A Matter of Principles: Towards the Largest DLP Possible*

Markus Krötzsch and Sebastian Rudolph

Institut AIFB, Universität Karlsruhe, DE

Abstract. Description Logic Programs (DLP) have been described as a description logic (DL) that is in the "expressive intersection" of DL and datalog. This is a very weak guideline for defining DLP in a way that can be claimed to be optimal or maximal in any sense. Moreover, other DL fragments such as $\mathcal{EL}$ and Horn-$\mathcal{SHIQ}$ have also been "expressed" using datalog. Is DLP just one out of many equal DLs in this "expressive intersection"? This paper attempts to clarify these issues by characterising DLP with various design principles that clearly distinguish it from other approaches. A consequent application of the introduced principles leads to the definition of a significantly larger variant of DLP which we conjecture to be maximal in a concrete sense. A preliminary report on the proof of this maximality is provided. While DLP is used as a concrete (and remarkably complex) example in this paper, we argue that similar approaches can be applied to find canonical definitions for other fragments of logical languages.

1 Introduction

Description Logic Programs (DLP) were introduced as a family of fragments of description logic (DL) that can be expressed in first-order Horn-logic [1,2]. Since common reasoning tasks are still undecidable for first-order Horn-logic, its function-free fragment *datalog* is of particular interest, and the term "DLP" today is most commonly used to refer to tractable DLs that can be translated to equisatisfiable datalog. This statement is slightly more concrete than describing DLP as a subset of the "expressive intersection" of DL and datalog [1], but it is still insufficient to characterise DLP. In particular, it is well-known that other tractable DLs such as $\mathcal{EL}$ can also be translated to equisatisfiable datalog programs [3,4]. The union of DLP and $\mathcal{EL}$ is an intractable DL (for some discussion, see [3]), but one may still wonder whether DLP is merely one among several equivalent subsets of the "expressive intersection" of DL and datalog.

But tractability was not among the original design goals of DLP, and one might also weaken this principle to require merely a polytime transformation to datalog. Since reasoning in datalog is still ExpTime complete, this would not preclude intractable logics. 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-$\mathcal{SHIQ}$ for which a satisfiability-preserving datalog transformation is known [5]. However, $\mathcal{EL}$ and DLP can be translated to datalog axiom-by-axiom, i.e. in a *modular* fashion, while the known datalog

* Research reported herein was supported by the EU in the IST project ACTIVE (IST-2007-215040), and by the German Research Foundation under the ReaSem project.

transformation for Horn-$\mathcal{SHIQ}$ needs to consider the whole knowledge base. But how can we be sure that there is no simpler transformation given that both data-complexity and combined complexity of datalog and Horn-$\mathcal{SHIQ}$ agree? The answer is given in Proposition 1 below.

In any case, it is obvious that the design principles for DLP – but also for $\mathcal{EL}$ and Horn-$\mathcal{SHIQ}$ – are not sufficiently well articulated to clarify the distinction between these formalisms. This paper thus gives explicit characterisation of DLP (Section 3), not in terms of concrete syntax but in terms of general design principles, which captures the specifics of the known DLP for datalog. An essential principle is *structurality* of the language: a formula should be in DLP based on its term structure, not based on concrete entity names that it uses. Moreover, we ask whether DLP could be defined as a larger, or even as the *largest*, DL language satisfying the design principles. A significantly larger variant of DLP is introduced in Section 4. This paper does not give a conclusive answer on whether or not this extended DLP is the largest possible, but we conjecture this to be true, and we sketch the proof currently under construction (Section 5). We conclude with an outlook on further application areas for the presented approach (Section 6). Proofs and other details omitted herein for lack of space can be found in [6].

2 Preliminaries

We use $\mathcal{SROIQ}^{\text{free}}$ to denote the DL $\mathcal{SROIQ}$ without simplicity and regularity constraints. Knowledge bases are defined over finite sets of individual names $\mathbf{I}$, concept names $\mathbf{A}$, and role names $\mathbf{R}$, and $\mathscr{S} = \langle \mathbf{I}, \mathbf{A}, \mathbf{R} \rangle$ then is a *signature*. For this paper, we assume that the universal role U is no special symbol (it can still be introduced by suitable axiomatisation), and we write $\exists R.A$ and $\forall R.A$ as $\geqslant 1\, R.A$ and $\leqslant 0\, R.\neg A$, respectively.

We write $\mathsf{NNF}(\mathrm{KB})$ for the negation normal form (NNF) of a knowledge base KB, defined as usual. By $\mathsf{DNF}(\mathrm{KB})$ we denote the disjunctive normal form, which is obtained by exhaustively replacing subconcepts of the form $(C \sqcup D) \sqcap E$ with $(C \sqcap E) \sqcup (D \sqcap E)$. Note that we do not distribute Boolean concept constructors over role restrictions, i.e. our DNF may still contain complex nested concepts.

$\mathcal{SROIQ}^{\text{free}}$ knowledge bases KB (and their signatures) can be expressed as semantically equivalent theories of first-order logic with equality $\mathbf{FOL}_{=}$ (with according signature). When using $\mathcal{SROIQ}$ in the context of $\mathbf{FOL}_{=}$, we will always assume some (arbitrary) such translation to be used. We consider *datalog* to be the Horn-fragment of $\mathbf{FOL}_{=}$ without function symbols; see [6] for further details.

