Polynomial Horn Rewritings for Description Logics Ontologies*

Mark Kaminski and Bernardo Cuenca Grau

Department of Computer Science, University of Oxford, UK

Abstract. We study the problem of rewriting an ontology $\mathcal{O}_1$ in a DL $\mathcal{L}_1$ into an ontology $\mathcal{O}_2$ in a Horn DL $\mathcal{L}_2$ such that $\mathcal{O}_1$ and $\mathcal{O}_2$ are equisatisfiable when extended with any dataset. After showing undecidability whenever $\mathcal{L}_1$ extends $\mathcal{ALCF}$, we focus on devising efficiently checkable conditions that ensure existence of a Horn rewriting. By lifting existing Datalog rewriting techniques for Disjunctive Datalog programs to first-order programs with function symbols, we identify a class of ontologies that admit Horn rewritings of polynomial size. Our experiments indicate that many real-world ontologies admit such polynomial Horn rewritings.

1 Introduction

Reasoning over ontology-enriched datasets is a key requirement in many applications. Standard reasoning tasks are, however, of high worst-case complexity. Satisfiability checking is 2NExpTime-complete for the DL $\mathcal{SROIQ}$ underpinning OWL 2 and NExpTime-complete for $\mathcal{SHOIN}$, which underpins OWL DL [13]. Reasoning is also co-NP-hard in *data complexity*—a key measure of complexity for applications involving large amounts of instance data [9].

Tractability in data complexity is typically associated with *Horn* DLs, where ontologies correspond to first-order Horn clauses [18, 9]. The more favourable computational properties of Horn DLs make them a natural choice for data-intensive applications, but they also come at the expense of a loss in expressive power. In particular, Horn DLs cannot capture disjunctive axioms, i.e., statements such as "every X is either a Y or a Z". Disjunctive axioms are common in real-world ontologies, like the NCI Thesaurus or the ontologies underpinning the EBI linked data platform (see http://www.ebi.ac.uk/rdf/platform).

In this paper we are interested in *Horn rewritability* of description logic ontologies; that is, whether an ontology $\mathcal{O}_1$ in a DL $\mathcal{L}_1$ can be restated as an ontology $\mathcal{O}_2$ in a Horn DL $\mathcal{L}_2$ such that $\mathcal{O}_1$ and $\mathcal{O}_2$ are equisatisfiable when extended with an arbitrary dataset. Ontologies admitting such rewritings are amenable to more efficient reasoning techniques that are tractable in data complexity.

Horn rewritability of DL ontologies is strongly related to the rewritability of Disjunctive Datalog programs into Datalog, where both the source and target

* Work supported by the Royal Society, the EPSRC projects Score!, MaSI^3 and DBOnto, and the FP7 project Optique.

languages for rewriting are function-free. Kaminski et al. [12] characterised Datalog rewritability of Disjunctive Datalog programs in terms of linearity: a restriction that requires each rule to contain at most one body atom that is IDB (i.e., whose predicate also occurs in head position in the program). It was shown that every linear Disjunctive Datalog program can be rewritten into plain Datalog (and vice versa) by means of *program transposition*—a polynomial transformation in which rules are "inverted" by shuffling all IDB atoms between head and body while replacing their predicates by auxiliary ones. Subsequently, Kaminski et al. [11] proposed the class of *markable* Disjunctive Datalog programs, where the linearity requirement is relaxed so that it applies only to a subset of "marked" atoms. Every markable program can be polynomially rewritten into Datalog by exploiting a variant of transposition where only marked atoms are affected.

Our contributions in this paper are as follows. In Section 3, we show undecidability of Horn rewritability for ontologies in $\mathcal{ALCF}$. This is in consonance with the related undecidability results by Bienvenu et al. [3] and Lutz and Wolter [17]. In Section 4, we lift the markability condition and the transposition transformation in [11] for Disjunctive Datalog to first-order programs with function symbols. We then show that all markable programs admit Horn rewritings of polynomial size. This result is rather general and has potential implications in areas such as theorem proving [19] and knowledge compilation [5]. The notion of markability for first-order programs easily transfers to ontologies via the standard FOL translation of DLs [2]. This is, however, of limited practical value since Horn programs obtained via transposition may not be expressible using standard DL constructors. In Section 5, we introduce an alternative satisfiability-preserving translation from $\mathcal{ALCHIF}$ ontologies to first-order programs and show in Section 6 that the corresponding transposed programs can be translated back into Horn-$\mathcal{ALCHIF}$ ontologies. Finally, we focus on complexity and show that reasoning over markable $\mathcal{L}$-ontologies is ExpTime-complete in combined complexity and PTime-complete w.r.t. data for each DL $\mathcal{L}$ between $\mathcal{ELU}$ and $\mathcal{ALCHIF}$. All our results immediately extend to DLs with transitive roles (e.g., $\mathcal{SHIF}$) by exploiting standard transitivity elimination techniques [2].

We have implemented markability checking and evaluated our techniques on a large ontology repository. Our results indicate that many real-world ontologies are markable and thus admit Horn rewritings of polynomial size.

All proofs are deferred to an extended version (see `arXiv:1504.05150`).

2 Preliminaries

We assume standard first-order syntax and semantics. We treat the universal truth $\top$ and falsehood $\bot$ symbols as well as equality ($\approx$) as ordinary predicates of arity one ($\top$ and $\bot$) and two ($\approx$), the meaning of which will be axiomatised.

Programs A *(first-order) rule* is a sentence $\forall \boldsymbol{x} \forall \boldsymbol{z}.[\varphi(\boldsymbol{x}, \boldsymbol{z}) \rightarrow \psi(\boldsymbol{x})]$ where variables $\boldsymbol{x}$ and $\boldsymbol{z}$ are disjoint, $\varphi(\boldsymbol{x}, \boldsymbol{z})$ is a conjunction of distinct atoms over $\boldsymbol{x} \cup \boldsymbol{y}$, and $\psi(\boldsymbol{x})$ is a disjunction of distinct atoms over $\boldsymbol{x}$. Formula φ is the *body* of r, and ψ is the *head*. Quantifiers are omitted for brevity, and safety is assumed

$T1.$	$\bigsqcap_{i=1}^{n} A_i \sqsubseteq \bigsqcup_{j=1}^{m} C_j$	$\bigwedge_{i=1}^{n} A_i(x) \rightarrow \bigvee_{j=1}^{m} C_j(x)$
$T2.$	$\exists R.A \sqsubseteq C$	$\mathsf{at}(R, x, y) \wedge A(y) \rightarrow C(x)$
$T3.$	$A \sqsubseteq \exists R.B$	$A(x) \rightarrow \mathsf{at}(R, x, f(x));\ A(x) \rightarrow B(f(x))$
$T4.$	$A \sqsubseteq \forall R.C$	$A(x) \wedge \mathsf{at}(R, x, y) \rightarrow C(y)$
$T5.$	$S \sqsubseteq R$	$S(x, y) \rightarrow \mathsf{at}(R, x, y)$
$T6.$	$A \sqsubseteq \leq 1\, R.B$	$A(z) \wedge \mathsf{at}(R, z, x_1) \wedge \mathsf{at}(R, z, x_2) \wedge B(x_1) \wedge B(x_2) \rightarrow x_1 \approx x_2$

Table 1. Normalised DL axioms. A, B are named or $\top$; C named or $\bot$; role S is named and R is a (possibly inverse) role.

