Semantic Web 6 (2015) 63–79
DOI 10.3233/SW-130126
IOS Press

A closer look at the semantic relationship between Datalog and description logics

Editor(s): Diego Calvanese, Free University of Bozen-Bolzano, Italy; Thomas Lukasiewicz, University of Oxford, UK
Solicited review(s): Riccardo Rosati, Sapienza Università di Roma, Italy; Marie-Christine Rousset,University of Grenoble, France; anonymous reviewer

Markus Krötzsch [a,*], Sebastian Rudolph [b] and Peter H. Schmitt [c,**]
[a] *Department of Computer Science, University of Oxford, UK*
E-mail: markus.kroetzsch@cs.ox.ac.uk
[b] *Institute of Artificial Intelligence, Technische Universität Dresden, Germany*
E-mail: sebastian.rudolph@tu-dresden.de
[c] *Institute for Theoretical Computer Science, Karlsruhe Institute of Technology, Germany*
E-mail: pschmitt@ira.uka.de

Abstract. Translations to (first-order) Datalog have been used in a number of inferencing techniques for description logics (DLs), yet the relationship between the semantic expressivities of function-free Horn logic and DL is understood only poorly. Although Description Logic Programs (DLP) have been described as DLs in the "expressive intersection" of DL and Datalog, it is unclear what an intersection of two syntactically incomparable logics is, even if both have a first-order logic semantics. In this work, we offer a characterisation for DL fragments that can be expressed, in a concrete sense, in Datalog. We then determine the largest such fragment for the DL $\mathcal{ALC}$, and provide an outlook on the extension of our methods to more expressive DLs.

Keywords: Description logic programs, OWL RL, conservative extension, knowledge representation and reasoning

1. Introduction

Ontologies and *rules* are two major paradigms of knowledge representation and reasoning. Both have been successfully applied in many areas, ranging from logic programming [5] over databases [1] to the Semantic Web [14]. In spite of conceptual and technical differences, both areas are overlapping in many places, and the combination of their respective strengths is a worthwhile and established field of research.

Ontological approaches are most commonly based on the logical framework of description logics (DLs) [3]. In particular, they are the basis for the Direct Semantics of the OWL ontology language [34]. Technically, DLs are a family of fragments of first-order logic, with different DLs obtained by including or excluding expressive features in order to obtain favourable decidability or complexity properties for common reasoning tasks. Formulae of DL (called *axioms*) are commonly denoted in a variable-free syntax. For example, the following set of DL axioms expresses that every supervisor of a student is a professor (1), every professor holds some PhD degree (2), and all professors are either full or associate professors (3):

$$\mathsf{Student} \sqsubseteq \forall \mathsf{supervisor}.\mathsf{Prof}, \tag{1}$$

$$\mathsf{Prof} \sqsubseteq \exists \mathsf{hasDegree}.\mathsf{PhD}, \tag{2}$$

$$\mathsf{Prof} \sqsubseteq \mathsf{FullProf} \sqcup \mathsf{AssociateProf}. \tag{3}$$

*Corresponding author.
**An earlier version of this paper, entitled "On the Semantic Relationship between Datalog and Description Logics," has been published in the proceedings of the 4th International Conference on Web Reasoning and Rule Systems (RR 2010), Springer 2010.

This example corresponds to the following first-order logic theory:

$$\forall x.\mathsf{Student}(x) \rightarrow (\forall y.\mathsf{supervisor}(x,y) \rightarrow \mathsf{Prof}(y)), \quad (4)$$

$$\forall x.\mathsf{Prof}(x) \rightarrow \exists y.\mathsf{hasDegree}(x,y) \wedge \mathsf{PhD}(y), \quad (5)$$

$$\forall x.\mathsf{Prof}(x) \rightarrow \mathsf{FullProf}(x) \vee \mathsf{AssociateProf}(x). \quad (6)$$

Rule-based approaches are rooted in deductive databases [1] and logic programming [5]. The common core of these fields are function-free Horn logic rules, known as *Datalog* in the context of deductive databases. Datalog rules are one of the simplest forms of logical rules. For example, the following example states that students are supervised by professors as above (7), and that a supervisor who reviews a paper authored by her student has a conflict of interest (8):

$$\mathsf{Student}(x) \wedge \mathsf{supervisor}(x,y) \rightarrow \mathsf{Prof}(y), \quad (7)$$

$$\mathsf{hasAuthor}(x,y) \wedge \mathsf{hasReviewer}(x,z) \wedge \mathsf{supervisor}(y,z) \rightarrow \mathsf{conflict}(z). \quad (8)$$

Such rules can be interpreted as implications under the semantics of first-order logic or under a least (Herbrand) model semantics that can be axiomatised in second-order logic. Fortunately, both semantics entail the same Datalog formulae [1], and in particular the same ground facts. In this work, we will therefore study Datalog under a first-order semantics that is compatible with DLs.

A natural question to ask is how DLs and Datalog – viewed as decidable fragments of first-order logic – relate to each other. One direction of research explores how either formalism could be extended with features of the other. Approaches to extending the expressivity of DLs with first-order rules include $\mathcal{AL}$-log [9], CARIN [26], SWRL [15,16], $\mathcal{DL}$+*log* [35], DL-safe rules [31], and DL Rules [12,24].[1] A dual approach is to extended Datalog with typical DL features, in particular with existential quantification, which results in formalisms such as Datalog$^{\pm}$ [6], $\forall\exists$-rules [4], and various related fragments of existential rules; see [22,32] for recent overviews.

These manifold research activities are based on the observation that DL and Datalog have distinct modelling capabilities that are not easily reconciled in a single formalism without sacrificing useful computational properties. For example, DLs feature existential quantification (2) and disjunction (3), while Datalog can capture dependency structures that are not expressible in DL axioms (8). However, DL and Datalog also have some overlapping expressivity. Formulae (1) and (7), e.g., are semantically equivalent. DLP ("Description Logic Programs") has been proposed as a family of DLs that can be faithfully expressed in first-order Horn-logic, and in particular in Datalog [13,36]. This bears computational advantages since rule-based reasoning methods can be applied, and indeed DLP in its simplest form became the basis of the W3C standard OWL RL [28].

This raises the core question of this paper:

> What is an appropriate exact definition of the "semantic intersection" of DL and Datalog, i.e., of a provably maximal logic that can be expressed in both?

Unfortunately, this question as such leaves room for interpretation. Due to the incomparable syntax, we cannot consider a syntactic intersection of both logics. Also when transforming DL syntax to first-order logic, the result is normally not in the form of Datalog rules, even for DL axioms that are easily expressible in such form. Neither (1) nor (4), e.g., are equal to (7) above.

Thus one needs to consider semantic criteria for defining the "intersection" of DL and Datalog. This, however, can lead to a language definition for which checking membership is of very high computational complexity. Indeed, every inconsistent ontology is semantically equivalent to an inconsistent set of Datalog rules.[2] So checking whether some DL ontology is semantically equivalent, or even merely equisatisfiable, to some set of Datalog rules is at least as hard as checking satisfiability for a DL knowledge base, i.e., typically at least ExpTime-hard.

On the other hand, restricting to DL knowledge bases that are equisatisfiable to some set of Datalog rules may still be insufficient to characterise the "intersection" of DL and Datalog. For example, it is well-known that other tractable DLs such as $\mathcal{EL}^{++}$ can also be translated into equisatisfiable sets of Datalog rules [18,20,25]. The union of DLP and $\mathcal{EL}^{++}$ is not a DL for which standard reasoning tasks are tractable (see [25] for some discussion), so DLP and $\mathcal{EL}^{++}$ may merely be two among several tractable subsets of the "expressive intersection" of DL and Datalog, without actually capturing the essence of this slogan.Indeed,

[1] Similar approaches exist for extending DLs with *non-monotonic* features from logic programming [10,11,29,30,35] which are interesting in their own right but not closely related to this work.

[2] We generally allow rules with head $\bot$, interpreted as *false*. Thus Datalog rules can be inconsistent.

tractability was not among the original design goals of DLP, although it is now considered a major practical advantage that motivated the use in OWL RL.

Could the union of DLP and $\mathcal{EL}^{++}$ then be considered as an extended version of DLP? Possibly yes, since it is contained in the DL Horn-$\mathcal{SHIQ}$ and the even more expressive Horn-$\mathcal{SHOIQ}$ for which satisfiability-preserving Datalog transformations are known [17,33]. However, for $\mathcal{EL}^{++}$ and DLP there exist *modular* (i.e., axiom-by-axiom) translations into Datalog. Opposed to this, the known Datalog transformation for Horn-$\mathcal{SHIQ}$ from [17] needs to process the whole knowledge base in an exponential compilation process to obtain the Datalog output.

The Horn-$\mathcal{SHOIQ}$ transformation described in [33] is "more modular" and time polynomial, but a closer look reveals that the signature used by the knowledge base needs to be fixed and known beforehand, whence this translation does not allow for axioms being translated independently from each other if the signature is not bounded.

How can we be sure that there is no simpler transformation given that both data complexity and combined complexity[3] of Datalog, Horn-$\mathcal{SHIQ}$, and Horn-$\mathcal{SHOIQ}$ agree? The answer is given in Proposition 4.4 later on. In any case, it is obvious from this discussion that the design principles for DLP – but also for $\mathcal{EL}$ and Horn-DLs – are not sufficiently well articulated to clarify the conceptual distinction between those formalisms.

This paper thus approaches an explicit characterisation of a maximal DLP-type logic. After introducing our basic definitions for description logics and Datalog in Section 2, we discuss what it means for a logical theory to be "expressible" in Datalog. To do this, we first develop concrete requirements for such a language, that capture the specifics of the original DLP proposal, in Section 4. The above discussion indicates that some care is needed to define such principles. Thereafter, we ask whether DLP could be defined as a larger, or even as the *largest*, DL language that satisfies our design principles. A positive answer to this question is given by defining such a "largest possible Datalog fragment" $\mathcal{DLP}_{\mathcal{ALC}}$ for the DL $\mathcal{ALC}$ in Section 5, and proving its maximality in Section 6. In Section 7, we consider the generalisation of this approach to $\mathcal{SROIQ}$, and discuss a related study by Krötzsch and Rudolph [21].

[3] Recall that *data complexity* is the complexity of reasoning with respect to the amount of facts/assertions, while *combined complexity* refers to the overall size of the input.

Table 1
Syntax and semantics of $\mathcal{ALC}$ concept expressions

	Syntax	Semantics
Atomic concept	A	$A^{\mathcal{I}}$
Top	$\top$	$\Delta^{\mathcal{I}}$
Bottom	$\bot$	$\emptyset$
Conjunction	$C \sqcap D$	$C^{\mathcal{I}} \cap D^{\mathcal{I}}$
Disjunction	$C \sqcup D$	$C^{\mathcal{I}} \cup D^{\mathcal{I}}$
Negation	$\neg C$	$\Delta^{\mathcal{I}} \setminus C^{\mathcal{I}}$
Role restrictions		
existential	$\exists R.C$	$\{x \mid \text{there is } y\text{: } \langle x,y\rangle \in R^{\mathcal{I}} \text{ and } y \in C^{\mathcal{I}}\}$
universal	$\forall R.C$	$\{x \mid \text{for all } \langle x,y\rangle \in R^{\mathcal{I}}\text{: } y \in C^{\mathcal{I}}\}$

2. Description logic and Datalog

We provide a brief introduction to our notation on description logics (DLs) [3] and Datalog [1]. We use $\mathbf{FOL}_{=}$ for referring to standard first-order logic with equality, and we use the term *theory* for a set of closed formulae (sentences) of $\mathbf{FOL}_{=}$ (or another logic that can be considered as a fragment thereof).

Description Logics DL knowledge bases are defined over finite sets of individual names (constants) $\mathbf{I}$, concept names $\mathbf{A}$, and roles $\mathbf{R}$. We call $\mathscr{S} = \langle \mathbf{I}, \mathbf{A}, \mathbf{R}\rangle$ a *signature*. A signature $\mathscr{S}' = \langle \mathbf{I}', \mathbf{A}', \mathbf{R}'\rangle$ is called an *extension* of $\mathscr{S}$, in symbols $\mathscr{S} \subseteq \mathscr{S}'$, if $\mathbf{I} \subseteq \mathbf{I}'$ and $\mathbf{A} \subseteq \mathbf{A}'$ and $\mathbf{R} \subseteq \mathbf{R}'$.

One of the most expressive DLs considered in the literature is $\mathcal{SROIQ}$ but we will only consider the simpler logic $\mathcal{ALC}$ in detail within this paper. *Concept expressions* (or simply *concepts*) of $\mathcal{ALC}$ are defined recursively as in Table 1. *Terminological axioms* (or *TBox axioms*) of $\mathcal{ALC}$ are general concept inclusions (GCIs) of the form $C \sqsubseteq D$ where C and D are $\mathcal{ALC}$ concepts. *Assertional axioms* (or *ABox axioms*) of $\mathcal{ALC}$ are expressions $C(a)$ or $R(a,b)$ where a, b are individuals, C is a concept expression, and R is a role. An $\mathcal{ALC}$ *knowledge base* is a set of (terminological and assertional) axioms of $\mathcal{ALC}$.