Let F be a $\mathbf{FOL}_{=}$ formula, or a $\mathcal{SROIQ}^{\text{free}}$ axiom or concept expression, and let $\mathscr{S}$ be a signature. An expression F' is a *renaming* 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 language $\mathbf{L}$ over a signature $\mathscr{S}$ is a subset of all $\mathcal{SROIQ}^{\text{free}}$ concept expressions and axioms over $\mathscr{S}$. Infinite signatures are not admissible when studying computational complexities, so we need notation to extend the signature of a DL language: A *language scheme* is a class $\mathcal{L}$ of languages such that (1) for every signature $\mathscr{S}$, there is a language $\mathcal{L}(\mathscr{S})$

over $\mathscr{S}$ in $\mathscr{L}$, and, (2) for any two signatures $\mathscr{S}$ and $\mathscr{S}'$, and every concept expression or axiom C from $\mathscr{L}(\mathscr{S})$, we find that $\mathscr{L}(\mathscr{S}')$ contains every concept expression or axiom C' obtained from C by (uniformly and one-to-one) replacing all occurrences of individual, concept and role names from $\mathscr{S}$ into such from $\mathscr{S}'$. We say that $\mathscr{L}$ *contains* an axiom of concept expression C if there is a signature $\mathscr{S}$ such that $C \in \mathscr{L}(\mathscr{S})$. The notation is extended to knowledge bases KB: we write KB $\in \mathbf{L}$ to express KB $\in 2^{\mathbf{L}}$.

We need a notion of semantic correspondence between logical theories of DL and datalog. Semantic equivalence is too strong – it does not allow the use of auxiliary symbols for expressing a logical relationship – while equisatisfiability is too weak – it allows complex semantic translations that are not tractable. The following is a more appropriate middle-ground:

Definition 1. *Let T and T' be two first-order logic theories and let $\mathscr{S}$ be the signature over which T is defined. We say that T'* emulates *T if for every* $\mathbf{FOL}_{=}$ *formula φ over $\mathscr{S}$, $T' \cup \{\varphi\}$ is satisfiable if and only if $T \cup \{\varphi\}$ is.*

This notion can be generalised by constraining the set of "test formulae" φ. Emulation is loosely related to conservative extensions [7]: every conservative extension of T emulates T. However, for a conservative extension T' of T we have $T \subseteq T'$, while we want the logical (sub)languages of T and T' to be different.

3 Considerations for Defining DLP

In this section, we discuss why defining DLP is not straightforward, and we specify design principles to guide our subsequent definition. The goal is a notion of DLP that is characterised by these principles, as opposed to DLP being some *ad hoc* fragment of DL that just happens to be expressible in datalog. The first design principle fixes our choice of syntax and underlying DL:

DLP 1 (DL Syntax) DLP knowledge bases should be $\mathcal{SROIQ}^{\text{free}}$ knowledge bases.

The second principle states that the semantics of a DLP knowledge base can be expressed in datalog. Since we want to allow the datalog transformation to introduce auxiliary predicate symbols, we require *emulation* (Definition 1) instead of semantic equivalence:

DLP 2 (Semantic Correspondence) There should be a transformation function datalog that maps a DLP knowledge base KB to a datalog program datalog(KB) such that datalog(KB) emulates KB.

DLP 2 is a strong requirement with many useful consequences since it implies that the entailment of any $\mathbf{FOL}_{=}$ formula over the signature of KB can be checked in datalog(KB). Note that this is a stronger requirement than preservation of assertional consequences.

The principles DLP 1 and DLP 2 do not yet provide sufficient details to attempt a definition of DLP. A naive approach could be to define a DL ontology to belong to DLP if it can be expressed by a semantically equivalent datalog program. Such a definition is of little practical use: every inconsistent ontology can trivially be expressed

in datalog, so a DL reasoner is needed to decide whether or not a knowledge base should be considered to be in DLP. This is undesirable from a practical viewpoint. It is thus preferable that a definition can be checked without complex semantic computations:

DLP 3 (Tractability) Containment of a knowledge base KB in a DLP language over some signature $\mathscr{S}$ should be decidable in polynomial time with respect to the size of KB and $\mathscr{S}$.

Syntactic language definitions are often subpolynomial, e.g. if they can be decided in logarithmic space, but polytime language definitions might still be acceptable. For example, simplicity constraints in $\mathcal{SROIQ}$ require polytime checks.

The downside of a syntactic approach is that semantically equivalent transformations on a knowledge base may change its status with respect to DLP. This is not a new problem – many DLs are not syntactically closed under semantic equivalence – but it imposes an additional burden on ontology engineers and implementers. To alleviate this problem, a reasonable further design principle is to require closure under at least some forms of equivalence preserving transformations, such as negation normal form and disjunctive normal form as defined earlier:

DLP 4 (Closure Under NNF and DNF) A knowledge base KB should be in DLP iff its negation normal form $\mathsf{NNF}(\mathrm{KB})$ and its disjunctive normal form $\mathsf{DNF}(\mathrm{KB})$ are.

Closure under NNF will turn out to be mostly harmless, while closure under DNF imposes some real restrictions to our subsequent treatment. We still include it here since it allows us to generally present DL concepts as disjunctions, such that the relationship to datalog rules (disjunctions of literals) is more direct.