(all variables in the rule occur in the body). We define the following sets of rules for a finite signature Σ: *(i)* $\mathcal{P}_\Sigma^\top$ consists of a rule $P(x_1, \ldots, x_n) \rightarrow \top(x_i)$ for each predicate $P \in \Sigma$ and each $1 \leq i \leq n$ and a rule $\rightarrow \top(a)$ for each constant $a \in \Sigma$; *(ii)* $\mathcal{P}_\Sigma^\bot$ consists of the rule with $\bot(x)$ in the body and an empty head; and *(iii)* $\mathcal{P}_\Sigma^\approx$ consists of the standard axiomatisation of $\approx$ as a congruence over Σ. A *program* is a finite set of rules $\mathcal{P} = \mathcal{P}_0 \cup \mathcal{P}_\Sigma^\top \cup \mathcal{P}_\Sigma^\bot \cup \mathcal{P}_\Sigma^\approx$ with Σ the signature of $\mathcal{P}_0$, where we assume w.l.o.g. that the body of each rule in $\mathcal{P}_0$ does not mention $\bot$ or $\approx$, and the head is non-empty and does not mention $\top$. We omit Σ for the components of $\mathcal{P}$ and write $\mathcal{P}^\top$, $\mathcal{P}^\bot$ and $\mathcal{P}^\approx$. A rule is *Horn* if its head consists of at most one atom, and a program is Horn if so are all of its rules. Finally, a *fact* is a ground, function-free atom, and a *dataset* is a finite set of facts.

Ontologies A signature Σ consists of disjoint countable sets of concept names Σ_C and role names Σ_R. A role is an element of $\Sigma_R \cup \{R^- \mid R \in \Sigma_R\}$. The function inv is defined over roles as follows, where $R \in \Sigma_R$: $\mathsf{inv}(R) = R^-$ and $\mathsf{inv}(R^-) = R$. W.l.o.g., we consider normalised axioms as on the left-hand side of Table 1. An $\mathcal{ALCHIF}$ ontology $\mathcal{O}$ is a finite set of axioms of type T1-T6 in Table 1. An ontology is *Horn* if it contains no axiom T1 where $m \geq 2$. Given $\mathcal{O}$, we write $\sqsubseteq^*$ for the minimal reflexive and transitive relation over roles in $\mathcal{O}$ s.t. $R_1 \sqsubseteq^* R_2$ and $\mathsf{inv}(R_1) \sqsubseteq^* \mathsf{inv}(R_2)$ hold whenever $R_1 \sqsubseteq R_2 \in \mathcal{O}$.

We refer to the DL where only axioms T1–T3 are available and inverse roles are disallowed as $\mathcal{ELU}$. The logic $\mathcal{ALC}$ extends $\mathcal{ELU}$ with axioms T4. We then use standard naming conventions for DLs based on the presence of inverses ($\mathcal{I}$), axioms T5 ($\mathcal{H}$) and axioms T6 ($\mathcal{F}$). An ontology is $\mathcal{EL}$ if it is $\mathcal{ELU}$ and Horn.

Table 1 also provides the standard translation π from normalised axioms into rules, where $\mathsf{at}(R, x, y)$ stands for $R(x, y)$ if R is named and $S(y, x)$ if $R = S^-$. We define $\pi(\mathcal{O})$ as the smallest program containing $\pi(\alpha)$ for each axiom α in $\mathcal{O}$. Given a dataset $\mathcal{D}$, we say that $\mathcal{O} \cup \mathcal{D}$ is *satisfiable* iff so is $\pi(\mathcal{O}) \cup \mathcal{D}$ in FOL.

3 Horn Rewritability

Our focus is on satisfiability-preserving rewritings. Standard reasoning tasks in DLs are reducible to unsatisfiability checking [2], which makes our results practically relevant. We start by formulating our general notion of rewriting.

Definition 1. *Let $\mathcal{F}, \mathcal{F}'$ be sets of rules. Then $\mathcal{F}'$ is a* rewriting *of $\mathcal{F}$ if it holds that $\mathcal{F} \cup \mathcal{D}$ is satisfiable iff so is $\mathcal{F}' \cup \mathcal{D}$ for each dataset $\mathcal{D}$ over predicates from $\mathcal{F}$.*

We are especially interested in computing Horn rewritings of ontologies—that is, rewritings where the given ontology $\mathcal{O}_1$ is expressed in a DL $\mathcal{L}_1$ and the rewritten ontology $\mathcal{O}_2$ is in a Horn DL $\mathcal{L}_2$ (where preferably $\mathcal{L}_2 \subseteq \mathcal{L}_1$). This is not possible in general: satisfiability checking is co-NP-complete in data complexity even for the basic logic $\mathcal{ELU}$ [14], whereas data complexity is tractable even for highly expressive Horn languages such as Horn-$\mathcal{SROIQ}$ [18]. Horn rewritability for DLs can be formulated as a decision problem as follows:

Definition 2. *The $(\mathcal{L}_1, \mathcal{L}_2)$-Horn rewritability problem for DLs $\mathcal{L}_1$ and $\mathcal{L}_2$ is to decide whether a given $\mathcal{L}_1$-ontology admits a rewriting expressed in Horn-$\mathcal{L}_2$.*

Our first result establishes undecidability whenever the input ontology contains at-most cardinality restrictions and thus equality. This result fits in with the related undecidability results by Bienvenu et al. [3] and Lutz and Wolter [17] for Datalog rewritability and non-uniform data complexity for $\mathcal{ALCF}$ ontologies.