The semantics of DLs are based on a Tarski-style model theory. An interpretation $\mathcal{I}$ over a domain $\Delta^{\mathcal{I}}$ assigns a set $A^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$ to each atomic concept $A \in \mathbf{A}$, a binary relation $R^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$ to each role $R \in \mathbf{R}$, and an element $a^{\mathcal{I}} \in \Delta^{\mathcal{I}}$ to each individual $a \in \mathbf{I}$. The interpretation of concept expressions is defined recursively as in Table 1. A GCI $C \sqsubseteq D$ is *satisfied* by $\mathcal{I}$, written $\mathcal{I} \models C \sqsubseteq D$, it $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$. An assertion $C(a)$ $(R(a,b))$ is *satisfied* by $\mathcal{I}$, written $\mathcal{I} \models C(a)$ $(\mathcal{I} \models R(a,b))$, if $a^{\mathcal{I}} \in C^{\mathcal{I}}$ $(\langle a^{\mathcal{I}}, b^{\mathcal{I}}\rangle \in R^{\mathcal{I}})$. A knowledge base KB is satisfied by $\mathcal{I}$, written $\mathcal{I} \models \mathrm{KB}$, if $\mathcal{I} \models \alpha$ for all ax-

Table 2
Transforming $\mathcal{ALC}$ axioms to first-order logic

Concept expressions
$\pi(\top, t) = \top$
$\pi(\bot, t) = \bot$
$\pi(A, t) = A(t)$
$\pi(\neg C, t) = \neg\pi(C, t)$
$\pi(C \sqcap D, t) = \pi(C, t) \wedge \pi(D, t)$
$\pi(C \sqcup D, t) = \pi(C, t) \vee \pi(D, t)$
$\pi(\forall R.C, t) = \forall x.(R(t, x) \rightarrow \pi(C, x))$
$\pi(\exists R.C, t) = \exists x.(R(t, x) \wedge \pi(C, x))$
Axioms
$\pi(C(a)) = \pi(C, a)$
$\pi(R(a, b)) = R(a, b)$
$\pi(C \sqsubseteq D) = \forall x.(\pi(C, x) \rightarrow \pi(D, x))$

ioms $\alpha \in \mathrm{KB}$. When an interpretation satisfies an axiom/a knowledge base, we also say that it is a *model* of that axiom/knowledge base.

Entailment is defined as usual. A knowledge base KB_1 *entails* a knowledge base KB_2, written $\mathrm{KB}_1 \models \mathrm{KB}_2$, if every model of KB_1 is a model of KB_2. A knowledge base is *unsatisfiable* (or *inconsistent*) if it has no models, and *satisfiable* (or *consistent*) otherwise. We use the same terminology for axioms, treated as singleton knowledge bases.

Every DL interpretation as defined above can be considered as an interpretation of $\mathbf{FOL}_{=}$ by considering atomic concepts as unary predicates, roles as binary predicates, and individuals as constant symbols. Indeed, most description logics can be viewed as fragments of first-order logic. Table 2 defines a standard mapping π from $\mathcal{ALC}$ axioms to first-order logic sentences. For a knowledge base KB, we define $\pi(\mathrm{KB})$ to be the conjunction $\bigwedge_{\alpha \in \mathrm{KB}} \pi(\alpha)$. It is easy to see that KB and $\pi(\mathrm{KB})$ have the same models, and thus also the same consequences. Knowing this, we sometimes treat DL axioms like first-order sentences, even without using π explicitly.

Datalog We use the term "*Datalog*" to refer to the function-free Horn logic fragment of $\mathbf{FOL}_{=}$. A *Datalog program* is a first-order theory which contains only formulae of the form $\forall \mathbf{x}.A_1 \wedge \cdots \wedge A_n \rightarrow B$ where A_i, B are atoms without function symbols of arity greater than 0, and $\mathbf{x}$ contains all variables occurring in these atoms. We generally omit the quantifier, we simply write B if $n = 0$. A special nullary atom $\bot$ is used to denote falsity. Rules with head $\bot$ are sometimes called (integrity) constraints, and are also written by omitting the head completely.

The semantics of Datalog is defined as for first-order logic, where $\bot$ is interpreted as a nullary atom with constant value *false*. Entailment and satisfiability are defined as usual for $\mathbf{FOL}_{=}$.

Note that we allow rules that contain variables in the head that do not occur in the body, i.e., rules that are not *safe* in the sense of the Datalog literature. Semantically, a rule like $\rightarrow B(x)$ represents the first-order formula $\forall x.B(x)$. When considering Datalog as a first-order language, such rules do not require any special treatment; all properties of Datalog remain unaffected. Given a fixed signature, one can transform unsafe Datalog into safe Datalog without affecting ground entailments. For this, one adds facts of the form $\top(c)$ for every constant $c \in \mathbf{I}$, and atoms $\top(x)$ to the body of every rule that contains a variable x. In this way, rule engines that are limited to safe Datalog programs can be used to compute first-order entailments of unsafe programs.

3. Semantic correspondences between logical theories

We are generally interested in DL knowledge bases the semantics of which can be expressed in a Datalog program. In this section, we introduce the kind of semantic correspondence that we find most appropriate for this task, and we observe a useful lemma that relates this notion to Datalog.

As discussed above, there are various notions of semantic correspondence that could be considered. For example, we could restrict to knowledge bases KB such that $\pi(\mathrm{KB})$ is semantically equivalent to a Datalog program. This, however, leads to a very strong requirement that excludes some interesting cases.

Example 3.1. The following $\mathcal{ALC}$ assertion states that Tom has a supervisor who is a professor:

$$\exists \mathsf{supervisor}.\mathsf{Prof}(\mathsf{tom}). \tag{9}$$

Datalog cannot express existential quantifiers in general. But this particular case requires the existence of only one individual (the supervisor of Tom), and the claimed existence of this individual can be captured with two facts using an auxiliary constant:

$$\mathsf{supervisor}(\mathsf{tom}, c_{\mathsf{tomsprof}}), \tag{10}$$

$$\mathsf{Prof}(c_{\mathsf{tomsprof}}). \tag{11}$$

Then (10) and (11) together are just another way of writing the Skolemisation of (9) with c_{tomsprof} used as

the nullary Skolem function symbol. Thus both forms are equisatisfiable, but they are not semantically equivalent.

This shows that semantic equivalence might turn out to be too restrictive, and that equisatisfiability might be a more suitable form of semantic correspondence. However, equisatisfiability is too weak, since it does not preserve relevant logical entailments. In particular, every satisfiable DL knowledge base is equisatisfiable to the empty Datalog program, yet this correspondence has no practical utility for using Datalog-based reasoning methods.

But Skolemisation actually leads to a stronger form of semantic correspondence that is certainly more useful as a middle-ground between equivalence and equisatisfiability:

Definition 3.2. Consider $\mathbf{FOL}_=$ theories T and T' with signatures $\mathscr{S} \subseteq \mathscr{S}'$. Then T' *semantically emulates* T if the following conditions hold:

(1) every model of T' becomes a model of T when restricted to the interpretations of symbols from $\mathscr{S}$,
(2) for every model $\mathcal{J}$ of T there is a model $\mathcal{I}$ of T' that has the same domain as $\mathcal{J}$, and that agrees with $\mathcal{J}$ on $\mathscr{S}$.

It is usually not necessary to mention the signatures of T and T' explicitly, since it is always possible to find minimal signatures for T and T' that satisfy condition (1) of Definition 3.2. Intuitively speaking, whenever a theory T' semantically emulates a theory T, we find that T' and T encode the same information *about the symbols* in T, and in particular that T' cannot be distinguished from T when restricting to those symbols.

The concept of semantic emulation is related to the notion of *conservative extensions* [27] which, however, additionally assumes $T \subseteq T'$ and hence requires syntactic compatibility of the involved logics. Another closely related notion is the model-theoretic version of $\mathscr{S}$*-inseparability* [19] which holds between two theories if their model classes coincide after being projected to a given signature $\mathscr{S}$. This justifies to introduce the new notion of semantic emulation for our setting. Also note that, in contrast to equivalence and equisatisfiability, semantic emulation is not a symmetric relation, since one of the theories introduces additional "internal" symbols to its signature. It is not hard to see that the Datalog program consisting of (10) and (11) above semantically emulates the DL fact (9) since we can always find a suitable interpretation for the fresh constant c_{tomsprof}.

To understand the consequence of Definition 3.2, we also consider a slightly weaker notion:

Definition 3.3. Consider $\mathbf{FOL}_=$ theories T and T' with signatures $\mathscr{S} \subseteq \mathscr{S}'$. Then T' *syntactically emulates* T if for every first-order formula φ over $\mathscr{S}$: $T \models \varphi$ iff $T' \models \varphi$.

Note that syntactic emulation of T by T' can equivalently be characterized by the requirement that for every formula φ over $\mathscr{S}$ the sets $T \cup \{\varphi\}$ and $T' \cup \{\varphi\}$ be equisatisfiable. It is easy to see that semantic emulation implies syntactic emulation. The converse is not true in general, and indeed it is not hard to show that semantic emulation is equivalent to the condition that one would obtain when considering second-order logic instead of first-order logic in Definition 3.3. In this work, we use the stronger notion, since it guarantees a maximal amount of semantic interoperability without depending on a particular logic. However, we conjecture that our results are not affected by this choice. Some related discussion can be found in Section 7.

An important goal in this work is to show that certain DL axioms can be semantically emulated in Datalog. This can be accomplished by noting model-theoretic properties of Datalog that are not generally shared by description logics. To this end, we recall two simple constructions: intersection and product [7].

Definition 3.4. Let $\mathcal{I}_1$ and $\mathcal{I}_2$ be interpretations over the same domain Δ, such that $c^{\mathcal{I}_1} = c^{\mathcal{I}_2}$ for all constants c. The *intersection* $\mathcal{I}_1 \cap \mathcal{I}_2$ of $\mathcal{I}_1$ and $\mathcal{I}_2$ is the interpretation with domain Δ that interprets constants c as $c^{\mathcal{I}_1 \cap \mathcal{I}_2} := c^{\mathcal{I}_1}$, and predicates p as $p^{\mathcal{I}_1 \cap \mathcal{I}_2} := p^{\mathcal{I}_1} \cap p^{\mathcal{I}_2}$.

Definition 3.5. Let $\mathcal{I}_1$ and $\mathcal{I}_2$ be interpretations. The *product* $\mathcal{I}_1 \times \mathcal{I}_2$ of $\mathcal{I}_1$ and $\mathcal{I}_2$ is the interpretation with domain $\Delta^{\mathcal{I}_1} \times \Delta^{\mathcal{I}_2}$ that interprets constants c as $c^{\mathcal{I}_1 \times \mathcal{I}_2} := \langle c^{\mathcal{I}_1}, c^{\mathcal{I}_2} \rangle$, and predicates p as $p^{\mathcal{I}_1 \times \mathcal{I}_2} := p^{\mathcal{I}_1} \times p^{\mathcal{I}_2}$.

Models of Datalog are closed under both operations:

Proposition 3.6. *Consider interpretations $\mathcal{I}_1$ and $\mathcal{I}_2$, and a Datalog program P with $\mathcal{I}_1 \models P$ and $\mathcal{I}_2 \models P$.*

1. *If $\mathcal{I}_1 \cap \mathcal{I}_2$ is defined, then $\mathcal{I}_1 \cap \mathcal{I}_2 \models P$.*
2. *$\mathcal{I}_1 \times \mathcal{I}_2 \models P$.*

Proof. The arguments for both cases are similar. We illustrate the case of $\times$, which is the less widely known construction. Consider an arbitrary rule $A_1 \wedge \cdots \wedge A_n \rightarrow B$ in P. Assume that there is a variable assignment $\mathcal{Z}$ for $\mathcal{I}_1 \times \mathcal{I}_2$ such that $\mathcal{I}_1 \times \mathcal{I}_2, \mathcal{Z} \models A_1 \wedge \cdots \wedge A_n$.

Let $\mathcal{Z}_i$ be the variable assignment for $\mathcal{I}_i$ that maps x to the ith component of $\mathcal{Z}(x)$. By definition of $\mathcal{I}_1 \times \mathcal{I}_2$, we find $\mathcal{I}_i, \mathcal{Z}_i \models A_1 \wedge \cdots \wedge A_n$. Since $\mathcal{I}_i \models P$, we obtain $\mathcal{I}_i, \mathcal{Z}_i \models B$. By definition of $\mathcal{I}_1 \times \mathcal{I}_2$, this implies $\mathcal{I}_1 \times \mathcal{I}_2, \mathcal{Z} \models B$, as required. □

4. The DLP fragment of a description logic

In this section, we discuss and motivate a generic definition for DLP fragments of a description logic. This will provide a meaningful definition for the "intersection of DL and Datalog" that we will use in the rest of this paper.

