Finding the Largest Datalog Fragment of Description Logic

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. So 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 show to be maximal in a concrete sense. 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, such as the "maximal" fragment of SWRL rules that can be expressed in the DL $\mathcal{SROIQ}$.

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 [5,15]. 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 [5], 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 [11,8]. It is known that the union of DLP and $\mathcal{EL}$ is an intractable DL (for some discussion, see [11]), 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 semantics-preserving transformation to datalog. 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 [7]. However, $\mathcal{EL}$ and DLP can be translated to datalog axiom-by-axiom, i.e. in a *modular* fashion, while the known datalog 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 2 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 those formalisms. This paper thus approaches an explicit characterisation of DLP, 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 that satisfies our design principles. A positive answer to this question is given by introducing a significantly larger variant of DLP that is proven to be a maximal DLP description logic in the sense of this work.

This paper begins with some preliminary definitions in Section 2. In Section 3, we discuss the problems of characterising DLP and provide some fundamental results. Section 4 provides a simplified version of the main results by restricting attention to the smaller description logic $\mathcal{ALC}$, where it is significantly easier to define a DLP fragment and prove its maximality. These simplifications allow us to outline the general proof structure and some relevant methods, but they do neither cover all relevant parts of earlier DLP definitions nor all relevant proof techniques needed in the general case. A full definition for an extended language $\mathcal{DLP}$ is then provided in Section 5. In Section 6, we show how $\mathcal{DLP}$ can be expressed using datalog. Section 7 discussed some important model-theoretic constructions for characterising fragments of first-order logic that can be expressed in datalog. These constructions are then used as a basis for showing maximality of $\mathcal{DLP}$ in Section 8. Section 9 provides a short outlook on further application areas for the presented approaches.

2 Preliminaries

We consider the well-known description logic $\mathcal{SROIQ}$ as defined in [6]. As we are mainly interested in the syntax of the language to be defined, we consider $\mathcal{SROIQ}^{\text{free}}$ denoting $\mathcal{SROIQ}$ without simplicity and regularity constraints. In particular $\mathcal{SROIQ}^{\text{free}}$ allows arbitrary role inclusion axioms $R_1 \circ \ldots \circ R_n \sqsubseteq R$. Clearly, the semantics of $\mathcal{SROIQ}^{\text{free}}$ follows from that of $\mathcal{SROIQ}$. As usual, $\mathcal{SROIQ}^{\text{free}}$ knowledge bases are defined over finite sets of individual names $\mathbf{I}$, concept names $\mathbf{A}$, and roles $\mathbf{R}$. For the purpose of this paper, we assume that $\mathbf{R}$ includes inverse roles, i.e. that for each $R \in \mathbf{R}$ there is an inverse $\text{Inv}(R) \in \mathbf{R}$ such that $\text{Inv} : \mathbf{R} \to \mathbf{R}$ is bijective, symmetric, and irreflexive. We call $\mathscr{S} = \langle \mathbf{I}, \mathbf{A}, \mathbf{R} \rangle$ *signature*, and all signatures are assumed to be finite in this paper. A signature $\mathscr{S}' = \langle \mathbf{I}', \mathbf{A}', \mathbf{R}' \rangle$ is called an *extension* of $\mathscr{S}$, if $\mathbf{I} \subseteq \mathbf{I}'$ and $\mathbf{A} \subseteq \mathbf{A}'$ and $\mathbf{R} \subseteq \mathbf{R}'$.

Our work leads to rather complex syntactic descriptions of DL languages, so it is desirable to simplify syntax as much as possible early on. Unfortunately, expressive features that can be considered as syntactic sugar in $\mathcal{SROIQ}$ may not be syntactic sugar in the restricted DL fragments we study. For example, the symbol $\top$ cannot be expressed by $A \sqcup \neg A$ in DLP. Thus, in general, the precise set of available operators influences the definition and expressivity of DLP. Yet, we do assume within this paper that the universal role U is *not* a specific logical symbol, but that it is only available through axiomatisation.[1] Omitting U as a language construct significantly simplifies the complexity of the definitions we arrive at. Some further syntactic simplification can be assumed without any reservations: we always write $\exists R.A$ and $\forall R.A$ as $\geqslant 1\, R.A$ and $\leqslant 0\, R.\neg A$, respectively, and we omit syntactic forms that can be derived by exchanging operators in conjunctions and disjunctions, i.e. we specify grammars only up to commutativity and associativity of $\sqcap$ and $\sqcup$.