The above principles still allow DLP to be defined in such a way that some DLP knowledge base subsumes another knowledge base that is not in DLP. This behaviour is undesirable since it requires implementations and knowledge engineers to consider all axioms of a knowledge base to check if it is in DLP, which motivates the following principle:

DLP 5 (Modularity) Consider two knowledge bases KB_1 and KB_2. Then $\mathrm{KB}_1 \cup \mathrm{KB}_2$ should be in DLP if and only if both KB_1 and KB_2 are. Moreover, in this case the datalog transformation should be $\mathsf{datalog}(\mathrm{KB}_1 \cup \mathrm{KB}_2) = \mathsf{datalog}(\mathrm{KB}_1) \cup \mathsf{datalog}(\mathrm{KB}_2)$.

Modularity changes our goal from defining DLP *knowledge bases* to defining DLP *axioms*. Note that $\mathcal{SROIQ}$ with global constraints (regularity, simplicity) does not satisfy DLP 5 (to see this, set $\mathrm{KB}_1 = \{\mathsf{Tra}(R)\}$ and $\mathrm{KB}_2 = \{\top \sqsubseteq \geqslant 1\, R.\top\}$) which is why we consider $\mathcal{SROIQ}^{\text{free}}$ instead. The above principles already suffice to establish an interesting complexity result:

Proposition 1. *Consider a class K of knowledge bases that belong to a language satisfying DLP 1 to DLP 5, and such that the maximal size of axioms in K is bounded. Then deciding satisfiability of knowledge bases in K is possible in polynomial time.*

Proof. By DLP 2, satisfiability of $\mathrm{KB} \in K$ can be decided by checking satisfiability of $\mathsf{datalog}(\mathrm{KB})$. Assume that the size of axioms in knowledge bases in K is at most n. Up to renaming of symbols, there is only a finite number of different axioms of size n. We can

assume without loss of generality that the transformation datalog produces structurally similar datalog for structurally similar axioms, so that there are only a finite number of structurally different datalog theories datalog($\{\alpha\}$) that can be obtained from axioms α in K. The maximal number of variables occurring within these datalog programs is bounded by some m. By DLP 5, the same holds for all programs datalog(KB) with KB $\in K$. Satisfiability of datalog with at most m variables per rule can be decided in time polynomial in 2^m [8]. Since m is a constant, this yields a polynomial time upper bound for deciding satisfiability of knowledge bases in K. □

The previous result states that reasoning in any DLP language is "almost" tractable. 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, but we will see that not all DLP axioms can be decomposed in DLP. Yet, Proposition 1 shows why Horn-$\mathcal{SHIQ}$ 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 [9].

None of the above principles actually require DLP to contain any knowledge base at all. An obvious approach thus is to define DLP to be the largest language that adheres to all of the chosen design principles. The question to ask at this point is whether this is actually possible: is there a definition of DLP that adheres to the above principles and that includes as many DL axioms as possible? The answer is a resounding no:

Proposition 2. *Consider a language* $\mathbf{L}_{DLP}$ *that adheres to the principles DLP 1 to DLP 5. There is a language* $\mathbf{L}'_{DLP}$ *that adheres to DLP 1 to DLP 5 while covering more knowledge bases, i.e.* $\mathbf{L}_{DLP} \subset \mathbf{L}'_{DLP}$.

This shows that any attempt to arrive at a maximal definition of DLP based on the above design principles must fail. Any definition still requires further choices that, lacking concrete guidelines, are necessarily somewhat arbitrary. While it is certainly useful to capture some general requirements in explicit principles, our approach of defining DLP would not improve over existing *ad hoc* approaches.

The proof of Proposition 2 exploits complexity differences between datalog and DL. Intuitively speaking, a definition of DLP cannot reach the desired maximum since the computations required in this case would no longer be polynomial. DLP 5 does ameliorate the situation slightly by restricting attention to axioms, but DLs can encode complex semantic relationships even within single axioms. The core of Proposition 2 in this sense is that there is no polytime procedure for deciding whether a $\mathcal{SROIQ}$ axiom can be expressed in datalog.

These considerations highlight a strategy for further constraining DLP to obtain a clearly defined canonical definition. Namely, it is necessary to avoid the complicated semantic effects that may arise when considering even single DL axioms. An intuitive reason for the high complexity of evaluating single axioms is that parts of an axiom, even if structurally separated, may semantically affect each other. An important observation is that the semantic interplay of parts of an axiom usually requires entity names to be reused. For example, $\top \sqsubseteq A \sqcap \neg A$ is unsatisfiable since the concept A is used in both conjuncts, while the structurally similar formula $\top \sqsubseteq A \sqcap \neg B$ is satisfiable. So, to prevent such semantic effects from affecting DLP, we can require DLP to be closed under the exchange of entities:

DLP 6 (Structurality) Consider knowledge bases KB and KB′ such that KB′ is an arbitrary renaming KB. Then KB is in DLP iff KB′ is.

Note that renamings need not be uniform, so $A \sqcap \neg B$ can be obtained from $A \sqcap \neg A$. This is a very strong requirement since it forces DLP to be based on the syntactic structure of axioms rather than on the semantic effects that occur for one particular axiom only. Together with modularity (DLP 5), this principle captures the essential difference between a "syntactic" and a "semantic" transformation from DL to datalog. Adhering to DLP 5 and DLP 6, DLP can only include axioms for which all potential semantic effects can be faithfully captured by datalog. Semantic computations for checking satisfiability must be accomplished in datalog, and not during the translation. This intuition turns out to be quite accurate, but more is needed to establish formal results.