Since DL and Datalog use a different syntax, this "intersection" is necessarily asymmetrical in the sense that DLP must be a fragment of either DL or of Datalog. In the tradition of the original DLP proposal, we choose the former [13]. A second defining property of DLP is the semantic correspondence with some Datalog program. As discussed in the previous section, the notion of semantic emulation provides a suitable notion for this correspondence.

These requirements alone, however, do not give rise to viable language definitions yet. As discussed in the introduction, deciding whether a knowledge base meets the semantic criteria of being expressible in Datalog may involve complex reasoning. In particular, every inconsistent knowledge base can be semantically emulated by some Datalog program. Therefore, some additional criterion is needed to ensure that containment in the language is easy to check.

A powerful tool for obtaining this criterion is the construction of variants of logical expressions which preserve only the logical structure but may modify concrete signature symbols:

Definition 4.1. Let F be a $\mathbf{FOL}_=$ formula, a DL axiom, or a DL concept expression, and let $\mathscr{S}$ be a signature. An expression F' is a *variant* of F in $\mathscr{S}$ if F' can be obtained from F by replacing each occurrence of a role/concept/individual name with some role/concept/individual name in $\mathscr{S}$. Multiple occurrences of the same entity name in F need *not* be replaced by the same entity name of $\mathscr{S}$ in this process.

A knowledge base KB′ is a *variant* of a knowledge base KB if it is obtained from KB by replacing each axiom with a variant.

Note that we do not require all occurrences of an entity name to be renamed together, so it is indeed possible to obtain $A \sqcap \neg B$ from $A \sqcap \neg A$. Considering all variants of a formula or axiom allows us to study the semantics and expressivity of formulae based on their syntactic structure, disregarding any possible interactions between signature symbols. We call a $\mathbf{FOL}_=$ formula, DL axiom, or DL concept expression F *name-separated* if no signature symbol occurs more than once in F. Intuitively speaking, disallowing symbols to occur in multiple positions in name-separated axioms prevent most of the complex semantic effects that could require reasoning, i.e., a name-separated axiom that can only be expressed in Datalog if its formula structure can generally be captured using rules.

Combining these ideas, we can formally define DLP fragments:

Definition 4.2. Given description logics $\mathcal{L}$ and $\mathcal{D}$, we call $\mathcal{D}$ a *DLP fragment* of $\mathcal{L}$ if

(1) every axiom of $\mathcal{D}$ is an axiom of $\mathcal{L}$,
(2) there is a transformation function datalog that maps every $\mathcal{D}$ axiom α to a Datalog program datalog(α) such that datalog(α) semantically emulates α,
(3) $\mathcal{D}$ is closed under variants, i.e., given any axiom α and an arbitrary variant α' of α, we find α is in $\mathcal{D}$ iff α' is.

As discussed above, item (1) of this definition fixes the syntactic framework for DLP fragments. Item (2) states the property that motivates the study of DLP languages: every axiom of a DLP fragment can be expressed in Datalog. DLP languages as discussed in the literature may require the use of auxiliary symbols for the translation to Datalog [36], and the Datalog program can no longer be semantically equivalent to the original knowledge base in this case, even if all consequences with respect to the original predicates are still the same. This motivates the use of semantic emulation as introduced in Definition 3.2. Note that we do not require the transformation function datalog to be computable, although it will turn out to be computable rather easily in the case studied in this paper.

Item (3) of Definition 4.2 reflects our desire to obtain fragments that correspond to well-behaved logical languages as opposed to being arbitrary collections of axioms. An obvious way to implement this would be to require DLP fragments to be described by a context-free grammar. A typical feature of grammars for logical languages is that they are parametrised by a logical signature that can be modified without changing the essential structural features of the language. This effect is mirrored by the requirement of item (3) without

introducing detailed requirements on a suitable logical grammar. We will find grammatical descriptions in the cases we consider, though item (3) as such does not imply that this is possible.

The original motive for item (3) in Definition 4.2 was to obtain DLP fragments for which membership can be checked without complex reasoning. A natural alternative would thus be to require that membership in a fragment can be decided efficiently, say in polynomial time. However, Proposition 4.3 below shows that in this case no *maximal* DLP fragment can exist. Definition 4.2, in contrast, does not impose any restriction on the complexity of checking the membership relation, but it admits a maximal DLP fragment for $\mathcal{ALC}$ that can be described by a context-free language (Section 5), and thus is efficiently recognisable.

Proposition 4.3. *Given description logics $\mathcal{L}$ and $\mathcal{D}$, we call $\mathcal{D}$ a* P-DLP fragment *of $\mathcal{L}$ if items (1) and (2) of Definition 4.2 are satisfied, and in addition there is a polynomial procedure for deciding $\alpha \in \mathcal{D}$ for any DL axiom α.*

Unless the complexity classes P *and* PSpace *coincide, there is no maximal* P*-DLP fragment of $\mathcal{ALC}$: given any* P*-DLP fragment $\mathcal{D}$ of $\mathcal{ALC}$, there is a* P*-DLP fragment $\mathcal{D}'$ of $\mathcal{ALC}$ that covers more axioms, i.e, $\mathcal{D} \subset \mathcal{D}'$.*

Proof. We start with an auxiliary construction: if the concept expression C is satisfiable and does not contain the symbols R, A_1, A_2, and c, then no Datalog program semantically emulates the expression $\alpha_C := (C \sqcap \exists R.(A_1 \sqcup A_2))(c)$. For a contradiction, suppose that α_C is semantically emulated by a Datalog theory $\mathsf{datalog}(\alpha_C)$. By construction, α_C is satisfiable, and so is $\{\alpha_C, A_i \sqsubseteq \bot\}$ for each $i = 1, 2$. By Definition 3.3, we find that $\mathsf{datalog}(\alpha_C) \cup \{A_i \sqsubseteq \bot\}$ is satisfiable, too. Thus, there are models $\mathcal{I}_i$ of $\mathsf{datalog}(\alpha_C)$ such that $A_i^{\mathcal{I}_i} = \emptyset$. Thanks to Proposition 3.6, we find a model $\mathcal{I} = \mathcal{I}_1 \cap \mathcal{I}_2$ of $\mathsf{datalog}(\alpha_C)$ such that $A_1^{\mathcal{I}} = A_2^{\mathcal{I}} = \emptyset$. But then $\mathsf{datalog}(\alpha_C) \cup \{A_1 \sqcup A_2 \sqsubseteq \bot\}$ is satisfiable although $\{\alpha, A_1 \sqcup A_2 \sqsubseteq \bot\}$ is not, contradicting the supposed semantic emulation.

Let us now assume for the sake of a contradiction that $\mathcal{D}$ contains all unsatisfiable $\mathcal{ALC}$ axioms of the form of α_C. This would give a polynomial decision procedure for deciding satisfiability of $\mathcal{ALC}$ concept expressions C: construct α_C from C (clearly polynomial) decide $\alpha_C \in \mathcal{D}$ (was assumed to be of polynomial complexity). This contradicts the fact that deciding (un)satisfiability of $\mathcal{ALC}$ concept expressions is PSpace-hard.

Therefore, there is an unsatisfiable expression α with $\alpha \notin \mathcal{D}$. Now let $\mathcal{D}'$ be defined as $\mathcal{D} \cup \{\alpha\}$. The transformation is given by $\mathsf{datalog}'(\beta) = \mathsf{datalog}(\beta)$ if $\beta \in \mathcal{D}$, and $\mathsf{datalog}'(\beta) = \{\top \to A(x), A(x) \to \bot\}$ otherwise, where A is a new predicate symbol. It is immediate that $\mathcal{D}'$ P-DLP fragment of $\mathcal{ALC}$ strictly greater than $\mathcal{D}$. □

This proof exemplifies a general problem that occurs when trying to define DLP: the question whether an axiom is expressible in Datalog is typically computationally harder than one would like to admit for a language definition. This result carries over to more expressive DLs, and remains valid even if requirements such as closure under common normal form transformations are added to the definition of fragments. The fact that this problem is avoided by item (3) in Definition 4.2 confirms our intuition that this requirement closely relates to the possibility of representing DLP fragments syntactically, i.e., without referring to complex semantic conditions.

Next, we establish an interesting general result on the complexity of reasoning in DLP fragments. By the *size of an axiom*, we mean the number of symbols that are necessary to write it.

Proposition 4.4. *Consider a class K of knowledge bases that belong to a DLP fragment of some description logic, and such that the maximal size of axioms in K is bounded. Deciding satisfiability of knowledge bases in K is possible in polynomial time.*

Proof. Let the maximal size of axioms be bounded by N. Let V be a vocabulary with N concept, role and constant symbols. By assumption we know that for every of the finitely many axioms α of size less than N there is a translation $\mathsf{datalog}(\alpha)$. We will use this as a (finite) look-up table for finding a Datalog transformation for axioms β in $\mathrm{KB} \in K$. Note that we do not need to specify how the translations $\mathsf{datalog}(\alpha)$ were computed, since we only need to show that there is a polynomial time algorithm, not how it can be found.

We define a Datalog transformation $\mathsf{datalog}_K(\beta)$ for all axioms $\beta \in \mathrm{KB}$ that occur in some knowledge base $\mathrm{KB} \in K$. By the assumption on K, there are at most N signature symbols in β. Hence there some axiom α over the vocabulary of V and a 1-1 renaming σ of symbols in α such that $\sigma(\alpha) = \beta$. We thus define $\mathsf{datalog}_K(\beta) := \sigma(\mathsf{datalog}(\alpha))$. It is easy to see that $\mathsf{datalog}_K(\beta)$ still satisfies item (2) of Definition 4.2.

Thus satisfiability of $\mathrm{KB} \in K$ can be decided by checking satisfiability of $\bigcup_{\beta \in \mathrm{KB}} \mathsf{datalog}_K(\beta)$. The

Concepts necessarily equivalent to $\top$:	$\mathbf{L}_\top ::= \top \mid \forall \mathbf{R}.\mathbf{L}_\top \mid \mathbf{L}_\top \sqcap \mathbf{L}_\top \mid \mathbf{L}_\top \sqcup \mathbf{C}$
Concepts necessarily equivalent to $\bot$:	$\mathbf{L}_\bot ::= \bot \mid \exists \mathbf{R}.\mathbf{L}_\bot \mid \mathbf{L}_\bot \sqcap \mathbf{C} \mid \mathbf{L}_\bot \sqcup \mathbf{L}_\bot$
Body ($C \in \mathbf{L}_B$ iff $\neg C \sqsubseteq A$ in $\mathcal{DLP}_{\mathcal{ALC}}$):	$\mathbf{L}_B ::= \mathbf{L}_\top \mid \mathbf{L}_\bot \mid \neg \mathbf{A} \mid \forall \mathbf{R}.\mathbf{L}_B \mid \mathbf{L}_B \sqcap \mathbf{L}_B \mid \mathbf{L}_B \sqcup \mathbf{L}_B$
Head ($C \in \mathbf{L}_H$ iff $A \sqsubseteq C$ in $\mathcal{DLP}_{\mathcal{ALC}}$):	$\mathbf{L}_H ::= \mathbf{L}_B \mid \mathbf{A} \mid \forall \mathbf{R}.\mathbf{L}_H \mid \mathbf{L}_H \sqcap \mathbf{L}_H \mid \mathbf{L}_H \sqcup \mathbf{L}_B$
Assertions ($C \in \mathbf{L}_a$ iff $C(a)$ in $\mathcal{DLP}_{\mathcal{ALC}}$):	$\mathbf{L}_a ::= \mathbf{L}_H \mid \exists \mathbf{R}.\mathbf{L}_a \mid \mathbf{L}_a \sqcap \mathbf{L}_a \mid \mathbf{L}_a \sqcup \mathbf{L}_B$

Fig. 1. $\mathcal{DLP}_{\mathcal{ALC}}$ concepts in negation normal form.

maximal number of variables occurring within these Datalog programs can be bounded by an integer M. Indeed, M can be taken to be the (finite) number of variables in $\bigcup_\alpha \mathsf{datalog}(\alpha)$, where this is the union over all axioms α for which $\mathsf{datalog}(\alpha)$ was defined above. Note that M depends only on the choice of N since we can assume w.l.o.g. that the translations $\mathsf{datalog}(\alpha)$ are such that M is minimal.