Theorem 3. *$(\mathcal{L}_1, \mathcal{L}_2)$-Horn rewritability is undecidable for $\mathcal{L}_1 = \mathcal{ALCF}$ and $\mathcal{L}_2$ any DL between $\mathcal{ELU}$ and $\mathcal{ALCHIF}$. This result holds if* PTIME≠NP.

Intractability results in data complexity rely on the ability of non-Horn DLs to encode co-NP-hard problems, such as non-3-colourability [14, 9]. In practice, however, it can be expected that ontologies do not encode such problems. Thus, our focus from now onwards will be on identifying classes of ontologies that admit (polynomial size) Horn rewritings.

4 Program Markability and Transposition

In this section, we introduce the class of *markable* programs and show that every markable program can be rewritten into a Horn program by means of a polynomial transformation, which we refer to as *transposition*. Roughly speaking, transposition inverts the rules in a program $\mathcal{P}$ by moving certain atoms from head to body and vice versa while replacing their corresponding predicates with fresh ones. Markability of $\mathcal{P}$ ensures that we can pick a set of predicates (a *marking*) such that, by shuffling only atoms with a marked predicate, we obtain a Horn rewriting of $\mathcal{P}$. Our results in this section generalise the results by Kaminski et al. [11] for Disjunctive Datalog to first-order programs with function symbols.

To illustrate our definitions throughout this section, we use an example program $\mathcal{P}_{\mathsf{ex}}$ consisting of the following rules:

$$A(x) \to B(x) \qquad B(x) \to C(x) \vee D(x)$$
$$C(x) \to \bot(x) \qquad D(x) \to C(f(x))$$

Markability. The notion of markability involves a partitioning of a program's predicates into *Horn* and *disjunctive*: the extension of Horn predicates for all

datasets depends only on the Horn rules in the program while the extension of disjunctive predicates may depend on a disjunctive rule. This intuition can be formalised using the standard notion of a dependency graph in logic programming.

Definition 4. *The* dependency graph $G_{\mathcal{P}} = (V, E, \mu)$ *of a program* $\mathcal{P}$ *is the smallest edge-labeled digraph such that: (i)* V *contains all predicates in* $\mathcal{P}$*; (ii)* $r \in \mu(P, Q)$ *whenever* $r \in \mathcal{P}$*,* P *is in the body of* r*, and* Q *is in the head of* r*; and (iii)* $(P, Q) \in E$ *whenever* $\mu(P, Q) \neq \emptyset$*. A predicate* Q depends on $r \in \mathcal{P}$ *if* $G_{\mathcal{P}}$ *has a path ending in* Q *and involving an* r*-labeled edge. Predicate* Q *is* Horn *if it depends only on Horn rules; otherwise,* Q *is* disjunctive*.*

For instance, predicates C, D, and $\bot$ are disjunctive in our example program $\mathcal{P}_{\mathsf{ex}}$, whereas A and B are Horn. We can now introduce the notion of a *marking*—a subset of the disjunctive predicates in a program $\mathcal{P}$ ensuring that the transposition of $\mathcal{P}$ where only marked atoms are shuffled between head and body results in a Horn program.

Definition 5. *A* marking *of a program* $\mathcal{P}$ *is a set* M *of disjunctive predicates in* $\mathcal{P}$ *satisfying the following properties, where we say that an atom is marked if its predicate is in* M*: (i) each rule in* $\mathcal{P}$ *has at most one marked body atom; (ii) each rule in* $\mathcal{P}$ *has at most one unmarked head atom; and (iii) if* $Q \in M$ *and* P *is reachable from* Q *in* $G_{\mathcal{P}}$*, then* $P \in M$*. A program is* markable *if it has a marking.*

Condition *(i)* in Def. 5 ensures that at most one atom is moved from body to head during transposition. Condition *(ii)* ensures that all but possibly one head atom are moved to the body. Finally, condition *(iii)* requires that all predicates depending on a marked predicate are also marked. We can observe that our example program $\mathcal{P}_{\mathsf{ex}}$ admits two markings: $M_1 = \{C, \bot\}$ and $M_2 = \{C, D, \bot\}$.

Markability can be efficiently checked via a 2-SAT reduction, where we assign to each predicate Q in $\mathcal{P}$ a variable X_Q and encode the constraints in Def. 5 as 2-clauses. For each rule $\varphi \wedge \bigwedge_{i=1}^{n} P_i(\boldsymbol{s}_i) \rightarrow \bigvee_{j=1}^{m} Q_j(\boldsymbol{t}_j)$, with φ the conjunction of all Horn atoms in the head, we include clauses *(i)* $\neg X_{P_i} \vee \neg X_{P_j}$ for all $1 \leq i < j \leq n$, which enforce at most one body atom to be marked; *(ii)* $X_{Q_i} \vee X_{Q_j}$ for $1 \leq i < j \leq m$, which ensure that at most one head atom is unmarked; and *(iii)* $\neg X_{P_i} \vee X_{Q_j}$ for $1 \leq i \leq n$ and $1 \leq j \leq m$, which close markings under rule dependencies. Each model of the resulting clauses yields a marking of $\mathcal{P}$.

Transposition. Before defining transposition, we illustrate the main intuitions using program $\mathcal{P}_{\mathsf{ex}}$ and marking M_1.

The first step to transpose $\mathcal{P}_{\mathsf{ex}}$ is to introduce fresh unary predicates $\overline{C}$ and $\overline{\bot}$, which stand for the negation of the marked predicates C and $\bot$. To capture the intended meaning of these predicates, we introduce rules $X(x) \rightarrow \overline{\bot}(x)$ for $X \in \{A, B, C, D\}$ and a rule $\overline{\bot}(x) \rightarrow \overline{\bot}(f(x))$ for the unique function symbol f in $\mathcal{P}_{\mathsf{ex}}$. The first rules mimick the usual axiomatisation of $\top$ and ensure that an atom $\overline{\bot}(c)$ holds in a Herbrand model of the transposed program whenever $X(c)$ also holds. The last rule ensures that $\overline{\bot}$ holds for all terms in the Herbrand universe of the transposed program—an additional requirement that is consistent with the intended meaning of $\overline{\bot}$, and critical to the completeness of transposition

in the presence of function symbols. Finally, a rule $\overline{\bot}(z) \land C(x) \land \overline{C}(x) \to \bot(z)$ ensures that the fresh predicate $\overline{C}$ behaves like the negation of C.