Every $\mathcal{SROIQ}^{\text{free}}$ GCI $C \sqsubseteq D$ can be expressed by $\top \sqsubseteq \neg C \sqcup D$ (i.e. by stating that the concept $\neg C \sqcup D$ is universally valid). In the following, we will often tacitly assume that GCIs are expressed as universally valid concepts. For further simplification, we consider various syntactic normal forms. 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. Our later definition of DLP is not generally closed under such transformations: requiring closure under stronger normalisations reduces the amount of knowledge bases that the definition covers.

Obviously, $\mathcal{SROIQ}^{\text{free}}$ knowledge bases KB can be expressed as semantically equivalent theories of first-order logic with equality ($\mathbf{FOL}_=$), where $\mathbf{I}$, $\mathbf{A}$, $\mathbf{R}$ take the rôles of constants, unary predicates and binary predicates, respectively. We will use $\pi(\mathrm{KB})$ to denote one (arbitrary) such translation, and we will consider signatures of $\mathcal{SROIQ}$ as $\mathbf{FOL}_=$ signatures when convenient, but we will assume that only individual names (constants), concept names (unary predicates), and roles (binary predicates) are present in any considered $\mathbf{FOL}_=$ signature.

We use the term "*datalog*" to refer to the function-free Horn logic fragment of $\mathbf{FOL}_=$.[2] A *datalog program* over a first-order signature $\mathscr{S}$ is a first-order theory over $\mathscr{S}$ which contains only Horn clauses, i.e. $\mathbf{FOL}_=$ formulae of one of the forms

$$\begin{array}{l}\forall \mathbf{x}.(A_1 \wedge \ldots \wedge A_n \to A)\\ \forall \mathbf{x}.(A_1 \wedge \ldots \wedge A_n \to \bot)\\ \forall \mathbf{x}.(\top \to A),\end{array}$$

where $A_{(i)}$ are atoms over $\mathscr{S}$ that contain no function symbols, and $\forall \mathbf{x}$ quantifies over all variables occurring in the implications. For clarity, we use the nullary operators $\top$ and $\bot$ that are interpreted as *true* and *false*, respectively. We will follow the common practice of omitting the quantifiers, and of writing facts $\top \to A$ as A.

Our discussion is necessarily based on a notion of semantic correspondence between different logical theories of DL and datalog. It turns out, however, that semantic

[1] This is easy in $\mathcal{SROIQ}$ since U is not required to be simple: $\top \sqsubseteq \exists R.\{a\}$, $\top \sqsubseteq \exists S^-.\{a\}$, $R \circ S \sqsubseteq U$; we will see later that the same can be done in DLP.

[2] Please note that it is also common to study datalog under a higher-order semantics. The first-order and higher-order view are closely related in some respects [1], yet it is crucial to not confuse the approaches. Throughout this work, we will only study logics with a first-order semantics.

equivalence is too strong – it does not allow the use of auxiliary symbols for expressing a logical relationship – while equisatisfiability is too weak – it does not preserve relevant logical entailments. The following notion turns out to be a more appropriate middle-ground:

Definition 1. *Given* $\mathbf{FOL}_=$ *theories* T *and* T' *with signatures* $\mathscr{S}$ *and* $\mathscr{S}'$*, then* T' semantically emulates T *if*

(1) $\mathscr{S}'$ *extends* $\mathscr{S}$*,*
(2) every model of T' *becomes a model of* T *when restricted to the interpretations of symbols from* $\mathscr{S}$*, and*
(3) 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 coincides with* $\mathcal{J}$ *on all symbols of* $\mathscr{S}$*.*

Given a $\mathcal{SROIQ}$ *knowledge base* KB *and a datalog program* P*, we say that* P *emulates* KB *if* P *emulates* $\pi(\mathrm{KB})$*.*

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 would be possible to establish more general notions that are based on arbitrary incomplete mappings between two signatures, but we found the basic definition above to be adequate for this work. We also point out that 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 1.

Given a situation as in Definition 1, we find that a first-order formula φ over $\mathscr{S}$ is a logical consequence of T if and only if it is a logical consequence of T'. This illustrates how strong this form of correspondence is, and it hints at the practical relevance of this condition for knowledge representation: 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 in any application that restricts to those symbols. In a sense, T' thus really "simulates" the behaviour of T in arbitrary contexts, but possibly by means of rather different syntactic structures.[3] If the required "interface" is restricted not only to a particular set of symbols but also to a particular logic, then the following definition may seem more natural.