Satisfiability of Datalog with at most M variables per rule can be decided in time polynomial in 2^M [8]. The renamings σ can be found in time polynomial in 2^N. Since N and M are constants, this yields a polynomial time upper bound for deciding satisfiability of knowledge bases in K. □

It is interesting that the previous result does not require any assumptions on the computational complexity of recognising or translating DLP axioms. Intuitively, Proposition 4.4 states that reasoning in any DLP language is necessarily "almost" tractable. Indeed, many DLs allow complex axioms to be decomposed into a number of simpler normal forms of bounded size, and in any such case tractability is obtained. Moreover, Proposition 4.4 clarifies why Horn-$\mathcal{SHIQ}$ (and thus also Horn-$\mathcal{SHOIQ}$) cannot be in DLP: ExpTime worst-case complexity of reasoning can be proven for a class K of Horn-$\mathcal{SHIQ}$ knowledge bases as in the above proposition (see [23], noting that remaining complex axioms can be decomposed in Horn-$\mathcal{SHIQ}$). In fact the same argument already holds for the much weaker DL Horn-$\mathcal{FLE}$ [23].

5. The DLP fragment of $\mathcal{ALC}$

Using Definition 4.2, it is now possible to investigate DLP fragments of relevant description logics. In this paper, we detail this approach for $\mathcal{ALC}$; some remarks on the more complex case of $\mathcal{SROIQ}$ are given in Section 7 below. It turns out that the largest DLP fragment of $\mathcal{ALC}$ exists, and can be defined as follows, where we use the negation normal form NNF for simplifying our presentation.

Definition 5.1. The description logic $\mathcal{DLP}_{\mathcal{ALC}}$ consists of all knowledge bases that contain only $\mathcal{ALC}$ axioms which are

- GCIs $C \sqsubseteq D$ such that $\mathsf{NNF}(\neg C \sqcup D)$ is an $\mathbf{L}_H$ concept as defined in Fig. 1, or
- ABox axioms $C(a)$ where $\mathsf{NNF}(C)$ is an $\mathbf{L}_a$ concept as defined in Fig. 1.
- ABox axioms $R(a, b)$ where $R \in \mathbf{R}$ is a role name.

The headings in Fig. 1 give the basic intuition about the significance of the various concept languages. The distinction of head and body concepts is typical for many works on DLP and Horn DLs, while our use of additional assertional concepts takes into account that emulation allows for some forms of Skolemisation.

Example 5.2. Some typical example representatives of the head, body, and assertion grammars in Fig. 1 are as follows:

$$\neg A \sqcap \forall R.(\neg B \sqcup \neg C) \in \mathbf{L}_B, \tag{12}$$

$$\neg A \sqcup (B \sqcap \forall R.C) \in \mathbf{L}_H, \tag{13}$$

$$\neg A \sqcup \exists R.B \in \mathbf{L}_a. \tag{14}$$

Concept (13) corresponds to the $\mathcal{DLP}_{\mathcal{ALC}}$ axiom $A \sqsubseteq B \sqcap \forall R.C$, while (14) can be used in assertions such as $(\neg A \sqcup \exists R.B)(c)$. Typical examples of axioms that are not in $\mathcal{DLP}_{\mathcal{ALC}}$ include $A \sqsubseteq B \sqcup C$ and $(B \sqcup C)(a)$, and also $A \sqsubseteq \exists R.B$. In contrast, $(\exists R.B)(a)$ is in $\mathcal{DLP}_{\mathcal{ALC}}$.

Though name separation prevents most forms of semantic interactions within concepts, we still require grammars for $\mathbf{L}_\top$ and $\mathbf{L}_\bot$ to characterise concepts all variants of which are equivalent to $\top$ and $\bot$, respectively. This includes concept expressions such as $A \sqcap \exists R.\bot$ and $B \sqcup \forall R.\top$.

We start with an easy observation on Definition 5.1. This result will not explicitly be used later on but might add to the understanding of this definition.

$$\mathsf{dlg}_B(\neg A \sqsubseteq C) = \begin{cases} \{\} & \text{if } C \in \mathbf{L}_\top \\ \{A(x)\} & \text{if } C \in \mathbf{L}_\bot \\ \{B(x) \rightarrow A(x)\} & \text{if } C = \neg B \\ \mathsf{dlg}_B(\neg X \sqsubseteq D) \cup \{R(x,y) \wedge X(y) \rightarrow A(x)\} & \text{if } C = \forall R.D \\ \mathsf{dlg}_B(\neg A \sqsubseteq D_1) \cup \mathsf{dlg}_B(\neg A \sqsubseteq D_2) & \text{if } C = D_1 \sqcap D_2 \in (\mathbf{L}_B \sqcap \mathbf{L}_B) \\ \mathsf{dlg}_B(\neg X_1 \sqsubseteq D_1) \cup \mathsf{dlg}_B(\neg X_2 \sqsubseteq D_2) \cup \{X_1(x) \wedge X_2(x) \rightarrow A(x)\} & \text{if } C = D_1 \sqcup D_2 \in (\mathbf{L}_B \sqcup \mathbf{L}_B) \end{cases}$$

Fig. 2. Transforming axioms $\neg A \sqsubseteq C$ to Datalog, where $A \in \mathbf{A}$, $C \in \mathbf{L}_B$, and $X_{(i)}$ are fresh concept names.

$$\mathsf{dlg}_H(A \sqsubseteq C) = \begin{cases} \mathsf{dlg}_B(\neg X \sqsubseteq C) \cup \{A(x) \wedge X(x) \rightarrow \bot\} & \text{if } C \in \mathbf{L}_B \\ \{A(x) \rightarrow C(x)\} & \text{if } C \in \mathbf{A} \\ \mathsf{dlg}_H(X \sqsubseteq D) \cup \{A(x) \wedge R(x,y) \rightarrow X(y)\} & \text{if } C = \forall R.D \\ \mathsf{dlg}_H(A \sqsubseteq D_1) \cup \mathsf{dlg}_H(A \sqsubseteq D_2) & \text{if } C = D_1 \sqcap D_2 \\ \mathsf{dlg}_H(X_2 \sqsubseteq D_1) \cup \mathsf{dlg}_B(\neg X_1 \sqsubseteq D_2) \cup \{A(x) \wedge X_1(x) \rightarrow X_2(x)\} & \text{if } C = D_1 \sqcup D_2 \in (\mathbf{L}_H \sqcup \mathbf{L}_B) \end{cases}$$

Fig. 3. Transforming axioms $A \sqsubseteq C$ to Datalog, where $A \in \mathbf{A}$, $C \in \mathbf{L}_H$, and $X_{(i)}$ are fresh concept names.

Lemma 5.3. *Consider arbitrary $\mathcal{ALC}$ concept expressions C that do not contain quantifiers $\forall$, $\exists$, and the symbols $\top$ and $\bot$.*

1. *If $C \in \mathbf{L}_B$ then C has a conjunctive normal form $\bigsqcap_i \bigsqcup_j C_{i,j}$ with $C_{i,j}$ a negated atom for all i, j.*
2. *If $C \in \mathbf{L}_H$ or $C \in \mathbf{L}_a$ then C has a conjunctive normal form $\bigsqcap_i \bigsqcup_j C_{i,j}$ with $C_{i,j}$ negated or non-negated atoms and for every i there is at most one j such that $C_{i,j}$ is an non-negated atom.*
   (Since the assumptions require that C does not contain quantifiers there is no difference here between $C \in \mathbf{L}_H$ and $C \in \mathbf{L}_a$.)

Proof. Notice that $C \notin \mathbf{L}_\top$ and $C \notin \mathbf{L}_\bot$ since neither $\top$ nor $\bot$ occur in C. For item (1), note that if $C \in \mathbf{L}_B$ then either C is a negated atom, or $C = C_1 \sqcap C_2$ or $C = C_1 \sqcup C_2$ with $C_i \in \mathbf{L}_B$. The claim now follows easily from the induction hypothesis on C_1, C_2.

For item (2), by the assumptions on C we have $C \in \mathbf{L}_H$ if one of the following cases holds true:

1. $C \in \mathbf{L}_B$. Then the claim follows from part (1) of the lemma.
2. C is an atom. Then the claim is obviously true.
3. $C = C_1 \sqcap C_2$ with $C_i \in \mathbf{L}_H$. If C_i' is a conjunctive normal form of C_i satisfying the claim then $C_1' \sqcap C_2'$ is a conjunctive normal form of C satisfying the claim.
4. $C = C_1 \sqcup C_2$ with $C_i \in \mathbf{L}_H$ and $C_1 \in \mathbf{L}_B$. Let $\bigsqcap_i \bigsqcup_j C^1_{i,j}$ and $\bigsqcap_m \bigsqcup_n C^2_{m,n}$ be the conjunctive normal forms that exist by induction hypothesis satisfying the respective claims. A conjunctive normal form of $C = C_1 \sqcup C_2$ is obtained as the conjunction of all $\bigsqcup_j C^1_{i,j} \sqcup \bigsqcup_n C^2_{m,n}$ for all combinations of i, m. Since $\bigsqcup_j C^1_{i,j}$ contains at most one positive atom and $\bigsqcup_n C^2_{m,n}$ contains only negative atoms we are finished. □

It is obvious that $\mathcal{DLP}_{\mathcal{ALC}}$ satisfies items (1) and (3) of Definition 4.2, so what remains to show is that $\mathcal{DLP}_{\mathcal{ALC}}$ knowledge bases can indeed be expressed in Datalog. Following the grammatical structure of $\mathcal{DLP}_{\mathcal{ALC}}$, we specify various functions for constructing Datalog programs to semantically emulate a $\mathcal{DLP}_{\mathcal{ALC}}$ knowledge base. GCIs are handled by the functions dlg_B and dlg_H, defined recursively in Figs 2 and 3. The function dlg_B constructs Datalog to semantically emulate GCIs of the form $\neg A \sqsubseteq C$ with $C \in \mathbf{L}_B$, while dlg_H allows us to semantically emulate GCIs $A \sqsubseteq C$ with $C \in \mathbf{L}_H$.

Example 5.4. Let E be the $\mathbf{L}_H$ concept $\neg A \sqcup (B \sqcap \forall R.C)$ as in (13). The rules of $\mathsf{dlg}_H(D \sqsubseteq E)$ are as follows:

$$D(x) \wedge X_1(x) \rightarrow X_2(x)$$
$$A(x) \rightarrow X_1(x)$$
$$X_2(x) \rightarrow B(x)$$
$$X_2(x) \wedge R(x,y) \rightarrow X_3(x)$$
$$X_3(x) \rightarrow C(x)$$

$$\mathsf{dlg}_a(C(a), E) = \begin{cases} \mathsf{dlg}_H(X \sqsubseteq C \sqcup E) \cup \{X(a)\} & \text{if } C \in \mathbf{L}_H \\ \mathsf{dlg}_a(D_1(a), E) \cup \mathsf{dlg}_a(D_2(a), E) & \text{if } C = D_1 \sqcap D_2 \\ \mathsf{dlg}_B(\neg X \sqsubseteq D_2) \cup \mathsf{dlg}_a(D_1(a), E \sqcup \neg X) & \text{if } C = D_1 \sqcup D_2 \in (\mathbf{L}_a \sqcup \mathbf{L}_B) \\ \mathsf{dlg}_B(\neg X \sqsubseteq E) \cup \mathsf{dlg}_a(D(b), \neg Y) \cup \{X(a) \to R(a,b), X(a) \to Y(b)\} & \text{if } C = \exists R.D \end{cases}$$

Fig. 4. Transforming axioms $C(a)$ to Datalog, where $C \in \mathbf{L}_a$, $E \in \mathbf{L}_B$, X, Y are fresh concept names, and b is a fresh constant.

Clearly, this rule set could be further simplified to obtain three rules $D(x) \wedge A(x) \to X_2(x)$, $X_2(x) \to B(x)$, and $X_2(x) \wedge R(x,y) \to C(x)$, which are easily seen to semantically emulate $D \sqsubseteq E$.

The correctness of dlg_B and dlg_H is shown in the following two lemmas.

Lemma 5.5. *Given a concept name A, and a concept $C \in \mathbf{L}_B$, Fig. 2 recursively defines a Datalog program $\mathsf{dlg}_B(\neg A \sqsubseteq C)$ that semantically emulates $\neg A \sqsubseteq C$.*

Proof. The claim is shown by induction over the structure of C. We illustrate one case.

Consider the case $C = D_1 \sqcup D_2$. To show Definition 3.2 (1), assume that $\mathcal{I} \models \mathsf{dlg}_B(\neg A \sqsubseteq C)$. In particular, $\mathcal{I} \models \mathsf{dlg}_B(\neg X_1 \sqsubseteq D_1)$ and $\mathcal{I} \models \mathsf{dlg}_B(\neg X_2 \sqsubseteq D_2)$. By the induction hypothesis, $\mathcal{I} \models \neg X_1 \sqsubseteq D_1$ and $\mathcal{I} \models \neg X_2 \sqsubseteq D_2$. Since $\mathcal{I} \models X_1(x) \wedge X_2(x) \to A(x)$, for any $\delta \in A^{\mathcal{I}}$, we find that $\delta \notin X_1^{\mathcal{I}}$ for some $i \in \{1,2\}$. Thus, $\delta \in D_i^{\mathcal{I}}$ follows from $\mathcal{I} \models \neg X_i \sqsubseteq D_i$. Since δ was arbitrary, $\mathcal{I} \models \neg A \sqsubseteq D_1 \sqcup D_2$.