The key step of transposition is to invert the rules involving the marked predicates by shuffling marked atoms between head and body while replacing their predicate with the corresponding fresh one. In this way, rule $B(x) \to C(x) \lor D(x)$ yields $B(x) \land \overline{C}(x) \to D(x)$, and $C(x) \to \bot(x)$ yields $\overline{\bot}(x) \to \overline{C}(x)$. Additionally, rule $D(x) \to C(f(x))$ is transposed as $\overline{\bot}(z) \land D(x) \land \overline{C}(f(x)) \to \bot(z)$ to ensure safety. Finally, transposition does not affect rules containing only Horn predicates, e.g., rule $A(x) \to B(x)$ is included unchanged.

Definition 6. *Let M be a marking of a program $\mathcal{P}$. For each disjunctive predicate P in $\mathcal{P}$, let $\overline{P}$ be a fresh predicate of the same arity. The M-transposition of $\mathcal{P}$ is the smallest program $\Xi_M(\mathcal{P})$ containing every rule in $\mathcal{P}$ involving only Horn predicates and all rules given next, where φ is the conjunction of all Horn atoms in a rule, $\varphi_\top$ is the least conjunction of $\overline{\bot}$-atoms making a rule safe:*

1. $\varphi_\top \land \varphi \land \bigwedge_{j=1}^{m} Q_j(\boldsymbol{t}_j) \land \bigwedge_{i=1}^{n} \overline{P}_i(\boldsymbol{s}_i) \to \overline{Q}(\boldsymbol{t})$ *for each rule in $\mathcal{P}$ of the form* $\varphi \land Q(\boldsymbol{t}) \land \bigwedge_{j=1}^{m} Q_j(\boldsymbol{t}_j) \to \bigvee_{i=1}^{n} P_i(\boldsymbol{s}_i)$ *where $Q(\boldsymbol{t})$ is the only marked body atom;*
2. $\overline{\bot}(x) \land \varphi \land \bigwedge_{j=1}^{m} Q_j(\boldsymbol{t}_j) \land \bigwedge_{i=1}^{n} \overline{P}_i(\boldsymbol{s}_i) \to \bot(x)$, *where x a fresh variable, for each rule in $\mathcal{P}$ of the form* $\varphi \land \bigwedge_{j=1}^{m} Q_j(\boldsymbol{t}_j) \to \bigvee_{i=1}^{n} P_i(\boldsymbol{s}_i)$, *with no marked body atoms and no unmarked head atoms;*
3. $\varphi \land \bigwedge_{j=1}^{m} Q_j(\boldsymbol{t}_j) \land \bigwedge_{i=1}^{n} \overline{P}_i(\boldsymbol{s}_i) \to P(\boldsymbol{s})$ *for each rule in $\mathcal{P}$ of the form* $\varphi \land \bigwedge_{j=1}^{m} Q_j(\boldsymbol{t}_j) \to P(\boldsymbol{s}) \lor \bigvee_{i=1}^{n} P_i(\boldsymbol{s}_i)$ *where $P(\boldsymbol{s})$ is the only unmarked head atom;*
4. $\overline{\bot}(z) \land P(\boldsymbol{x}) \land \overline{P}(\boldsymbol{x}) \to \bot(z)$ *for marked predicate P;*
5. $P(x_1, \ldots, x_n) \to \overline{\bot}(x_i)$ *for each P in $\mathcal{P}$ and $1 \leq i \leq n$;*
6. $\overline{\bot}(x_1) \land \ldots \land \overline{\bot}(x_n) \to \overline{\bot}(f(x_1, \ldots, x_n))$ *for each n-ary function symbol f in $\mathcal{P}$.*

Clearly, $\mathcal{P}_{\mathsf{ex}}$ is unsatisfiable when extended with fact $A(a)$. To see that $\Xi_{M_1}(\mathcal{P}_{\mathsf{ex}}) \cup \{A(a)\}$ is also unsatisfiable, note that $B(a)$ is derived by the unchanged rule $A(x) \to B(x)$. Fact $\overline{C}(a)$ is derived using $A(x) \to \overline{\bot}(x)$ and the transposed rule $\overline{\bot}(x) \to \overline{C}(x)$. We derive $D(a)$ using $B(x) \land \overline{C}(x) \to D(x)$. But then, to derive a contradiction we need to apply rule $\overline{\bot}(z) \land D(x) \land \overline{C}(f(x)) \to \bot(z)$, which is not possible unless we derive $\overline{C}(f(a))$. For this, we first use $\overline{\bot}(x) \to \overline{\bot}(f(x))$, which ensures that $\overline{\bot}$ holds for $f(a)$, and then $\overline{\bot}(x) \to \overline{C}(x)$.

Theorem 7. *Let M be a marking of a program $\mathcal{P}$. Then $\Xi_M(\mathcal{P})$ is a polynomial-size Horn rewriting of $\mathcal{P}$.*

It follows that every markable set of non-Horn clauses $\mathcal{N}$ can be polynomially transformed into a set of Horn clauses $\mathcal{N}'$ such that $\mathcal{N} \cup \mathcal{D}$ and $\mathcal{N}' \cup \mathcal{D}$ are equisatisfiable for every set of facts $\mathcal{D}$. This result is rather general and has potential applications in first-order theorem proving, as well as in knowledge compilation, where Horn clauses are especially relevant [5, 6].

5 Markability of DL Ontologies

The notion of markability is applicable to first-order programs and hence can be seamlessly adapted to ontologies via the standard translation π in Table 1.