Definition 2. *Let* T *and* T' *be two* $\mathbf{FOL}_=$ *theories, let* $\mathscr{S}$ *be the signature over which* T *is defined, and let* $\mathcal{L}$ *be some fragment of* $\mathbf{FOL}_=$*. We say that* T' $\mathcal{L}$-emulates T *if for every* $\mathcal{L}$ *formula* φ *over* $\mathscr{S}$*, we find that* $T' \cup \{\varphi\}$ *and* $T \cup \{\varphi\}$ *are equisatisfiable.*

In particular, this provides us with a notion of $\mathbf{FOL}_=$-emulation that describes a situation where two theories behave equivalent in the context of any first-order theory over the given signature. To avoid confusion, formal results will always be explicit about the intended type of emulation, although we will sometimes speak of "emulation" to refer to semantic emulation in informal discussions. It is not hard to see that semantic emulation implies $\mathbf{FOL}_=$-emulation.

[3] We generally avoid the term "simulation" here since it is already common in the context of model-theoretic relationships in modal logic [3].

Proposition 1. *For any fragment $\mathcal{L}$ of first-order logic with equality and theories T and T', if T' semantically emulates T then T' $\mathcal{L}$-emulates T.*

Proof. It suffices to show the claim for the case that $\mathcal{L}$ is $\mathbf{FOL}_=$. Consider two theories T' and T such that T' semantically emulates T. We need to show that T' $\mathbf{FOL}_=$-emulates T. A simple induction on the structure of $\mathbf{FOL}_=$ formulae can be used to show that the validity of a $\mathbf{FOL}_=$ formula φ w.r.t. any first-order interpretation is independent of the interpretation of the signature elements not occurring in φ (†). To show the claim, suppose the conditions of Definition 1 hold but T does not $\mathbf{FOL}_=$-emulate T'. Hence, there is a $\mathbf{FOL}_=$ formula φ over $\mathscr{S}$ such that $T \cup \{\varphi\}$ and $T' \cup \{\varphi\}$ are not equisatisfiable. However, if $T \cup \{\varphi\}$ has some model $\mathcal{I}$, then we can apply condition (3) of Definition 1 to obtain an extended model $\mathcal{I}'$ such that $\mathcal{I}' \models T'$. But since φ contains only symbols that are interpreted in the same way by $\mathcal{I}$ and $\mathcal{I}'$, we obtain $\mathcal{I}' \models \varphi$ from (†). Conversely, if $T' \cup \{\varphi\}$ has a model $\mathcal{J}$, then condition (2) implies that the restriction $\mathcal{I}$ of $\mathcal{J}$ to the signature of T is such that $\mathcal{J} \models T$. As before, (†) implies $\mathcal{J} \models \varphi$. □

3 Considerations for Defining DLP

In this section, we discuss why defining DLP is not straightforward, and we specify various design principles to guide our subsequent definition. The goal is to arrive at a notion of DLP that is characterised by these principles, as opposed to DLP being some *ad hoc* fragment of description logic that happens to be expressible in datalog without being maximal or canonical in any sense. 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 every DLP knowledge base can be expressed in datalog. We will see below that it is sometimes useful to introduce auxiliary symbols during the translation to datalog. If this is done, the datalog program can no longer be semantically equivalent to the original knowledge base, even if all consequences with respect to the original predicates are still the same. Yet, *equisatisfiability* – the requirement that a DLP knowledge base is satisfiable iff its datalog translation is – turns out to be too weak for many purposes. A suitable compromise is the notion of *emulation* introduced in Definition 2:

DLP 2 (Semantic Correspondence) There should be a transformation function $\mathsf{datalog}$ that maps a DLP knowledge base KB to a datalog program $\mathsf{datalog}(\mathrm{KB})$ such that $\mathsf{datalog}(\mathrm{KB})$ $\mathbf{FOL}_=$-emulates KB.

It turns out that DLP 2 is a strong requirement with many useful consequences. For example, it ensures us that instance retrieval queries can directly be answered over datalog, without needing to know the details of the datalog transformation: to find out whether KB entails $C(a)$, it suffices to check if $\mathsf{datalog}(\mathrm{KB})$ entails $C(a)$. But DLP 2 is much stronger than the requirement of preserving such atomic consequences, since the entailment of any $\mathbf{FOL}_=$ formula over the signature of KB can be checked in datalog(KB).