To show Definition 3.2 (2), assume that $\mathcal{I} \models \neg A \sqsubseteq C$. An interpretation $\mathcal{I}'$ over the extended signature is defined by setting $X_i^{\mathcal{I}'} := \Delta^{\mathcal{I}} \setminus D_i^{\mathcal{I}}$ for $i \in \{1,2\}$. It is easy to see that $\mathcal{I}' \models \{\neg X_i \sqsubseteq D_i \mid i \in \{1,2\}\} \cup \{X_1(x) \wedge X_2(x) \to A(x)\}$. By the induction hypothesis, we can find an interpretation $\mathcal{I}_1$ that extends $\mathcal{I}'$ and such that $\mathcal{I}_1 \models \mathsf{dlg}_B(\neg X_1 \sqsubseteq D_1)$. Another application of the hypothesis yields a model $\mathcal{I}_2 \models \mathsf{dlg}_B(\neg A \sqsubseteq C)$ as required to show the claim. □

Lemma 5.6. *Given a concept name A, and a concept $C \in \mathbf{L}_H$, Fig. 3 recursively defines a Datalog program $\mathsf{dlg}_H(A \sqsubseteq C)$ that semantically emulates $A \sqsubseteq C$.*

Proof. First note that $\mathsf{dlg}_H(A \sqsubseteq C)$ is well-defined. In particular, programs $\mathsf{dlg}_B(\neg B \sqsubseteq D)$ are only used if $D \in \mathbf{L}_B$. The inductive proof of the claim is similar to the proof of Lemma 5.5, so we omit the details. □

For translating concept assertions to Datalog, we define a function dlg_a in Fig. 4. The construction of Fig. 4 uses a "guard" concept E that is used to defer the encoding of $\mathbf{L}_B$ disjunctions: $\mathsf{dlg}_H(C(a), E)$ semantically emulates $(C \sqcup E)(a)$. To encode an assertion $C(a)$, the program $\mathsf{dlg}_H(C(a), \bot)$ is used. As before, this transformation is designed for a concise definition, not for optimised output.

Example 5.7. Let E be the $\mathbf{L}_a$ concept $\neg A \sqcup \exists R.B$ as in (14). Then $\mathsf{dlg}_H(E(a), \bot)$ consists of the following rules (X_i and Y indicating fresh concept names as in the definition of the transformation):

$$\begin{array}{rcl@{\qquad}rcl} A(x) & \to & X_1(x) & X_2(a) & \to & R(a,b) \\ X_2(a) & \to & Y(b) & X_3(x) \wedge X_4(x) & \to & X_2(x) \\ & \to & X_3(x) & X_1(x) & \to & X_4(x) \\ & \to & X_5(b) & X_5(x) \wedge X_6(x) & \to & X_7(x) \\ X_7(x) & \to & B(x) & Y(x) & \to & X_6(x) \end{array}$$

As before, this rule set can be simplified significantly by eliminating most of the introduced auxiliary concept symbols. Doing this, we obtain the three rules $A(x) \to X_2(x)$, $X_2(a) \to R(a,b)$, and $X_2(a) \to B(b)$, which again are easily seen to semantically emulate $E(a)$ as claimed. Here, the fresh constant symbol b acts as a Skolem constant that represents the individual that the existential concept expression may require to exist.

Note that some Datalog rules created in Example 5.7 are not safe. As explained in Section 2, this is not a principal issue.

Lemma 5.8. *Given a constant a and a concept $C \in \mathbf{L}_a$, Fig. 4 recursively defines a Datalog program $\mathsf{dlg}_H(C(a), \bot)$ that semantically emulates $C(a)$.*

Proof. As before, the proof works by induction. The induction claim is that, for every $E \in \mathbf{L}_B$, $C \in \mathbf{L}_a$, and $a \in \mathbf{I}$, the program $\mathsf{dlg}_H(C(a), E)$ semantically emulates $(C \sqcup E)(a)$.

The concept E is processed in case $C \in \mathbf{L}_H$ by using dlg_H. Another more interesting case is $C = \exists R.D$. The basic encoding works by standard Skolemisation, but the guard concept is also processed and a new guard $\neg Y$ is created for the Skolem constant d. It is not hard to show semantic emulation in all cases. □

Combining the previous lemmata, we obtain the emulation theorem for $\mathcal{DLP}_{\mathcal{ALC}}$.

Theorem 5.9. *For every $\mathcal{DLP}_{\mathcal{ALC}}$ axiom α as in Definition 5.1, one can construct a Datalog program* $\mathsf{dlg}(\alpha)$ *that semantically emulates α.*

Proof. If $\alpha = C \sqsubseteq D$ is a TBox axiom, define $\mathsf{datalog}(\alpha) := \mathsf{dlg}_H(A \sqsubseteq \mathsf{NNF}(\neg C \sqcup D)) \cup \{A(x)\}$. If $\alpha = C(a)$ is an ABox axiom, define $\mathsf{datalog}(\alpha) := \mathsf{dlg}_a(C(a), \bot)$. The result then follows by Lemma 5.6 and 5.8. □

6. Maximality of $\mathcal{DLP}_{\mathcal{ALC}}$

It remains to show that $\mathcal{DLP}_{\mathcal{ALC}}$ is indeed the largest DLP fragment of $\mathcal{ALC}$. In general, Proposition 3.6 can be used to show that a logical theory T cannot be semantically emulated in Datalog as follows: based on the assumption that T is semantically emulated by a Datalog program P_T, one constructs interpretations $\mathcal{I}_1$ and $\mathcal{I}_2$ such that $\mathcal{I}_1 \models P_T$ and $\mathcal{I}_2 \models P_T$ and shows that either $\mathcal{I}_1 \cap \mathcal{I}_2 \not\models P_T$ or $\mathcal{I}_1 \times \mathcal{I}_2 \not\models P_T$, contradicting Proposition 3.6. We use $\mathcal{I}_1 \cap \mathcal{I}_2$ to show that DLP cannot contain existential axioms like $\exists R.A$ (Lemma 6.7), and $\mathcal{I}_1 \times \mathcal{I}_2$ to show that it cannot contain disjunctive axioms like $A \sqcup B$ (Lemma 6.10).

We can simplify our arguments by means of the following syntactic simplification for concepts of the form $\mathbf{L}_\top$ and $\mathbf{L}_\bot$.

Definition 6.1. Let C be an $\mathcal{ALC}$ concept expression in negation normal form, i.e., such that $C = \mathsf{NNF}(C)$. The expression $\mathsf{etb}(C)$ (**e***liminate* **t***op and* **b***ottom*) is obtained from C by applying exhaustively the following rewrite rules:

$$\begin{array}{ll} \top \sqcap D \mapsto D & \bot \sqcup D \mapsto D \\ \top \sqcup D \mapsto \top & \bot \sqcap D \mapsto \bot \\ D \sqcap \top \mapsto D & D \sqcup \bot \mapsto D \\ D \sqcup \top \mapsto \top & D \sqcap \bot \mapsto \bot \\ \forall R.\top \mapsto \top & \exists R.\bot \mapsto \bot \end{array}$$

Note, that $\mathsf{etb}(C)$ may still contain subexpressions of the form $\forall \mathbf{R}.\bot$ and $\exists \mathbf{R}.\top$. The next lemma summarises some easy observations on etb.

Lemma 6.2. *For any $\mathcal{ALC}$ concept $C = \mathsf{NNF}(C)$:*

1. $\mathsf{etb}(C)$ *is logically equivalent to C, i.e., for any interpretation $\mathcal{I}$, we have $C^{\mathcal{I}} = \mathsf{etb}(C)^{\mathcal{I}}$;*
2. *for every $\mathbf{L} \in \{\mathbf{L}_\top, \mathbf{L}_\bot, \mathbf{L}_a, \mathbf{L}_B, \mathbf{L}_H\}$, we have $C \in \mathbf{L}$ iff $\mathsf{etb}(C) \in \mathbf{L}$;*
3. *if C does not contain subexpressions of the form $\forall \mathbf{R}.\bot$ or $\exists \mathbf{R}.\top$ then $\mathsf{etb}(C) = \bot$, or $\mathsf{etb}(C) = \top$, or $\mathsf{etb}(C)$ does neither contain $\bot$ nor $\top$.*

We now introduce some simple ways of constructing interpretations that assign specific extensions to name-separated concepts of the form $\mathsf{etb}(\mathsf{NNF}(C))$. For an arbitrary concept C, we write $C^{\mathcal{I}} := C^{\mathcal{J}}$ to express that $\mathcal{I}$ is defined for all symbols X that occur in C by setting $X^{\mathcal{I}} := X^{\mathcal{J}}$. First, we define interpretations that map concepts to the whole domain and to the empty set, respectively.

Definition 6.3. Consider a set Δ, and a name-separated concept $C \neq \bot$ such that $C = \mathsf{etb}(\mathsf{NNF}(C))$. We recursively define an interpretation $\mathcal{I}_\top(C)$ over Δ that is defined on all signature symbols in C:

- If $C = \top$, then $\mathcal{I}_\top(C)$ is defined on no symbol.
- If $C = A$ is an atomic concept, then $A^{\mathcal{I}_\top(C)} := \Delta$.
- If $C = \neg A$ with A atomic, then $A^{\mathcal{I}_\top(C)} := \emptyset$.
- If $C = C_1 \sqcap C_2$ or $C = C_1 \sqcup C_2$, then $C_1^{\mathcal{I}_\top(C)} := C_1^{\mathcal{I}_\top(C_1)}$ and $C_2^{\mathcal{I}_\top(C)} := C_2^{\mathcal{I}_\top(C_2)}$.
- If $C = \exists R.D$, then $R^{\mathcal{I}_\top(C)} := \{\langle \delta, \delta \rangle \mid \delta \in \Delta\}$ and $D^{\mathcal{I}_\top(C)} := D^{\mathcal{I}_\top(D)}$.
- If $C = \forall R.D$, then $R^{\mathcal{I}_\top(C)} := \emptyset$ and the interpretation of symbols in D is arbitrary.

Moreover, for a name-separated concept $C' \neq \top$ with $C' = \mathsf{etb}(\mathsf{NNF}(C'))$, we define an interpretation $\mathcal{I}_\bot(C') := \mathcal{I}_\top(\mathsf{NNF}(\neg C'))$.

Note that $\mathcal{I}_\top(C)$ is well-defined: the case $C = \bot$ cannot occur, and in the cases of $C = C_1 \sqcap C_2$, $C = C_1 \sqcup C_2$, and $C = \exists R.D$, we find $C_1 \neq \bot$, $C_2 \neq \bot$, and $D \neq \bot$, respectively, since $C = \mathsf{etb}(C)$. The following is immediate from the definition.

Lemma 6.4. *Consider a concept $C = \mathsf{etb}(\mathsf{NNF}(C))$. If $C \neq \bot$, then $C^{\mathcal{I}_\top(C)} = \Delta$. If $C \neq \top$, then $C^{\mathcal{I}_\bot(C)} = \emptyset$.*

Next, we define an interpretation $\mathcal{I}[C \rightsquigarrow D]$ that extends an interpretation $\mathcal{I}$ over a concept D to a larger concept C of which D is a subconcept in such a way that $C^{\mathcal{I}[C \rightsquigarrow D]} = D^{\mathcal{I}}$.

Definition 6.5. Consider a name-separated concept C with a non-negated occurrence of a subconcept D. Given an interpretation $\mathcal{I}$ that is defined for symbols in D, we recursively define an interpretation $\mathcal{I}[C \rightsquigarrow D]$ as follows:

- If $C = D$, then $\mathcal{I}[C \rightsquigarrow D] := \mathcal{I}$.
- If $C = C_1 \sqcap C_2$ and D occurs in C_2, then $C_1^{\mathcal{I}[C \rightsquigarrow D]} := C_1^{\mathcal{I}_\top(C_1)}$ and $C_2^{\mathcal{I}[C \rightsquigarrow D]} := C_2^{\mathcal{I}[C_2 \rightsquigarrow D]}$. The case where D occurs in C_1 is analogous.

- If $C = C_1 \sqcup C_2$ and D occurs in C_2, then $C_1^{\mathcal{I}[C\rightsquigarrow D]} := C_1^{\mathcal{I}_\bot(C_1)}$ and $C_2^{\mathcal{I}[C\rightsquigarrow D]} := C_2^{\mathcal{I}[C_2\rightsquigarrow D]}$. The case where D occurs in C_1 is analogous.
- If $C = \exists R.E$ or $C = \forall R.E$, and D occurs in E, then $R^{\mathcal{I}[C\rightsquigarrow D]} := \{\langle\delta,\delta\rangle \mid \delta \in \Delta^{\mathcal{I}}\}$ and $E^{\mathcal{I}[C\rightsquigarrow D]} := E^{\mathcal{I}[E\rightsquigarrow D]}$.

