description logic programs in the Semantic Web', in *Proc. RuleML*, pp. 81–97, (2004). Full paper *ACM TOCL*, (to appear).

[10] A. Van Gelder, K. Ross, and J. S. Schlipf, 'The well-founded semantics for general logic programs', *JACM*, **38**(3), 620–650, (1991).

[11] M. Gelfond and V. Lifschitz, 'The stable model semantics for logic programming', in *Proc. ICLP*, pp. 1070–1080. The MIT Press, (1988).

[12] B. N. Grosof, I. Horrocks, R. Volz, and S. Decker, 'Description logic programs: Combining logic programs with description logic', in *Proc. WWW 2003*, pp. 48–57. ACM, (2003).

[13] S. Heymans, D. Van Nieuwenborgh, and D. Vermeir, 'Open answer set programming with guarded programs', *ToCL*, **9**(4), 1–53, (2008).

[14] U. Hustadt, B. Motik, and U. Sattler, 'Reducing $\mathcal{SHIQ}^{-}$ description logic to disjunctive datalog programs', in *Proc. of KR*, pp. 152–162. AAAI Press, (2004).

[15] M. Krötzsch, S. Rudolph, and P. Hitzler, 'Description logic rules', in *Proc. ECAI*, pp. 80–84. IOS Press, (2008).

[16] M. Krötzsch, S. Rudolph, and P. Hitzler, 'ELP: Tractable rules for OWL 2', in *Proc. ISWC 2008*, pp. 649–664, (2008).

[17] *OWL 2 Web Ontology Profiles*, eds., B. Motik, B. Cuenca Grau, I. Horrocks, Z. Wu, A. Fokoue, and C. Lutz, 2008. W3C Rec. 27 Oct. 2009.

[18] *OWL 2 Web Ontology Language: Structural Specification and Functional-Style Syntax*, eds., B. Motik, P. F. Patel-Schneider, and B. Parsia, 2008. W3C Working Draft April 2009.

[19] B. Motik and R. Rosati, 'A faithful integration of description logics with logic programming', in *Proc. IJCAI*, pp. 477–482, (2007).

[20] B. Motik, U. Sattler, and R. Studer, 'Query answering for OWL-DL with rules', *Journal of Web Semantics*, **3**(1), 41–60, (July 2005).

[21] R. Rosati, 'On the decidability and complexity of integrating ontologies and rules', *Journal of Web Semantics*, **3**(1), 41–60, (2005).

[22] R. Rosati, 'DL+log: Tight integration of description logics and disjunctive datalog', in *Proc. KR*, pp. 68–78, (2006).

[23] T. Swift, 'Deduction in ontologies via ASP', in *Proc. of LPNMR*, pp. 275–288, (2004).

[24] A. Tarski, 'A lattice-theoretical fixpoint theorem and its applications', *Pacific Journal of Mathematics*, **5**, 285–309, (1955).