The principles DLP 1 and DLP 2 set the stage for defining DLP but they do not yet provide sufficient details to attempt a definition. The description of DLP as the "intersection" of DL and datalog is not a useful basis for defining DLP: the syntactic intersection of the two formalisms contains no terminological axioms at all. This raises the question of how to define DLP in a canonical way. A naive approach would be to define a DL ontology to belong to DLP if it can be expressed by a semantically equivalent datalog program. Such a definition would be of little practical use: every inconsistent ontology can trivially be expressed in datalog, and therefore a DL reasoner is needed to decide whether or not a knowledge base should be considered to be in DLP. This is certainly undesirable from a practical viewpoint. It is therefore preferable to give a definition that can be checked without complex semantic computations:

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

Note that typical syntactic language definitions are often subpolynomial, e.g. if they can be decided in logarithmic space (which leads to a linear time algorithm that can be parallelised). Yet, polynomial time language definitions might still be acceptable: for example, every decidable DL with transitive roles, number restrictions, and role hierarchies already requires polynomial time for checking simplicity of roles.

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 semantically equivalent transformations, e.g. due to simplicity restrictions – 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 or satisfiability preserving transformations. Particularly common transformations are the construction of 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 NNF(KB) and its disjunctive normal form DNF(KB) are in DLP.

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. In other words, it might happen that adding axioms to a non-DLP knowledge base turns it into a DLP knowledge base. This "nonmonotonic" behaviour is undesirable since it requires implementations and knowledge engineers to consider all axioms of a knowledge base in order to check if it is in DLP. The following principle requires definitions to be more well-behaved:

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 ensures that one can decide for each axiom of a knowledge base whether or not it belongs to DLP without regarding any other axioms. The goal thus has changed 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 our actual reason to consider $\mathcal{SROIQ}^{\mathrm{free}}$ instead of $\mathcal{SROIQ}$. The above principles already suffice to establish an interesting result about tractability of reasoning in DLP:

Proposition 2. *Consider a class K of knowledge bases that belong to a DL for which DLP 1 to DLP 5 are satisfied, 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 $\mathsf{datalog}$ produces structurally similar datalog for structurally similar axioms, so that there are only a finite number of structurally different datalog theories $\mathsf{datalog}(\{\alpha\})$ 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 $\mathsf{datalog}(\mathrm{KB})$ with $\mathrm{KB} \in K$. Satisfiability of datalog with at most m variables per rule can be decided in time polynomial in 2^m [4]. 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 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, but it turns out that there are arbitrarily large DLP axioms that cannot be decomposed in DLP. Yet, Proposition 2 clarifies 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], noting that remaining complex axioms can be decomposed in Horn-$\mathcal{SHIQ}$).

Note that 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 DL 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 ontologies as possible? The answer is a resounding no:

Proposition 3. *Consider a description logic $\mathcal{L}_{DLP}$ that adheres to the principles DLP 1 to DLP 5. There is a description logic $\mathcal{L}'_{DLP}$ that adheres to DLP 1 to DLP 5 while covering more knowledge bases, i.e. $\mathcal{L}_{DLP} \subset \mathcal{L}'_{DLP}$.*

Proof. We first need to argue that, even with unlimited resources for the datalog translation, it is not possible that DLP supports all $\mathcal{SROIQ}$ axioms. We show that, if the concept expression C is satisfiable and does not contain the symbols R, A_1, A_2, c and d, then the axiom $\alpha \coloneqq \{c\} \sqsubseteq C \sqcap \exists R.(d \sqcap (A_1 \sqcup A_2))$ cannot be emulated by any datalog program. For a contradiction, suppose that α is emulated by a datalog theory $\mathsf{datalog}(\alpha)$. By construction, α is satisfiable, and so is $\{\alpha, A_i \sqsubseteq \bot\}$ for each $i = 1, 2$. By Definition 2, we find that $\mathsf{datalog}(\alpha) \cup \{A_i \sqsubseteq \bot\}$ is satisfiable, too. Thus, there are models $\mathcal{I}_i$ of $\mathsf{datalog}(\alpha)$ such that $A_i^{\mathcal{I}_i} = \emptyset$. By the least model property of datalog, there is also a model $\mathcal{I}$ of $\mathsf{datalog}(\alpha)$ such that $A_1^{\mathcal{I}} = A_2^{\mathcal{I}} = \emptyset$. But then $\mathsf{datalog}(\alpha) \cup \{A_1 \sqcup A_2 \sqsubseteq \bot\}$ is satisfiable although $\{\alpha, A_1 \sqcup A_2 \sqsubseteq \bot\}$ is not, contradicting the supposed emulation.