Ontology $\mathcal{O}_{\mathsf{ex}}$	Rule translation $\xi(\mathcal{O}_{\mathsf{ex}})$	Transposition $\Xi_{M_{\mathsf{ex}}}(\xi(\mathcal{O}_{\mathsf{ex}}))$	Horn DL rewriting
$\alpha_1\ A \sqsubseteq B \sqcup C$	$A(x) \rightarrow B(x) \vee C(x)$	$A(x) \wedge \overline{B}(x) \rightarrow C(x)$	$A \sqcap \overline{B} \sqsubseteq C$
$\alpha_2\ B \sqsubseteq \exists R.D$	$B(x) \rightarrow D(f_{R,D}(x))$	$\overline{D}(f_{R,D}(x)) \rightarrow \overline{B}(x)$	$\exists R_D.\overline{D} \sqsubseteq \overline{B}$
$\alpha_3\ \exists R.D \sqsubseteq D$	$R(x,y) \wedge D(y) \rightarrow D(x)$	$R(x,y) \wedge \overline{D}(x) \rightarrow \overline{D}(y)$	$\overline{D} \sqsubseteq \forall R.\overline{D}$
	$D(f_{R,D}(x)) \rightarrow D(x)$	$\overline{D}(x) \rightarrow \overline{D}(f_{R,D}(x))$	$\overline{D} \sqsubseteq \forall R_D\overline{D}$
	$D(f_{R,B}(x)) \rightarrow D(x)$	$\overline{D}(x) \rightarrow \overline{D}(f_{R,B}(x))$	$\overline{D} \sqsubseteq \forall R_B\overline{D}$
$\alpha_4\ C \sqsubseteq \exists R.B$	$C(x) \rightarrow B(f_{R,B}(x))$	$\overline{\bot}(z) \wedge C(x) \wedge \overline{B}(f_{R,B}(x)) \rightarrow \bot(z)$	$C \sqcap \exists R_B.\overline{B} \sqsubseteq \bot$
$\alpha_5\ D \sqcap E \sqsubseteq \bot$	$D(x) \wedge E(x) \rightarrow \bot(x)$	$E(x) \wedge \overline{\bot}(x) \rightarrow \overline{D}(x)$	$E \sqcap \overline{\bot} \sqsubseteq \overline{D}$
		$X(x) \rightarrow \overline{\bot}(x),\ X \in \{A,B,C,D,E\}$	$X \sqsubseteq \overline{\bot}$
		$R(x_1,x_2) \rightarrow \overline{\bot}(x_i),\ 1 \leq i \leq 2$	$\top \sqsubseteq \forall R.\overline{\bot},\ \exists R.\top \sqsubseteq \overline{\bot}$
		$\overline{\bot}(x) \rightarrow \overline{\bot}(f_{R,Y}(x)),\ Y \in \{B,D\}$	$\overline{\bot} \sqsubseteq \exists R_Y.\overline{\bot}$

Table 2. Rewriting the example $\mathcal{ELU}$ ontology $\mathcal{O}_{\mathsf{ex}}$ into a Horn-$\mathcal{ALC}$ ontology using the marking $M_{\mathsf{ex}} = \{B, D, \bot\}$.

This, however, would be of limited value since the Horn programs resulting from transposition may not be expressible in Horn-$\mathcal{ALCHIF}$.

Consider any ontology with an axiom $\exists R.A \sqsubseteq B$ and any marking M involving R. Rule $R(x,y) \wedge A(y) \rightarrow B(x)$ stemming from π would be transposed as $\overline{B}(x) \wedge A(y) \rightarrow \overline{R}(x,y)$, which cannot be captured in $\mathcal{ALCHIF}$.[1]

To address this limitation we introduce an alternative translation ξ from DL axioms into rules, which we illustrate using the example ontology $\mathcal{O}_{\mathsf{ex}}$ in Table 2. The key idea is to encode existential restrictions in axioms T3 as unary atoms over functional terms. For instance, axiom α_2 in $\mathcal{O}_{\mathsf{ex}}$ would yield $B(x) \rightarrow D(f_{R,D}(x))$, where the "successor" relation between an instance b of B and some instance of D in a Herbrand model is encoded as a term $f_{R,D}(b)$, instead of a binary atom of the form $R(b, g(b))$. This encoding has an immediate impact on markings: by marking B we are only forced to also mark D (rather than both R and D). In this way, we will ensure that markings consist of unary predicates only.

To compensate for the lack of binary atoms involving functional terms in Herbrand models, we introduce new rules when translating axioms T2, T4, and T6 using ξ. For instance, $\xi(\alpha_3)$ yields the following rules in addition to $\pi(\alpha_3)$: a rule $D(f_{R,D}(x)) \rightarrow D(x)$ to ensure that all objects c with an R-successor $f_{R,D}(c)$ generated by $\xi(\alpha_2)$ are instances of D; a rule $D(f_{R,B}(x)) \rightarrow D(x)$, which makes sure that an object whose R-successor generated by $\xi(\alpha_4)$ is an instance of D is also an instance of D. Finally, axioms α_1 and α_5, which involve no binary predicates, are translated as usual.

Definition 8. *Let $\mathcal{O}$ be an ontology. For each concept $\exists R.B$ in an axiom of type T3, let $f_{R,B}$ be a unary function symbol, and Φ the set of all such symbols. We define $\xi(\mathcal{O})$ as the smallest program containing $\pi(\alpha)$ for each axiom α of type T1–T2 and T4–T6, as well as the following rules:*

– *$A(x) \rightarrow B(f_{R,B}(x))$ for each axiom T3;*

[1] Capturing such a rule would require a DL that can express products of concepts [20].

– $A(f_{R',Y}(x)) \to C(x)$ *for each axiom T2 and* R', Y *s.t.* $f_{R',Y} \in \Phi$ *and* $R' \sqsubseteq^* R$*;*
– $A(f_{\mathsf{inv}(R'),Y}(x)) \to C(x)$ *for each ax. T4 and* R', Y *s.t.* $f_{\mathsf{inv}(R'),Y} \in \Phi$*,* $R' \sqsubseteq^* R$*;*
– $A(x) \wedge Y(f_{\mathsf{inv}(R'),Y}(x)) \to C(f_{\mathsf{inv}(R'),Y}(x))$ *for each axiom T2 and* R', Y *s.t.* $f_{\mathsf{inv}(R'),Y} \in \Phi$ *and* $R' \sqsubseteq^* R$*;*
– $A(x) \wedge Y(f_{R',Y}(x)) \to C(f_{R',Y}(x))$ *for each axiom T4 and* R', Y *s.t.* $f_{R',Y} \in \Phi$ *and* $R' \sqsubseteq^* R$*;*
– $A(z) \wedge B(f_{R',Y}(z)) \wedge \mathsf{at}(R, z, x) \wedge B(x) \to f_{R',Y}(z) \approx x$ *for each ax. T6 and* R', Y *s.t.* $f_{R',Y} \in \Phi$ *and* $R' \sqsubseteq^* R$*;*
– $A(f_{\mathsf{inv}(R'),Y}(x)) \wedge B(x) \wedge \mathsf{at}(R, f_{\mathsf{inv}(R'),Y}(x), y) \wedge B(y) \to x \approx y$ *for each axiom T6 and* R', Y *s.t.* $f_{\mathsf{inv}(R'),Y} \in \Phi$ *and* $R' \sqsubseteq^* R$*;*
– $A(z) \wedge B(f_{R'_1,Y_1}(z)) \wedge B(f_{R'_2,Y_2}(z)) \to f_{R'_1,Y_1}(z) \approx f_{R'_2,Y_2}(z)$ *for each axiom T6 and* $f_{R'_i,Y_i} \in \Phi$ *s.t.* $R'_i \sqsubseteq^* R$*;*
– $A(f_{\mathsf{inv}(R'_1),Y_1}(x)) \wedge B(x) \wedge B(f_{R'_2,Y_2}(f_{\mathsf{inv}(R'_1),Y_1}(x))) \to x \approx f_{R'_2,Y_2}(f_{\mathsf{inv}(R'_1),Y_1}(x))$ *for each axiom T6 and* R'_i, Y_i *s.t.* $\{f_{\mathsf{inv}(R'_1),Y_1}, f_{R'_2,Y_2}\} \subseteq \Phi$ *and* $R'_i \sqsubseteq^* R$*.*