Structurality also interacts with normal form transformations. For example, the concept $(\neg A \sqcup \neg B) \sqcap C$ can be emulated in datalog using rules $\rightarrow C(x)$ and $A(x), \wedge B(x) \rightarrow$. But its DNF $(\neg A \sqcap C) \sqcup (\neg B \sqcap C)$ is only in DLP if its renaming $(\neg A \sqcap C) \sqcup (\neg B \sqcap D)$ is, which turns out to be not the case. Therefore, the knowledge base $\{\neg A \sqcup \neg B, C\}$ is in DLP but the knowledge base $\{(\neg A \sqcup \neg B) \sqcap C\}$ is not. We have discussed above why such effects are not avoidable in general. The more transformations are allowed for DLP, the less knowledge bases are contained in DLP. Note that such effects do not occur for negation normal forms.

4 Defining DLP

We now provide a direct definition of DLP and show that the resulting language can be emulated in datalog. We call a language scheme $\mathcal{L}$ a *DLP language scheme* if it adheres to the principles DLP 1–DLP 6 of Section 3. A *DLP language* is a language of the form $\mathcal{L}(\mathscr{S})$ for a signature $\mathscr{S}$ and DLP language scheme $\mathcal{L}$. Without loss of generality, it is assumed that datalog(KB) is independent of the DLP language that KB is taken from.

It turns out that the above characterisation leads to a prohibitively complex syntactic description of the language. Our first goal in this section therefore is to identify ways of simplifying its presentation. Note that it is not desirable to simply eliminate "syntactic sugar" in general, since the very goal of this work is to characterise which $\mathcal{SROIQ}$ knowledge bases can be considered as syntactic sugar for datalog.

Restricting to axioms in negation normal form seems to free us from the burden of explicitly considering negative occurrences of non-atomic concepts. But NNF does not allow for this simplification, since concepts of the form $\leqslant n\, R.D$ still contain D in negative polarity. A modified NNF is more adequate. We thus say that a $\mathcal{SROIQ}^{\text{free}}$ concept expression C is in *positive negation normal form* (pNNF) if (1) all its subexpressions $\leqslant n\, R.D$ have the form $\leqslant n\, R.\neg D'$, and (2) every other occurrence of $\neg$ in C is part of a subconcept $\neg A$ or $\neg\{a\}$ with $A \in \mathbf{A}$, $a \in \mathbf{I}$. Concept expressions C can be transformed into semantically equivalent concept expressions pNNF(C) in positive negation normal form in linear time.

While pNNF effectively reduces the size of a DLP definition by half, the definition is still exceedingly complex. The construction of disjunctive normal forms is compatible with pNNF, so we can additionally require this form of normalisation. Another source of complexity is the fact that $\mathcal{SROIQ}$ features many concept expressions for which

Body concepts: for C in normal form, $C \in \mathbf{D}_B$ iff $C \sqcup A$ (or $\neg C \sqsubseteq A$) is in DLP
$\mathbf{C}_B ::= \neg\mathbf{A} \mid \neg\{\mathbf{I}\} \mid \neg\exists\mathbf{R}.\mathsf{Self} \mid \leqslant 0\,\mathbf{R}.\neg(\mathbf{D}_B \cup \{\bot\}) \mid \mathbf{C}_B \sqcap \mathbf{C}_B$
$\mathbf{D}_B ::= \mathbf{C}_B \mid \mathbf{D}_B \sqcup \mathbf{D}_B$
Head concepts: for C in normal form, $C \in \mathbf{D}_H$ iff $A \sqsubseteq C$ is in DLP
$\mathbf{C}_H ::= \mathbf{C}_B \mid \mathbf{A} \mid \{\mathbf{I}\} \mid \exists\mathbf{R}.\mathsf{Self} \mid \geqslant n\,\mathbf{R}.\mathbf{D}_{n!} \mid \leqslant 0\,\mathbf{R}.\neg\mathbf{D}_H \mid \leqslant 1\,\mathbf{R}.\neg(\mathbf{D}_B \cup \{\bot\}) \mid \mathbf{C}_H \sqcap \mathbf{C}_H \mid \mathbf{D}_{1!}$
$\mathbf{D}_H ::= \mathbf{C}_H \mid \mathbf{D}_H \sqcup \mathbf{D}_B \mid \mathbf{D}_a \sqcup \mathbf{C}_{\geq}$
Assertional concepts: for C in normal form, $C \in \mathbf{D}_a$ iff $\{a\} \sqsubseteq C$ is in DLP
$\mathbf{C}_a ::= \mathbf{C}_H \mid \geqslant n\,\mathbf{R}.\mathbf{D}^{\geqslant n} \mid \mathbf{C}_a \sqcap \mathbf{C}_a$
$\mathbf{D}_a ::= \mathbf{C}_a \mid \mathbf{D}_a \sqcup \mathbf{D}_B$
Disjunctions of nominal assertions of the form $\{\mathbf{I}\} \sqcap \mathbf{C}_a$
$\mathbf{D}_{1!} ::= \{\mathbf{I}\} \sqcap \mathbf{C}_a$
$\mathbf{D}_{m+1!} ::= \mathbf{D}_{m!} \sqcup \mathbf{D}_{1!}$
Conjunction of negated nominals, i.e. complements of some nominal disjunction
$\mathbf{C}_{\geq} ::= \neg\{\mathbf{I}\} \mid \mathbf{C}_{\geq} \sqcap \mathbf{C}_{\geq}$
Filler concepts for $\geqslant n$ in $\mathbf{D}_a$
$\mathbf{D}^{\geqslant n} ::= \top \mid \mathbf{C}_{\geq} \sqcup \mathbf{D}_a^+ \mid \mathbf{D}_B \sqcup (\mathbf{D}_{\leq m} \cap \mathbf{D}_a^+) \quad (m < n) \mid$ $\mathbf{D}_a \sqcup (\mathbf{D}_{\leq m} \cap \mathbf{D}_a^+) \sqcup \mathbf{D}_{l!}$ (for $r := n - (m + l)$ we have $r > 0$ and $r(r-1) \geq m$) where no disjuncts are added for $\mathbf{D}_{\leq 0} \cap \mathbf{D}_a^+$ and $\mathbf{D}_{0!}$
Extended concepts with restricted forms of ("local") disjunctions, used in $\mathbf{D}^{\geqslant n}$ only
$\mathbf{C}_H^+ ::= \mathbf{C}_H \mid \geqslant n\,\mathbf{R}.\mathbf{D}_{n!}^+ \mid \leqslant 0\,\mathbf{R}.\neg\mathbf{D}_H^+ \mid \leqslant n\,\mathbf{R}.\neg(\mathbf{D}_{\geq\omega-m} \cap \mathbf{D}_a^+) \mid \mathbf{C}_H^+ \sqcap \mathbf{C}_H^+ \mid \mathbf{D}_{1!}^+$
$\mathbf{D}_H^+ ::= \mathbf{C}_H^+ \mid \mathbf{D}_H^+ \sqcup \mathbf{D}_B \mid \mathbf{D}_a^+ \sqcup \mathbf{C}_{\geq}$
$\mathbf{C}_a^+ ::= \mathbf{C}_H^+ \mid \geqslant n\,\mathbf{R}.(\mathbf{D}_a^+ \cup \{\top\}) \mid \mathbf{C}_a^+ \sqcap \mathbf{C}_a^+$
$\mathbf{D}_a^+ ::= \mathbf{C}_a^+ \mid \mathbf{D}_a^+ \sqcup \mathbf{D}_a^+$
$\mathbf{D}_{1!}^+ ::= \{\mathbf{I}\} \sqcap \mathbf{C}_a^+$
$\mathbf{D}_{m+1!}^+ ::= \mathbf{D}_{m!}^+ \sqcup \mathbf{D}_{1!}^+$
$\mathbf{D}_{\leq n}$ ($\mathbf{D}_{\geq\omega-n}$) concepts contain (exclude) at most n domain elements, see [6].