We can now show that there is some unsatisfiable axiom that is not in $\mathcal{L}_{DLP}$. To this end, recall that deciding (un)satisfiability of $\mathcal{SHOIQ}$ concept expressions is NExpTime hard. This follows from NExpTime hardness of deciding consistency of $\mathcal{SHOIQ}$ knowledge bases [14] together with the fact that knowledge base satisfiability in $\mathcal{SROIQ}$ can be reduced to concept satisfiability [13]. However, we just showed that, if the axiom $\alpha = \{c\} \sqsubseteq C \sqcap \exists R.(\{d\} \sqcap (A_1 \sqcup A_2))$ is in $\mathcal{L}_{DLP}$ with symbols R, A_1, A_2, c, d not in C, then the concept C is unsatisfiable. Thus, if $\mathcal{L}_{DLP}$ contains all unsatisfiable $\mathcal{SHOIQ}$ axioms of the form of α, then deciding whether $\alpha \in \mathcal{L}_{DLP}$ is equivalent to deciding whether C is unsatisfiable (since one can clearly construct α from C in polynomial time). By DLP 3, this would yield a polynomial decision procedure for $\mathcal{SHOIQ}$ concept satisfiability – a contradiction.

Therefore, there is an unsatisfiable axiom α with $\alpha \notin \mathcal{L}_{DLP}$. Now let $\mathcal{L}'_{DLP}$ be defined as $\{\mathrm{KB} \mid \mathsf{DNF}(\mathsf{NNF}(\mathrm{KB})) \setminus \{\mathsf{DNF}(\mathsf{NNF}(\alpha))\} \in \mathcal{L}_{DLP}\}$. The transformation is given by $\mathsf{datalog}'(\mathrm{KB}) = \mathsf{datalog}(\mathrm{KB})$ if $\mathrm{KB} \in \mathcal{L}_{DLP}$, and $\mathsf{datalog}'(\mathrm{KB}) = \{\top \to A(x), A(x) \to \bot\} \cup \mathsf{datalog}(\mathsf{DNF}(\mathsf{NNF}(\mathrm{KB})) \setminus \{\mathsf{DNF}(\mathsf{NNF}(\alpha))\})$ otherwise, where A is a new predicate symbol. It is immediate that this defines a DL fragment (DLP 1), and that this definition is tractable (DLP 3). Equisatisfiability (DLP 2) follows since any knowledge base containing an axiom that is equivalent to α is unsatisfiable. Closure under negation normal form (DLP 4) and modularity (DLP 5) are immediate. □

This shows that any attempt to arrive at a maximal definition of DLP based on the above design principles must fail. Summing up, the above design principles are still too weak for characterising DLP: any concrete definition requires further choices that, lacking concrete guidelines, are necessarily somewhat arbitrary. Thus, while it is certainly useful to capture some general requirements in explicit principles, the resulting approach of defining DLP would not be a significant improvement over existing *ad hoc* approaches.

Analysing the proof of Proposition 3 reveals the reason why DLP 1 to DLP 5 are still insufficient. Intuitively, a definition of DLP cannot reach the desired maximum since the computations that were required in this case would no longer be polynomial (DLP 3). Even DLP 5 does not ameliorate the situation, since expressive DLs can encode complex semantic relationships within single axioms. The core of the argument underlying Proposition 3 in this sense is the fact that there is no polynomial time procedure for deciding whether or not a single $\mathcal{SROIQ}$ axiom can be expressed in datalog.

These considerations highlight a strategy for further constraining DLP to obtain a clearly defined canonical definition instead of infinitely many non-optimal choices. Namely, it is necessary to prevent complicated semantic effects that may arise when considering even single DL axioms from having any impact on the definition of DLP.

Intuitively speaking, the reason for the high complexity of evaluating single axioms is that individual parts of an axiom, even if they are structurally separated, may semantically affect each other. In expressive DLs, individual parts of an axiom can capture the semantics of arbitrary terminological axioms: the TBox can be *internalised* into a single axiom.

An important observation now is that the semantic interplay of parts of an axiom usually requires entity names to be reused. For example, the axiom $\top \sqsubseteq A \sqcap \neg A$ is unsatisfiable because the concept name A is used in both conjuncts, while the structurally similar formula $\top \sqsubseteq A \sqcap \neg B$ is satisfiable. So, in order to disallow complex semantic effects within single axioms to affect DLP, we can require DLP to be closed under the exchange of entities in the following sense:

Definition 3. *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* 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 knowledge base KB′ *is a* renaming *of a knowledge base* KB *if it is obtained from* KB *by replacing each axiom with a renaming.*

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 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 $A \sqcap \neg A$. This is clearly 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 that has this structure. We will thus study the semantics and expressivity of formulae based on their syntactic structure, disregarding any possible interactions between signature symbols. We therefore call a $\mathbf{FOL}_=$ formula, DL axiom, or DL concept expression F *structural* if no signature symbols occur more than once in F.