Again, it is easy to see that $\mathcal{I}[C \rightsquigarrow D]$ is well-defined. In particular, the occurrence of D is unique since C is name-separated. In cases $C = C_1 \sqcap C_2$ and $C = C_1 \sqcup C_2$, we find that $C_1, C_2 \notin \{\bot, \top\}$ since $C = \mathrm{etb}(C)$. Moreover, the cases $C = \top$, $C = \bot$, $C = A$, and $C = \neg A$ are subsumed by the case $C = D$. For $C = \neg A$, this follows since D does not occur in a negation.

Lemma 6.6. *If C, D, and $\mathcal{I}$ are as in Definition 6.5, then $C^{\mathcal{I}[C\rightsquigarrow D]} = D^{\mathcal{I}}$.*

Proof. The proof is again immediate in most cases. For the cases of $C = \exists R.E$ and $C = \forall R.E$, it is useful to note:

$$\begin{aligned}
&(\forall R.E)^{\mathcal{I}[C\rightsquigarrow D]}\\
&\quad = \{a \in \Delta \mid b \in E^{\mathcal{I}[E\rightsquigarrow D]} \text{ for all } \langle a,b\rangle \in R^{\mathcal{I}[C\rightsquigarrow D]}\}\\
&\quad = \{a \in \Delta \mid a \in E^{\mathcal{I}[E\rightsquigarrow D]}\} = E^{\mathcal{I}[E\rightsquigarrow D]},\\
&(\exists R.E)^{\mathcal{I}[C\rightsquigarrow D]}\\
&\quad = \{a \in \Delta \mid \text{there is } \langle a,b\rangle \in R^{\mathcal{I}[C\rightsquigarrow D]}\\
&\qquad\qquad \text{with } b \in E^{\mathcal{I}[E\rightsquigarrow D]}\\
&\quad = \{a \in \Delta \mid a \in E^{\mathcal{I}[E\rightsquigarrow D]}\} = E^{\mathcal{I}[E\rightsquigarrow D]}.
\end{aligned}$$

In either case, the claim follows by induction. □

To show that $\mathcal{DLP}_{\mathcal{ALC}}$ is maximal with respect to concept inclusions, we first observe that the use of existential quantification is severely restricted in the concept inclusions of any DLP fragment of $\mathcal{ALC}$. In essence, the following lemma states that etb is able to eliminate all existential quantifiers.

Lemma 6.7. *Let C be a name-separated concept with $C = \mathrm{etb}(\mathsf{NNF}(C))$ that contains a subconcept of the form $\exists R.D$. Then $\top \sqsubseteq C$ cannot be semantically emulated in Datalog.*

Proof. Suppose for a contradiction that $\top \sqsubseteq C$ can be semantically emulated in Datalog. Let Δ be an infinite domain with enumerated elements $\Delta = \{\delta_1, \delta_2, \dots\}$. We define an interpretation $\mathcal{I}$ over Δ by setting $D^{\mathcal{I}} := D^{\mathcal{I}_\top(D)}$ and $R^{\mathcal{I}} := \{\langle\delta_i, \delta_{i+1}\rangle \mid i \geqslant 1\}$. We extend $\mathcal{I}$ to C to obtain an interpretation $\mathcal{J} := \mathcal{I}[C \rightsquigarrow \exists R.D]$. By Lemma 6.4, $D^{\mathcal{I}} = D^{\mathcal{J}} = \Delta$, and thus also $\exists R.D^{\mathcal{I}} = \exists R.D^{\mathcal{J}} = \Delta$. By Lemma 6.6, $C^{\mathcal{J}} = \exists R.D^{\mathcal{I}} = \Delta$.

By assumption, $\top \sqsubseteq C$ can be semantically emulated by some Datalog program P_C. Thus there is an extension $\mathcal{J}_1$ of $\mathcal{J}$ such that $\mathcal{J}_1 \models P_C$. By Definition 3.2, $C^{\mathcal{J}_1} = C^{\mathcal{J}}$ and $R^{\mathcal{J}_1} = R^{\mathcal{J}}$. An element $\delta \in \Delta$ is *anonymous* if no constant (of the extended signature of P_C) is interpreted as δ. Since Δ is infinite, there are infinitely many anonymous elements. Let δ_i and δ_j be two anonymous elements for which there are elements δ_i' and δ_j' such that $\langle\delta_i,\delta_i'\rangle, \langle\delta_j,\delta_j'\rangle \in R^{\mathcal{J}}$ and $\delta_i, \delta_i', \delta_j, \delta_j'$ are pairwise distinct. Such elements exist by construction of $R^{\mathcal{J}}$ and since Δ is infinite.

We define an interpretation $\mathcal{J}_2$ that agrees with $\mathcal{J}_1$ in all aspects, other than the definition of $R^{\mathcal{J}_2}$ which is

$$R^{\mathcal{J}_2} := R^{\mathcal{J}} \cup \{\langle\delta_i,\delta_j'\rangle, \langle\delta_j,\delta_i'\rangle\} \setminus \{\langle\delta_i,\delta_i'\rangle, \langle\delta_j,\delta_j'\rangle\}.$$

It is easy to see that $\mathcal{J}_2 \models P_C$. Indeed, $\mathcal{J}_1$ and $\mathcal{J}_2$ cannot be distinguished by any first-order logic formula. By construction, we still have $C^{\mathcal{J}_2} = \exists R.D^{\mathcal{J}_2} = \Delta$.

By Proposition 3.6, $\mathcal{J}_1 \cap \mathcal{J}_2 \models P_C$. Since $\mathcal{J}_1$ and $\mathcal{J}_2$ agree on all symbols other than R, we still have $C^{\mathcal{J}_1\cap\mathcal{J}_2} = \exists R.D^{\mathcal{J}_1\cap\mathcal{J}_2}$. However, $R^{\mathcal{J}_1\cap\mathcal{J}_2}$ does not contain any pair of the form $\langle\delta_i,\epsilon\rangle$ or $\langle\delta_j,\epsilon\rangle$. Thus, $\delta_i, \delta_j \notin \exists R.D^{\mathcal{J}_1\cap\mathcal{J}_2}$, and hence $C^{\mathcal{J}_1\cap\mathcal{J}_2} \neq \Delta$. Therefore, $\mathcal{J}_1 \cap \mathcal{J}_2$ is a model of P_C that is no model of C, which contradicts Definition 3.2. □

The previous proof uses the intersection of $\mathcal{J}_1$ and $\mathcal{J}_2$ to show that a formula C cannot be semantically emulated in Datalog. Interpretation $\mathcal{J}_1$ is obtained by semantic emulation from a model $\mathcal{J} \models \top \sqsubseteq C$, while $\mathcal{J}_2$ is constructed by modifying $\mathcal{J}_1$. It would be easier to construct two models $\mathcal{I}_1$ and $\mathcal{I}_2$ or C such that $\mathcal{I}_1 \cap \mathcal{I}_2 \not\models \top \sqsubseteq C$, but this would not show the claim, as illustrated by the next example.

Example 6.8. The axiom $\exists R.\top(c)$ is in $\mathcal{DLP}_{\mathcal{ALC}}$, and indeed it can be semantically emulated by the Datalog program $R(c,d)$ where d is a fresh constant. However, there are models $\mathcal{I}_1 \models \exists R.\top(c)$ and $\mathcal{I}_2 \models \exists R.\top(c)$ that agree on constants, whereas $\mathcal{I}_1 \cap \mathcal{I}_2 \not\models \exists R.\top(c)$. For example, let $\Delta := \{c, d_1, d_2\}$, $R^{\mathcal{I}_1} := \{\langle c, d_1\rangle\}$, and $R^{\mathcal{I}_2} := \{\langle c, d_2\rangle\}$. For both $i = 1$ and $i = 2$, $\mathcal{I}_i$ extends to a model $\mathcal{J}_i$ of $R(c,d)$ by setting $d^{\mathcal{J}_i} := d_i$. Proposition 3.6 is not applicable since $\mathcal{J}_1$ and $\mathcal{J}_2$ do not agree on the interpretation of the constant d.

Lemma 6.7 allows us to exclude existential quantification from DLP. It remains to exclude disjunctive in-

formation. Disjunctions of the form $\mathbf{L}_B \sqcup \mathbf{L}_H$ are not problematic – they include DLP axioms like $\neg A \sqcup B$. Our interest is is therefore in disjunctions $C_1 \sqcup C_2$ where neither C_1 nor C_2 is in $\mathbf{L}_B$. The next lemma shows that any such concept contains a positive occurrence of an atomic concept or role.

Lemma 6.9. *Let C be a concept with $C = \mathsf{etb}(\mathsf{NNF}(C))$ such that $C \notin \mathbf{L}_B$. Then C contains a non-negated subconcept that is either an atomic concept A or of the form $\exists R.D$.*

Proof. We show the contrapositive: every concept C with $C = \mathsf{etb}(\mathsf{NNF}(C))$ that contains no subconcept as in the claim is in $\mathbf{L}_B$. This follows from an easy induction over the structure of $\mathcal{ALC}$ concepts. If $C = A$ is atomic, then it has a non-negated atomic subconcept, contradicting our assumption. If $C = \top$, $C = \bot$, or $C = \neg A$ for an atom A, then the claim holds by definition of $\mathbf{L}_B$. If C is of the form $C_1 \sqcap C_2$, $C_1 \sqcup C_2$, or $\forall R.C_1$, then the claim holds for $C_1, C_2 \in \mathbf{L}_B$ by induction hypothesis; thus $C \in \mathbf{L}_B$ by definition of $\mathbf{L}_B$. If $C = \exists R.D$, then it has a non-negated subconcept of the form $\exists R.D$, contradicting our assumption. □

We can now show that DLP cannot express disjunctions in GCIs or concept assertions.

Lemma 6.10. *Let C be a name-separated concept with $C = \mathsf{etb}(\mathsf{NNF}(C))$ that contains a subconcept of the form $C_1 \sqcup C_2$ with $C_1, C_2 \notin \mathbf{L}_B$. Then neither $\top \sqsubseteq C$ nor $C(a)$ can be semantically emulated in Datalog.*

Proof. By Lemma 6.9, each of the concepts C_i ($i \in \{1,2\}$) contains a non-negated subconcept D_i that is either atomic or of the form $\exists R.D$. In the latter case we assume without loss of generality, that D_i is an outermost existential subconcept in C_i, that is, D_i does not occur in a concept E for which $\exists S.E$ is a subconcept of C_i. Let Δ be an arbitrary nonempty domain, and let $\mathcal{I}_i := \mathcal{I}_\top(D_i)$ ($i \in \{1,2\}$). By Lemma 6.4, $D_i^{\mathcal{I}_i} = \Delta$. We extend $\mathcal{I}_i$ to C by defining $\mathcal{J}_i := \mathcal{I}[C \rightsquigarrow D_i]$. By Lemma 6.6, $C^{\mathcal{J}_i} = D_i^{\mathcal{I}_i} = \Delta$. Clearly $\mathcal{J}_i \models \top \sqsubseteq C$ and $\mathcal{J}_i \models C(a)$, where the interpretation $a^{\mathcal{J}_i}$ can be arbitrary.

Suppose for a contradiction that $\top \sqsubseteq C$ can be semantically emulated by a Datalog program P_C. Thus there is an extension $\mathcal{J}_i'$ of $\mathcal{J}_i$ such that $\mathcal{J}_i' \models P_C$. By Definition 3.2, $C^{\mathcal{J}_i'} = C^{\mathcal{J}_i}$. By Proposition 3.6, $\mathcal{J}_1' \times \mathcal{J}_2' \models P_C$.

We claim that $D_1^{\mathcal{J}_2'} = \emptyset$. The definition of $\mathcal{I}[C \rightsquigarrow D_2]$ ensures $(C_1 \sqcup C_2)^{\mathcal{I}[C \rightsquigarrow D_2]} = D_2^{\mathcal{I}_2}$; this is achieved by setting $C_1^{\mathcal{I}[C \rightsquigarrow D_2]} := C_1^{\mathcal{I}_\bot(C_1)}$. We have $\mathcal{I}_\bot(C_1) = \mathcal{I}_\top(\mathsf{NNF}(\neg C_1))$. If D_1 is atomic, it occurs as $\neg D_1$ in $\mathsf{NNF}(\neg C_1)$, and we find $D_1^{\mathcal{I}_\top(\neg D_1)} = D_1^{\mathcal{I}_\bot(\mathsf{NNF}(\neg C_1))} = D_1^{\mathcal{J}_2} = D_1^{\mathcal{J}_2'} = \emptyset$. If D_1 is of the form $\exists R.D$, then it leads to a subconcept $\forall R.\mathsf{NNF}(\neg D)$ in $\mathsf{NNF}(\neg C_1)$, and we find $R^{\mathcal{I}_\bot(\mathsf{NNF}(\neg C_1))} = R^{\mathcal{J}_2} = R^{\mathcal{J}_2'} = \emptyset$. Again, we obtain $D_1^{\mathcal{J}_2'} = \emptyset$. Analogously, one can show $D_2^{\mathcal{J}_1'} = \emptyset$.