Fig. 1. Grammars for defining DLP concepts

all possible renamings are necessarily equivalent to $\top$ or $\bot$. Examples include $\top \sqcup C$, $\geqslant 0\,R.C$, but also $\leqslant 3\,R.\{a\} \sqcup \{b\}$.

Many unexpected DLP axioms are based on the observation that some DL concepts do not allow for arbitrary interpretations. This is especially the case for concepts that use nominals, but even DLs without nominals admit such constrained concept expressions. An important special case are concepts that can only represent $\top$ or $\bot$ in any renaming. We say that a $\mathcal{SROIQ}$ concept expression C is *structurally valid* (*structurally unsatisfiable*) if $\top \sqsubseteq C'$ ($C' \sqsubseteq \bot$) is valid for every renaming (over arbitrary alphabets) C' of C. Moreover, C is *structurally refutable* (*structurally satisfiable*) if it is not structurally valid (structurally unsatisfiable). It can be shown that the sets of all $\mathcal{SROIQ}$ concept expressions in pNNF that are structurally valid, unsatisfiable, refutable, or satisfiable can be recognised in polynomial time (see [6]). Thus we can also eliminate such concepts from our considerations.

Definition 2. *A concept expression C is in* DLP normal form *if* $C = \mathsf{DNF}(\mathsf{pNNF}(C))$ *and*

- *if C has a structurally valid subconcept D, then $D = \top$ and either $C = D$ or D occurs in a subconcept of the form $\geqslant n\, R.D$,*
- *if C has a structurally unsatisfiable subconcept D, then $D = \bot$ and either $C = D$ or D occurs in a subconcept of the form $\leqslant n\, R.\neg D$.*

A $\mathcal{SROIQ}$ concept expression can be transformed into an equivalent expression in DLP normal form in polynomial time. The following definitions of DLP thus restrict to concepts in DLP normal form. Commutativity and associativity of $\sqcap$ and $\sqcup$ are exploited for further simplification.

Before providing the full definition of a large DLP language scheme, we provide some interesting examples to sketch the complexities of this endeavour (datalog emulations are provided in parentheses). DLP expressions of the form $A \sqcap \exists R.B \sqsubseteq \forall S.C$ ($A(x) \wedge R(x,y) \wedge B(y) \wedge S(x,z) \rightarrow C(z)$) are well-known. The same is true for $A \sqsubseteq \exists R.\{c\}$ ($A(x) \rightarrow R(x,c)$) but hardly for $A \sqsubseteq \geqslant 2\, R.(\{c\} \sqcup \{d\})$ ($A(x) \rightarrow R(x,c)$, $A(x) \rightarrow R(x,d)$). Another unusual form of DLP axioms arises when Skolem constants (not functions) can be used as in the case $\{c\} \sqsubseteq \geqslant 2\, R.A$ ($R(c,s)$, $R(c,s')$, $A(s)$, $A(s')$, $s \approx s' \rightarrow \bot$ with fresh s, s') and $A \sqsubseteq \exists R.(\{c\} \sqcap \exists S.\top)$ ($A(x) \rightarrow R(x,c)$, $S(c,s)$ with fresh s). Besides these simple cases, there are various DLP axioms for which the emulation in datalog is significantly more complicated, typically requiring an exponential number of rules. Examples are $\{c\} \sqsubseteq \geqslant 2\, R.(\neg\{a\} \sqcup A \sqcup B)$ and $\{c\} \sqsubseteq \geqslant 5\, R.(A \sqcup \{a\} \sqcup (\{b\} \sqcap \leqslant 1\, S.(\{c\} \sqcup \{d\})))$. These cases are based on the complex semantic interactions between nominals and atleast-restrictions.