Together with modularity (DLP 5), this principle captures the essential difference between a "syntactic" and a "semantic" transformation from DL to datalog. Indeed, if DLP adheres to DLP 5 and DLP 6, then it may only include knowledge bases for which all potential semantic effects can be faithfully represented in datalog. The datalog transformation thus needs to take into account that additional axioms may be added (DLP 5) to state that certain entity names are semantically equivalent, while hardly any semantic consequences can be computed in advance without knowing about these equivalences. In consequence, the semantic computations that determine satisfiability must be accomplished in datalog, and not during the translation. This intuition will turn out to be quite accurate – but a lot more is needed to establish formal results below.

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 $\top \rightarrow C(x)$ and $A(x) \wedge B(x) \rightarrow \bot$. 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

Concepts that are necessarily equivalent to $\top$ and $\bot$
$\mathbf{L}^{\mathcal{A}}_{\top} ::= \top \mid \forall\mathbf{R}.\mathbf{L}^{\mathcal{A}}_{\top} \mid \mathbf{L}^{\mathcal{A}}_{\top} \sqcap \mathbf{L}^{\mathcal{A}}_{\top} \mid \mathbf{L}^{\mathcal{A}}_{\top} \sqcup \mathbf{C}$ $\mathbf{L}^{\mathcal{A}}_{\bot} ::= \bot \mid \exists\mathbf{R}.\mathbf{L}^{\mathcal{A}}_{\bot} \mid \mathbf{L}^{\mathcal{A}}_{\bot} \sqcap \mathbf{C} \mid \mathbf{L}^{\mathcal{A}}_{\bot} \sqcup \mathbf{L}^{\mathcal{A}}_{\bot}$
Body concepts: for C in normal form, $C \in \mathbf{L}^{\mathcal{A}}_{B}$ iff $C \sqcup A$ (or $\neg C \sqsubseteq A$) is in $\mathcal{DLP}_{\mathcal{ALC}}$
$\mathbf{L}^{\mathcal{A}}_{B} ::= \mathbf{L}^{\mathcal{A}}_{\top} \mid \mathbf{L}^{\mathcal{A}}_{\bot} \mid \neg\mathbf{A} \mid \forall\mathbf{R}.\mathbf{L}^{\mathcal{A}}_{B} \mid \mathbf{L}^{\mathcal{A}}_{B} \sqcap \mathbf{L}^{\mathcal{A}}_{B} \mid \mathbf{L}^{\mathcal{A}}_{B} \sqcup \mathbf{L}^{\mathcal{A}}_{B}$
Head concepts: for C in normal form, $C \in \mathbf{L}^{\mathcal{A}}_{H}$ iff $A \sqsubseteq C$ is in $\mathcal{DLP}_{\mathcal{ALC}}$
$\mathbf{L}^{\mathcal{A}}_{H} ::= \mathbf{L}^{\mathcal{A}}_{B} \mid \mathbf{A} \mid \forall\mathbf{R}.\mathbf{L}^{\mathcal{A}}_{H} \mid \mathbf{L}^{\mathcal{A}}_{H} \sqcap \mathbf{L}^{\mathcal{A}}_{H} \mid \mathbf{L}^{\mathcal{A}}_{H} \sqcup \mathbf{L}^{\mathcal{A}}_{B}$
Assertional concepts: for C in normal form, $C \in \mathbf{L}^{\mathcal{A}}_{a}$ iff $C(a)$ is in $\mathcal{DLP}_{\mathcal{ALC}}$
$\mathbf{L}^{\mathcal{A}}_{a} ::= \mathbf{L}^{\mathcal{A}}_{H} \mid \exists\mathbf{R}.\mathbf{L}^{\mathcal{A}}_{a} \mid \mathbf{L}^{\mathcal{A}}_{a} \sqcap \mathbf{L}^{\mathcal{A}}_{a} \mid \mathbf{L}^{\mathcal{A}}_{a} \sqcup \mathbf{L}^{\mathcal{A}}_{B}$

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

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 The $\mathcal{ALC}$ Fragment of DLP

Our investigations in later sections show that the definition of a maximal DLP fragment of $\mathcal{SROIQ}^{\text{free}}$ is surprisingly complex, and the required proofs for showing its maximality are rather intricate. For this reason, we first characterise the maximal DLP fragment of the much simpler description logic $\mathcal{ALC}$. The absence of nominals and cardinality restrictions simplifies the required constructions significantly. Various basic aspects of the relationship between DL and datalog can also be found in this simpler case, but there are also a number of aspects that are not touched at all.