We thus find $D_1^{\mathcal{J}_1' \times \mathcal{J}_2'} = D_2^{\mathcal{J}_1' \times \mathcal{J}_2'} = \emptyset$. By induction on the structure of C_i and the corresponding cases in the definition of $\mathcal{I}[C_i \rightsquigarrow D_i]$, one can show $C_1^{\mathcal{J}_1' \times \mathcal{J}_2'} = C_2^{\mathcal{J}_1' \times \mathcal{J}_2'} = \emptyset$. Thus, $(C_1 \sqcup C_2)^{\mathcal{J}_1' \times \mathcal{J}_2'} = \emptyset$. This implies $C^{\mathcal{J}_1' \times \mathcal{J}_2'} = \emptyset$, again by induction on the structure of C.

Thus, $\mathcal{J}_1' \times \mathcal{J}_2' \models P_C$ but $\mathcal{J}_1' \times \mathcal{J}_2' \not\models \top \sqsubseteq C$, contradicting the supposed semantic emulation. The proof for the case of $C(a)$ is similar. In fact, we can use the same interpretations: clearly $\mathcal{J}_1 \models C(a)$, $\mathcal{J}_2 \models C(a)$, and $\mathcal{J}_1' \times \mathcal{J}_2' \not\models C(a)$. □

The main result of this section can now be established by noting that every $\mathcal{ALC}$ axiom that does not contain either of the problematic cases discussed in Lemmas 6.7 and 6.10 is an axiom of $\mathcal{DLP}_{\mathcal{ALC}}$.

Theorem 6.11. *$\mathcal{DLP}_{\mathcal{ALC}}$ is the largest DLP fragment of $\mathcal{ALC}$.*

Proof. Let $\mathcal{F}$ be an arbitrary DLP fragment of $\mathcal{ALC}$, and let $\alpha \in \mathcal{F}$ a name-separated axiom of $\mathcal{F}$. In particular, α can be semantically emulated in Datalog. We show that $\alpha \in \mathcal{DLP}_{\mathcal{ALC}}$. Since α is arbitrary and DLP is closed under variants, this shows that $\mathcal{F} \subseteq \mathcal{DLP}_{\mathcal{ALC}}$.

If α is a GCI, then, by Lemmas 6.7 and 6.10, $\mathsf{etb}(\mathsf{NNF}(\alpha))$ does not contain an existential quantifier or a subconcept $C_1 \sqcup C_2$ with $C_1, C_2 \notin \mathbf{L}_B$. We show that this implies $\mathsf{etb}(\mathsf{NNF}(\alpha)) \in \mathbf{L}_H$, which implies $\alpha \in \mathbf{L}_H$ by Lemma 6.2 and the definition of $\mathcal{DLP}_{\mathcal{ALC}}$. The proof is by induction on the structure of $C = \mathit{etb}(\mathsf{NNF}(\alpha))$. If $C = \top$, $C = \bot$, $C = A$, or $C = \neg A$ for an atomic concept A, then $C \in \mathbf{L}_H$ by the grammar. If $C = C_1 \sqcap C_2$, then, by the induction hypothesis, $C_1, C_2 \in \mathbf{L}_H$; hence $C \in \mathbf{L}_H$. If $C = C_1 \sqcup C_2$, then again $C_1, C_2 \in \mathbf{L}_H$ by induction. In addition, by the assumption on C, we have $C_1 \in \mathbf{L}_B$ or $C_2 \in \mathbf{L}_B$; thus $C \in \mathbf{L}_H$. If $C = \forall R.D$, the claim is again immediate by induction. The case $C = \exists R.D$ cannot occur by our assumption on C.

The proof for the case where α is a concept assertion $C(a)$ is similar, but using only Lemma 6.10 to ensure that $\mathsf{etb}(\mathsf{NNF}(C))$ does not contain a subconcept $C_1 \sqcup C_2$ with $C_1, C_2 \notin \mathbf{L}_B$. Then $\mathsf{etb}(\mathsf{NNF}(C)) \in \mathbf{L}_a$ follows

again by induction, where the case $C = \exists R.D$ is now immediate from the definition of $\mathbf{L}_a$.

Finally, if α is a role assertion $R(a, b)$, then it is clearly in $\mathcal{DLP}_{\mathcal{ALC}}$, too. □

7. The Datalog fragment of $\mathcal{SROIQ}$

The previous sections showed that syntactic descriptions tend to become rather complex when maximising languages in a canonical way, but the situation is substantially more intricate when considering $\mathcal{SROIQ}$ instead of $\mathcal{ALC}$ as an underlying DL. In this section, we discuss some of the related problems by means of examples, and we review related work by Krötzsch and Rudolph, who have identified a DLP fragment of $\mathcal{SROIQ}$ that is maximal under some additional restrictions [21].

Among the additional features of $\mathcal{SROIQ}$, there are two that make the relationship to Datalog highly complicated. *Nominals* are concepts of the form $\{c\}$ for a constant $c \in \mathbf{I}$, which are interpreted as singleton sets $\{c\}^{\mathcal{I}} = \{c^{\mathcal{I}}\}$. Unions of nominals can express concepts of a certain maximal cardinality. *Number restrictions* are concepts of the form $\geqslant n\, R.C$ and $\leqslant n\, R.C$, where n is a non-negative number, and which denote the set of individuals that have at least n and at most n R-successors in C, respectively. In particular, one can express $\exists R.C$ as $\geqslant 1\, R.C$, and $\forall R.C$ as $\leqslant 0\, R.\neg C$.

Example 7.1. The concept $\{a\} \sqcup \{b\} \sqcup \{c\}$ has at most three instances, possibly less if the interpretations of two of the constant symbols coincide. Therefore, the expression $\geqslant 4\, R.(\{a\} \sqcup \{b\} \sqcup \{c\})$ is equivalent to $\bot$, and can thus be expressed in Datalog. The negated concept $\neg \geqslant 4\, R.(\{a\} \sqcup \{b\} \sqcup \{c\})$, which is equivalent to $\leqslant 3\, R.(\{a\} \sqcup \{b\} \sqcup \{c\})$, is therefore equivalent to $\top$.

Note that all concepts in the previous example are name-separated. This illustrates that more elaborate grammars are needed to define classes $\mathbf{L}_\bot$ and $\mathbf{L}_\top$ for $\mathcal{SROIQ}$.

The interaction of nominals, unions, and number restrictions becomes more complex in cases where the cardinality of a nominal equals the cardinality expressed in a restriction.

Example 7.2. The axiom $A \sqsubseteq \exists R.\{c\}$ can be expressed by the Datalog rule $A(x) \rightarrow R(x, c)$. This case is widely known – the ontology language OWL RL even includes a dedicated syntactic construct `ObjectHasValue` for concept expressions of the form $\exists R.\{c\}$ [28].

It is less known that this can be generalised to larger numbers. For instance, the axiom $A \sqsubseteq \geqslant 2\, R.(\{c\} \sqcup \{d\})$ can be expressed by the Datalog program

$$\{A(x) \rightarrow R(x, c),\ A(x) \rightarrow R(x, d),\ A(x) \wedge c \approx d \rightarrow \bot\},$$

which uses the equality predicate $\approx$. It is well-known that $\approx$ can easily be axiomatised in Datalog using a standard equality theory.

Even more intricate types of interaction between nominals and unions can occur in concept assertions.

Example 7.3. In concept assertions, nominals are not necessary to express $\geqslant$-restrictions. For example, $(\geqslant 2\, R.A)(c)$ is semantically emulated by the program

$$\{R(c, s_1),\ R(c, s_2),\ A(s_1),\ A(s_2),\ s_1 \approx s_2 \rightarrow \bot\},$$

where s_1 and s_2 are fresh Skolem constants. This generalises the case of $\exists$ in $\mathcal{DLP}_{\mathcal{ALC}}$ assertions. However, the filler concept A can be significantly more complex and even include disjunction. For example, the assertion $(\geqslant 2\, R.(\neg\{a\} \sqcup A \sqcup B))(c)$ is expressed by the following program

$$\{R(c, s_1),\ R(c, s_2),\ s_1 \approx s_2 \rightarrow \bot,$$
$$a \approx s_1 \rightarrow A(s_1),\ a \approx s_2 \rightarrow B(s_2)\},$$

which allows limited forms of disjunction by using the fact that only at most one of the two Skolem constants can be equal to a. Clearly, there are two distinct R-successors of c in every model. At most one of them can be equal to a (and thus fail to be an instance of $\neg\{a\}$): if $s_1 \approx a$, then $A(s_1)$; if $s_2 \approx a$, then $A(s_2)$. This reasoning is strictly deterministic, yet it expresses a certain form of disjunction. By introducing further pairs of distinct successor constants s_1' and s_2' (not necessarily distinct to s_1 and s_2), one can allow arbitrarily many disjunctive cases, e.g., to encode $(\geqslant 2\, R.(\neg\{a\} \sqcup A \sqcup B \sqcup C))(c)$.

The proof of Lemma 6.10 is not applicable to Example 7.3, since we cannot construct an interpretation $\mathcal{I}_\bot(\neg\{a\})$ with $\neg\{a\}^{\mathcal{I}_\bot(\neg\{a\})} = \emptyset$. More generally, the utility of product models is limited when dealing with nominals. In the above example, $\mathcal{I}_1 \times \mathcal{I}_2 \models s_1 \approx a$ holds only if both $\mathcal{I}_1 \models s_1 \approx a$ and $\mathcal{I}_2 \models s_1 \approx a$. So whenever $\mathcal{I}_1 \times \mathcal{I}_2 \models s_1 \approx a$, we find that $\mathcal{I}_1 \models A(s_1)$ and $\mathcal{I}_2 \models A(s_1)$, and hence $\mathcal{I}_1 \times \mathcal{I}_2 \models A(s_1)$.

Krötzsch and Rudolph consider more complex forms of product constructions to show the limits of

the encoding approach of Example 7.3 [21]. This allows them to show that an expression of the form $(\geqslant n\, R.((\neg\{a_1\}\sqcap\cdots\sqcap\neg\{a_m\})\sqcup A\sqcup B))(c)$ can be semantically emulated in Datalog exactly if $m \leqslant n^2 - n$. Moreover, they discover further kinds of encodings that are different from Example 7.3. A consequence of their findings is, for example, that the assertion $C(e)$ can be expressed in Datalog for $C = \geqslant 4\, R.(A \sqcup \{a\} \sqcup (\{b\} \sqcap \leqslant 1\, S.(\{c\} \sqcup \{d\})))$ but not for $C = \geqslant 3\, R.(A \sqcup \{a\} \sqcup (\{b\} \sqcap \leqslant 1\, S.(\{c\} \sqcup \{d\})))$, although the two cases only differ in the initial number restriction.

Another special case that needs to be considered occurs when a $\mathcal{SROIQ}$ axiom restricts the overall size of the domain.

Example 7.4. The axiom $\top \sqsubseteq \{a\}$ can only be satisfied by models with a domain that has exactly one element, denoted by a. In such a case, every expression of the form $\top \sqsubseteq \leqslant n\, R.C$ with $n \geqslant 1$ is equivalent to $\top$. Thus, the axiom $\top \sqsubseteq \{a\} \sqcap \leqslant 3\, R.(A \sqcup B)$ can be expressed by the (unsafe) Datalog rule $a \approx x$.

Krötzsch and Rudolph systematically study all of these cases to arrive at a grammatical description of a DLP fragment of $\mathcal{SROIQ}$ [21]. The resulting grammar is significantly more complicated than in our case. To curtail this complexity, they additionally require DLP to be closed under the computation of disjunctive normal forms. Clearly, if an axiom can be semantically emulated in Datalog, the same is true for its disjunctive normal form, but not necessarily for all variants (in the sense of Definition 4.1) of this normal form. It is conjectured that this additional requirement is not essential for the existence of a maximal DLP fragment.

Another difference to our work is that $\mathbf{FOL}_{=}$-emulation is used instead of semantic emulation as the main semantic criterion for defining DLP. We conjecture that this does not have any impact on the resulting fragments. It mainly affects the proofs: instead of comparing models of DL axioms and Datalog, one needs to compare entailments. Our model-theoretic constructions, such as $\mathcal{I}_\perp(C)$, are replaced by auxiliary Datalog programs, such as the program $[\![C \sqsubseteq \perp]\!]_{\mathcal{A}}$ that entails that C is empty.

Besides this, the main difference to our work is the significant increase in complexity that makes it harder to follow the essential ideas. The restriction to name-separated axioms in Definition 4.2 cannot prevent this complexity, but it still suffices to ensure the existence of a maximal DLP fragment that can be described by a context-free grammar.