Definition 3. *We define the language scheme $\mathcal{DLP}$ as follows. For a signature $\mathcal{S}$, the language $\mathcal{DLP}(\mathcal{S})$ contains all axioms which are $\mathcal{SROIQ}^{\text{free}}$ RBox axioms over $\mathcal{S}$, or GCIs $C \sqsubseteq D$ over $\mathcal{S}$ where* $\mathsf{pNNF}(\neg C \sqcup D)$ *is a $\mathbf{C}_{DLP}$ concept as defined in the following grammars, with $\mathbf{C}_H$ as defined in Fig. 1, and $\mathbf{D}^{=n}$ and $\mathbf{C}_{\neq\top}$ as defined in Fig. 2:*

$$\mathbf{C}_{DLP} ::= \top \mid \bot \mid \mathbf{C}_H \mid \mathbf{D}^{=n} \mid \mathbf{C}_{\neq\top}$$

In spite of the immense simplifications that DLP normal form provides, the definition of $\mathcal{DLP}$ still turns out to be extremely complex. We have not succeeded in simplifying the presentation any further without loosing substantial expressive features. Some intuitive explanations help to understand the underlying ideas. It is instructive to also compare these intuitions to the above examples.

The core language elements are in Fig. 1. Since all concepts are in DNF, each sublanguage consists of a conjunctive part **C** and a disjunctive part **D**. Definitions of DLP typically distinguish between "head" and "body" concepts, and $\mathbf{C}_H$ and $\mathbf{C}_B$ play a similar role in our definition. $\mathbf{C}_H$ represents concepts that carry the full power of a DLP GCI and that can serve as right hand sides ("heads") of DLP GCIs. $\mathbf{C}_B$ concepts can be seen as negated generic left hand sides ("bodies") of GCIs. However, these basic classes are not sufficient for defining a maximal DLP. $\mathbf{C}_a$ characterises concept expressions which can be asserted for named individuals – these are even more expressive than $\mathbf{C}_H$ in that existential restrictions are allowed (intuitively, this is possible as in the context of known individuals the existentially asserted role neighbours can be expressed by

Additional concepts based on global domain size restrictions	
$\mathbf{D}^{=1}$	$::= \{\mathbf{I}\} \sqcap \mathbf{C}_H^p$
$\mathbf{D}^{=m+1}$	$::= \mathbf{D}^{=m} \sqcup (\{\mathbf{I}\} \sqcap \mathbf{C}_\bot^{=m+1})$
Additional head and body concept expressions for unary domains ("propositional" case)	
$\mathbf{C}_B^p$	$::= \mathbf{C}_\bot^{=1} \mid \neg\mathbf{A} \mid \neg\exists\mathbf{R}.\mathsf{Self} \mid \mathbf{C}_B^p \sqcap \mathbf{C}_B^p \mid \leqslant 0\,\mathbf{R}.\neg(\mathbf{D}_B^p \cup \{\bot\}) \mid \leqslant n\,\mathbf{R}.\neg\mathbf{C}\ (n \geq 1)$
$\mathbf{D}_B^p$	$::= \mathbf{D}_B^p \mid \mathbf{D}_B^p \sqcup \mathbf{D}_B^p$
$\mathbf{C}_H^p$	$::= \mathbf{C}_B^p \mid \mathbf{A} \mid \{\mathbf{I}\} \mid \exists\mathbf{R}.\mathsf{Self} \mid \mathbf{C}_H^p \sqcap \mathbf{C}_H^p \mid \geqslant 1\,\mathbf{R}.\mathbf{D}_H^p \mid \leqslant 0\,\mathbf{R}.\neg\mathbf{D}_H^p$
$\mathbf{D}_H^p$	$::= \mathbf{C}_H^p \mid \mathbf{D}_H^p \sqcup \mathbf{D}_B^p$
Additional structurally unsatisfiable concepts for domains of restricted size	
$\mathbf{C}_\bot^{=1}$	$::= \neg\{\mathbf{I}\} \mid \mathbf{C}_\bot^{=1} \sqcap \mathbf{C} \mid \geqslant n\,\mathbf{R}.\mathbf{D}_\bot^{=1}\ (n \geq 0) \mid \geqslant n\,\mathbf{R}.\mathbf{D}\ (n \geq 2)$
$\mathbf{C}_\bot^{=m+1}$	$::= \mathbf{C}_\bot^{=m+1} \sqcap \mathbf{C} \mid \geqslant n\,\mathbf{R}.\mathbf{D}_\bot^{=m+1}\ (n \geq 0) \mid \geqslant n\,\mathbf{R}.\mathbf{D}\ (n \geq m+2)$
$\mathbf{D}_\bot^{=m}$	$::= \mathbf{C}_\bot^{=m} \mid \mathbf{D}_\bot^{=m} \sqcup \mathbf{D}_\bot^{=m}$
Concepts that can never hold for all individuals	
$\mathbf{C}_{\neq\top}$	$::= \neg\{\mathbf{I}\} \mid \mathbf{C}_{\neq\top} \sqcap \mathbf{C}$
D: concepts in DLP normal form that are not structurally valid or unsatisfiable	
C: concepts of **D** that are no disjunctions	

Fig. 2. Grammars for defining DLP concepts: special cases with restricted domain size

Skolem constants). $\mathbf{D}_{m!}$ concepts then can be viewed as collections of individual assertions (e.g. $\{a\} \sqcap B$). Another way of stating such assertions is to use $\mathbf{C}_{\geq}$ in a disjunction (e.g. $\neg\{a\} \sqcup B$).