The translation $\xi(\mathcal{O}_{\mathsf{ex}})$ of our example ontology $\mathcal{O}_{\mathsf{ex}}$ is given in the second column of Table 2. Clearly, $\mathcal{O}_{\mathsf{ex}}$ is unsatisfiable when extended with $A(a)$ and $E(a)$. We can check that $\xi(\mathcal{O}_{\mathsf{ex}}) \cup \{A(a), E(a)\}$ is also unsatisfiable.

Theorem 9. *For every ontology* $\mathcal{O}$ *and dataset* $\mathcal{D}$ *over predicates in* $\mathcal{O}$ *we have that* $\mathcal{O} \cup \mathcal{D}$ *is satisfiable iff so is* $\xi(\mathcal{O}) \cup \mathcal{D}$*.*

This translation has a clear benefit for markability checking: in contrast to $\pi(\mathcal{O})$, binary predicates in $\xi(\mathcal{O})$ do not belong to any minimal marking. In particular, $M_{\mathsf{ex}} = \{B, D, \bot\}$ is the only minimal marking of $\xi(\mathcal{O}_{\mathsf{ex}})$.

Proposition 10. *(i) If* $\approx$ *is Horn in* $\xi(\mathcal{O})$ *then so are all binary predicates in* $\xi(\mathcal{O})$*. (ii) If* $\xi(\mathcal{O})$ *is markable, it has a marking containing only unary predicates.*

Thus, we define markability of ontologies in terms of ξ rather than π. We can check that $\pi(\mathcal{O}_{\mathsf{ex}})$ is not markable, whereas $\xi(\mathcal{O}_{\mathsf{ex}})$ admits the marking M_{ex}.

Definition 11. *An ontology* $\mathcal{O}$ *is markable if so is* $\xi(\mathcal{O})$*.*

We conclude this section with the observation that markability of an ontology $\mathcal{O}$ can be efficiently checked by first computing the program $\xi(\mathcal{O})$ and then exploiting the 2-SAT encoding sketched in Section 4.

6 Rewriting Markable Ontologies

It follows from the correctness of transposition in Theorem 7 and ξ in Theorem 9 that every $\mathcal{ALCHIF}$ ontology $\mathcal{O}$ admitting a marking M has a Horn rewriting of polynomial size given as the program $\Xi_M(\xi(\mathcal{O}))$. In what follows, we show that this rewriting can be expressed within Horn-$\mathcal{ALCHIF}$.

Let us consider the transposition of $\xi(\mathcal{O}_{\mathsf{ex}})$ via the marking M_{ex}, which is given in the third column of Table 2. The transposition of α_1 and α_5 corresponds directly to DL axioms via the standard translation in Table 1. In contrast, the transposition of all other axioms leads to rules that have no direct correspondence in DLs. The following lemma establishes that the latter rules are restricted to the types T7–T20 specified on the left-hand side of Table 3.

$T7.$	$\overline{\bot}(z) \land B(x) \land R(x,y) \land A(y) \to \bot(z)$	$B \sqcap \exists R.A \sqsubseteq \bot$
$T8.$	$\overline{\bot}(z) \land A(f_{R,Y}(x)) \land B(x) \to \bot(z)$	$B \sqcap \exists R_Y.A \sqsubseteq \bot$
$T9.$	$\overline{\bot}(x) \to \overline{\bot}(f_{R,Y}(x))$	$\overline{\bot} \sqsubseteq \exists R_Y.\overline{\bot}$
$T10.$	$B(x) \to A(f_{R,Y}(x))$	$B \sqsubseteq \forall R_Y.A \quad$ if $A \neq \overline{\bot}$ or $B \neq \overline{\bot}$
$T11.$	$B(f_{R,Y}(x)) \to A(x)$	$\exists R_Y.B \sqsubseteq A$
$T12.$	$A(x) \land B(f_{R,Y}(x)) \to C(f_{R,Y}(x))$	$A \sqcap \exists R_Y.B \sqsubseteq \forall R_Y.C$
$T13.$	$\overline{\bot}(z) \land A(x) \land B(f_{R,Y}(x)) \land C(f_{R,Y}(x)) \to \bot(z)$	$A \sqcap \exists R_Y(B \sqcap C) \sqsubseteq \bot$
$T14.$	$B(f_{R,Y}(x)) \land C(f_{R,Y}(x)) \to A(x)$	$\exists R_Y(B \sqcap C) \sqsubseteq A$
$T15.$	$A(z) \land B(f_{R',Y}(z)) \land \mathsf{at}(R,z,x) \land B(x) \to f_{R',Y}(z) \approx x$	$R'_Y \sqsubseteq S_{\{R'_Y,R\}},\ R \sqsubseteq S_{\{R'_Y,R\}},$ $A \sqsubseteq \leq 1 S_{\{R'_Y,R\}}.B$
$T16.$	$A(f_{R',Y}(x)) \land B(x) \land \mathsf{at}(R, f_{R',Y}(x), y) \land B(y) \to x \approx y$	$\tilde{R}'_Y \sqsubseteq S_{\{\tilde{R}'_Y,R\}},\ R \sqsubseteq S_{\{\tilde{R}'_Y,R\}},$ $A \sqsubseteq \leq 1 S_{\{\tilde{R}'_Y,R\}}.B,\ \tilde{R}'_Y \equiv \mathsf{inv}(R'_Y)$
$T17.$	$A(z) \land B(f_{R,Y}(z)) \land B(f_{R',Z}(z)) \to f_{R,Y}(z) \approx f_{R',Z}(z)$	$R_Y \sqsubseteq S_{\{R_Y,R'_Z\}},\ R'_Z \sqsubseteq S_{\{R_Y,R'_Z\}},$ $A \sqsubseteq \leq 1 S_{\{R_Y,R'_Z\}}.B$
$T18.$	$A(f_{R,Y}(x)) \land B(x) \land B(f_{R',Z}(f_{R,Y}(x))) \to x \approx f_{R',Z}(f_{R,Y}(x))$	$\tilde{R}_Y \sqsubseteq S_{\{\tilde{R}_Y,R'_Z\}},\ R'_Z \sqsubseteq S_{\{\tilde{R}_Y,R'_Z\}},$ $A \sqsubseteq \leq 1 S_{\{\tilde{R}_Y,R'_Z\}}.B,\ \tilde{R}_Y \equiv \mathsf{inv}(R_Y)$
$T19.$	$R(x,y) \to \overline{\bot}(x)$	$\exists R.\top \sqsubseteq \overline{\bot}$
$T20.$	$R(x,y) \to \overline{\bot}(y)$	$\top \sqsubseteq \forall R.\overline{\bot}$