Throughout this section, we use $\exists$ and $\forall$ instead of $\geqslant 1$ and $\leqslant 0 \ldots \neg$, which yields a more natural syntax for $\mathcal{ALC}$. Exploiting DLP 4 we can simplify the definition of DLP by giving concepts in negation normal form only.

Definition 4. *We define the description logic $\mathcal{DLP}_{\mathcal{ALC}}$ to contain all knowledge bases consisting only of $\mathcal{SROIQ}^{\text{free}}$ axioms which are*

- *GCIs $C \sqsubseteq D$ such that $\mathsf{NNF}(\neg C \sqcup D)$ is an $\mathbf{L}^{\mathcal{A}}_{H}$ concept as defined in Fig. 1, or*
- *ABox axioms $C(a)$ where $\mathsf{NNF}(C)$ is an $\mathbf{L}^{\mathcal{A}}_{a}$ concept as defined in Fig. 1.*

Following the grammatical structure of $\mathcal{DLP}_{\mathcal{ALC}}$, we specify three auxiliary functions for constructing datalog programs to emulate a $\mathcal{DLP}_{\mathcal{ALC}}$ knowledge base.

Lemma 1. *Given a concept name A, and a concept $C \in \mathbf{L}^{\mathcal{A}}_{H}$, Fig. 2 recursively defines a datalog program $\mathsf{dlg}^{\mathcal{A}}_{H}(A \sqsubseteq C)$ that semantically emulates $A \sqsubseteq C$.*

Proof. First note that the definition of $\mathsf{dlg}^{\mathcal{A}}_{H}(A \sqsubseteq C)$ is well. In particular, programs $\mathsf{dlg}^{\mathcal{A}}_{B}(\neg B \sqsubseteq D)$ are only used if $D \in \mathbf{L}^{\mathcal{A}}_{B}$. The claim is shown by induction over the definitions of $\mathsf{dlg}^{\mathcal{A}}_{B}(\neg A \sqsubseteq C)$ and $\mathsf{dlg}^{\mathcal{A}}_{H}(A \sqsubseteq C)$, where the hypothesis for the former is

C	$\mathsf{dlg}^{\mathcal{A}}_{H}(A \sqsubseteq C)$
$D \in \mathbf{L}^{\mathcal{A}}_{B}$	$\mathsf{dlg}^{\mathcal{A}}_{B}(\neg X \sqsubseteq D) \cup \{A(x) \wedge X(x) \to \bot\}$
B	$\{A(x) \to B(x)\}$
$\forall R.D$	$\mathsf{dlg}^{\mathcal{A}}_{H}(X \sqsubseteq D) \cup \{A(x) \wedge R(x,y) \to X(y)\}$
$D_1 \sqcap D_2$	$\mathsf{dlg}^{\mathcal{A}}_{H}(A \sqsubseteq D_1) \cup \mathsf{dlg}^{\mathcal{A}}_{H}(A \sqsubseteq D_2)$
$D_1 \sqcup D_2 \in (\mathbf{L}^{\mathcal{A}}_{H} \sqcup \mathbf{L}^{\mathcal{A}}_{B})$	$\mathsf{dlg}^{\mathcal{A}}_{H}(X_2 \sqsubseteq D_1) \cup \mathsf{dlg}^{\mathcal{A}}_{B}(\neg X_1 \sqsubseteq D_2) \cup \{A(x) \wedge X_1(x) \to X_2(x)\}$