By far the most complex semantic interactions occur for atleast-restrictions in ABox assertions: $\mathbf{D}^{\geqslant n}$ and all subsequent definitions address this single case. For example, the $\mathcal{DLP}$ axiom $\{a\} \sqsubseteq \geqslant 2\,R.(\neg\{b\} \sqcup A \sqcup B)$ can be emulated by the following set of datalog rules, where c_i are auxiliary constants:

$$R(a, c_1), \quad R(a, c_2), \quad b \approx c_1 \rightarrow A(b), \quad b \approx c_2 \rightarrow B(b).$$

This emulation uses internal symbols to resolve apparently disjunctive cases in a deterministic way. The datalog program does not represent disjunctive information: its least model simply contains two successors that are not equal to b. The nested disjunction only becomes relevant in the context of some disjunctive $\mathbf{FOL}_{=}$ formula, such as $\forall x.x \approx a \vee x \approx b$. The considered theory is no longer datalog in this case, and the program simply "re-uses" the disjunctive expressive power provided by the external theory. The fact that the actual program is far from being semantically equivalent to the original axiom illustrates the motive and utility of our definition of emulation.

Many uses of nominals and atleast-restrictions lead to more complex interactions, some of which require completely different encodings. This is witnessed by the more complex arithmetic side condition used in $\mathbf{D}^{\geqslant n}$. Concepts in $\mathbf{D}_{\leq m} \cap \mathbf{D}_a^+$ correspond to disjunctions of m nominal classes, each of which is required to satisfy further disjunctive conditions, as e.g. $\{b\} \sqcap \geqslant 1\,R.(A \sqcup B)$. Now a disjunction of an atomic class and four such "disjunctive nominals" is allowed as a filler for $\geqslant 7$ (since $3 \times 2 \geq 4$) but not for $\geqslant 6$ (since $2 \times 1 < 4$). Also note that the disjunctive concepts like $\mathbf{D}_H^+$ and $\mathbf{D}_a^+$ that are allowed in fillers do not allow all types of disjunctive information but only a finite

amount of "local" disjunctions. For example, $\{a\} \sqcup B \sqcup C$ requires one "local" decision about a, whereas concepts like $\{a\} \sqcap \leqslant 0\, R.\neg(B \sqcup C)$ or $\{a\} \sqcap \leqslant 2\, R.\neg\bot$ require arbitrarily many decisions for all R successors.

The remaining grammars in Fig. 2 take care of less interesting special cases. Most importantly, $\mathbf{C}_H^p$ covers all concepts that can be emulated if the interpretation domain is restricted to contain just one individual. $\mathbf{C}_{\neq\top}$ contains axioms which make the knowledge base inconsistent as they deny the existence of a nominal.

Further details about the datalog transformation of $\mathcal{DLP}$ are provided in [6].

5 Maximality of DLP

We conjecture that the DL $\mathcal{DLP}$ of Definition 3 is the largest DLP language scheme. Proving this is not straightforward, since it requires us to ensure that no further kind of axioms could admit a datalog emulation. The definition of $\mathcal{DLP}$ is also a result of (sometimes surprising) failures in trying to prove this result. The earlier example emulations already hint at the complexity of the problem. The general proof technique is sketched in the following, and further updates on the status of the maximality conjecture are given in the technical report [6]. Structurality (DLP 6) is essential throughout the proof since it enables us to pick suitable renamings for each argument.

A useful observation is that any DLP language scheme can be extended to include all axioms of $\mathcal{DLP}$, since it can be shown that the union of any two DLP language schemes is still a DLP language scheme. Equipped with the basic toolbox of $\mathcal{DLP}$ axioms, it can then be shown that certain basic types of axioms can never be in DLP since their emulation would contradict basic properties of datalog. Examples of two basic such properties are the least model property and the complexity result of Proposition 1. The former entails that, for any datalog program P, if P has a model where the extension of predicate A is empty, and another model where the extension of predicate B is empty, then P has a model where both extensions are empty. This precludes datalog from emulating statements like $A \sqcup B$. Complexity properties can also be exploited:

Lemma 1. *No DLP language scheme contains axioms of the form $A \sqsubseteq \exists R.\top$.*

Proof. For a contradiction, suppose that there is a DLP language scheme that includes axioms of the form $A \sqsubseteq \exists R.\top$. We can assume that all $\mathcal{DLP}$ axioms are also available. The hardness proof given for Horn-$\mathcal{FL}^-$ in [9] can be adopted to show that deciding satisfiability for this DLP language is PSpace hard, even if axiom sizes are bounded. Since P $\neq$ PSpace, this contradicts Proposition 1. □

Both of these simple arguments do not extend to more general cases. More potent approaches are provided by model-theoretic properties that generalise the least model property to first-order interpretations: the *product model* and *product element* construction, see [6] for details. Applying these approaches to all cases requires a careful induction for various language definitions of $\mathcal{DLP}$. Individual proof steps can be intricate in some cases, e.g. to see why no datalog program can emulate $\{a\} \sqsubseteq \geqslant 2\, R.(A \sqcup (\{b\} \sqcap \geqslant 1\, S.(C \sqcup D))$ while it could emulate $\{a\} \sqsubseteq \geqslant 2\, R.(A \sqcup \{b\} \sqcup \{c\})$ and $\{a\} \sqsubseteq \geqslant 3\, R.(A \sqcup (\{b\} \sqcap \geqslant 1\, S.(C \sqcup D))$.