Table 3. Transformation Ψ from transposed rules to DLs. Role names $\tilde{R}$ are fresh for every R, and $S_{\{R,R'\}}$ for every $\{R, R'\}$.

Lemma 12. *Let $\mathcal{O}$ be an ontology and M a minimal marking of $\xi(\mathcal{O})$. Then $\Xi_M(\xi(\mathcal{O}))$ contains only Horn rules of type T1–T2 and T4–T6 in Table 1 as well as type T7–T20 in Table 3.*

We can now specify a transformation Ψ that allows us to translate rules T7–T20 in Table 3 back into DL axioms.

Definition 13. *We define Ψ as the transformation mapping (i) each Horn rule r of type T1–T2 and T4–T6 in Table 1 to the DL axiom $\pi^{-1}(r)$ (ii) each rule T7–T20 on the left-hand side of Table 3 to the axioms on the right-hand side.*[2]

Intuitively, Ψ works as follows: *(i)* Function-free rules are "rolled up" as usual into DL axioms (see e.g., T7). *(ii)* Unary atoms $A(f_{R,Y}(x))$ with $A \neq \overline{\bot}$ that involve a functional term are translated as existentially or universally quantified concepts depending on whether they occur in the body or in the head (e.g., T10, T11); in contrast, atoms $\overline{\bot}(f_{R,Y}(x))$ in rules $\overline{\bot}(x) \to \overline{\bot}(f_{R,Y}(x))$ are translated as $\exists R_Y.\overline{\bot}$ instead of $\forall R_Y.\overline{\bot}$ (T9). *(iii)* Rules T15–T18, which involve $\approx$ in the head and roles R' and R in the body, are rolled back into axioms of type T6 over the "union" of R and R', which is captured using fresh roles and role inclusions.

The ontology obtained by applying Ψ to our running example is given in the last column of Table 2. Correctness of Ψ and its implications for the computation of Horn rewritings are summarised in the following lemma.

[2] For succinctness, axioms resulting from T7, T8, T12, T13, T14, T16 and T18 are not given in normal form.

Lemma 14. *Let $\mathcal{O}$ be a markable $\mathcal{ALCHIF}$ ontology and let M be a marking of $\mathcal{O}$. Then the ontology $\Psi(\Xi_M(\xi(\mathcal{O})))$ is a Horn rewriting of $\mathcal{O}$.*

A closer look at our transformations reveals that our rewritings do not introduce constructs such as inverse roles and cardinality restrictions if these were not already present in the input ontology. In contrast, fresh role inclusions may originate from cardinality restrictions in the input ontology. As a result, our approach is language-preserving: if the input $\mathcal{O}_1$ is in a DL $\mathcal{L}_1$ between $\mathcal{ALC}$ and $\mathcal{ALCHI}$, then its rewriting $\mathcal{O}_2$ stays in the Horn fragment of $\mathcal{L}_1$; furthermore, if $\mathcal{L}_1$ is between $\mathcal{ALCF}$ and $\mathcal{ALCIF}$, then $\mathcal{O}_2$ may contain fresh role inclusions ($\mathcal{H}$). A notable exception is when $\mathcal{O}_1$ is an $\mathcal{ELU}$ ontology, in which case axioms T2 and T3 in $\mathcal{O}_1$ may yield axioms of type T4 in $\mathcal{O}_2$. The following theorem follows from these observations and Lemma 14.

Theorem 15. *Every markable $\mathcal{L}$ ontology is polynomially Horn-$\mathcal{L}$ rewritable whenever $\mathcal{L}$ is between $\mathcal{ALC}$ and $\mathcal{ALCHI}$. If $\mathcal{L}$ is between $\mathcal{ALCF}$ and $\mathcal{ALCHIF}$, every markable $\mathcal{L}$ ontology is polynomially rewritable into Horn-$\mathcal{LH}$. Finally, every markable $\mathcal{ELU}$ ontology is polynomially rewritable into Horn-$\mathcal{ALC}$.*

We conclude by establishing the complexity of satisfiability checking over markable ontologies. We first show that the problem is ExpTime-hard for markable $\mathcal{ELU}$ ontologies, which implies that it is not possible to polynomially rewrite every markable $\mathcal{ELU}$ ontology into $\mathcal{EL}$. Thus, our rewriting approach is optimal for $\mathcal{ELU}$ in the sense that introducing universal restrictions (or equivalently inverse roles) in the rewriting is unavoidable.

Lemma 16. *Satisfiability checking over markable $\mathcal{ELU}$ is* ExpTime*-hard.*

All Horn DLs from $\mathcal{ALC}$ to $\mathcal{ALCHIF}$ are ExpTime-complete in combined complexity and PTime-complete in data complexity [15]. By Theorem 15, the same holds for markable ontologies in DLs from $\mathcal{ALC}$ to $\mathcal{ALCHIF}$. Finally, Lemma 16 shows that these results extend to markable $\mathcal{ELU}$ ontologies.

Theorem 17. *Let $\mathcal{L}$ be in-between $\mathcal{ELU}$ and $\mathcal{ALCHIF}$. Satisfiability checking over markable $\mathcal{L}$-ontologies is* ExpTime*-complete and* PTime*-complete in data.*

7 Related Work

Horn logics are common target languages for knowledge compilation [5]. Selman and Kautz [21] proposed an algorithm for compiling a set of propositional clauses into a set of Horn clauses s.t. their Horn consequences coincide. This approach was generalised to FOL by Del Val [6], without termination guarantees.