8. Conclusions and outlook

DLP provides an interesting example of a general type of problem: given two knowledge representation (KR) formalisms that can be translated to first-order logic, how can we syntactically characterise all theories of the source formalism that can faithfully be represented in the target formalism? In this work, we proposed to interpret "faithful representation" by means of semantic emulation (a weaker notion of semantic equivalence), while "syntactic" has been realised by requiring closure under variants (non-uniform renamings of signature symbols). These two simple principles allowed us to show the existence of a largest DLP fragment for the DL $\mathcal{ALC}$. In this sense, we argue that our approach introduces a workable definition for the vague notion of the "intersection" of two KR formalisms.

Our rigorous definition of DLP fragments also clarifies the differences between DLP and the DLs $\mathcal{EL}$ and Horn-$\mathcal{SHIQ}$ which can both be expressed in terms of Datalog as well. Neither $\mathcal{EL}$ nor Horn-$\mathcal{SHIQ}$ can be semantically emulated in Datalog but both satisfy a weaker version of syntactic emulation that is obtained by restricting to variable-free formulae φ in Definition 3.3. Under such weaker requirements, a larger space of possible DL fragments is allowed, but it is unknown whether (finitely many) maximal languages exist in this case. There is clearly no largest such language, since both $\mathcal{EL}$ and $\mathcal{DLP}$ abide by the weakened principles whereas their (intractable) union does not (this follows from Proposition 4.4).

Even when weakening the requirements of DLP fragments like this, Horn-$\mathcal{FLE}$ and thus its prominent super-logics Horn-$\mathcal{SHIQ}$ and Horn-$\mathcal{SHOIQ}$ are still excluded by Proposition 4.4, which explains why Horn-$\mathcal{SHIQ}$ cannot be translated to Datalog axiom-by-axiom. In the presence of transitivity, Horn-$\mathcal{SHIQ}$ also is not really closed under variants, but this problem could be overcome by using distinct signature sets for simple and non-simple roles. Again, it is open which results can be established for Horn-$\mathcal{SHIQ}$-like DLs based on the remaining weakened principles.

This work also explicitly introduces a notion of *emulation* that appears to be novel, though loosely related to conservative extensions. In essence, it requires that a theory can take the place of another theory in all logical contexts, based on a given syntactic interface. Examples given in this paper illustrate that this can be very different from semantic equivalence. Yet, emulation can be argued to define minimal requirements

for preserving a theory's semantics even in combination with additional information, so it appears to be a natural tool for enabling information exchange in distributed knowledge systems. We think that the articulation of this notion is useful for studying the semantic interplay of heterogeneous logical formalisms in general.

Finally, the approach of this paper – seeking a logical fragment that is provably maximal under certain conditions – immediately leads to a number of further research questions. For example, what is the maximal fragment of SWRL ("Datalog ∪ $\mathcal{SROIQ}$," see [16]) that can be expressed in $\mathcal{SROIQ}$? Clearly, this fragment would contain DL Rules [24] and maybe some form of DL-safe rules [31]. But also the maximal $\mathbf{FOL}_{=}$ fragment that can be expressed in a well-known subset such as the Guarded Fragment [2] or the two-variable fragment might be of general interest. We argue that ultimate answers to such questions can indeed be obtained based on similar definitions of *fragments* as used for DLP in this work. At the same time, our study of $\mathcal{SROIQ}$ indicates that the required definitions and arguments can become surprisingly complex when dealing with a syntactically rich formalism like description logic. The main reason for this is that constructs that are usually considered "syntactic sugar" have non-trivial semantic effects when considering logical fragments that are closed under variants.

Acknowledgements

The first and the second author have been working at Karlsruhe Institute of Technology when conducting part of the research reported herein. This work was supported by DFG in project *ExpresST* and by EPSRC in projects *ConDOR* and *ExODA*.

References

[1] S. Abiteboul, R. Hull, and V. Vianu, *Foundations of Databases*, Addison Wesley, 1994.

[2] H. Andréka, J.F.A.K. van Benthem, and I. Németi, Modal languages and bounded fragments of predicate logic, *Journal of Philosophical Logic* **27**(3) (1998), 217–274.

[3] F. Baader, D. Calvanese, D. McGuinness, D. Nardi, and P. Patel-Schneider, eds, *The Description Logic Handbook: Theory, Implementation, and Applications*, 2nd edn, Cambridge University Press, 2007.

[4] J.-F. Baget, M. Leclère, M.-L. Mugnier, and E. Salvat, On rules with existential variables: Walking the decidability line, *Artif. Intell.* **175**(9–10) (2011), 1620–1654.

[5] C. Baral and M. Gelfond, Logic programming and knowledge representation, *J. Log. Program.* **19/20** (1994), 73–148.

[6] A. Calì, G. Gottlob, and T. Lukasiewicz, Datalog$^{\pm}$: A unified approach to ontologies and integrity constraints, in: *Proc. 12th Int. Conf. on Database Theory (ICDT'09)*, R. Fagin, ed, ACM, 2009, pp. 14–30.

[7] C.C. Chang and H.J. Keisler, *Model Theory*, 3rd edn, Studies in Logic and the Foundations of Mathematics, Vol. 73, North Holland, 1990.

[8] E. Dantsin, T. Eiter, G. Gottlob, and A. Voronkov, Complexity and expressive power of logic programming, *ACM Computing Surveys* **33**(3) (2001), 374–425.

[9] F.M. Donini, M. Lenzerini, D. Nardi, and A. Schaerf, $\mathcal{AL}$-log: Integrating datalog and description logics, *Journal of Intelligent and Cooperative Information Systems* **10**(3) (1998), 227–252.

[10] T. Eiter, G. Ianni, R. Schindlauer, and H. Tompits, A uniform integration of higher-order reasoning and external evaluations in answer-set programming, in: *Proc. 19th Int. Joint Conf. on Artificial Intelligence (IJCAI'05)*, L.P. Kaelbling and A. Saffiotti, eds, Professional Book Center, 2005, pp. 90–96.

[11] T. Eiter, T. Lukasiewicz, R. Schindlauer, and H. Tompits, Combining answer set programming with description logics for the Semantic Web, in: *Proc. 9th Int. Conf. on Principles of Knowledge Representation and Reasoning (KR'04)*, D. Dubois, C.A. Welty, and M.-A. Williams, eds, AAAI Press, 2004, pp. 141–151.

[12] F. Gasse, U. Sattler, and V. Haarslev, Rewriting rules into $\mathcal{SROIQ}$ axioms, in: *Proc. 21st Int. Workshop on Description Logics (DL'08)*, F. Baader, C. Lutz, and B. Motik, eds, CEUR Workshop Proceedings, Vol. 353, CEUR-WS.org, 2008.

[13] B.N. Grosof, I. Horrocks, R. Volz, and S. Decker, Description logic programs: Combining logic programs with description logic, in: *Proc. 12th Int. Conf. on World Wide Web (WWW'03)*, ACM, 2003, pp. 48–57.

[14] P. Hitzler, M. Krötzsch, and S. Rudolph, *Foundations of Semantic Web Technologies*. Chapman & Hall/CRC, 2009.

[15] I. Horrocks and P.F. Patel-Schneider, A proposal for an OWL rules language, in: *Proc. 13th Int. Conf. on World Wide Web (WWW'04)*, S.I. Feldman, M. Uretsky, M. Najork, and C.E. Wills, eds, ACM, 2004, pp. 723–731.

[16] I. Horrocks, P.F. Patel-Schneider, S. Bechhofer, and D. Tsarkov, OWL Rules: A proposal and prototype implementation, *Journal of Web Semantics* **3**(1) (2005), 23–40.

[17] U. Hustadt, B. Motik, and U. Sattler, Data complexity of reasoning in: Very expressive description logics, in: *Proc. 19th Int. Joint Conf. on Artificial Intelligence (IJCAI'05)*, L.P. Kaelbling and A. Saffiotti, eds, Professional Book Center, 2005, pp. 466–471.

[18] Y. Kazakov, Saturation-based decision procedures for extensions of the guarded fragment, PhD thesis, Universität des Saarlandes, Saarbrücken, Germany, 2006.

[19] B. Konev, C. Lutz, D. Walther, and F. Wolter, Semantic modularity and module extraction in description logics, in: *ECAI*, M. Ghallab, C.D. Spyropoulos, N. Fakotakis, and N. Avouris, eds, IOS Press, 2008, pp. 55–59.

[20] M. Krötzsch, Efficient rule-based inferencing for OWL EL, in: *Proc. 22nd Int. Conf. on Artificial Intelligence (IJCAI'09)*, T. Walsh, ed, IJCAI/AAAI, 2011, pp. 2668–2673.

[21] M. Krötzsch and S. Rudolph, Finding the largest datalog fragment of description logic, Technical Report 3002, Institute

AIFB, Karlsruhe Institute of Technology, 2009. Available online at http://www.aifb.kit.edu/web/Techreport3002.

[22] M. Krötzsch and S. Rudolph, Extending decidable existential rules by joining acyclicity and guardedness, in: *Proc. 22nd Int. Conf. on Artificial Intelligence (IJCAI'09)*, T. Walsh, ed, IJCAI/AAAI, 2011, pp. 963–968.

[23] M. Krötzsch, S. Rudolph, and P. Hitzler, Complexity boundaries for Horn description logics, in: *Proc. 22nd AAAI Conf. on Artificial Intelligence (AAAI'07)*, AAAI Press, 2007, pp. 452–457.

[24] M. Krötzsch, S. Rudolph, and P. Hitzler, Description logic rules, in: *Proc. 18th European Conf. on Artificial Intelligence (ECAI'08)*, M. Ghallab, C.D. Spyropoulos, N. Fakotakis, and N. Avouris, eds, IOS Press, 2008, pp. 80–84.

[25] M. Krötzsch, S. Rudolph, and P. Hitzler, ELP: Tractable rules for OWL 2, in: *Proc. 7th Int. Semantic Web Conf. (ISWC'08)*, A. Sheth, S. Staab, M. Dean, M. Paolucci, D. Maynard, T. Finin, and K. Thirunarayan, eds, LNCS, Vol. 5318, Springer, 2008, pp. 649–664.

[26] A.Y. Levy and M.-C. Rousset, Combining Horn rules and description logics in: CARIN, *Artificial Intelligence* **104**(1–2) (1998), 165–209.

[27] C. Lutz, D. Walther, and F. Wolter, Conservative extensions in: Expressive description logics, in: *Proc. 20th Int. Joint Conf. on Artificial Intelligence (IJCAI'07)*, M.M. Veloso, ed, IJCAI, 2007, pp. 453–458.

[28] B. Motik, B. Cuenca Grau, I. Horrocks, Z. Wu, A. Fokoue, and C. Lutz, eds, OWL 2 Web Ontology Language: Profiles, W3C Recommendation, 27 October 2009. Available at http://www.w3.org/TR/owl2-profiles/.

[29] B. Motik, I. Horrocks, R. Rosati, and U. Sattler, Can OWL and logic programming live together happily ever after? in: *Proc. 5th Int. Semantic Web Conf. (ISWC'06)*, I.F. Cruz, S. Decker, D. Allemang, C. Preist, D. Schwabe, P. Mika, M. Uschold, and L. Aroyo, eds, LNCS, Vol. 4273, Springer, 2006, pp. 501–514.

[30] B. Motik and R. Rosati, A faithful integration of description logics with logic programming, in: *Proc. 20th Int. Joint Conf. on Artificial Intelligence (IJCAI'07)*, M.M. Veloso, ed, IJCAI, 2007, pp. 477–482.

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

[32] M.-L. Mugnier, Ontological query answering with existential rules, in: *Proc. 5th Int. Conf. on Web Reasoning and Rule Systems (RR'11)*, S. Rudolph and C. Gutierrez, eds, LNCS, Vol. 6902, Springer, 2011, pp. 2–23.

[33] M. Ortiz, S. Rudolph, and M. Šimkus, Worst-case optimal reasoning for the Horn-DL fragments of OWL 1 and 2, in: *Proc. 12th Int. Conf. on Principles of Knowledge Representation and Reasoning (KR'10)*, F. Lin, U. Sattler, and M. Truszczynski, eds, AAAI Press, 2010, pp. 269–279.

[34] W3C OWL Working Group, OWL 2 Web Ontology Language: Document Overview, W3C Recommendation, 27 October 2009. Available at http://www.w3.org/TR/owl2-overview/.

[35] R. Rosati, $\mathcal{DL}$+*log*: A tight integration of description logics and disjunctive datalog, in: *Proc. 10th Int. Conf. on Principles of Knowledge Representation and Reasoning (KR'06)*, P. Doherty, J. Mylopoulos, and C.A. Welty, eds, AAAI Press, 2006, pp. 68–78.

[36] R. Volz, *Web Ontology Reasoning with Logic Databases*, PhD thesis, Universität Karlsruhe (TH), Germany, 2004.