6 Conclusions and Outlook

DLP provides an interesting example for the general problem of characterising syntactic fragments of a logic that are motivated by semantic properties. We derived and motivated a number of design principles for achieving such a characterisation for DLP, most notably the principles of *modularity* (closure under unions of knowledge bases) and *structurality* (closure under non-uniform renaming of signature symbols). We conjecture that the presented DLP language scheme is the largest one possible. Experiences with the ongoing proof of maximality confirm the utility of structurality in such proofs. Formalisms like our maximal DLP are unnecessarily large for practical applications, but understanding overall options and underlying design principles is indispensable for making an informed choice of DL for a concrete task.

Our results also clarify the differences between DLP and the DLs $\mathcal{EL}$ and Horn-$\mathcal{SHIQ}$ which can be expressed in datalog as well. First of all, neither $\mathcal{EL}$ nor Horn-$\mathcal{SHIQ}$ can be emulated in datalog (DLP 2). Instead, $\mathcal{EL}$ and Horn-$\mathcal{SHIQ}$ satisfy a weaker version of DLP 2 where Definition 1 is restricted to test formulae φ that are conjunctions of simple ABox facts. This weakening of DLP 2 allows for a larger space of possible DL fragments, but it is not clear 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. The weakened principles still exclude Horn-$\mathcal{SHIQ}$ that is not modular (DLP 5), as shown by Proposition 1. It is possible to define Horn-$\mathcal{SHIQ}$ as a structural language (DLP 6) 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 semantic *emulation* which 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 emulation can be very different from semantic equivalence. Yet, our criteria can be argued to define minimal requirements for preserving a theory's semantics even in combination with additional information, so emulation appears to be a natural tool for enabling information exchange in distributed knowledge systems. We expect that the explicit articulation of this notion will be useful for studying the semantic interplay of heterogeneous logical formalisms in general.

The general approach of this paper – seeking a structural logical fragment that is provably maximal under certain conditions – leads to a number of further research questions. For example, what is the maximal fragment of SWRL ("datalog $\cup$ $\mathcal{SROIQ}$") that can be expressed in $\mathcal{SROIQ}$? It should contain DL Rules [10] and some form of DL-safe rules [11]. But also the maximal $\mathbf{FOL}_{=}$ fragment that can be expressed in the guarded fragment or the two-variable fragment might be of general interest. Ultimate answers to such questions may indeed be obtained by a careful articulation of basic design principles.

References

1. Grosof, B., Horrocks, I., Volz, R., Decker, S.: Description Logic Programs: Combining logic programs with description logic. In: Proceedings of the 12th International World Wide Web Conference (WWW-03), Budapest, Hungary, ACM (2003) 48–57
2. Volz, R.: Web Ontology Reasoning with Logic Databases. PhD thesis, Universität Karlsruhe (TH), Germany (2004)
3. Krötzsch, M., Rudolph, S., Hitzler, P.: ELP: Tractable rules for OWL 2. Technical report, Universität Karlsruhe, Germany (May 2008) `http://korrekt.org/page/ELP`.
4. Kazakov, Y.: Saturation-Based Decision Procedures for Extensions of the Guarded Fragment. PhD thesis, Universität des Saarlandes, Germany (2006)
5. Hustadt, U., Motik, B., Sattler, U.: Data complexity of reasoning in very expressive description logics. In: Proc. 18th Int. Joint Conf. on Artificial Intelligence (IJCAI-05), Edinburgh, UK, Morgan-Kaufmann Publishers (2005) 466–471
6. Krötzsch, M., Rudolph, S.: The largest DLP possible. Technical report, Universität Karlsruhe, Germany (APR 2009) Available at `http://korrekt.org/page/MaxDLP`.
7. Lutz, C., Walther, D., Wolter, F.: Conservative extensions in expressive description logics. In Veloso, M.M., ed.: IJCAI. (2007) 453–458
8. Dantsin, E., Eiter, T., Gottlob, G., Voronkov, A.: Complexity and expressive power of logic programming. ACM Computing Surveys **33** (2001) 374–425
9. Krötzsch, M., Rudolph, S., Hitzler, P.: Complexity boundaries for Horn description logics. In: Proceedings of the 22nd AAAI Conference on Artficial Intelligence, Vancouver, British Columbia, Canada, AAAI Press (2007) 452–457
10. Krötzsch, M., Rudolph, S., Hitzler, P.: Description logic rules. In: Proc. 18th European Conf. on Artificial Intelligence (ECAI-08), IOS Press (2008) 80–84
11. Motik, B., Sattler, U., Studer, R.: Query answering for OWL DL with rules. J. of Web Semantics **3** (2005) 41–60