Bienvenu et al. [3] showed undecidability of Datalog rewritability for $\mathcal{ALCF}$ and decidability in NExpTime for $\mathcal{SHI}$. Cuenca Grau et al. [4] and Kaminski et al. [11] proposed practical techniques for computing Datalog rewritings of $\mathcal{SHI}$ ontologies based on a two-step process. First, $\mathcal{O}$ is rewritten using a resolution calculus Ω into a Disjunctive Datalog program $\Omega(\mathcal{O})$ of exponential

size [10]. Second, $\Omega(\mathcal{O})$ is rewritten into a Datalog program $\mathcal{P}$. For the second step, Kaminski et al. [11] propose the notion of markability of a Disjunctive Datalog program and show that $\mathcal{P}$ can be polynomially computed from $\Omega(\mathcal{O})$ using transposition whenever $\Omega(\mathcal{O})$ is markable. In contrast to our work, Kaminski et al. [11] focus on Datalog as target language for rewriting (rather than Horn DLs). Furthermore, their Datalog rewritings may be exponential w.r.t. the input ontology and cannot generally be represented in DLs.

Gottlob et al. [8] showed tractability in data complexity of fact entailment for the class of first-order rules with single-atom bodies, which is sufficient to capture most DLs in the DL-Lite$_{\mathsf{bool}}$ family [1].

Lutz and Wolter [17] investigated (non-uniform) data complexity of query answering w.r.t. *fixed* ontologies. They studied the boundary of PTime and co-NP-hardness and established a connection with constraint satisfaction problems. Finally, Lutz et al. [16] studied model-theoretic rewritability of ontologies in a DL $\mathcal{L}_1$ into a fragment $\mathcal{L}_2$ of $\mathcal{L}_1$. These rewritings are equivalence-preserving; this is in contrast to our approach, which requires only satisfiability preservation.

8 Proof of Concept

To assess the practical implications of our results, we have evaluated whether real-world ontologies are markable (and hence polynomially Horn rewritable). We analysed 120 non-Horn ontologies extracted from the Protege Ontology Library, BioPortal (http://bioportal.bioontology.org/), the corpus by Gardiner et al. [7], and the EBI linked data platform (http://www.ebi.ac.uk/rdf/platform). To check markability, we have implemented the 2-SAT reduction in Section 4 and a simple 2-SAT solver.

We found that a total of 32 ontologies were markable and thus rewritable into a Horn ontology, including some ontologies commonly used in applications, such as ChEMBL (see http://www.ebi.ac.uk/rdf/services/chembl/) and BioPAX Reactome (http://www.ebi.ac.uk/rdf/services/reactome/). When using π as first-order logic translation, we obtained 30 markable ontologies—a strict subset of the ontologies markable using ξ. However, only 27 ontologies were rewritable to a Horn DL since in three cases the marking contained a role.

9 Conclusion and Future Work

We have presented the first practical technique for rewriting non-Horn ontologies into a Horn DL. Our rewritings are polynomial, and our experiments suggest that they are applicable to widely-used ontologies. We anticipate several directions for future work. First, we would like to conduct an extensive evaluation to assess whether the use of our rewritings can significantly speed up satisfiability checking in practice. Second, we will investigate relaxations of markability that would allow us to capture a wider range of ontologies.

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. Baader, F., Calvanese, D., McGuinness, D.L., Nardi, D., Patel-Schneider, P.F.: The Description Logic Handbook: Theory, Implementation, and Applications. Cambridge University Press (2003)
3. Bienvenu, M., ten Cate, B., Lutz, C., Wolter, F.: Ontology-based data access: A study through disjunctive datalog, CSP, and MMSNP. ACM Tans. Database Syst. 39(4), 33 (2014)
4. Cuenca Grau, B., Motik, B., Stoilos, G., Horrocks, I.: Computing datalog rewritings beyond Horn ontologies. In: IJCAI. pp. 832–838 (2013)
5. Darwiche, A., Marquis, P.: A knowledge compilation map. J. Artif. Intell. Res. 17, 229–264 (2002)
6. Del Val, A.: First order LUB approximations: Characterization and algorithms. Artif. Intell. 162(1-2), 7–48 (2005)
7. Gardiner, T., Tsarkov, D., Horrocks, I.: Framework for an automated comparison of description logic reasoners. In: ISWC. pp. 654–667 (2006)
8. Gottlob, G., Manna, M., Morak, M., Pieris, A.: On the complexity of ontological reasoning under disjunctive existential rules. In: MFCS. pp. 1–18 (2012)
9. Hustadt, U., Motik, B., Sattler, U.: Data complexity of reasoning in very expressive description logics. In: IJCAI. pp. 466–471 (2005)
10. Hustadt, U., Motik, B., Sattler, U.: Reasoning in description logics by a reduction to disjunctive datalog. J. Autom. Reasoning 39(3), 351–384 (2007)
11. Kaminski, M., Nenov, Y., Cuenca Grau, B.: Computing datalog rewritings for disjunctive datalog programs and description logic ontologies. In: RR. pp. 76–91 (2014)
12. Kaminski, M., Nenov, Y., Cuenca Grau, B.: Datalog rewritability of disjunctive datalog programs and its applications to ontology reasoning. In: AAAI. pp. 1077–1083 (2014)
13. Kazakov, Y.: RIQ and SROIQ are harder than SHOIQ. In: KR. pp. 274–284 (2008)
14. Krisnadhi, A., Lutz, C.: Data complexity in the $\mathcal{EL}$ family of description logics. In: LPAR. pp. 333–347 (2007)
15. Krötzsch, M., Rudolph, S., Hitzler, P.: Complexities of Horn description logics. ACM Trans. Comput. Log. 14(1) (2013)
16. Lutz, C., Piro, R., Wolter, F.: Description logic TBoxes: Model-theoretic characterizations and rewritability. In: IJCAI. pp. 983–988 (2011)
17. Lutz, C., Wolter, F.: Non-uniform data complexity of query answering in description logics. In: KR. pp. 297–307 (2012)
18. Ortiz, M., Rudolph, S., Simkus, M.: Query answering in the Horn fragments of the description logics $\mathcal{SHOIQ}$ and $\mathcal{SROIQ}$. In: IJCAI. pp. 1039–1044 (2011)
19. Robinson, A., Voronkov, A. (eds.): Handbook of Automated Reasoning. Elsevier (2001)
20. Rudolph, S., Krötzsch, M., Hitzler, P.: All elephants are bigger than all mice. In: DL (2008)
21. Selman, B., Kautz, H.: Knowledge compilation and theory approximation. J. ACM 43(2), 193–224 (1996)