C	$\mathsf{dlg}^{\mathcal{A}}_{B}(\neg A \sqsubseteq C)$
$D \in \mathbf{L}^{\mathcal{A}}_{\top}$	$\{\}$
$D \in \mathbf{L}^{\mathcal{A}}_{\bot}$	$\{A(x)\}$
$\neg B$	$\{B(x) \to A(x)\}$
$\forall R.D$	$\mathsf{dlg}^{\mathcal{A}}_{B}(\neg X \sqsubseteq D) \cup \{R(x,y) \wedge X(y) \to A(x)\}$
$D_1 \sqcap D_2 \in (\mathbf{L}^{\mathcal{A}}_{B} \sqcap \mathbf{L}^{\mathcal{A}}_{B})$	$\mathsf{dlg}^{\mathcal{A}}_{B}(\neg A \sqsubseteq D_1) \cup \mathsf{dlg}^{\mathcal{A}}_{B}(\neg A \sqsubseteq D_2)$
$D_1 \sqcup D_2 \in (\mathbf{L}^{\mathcal{A}}_{B} \sqcup \mathbf{L}^{\mathcal{A}}_{B})$	$\mathsf{dlg}^{\mathcal{A}}_{B}(\neg X_1 \sqsubseteq D_1) \cup \mathsf{dlg}^{\mathcal{A}}_{B}(\neg X_2 \sqsubseteq D_2) \cup \{X_1(x) \wedge X_2(x) \to A(x)\}$
A, B concept names, R a role name, $X_{(i)}$ fresh concept names	

Fig. 2. Transforming axioms $\mathbf{A} \sqsubseteq \mathbf{L}^{\mathcal{A}}_{H}$ and $\neg\mathbf{A} \sqsubseteq \mathbf{L}^{\mathcal{A}}_{B}$ to datalog

that it semantically emulates $\neg A \sqsubseteq C$. The easy induction steps can directly be established by showing that any model of the datalog program can be restricted to a model of the corresponding DL axiom, and any model of the DL axiom can be extended to an interpretation that models the datalog program. We omit further details here. Examples of a very similar argument are found in the proofs of Lemma 6 and 7. □

Lemma 2. *Given a constant a and a concept $C \in \mathbf{L}^{\mathcal{A}}_{a}$, Fig. 3 recursively defines a datalog program $\mathsf{dlg}^{\mathcal{A}}_{H}(C(a), \bot)$ that semantically emulates $C(a)$.*

Proof. The construction of Fig. 3 uses a "guard" concept E that is used to defer the encoding of $\mathbf{L}^{\mathcal{A}}_{B}$ disjunctions. The induction claim thus is that, for every $E \in \mathbf{L}^{\mathcal{A}}_{B}$, $C \in \mathbf{L}^{\mathcal{A}}_{a}$, and $a \in \mathbf{I}$, the program $\mathsf{dlg}^{\mathcal{A}}_{H}(C(a), E)$ semantically emulates $(C \sqcup E)(a)$.

The concept E is processed in case $C \in \mathbf{L}^{\mathcal{A}}_{H}$ by using $\mathsf{dlg}^{\mathcal{A}}_{H}$. 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 and we omit further details and refer to the full proofs given in Section 6. □

We summarise these results in the emulation theorem for $\mathcal{DLP}_{\mathcal{ALC}}$.

Theorem 1. *For every $\mathcal{DLP}_{\mathcal{ALC}}$ axiom α as in Definition 4, one can construct a datalog program $\mathsf{dlg}(\alpha)$ that emulates α.*

C	$\mathsf{dlg}^{\mathcal{A}}_{a}(C(a), E)$
$D \in \mathbf{L}^{\mathcal{A}}_{H}$	$\mathsf{dlg}^{\mathcal{A}}_{H}(X \sqsubseteq D \sqcup E) \cup \{X(a)\}$
$D_1 \sqcap D_2$	$\mathsf{dlg}^{\mathcal{A}}_{a}(D_1(a), E) \cup \mathsf{dlg}^{\mathcal{A}}_{a}(D_2(a), E)$
$D_1 \sqcup D_2 \in (\mathbf{L}^{\mathcal{A}}_{a} \sqcup \mathbf{L}^{\mathcal{A}}_{B})$	$\mathsf{dlg}^{\mathcal{A}}_{B}(\neg X \sqsubseteq D_2) \cup \mathsf{dlg}^{\mathcal{A}}_{a}(D_1(a), E \sqcup \neg X)$
$\exists R.D$	$\mathsf{dlg}^{\mathcal{A}}_{B}(\neg X \sqsubseteq E) \cup \{X(a) \to R(a, b), X(a) \to Y(b)\} \cup \mathsf{dlg}^{\mathcal{A}}_{a}(D(b), \neg Y)$
$E \in \mathbf{L}^{\mathcal{A}}_{B}$, X, Y fresh concept names, b a fresh constant	

Fig. 3. Transforming axioms $C(a)$ with $C \in \mathbf{L}^{\mathcal{A}}_{a}$ to datalog

Proof. If $\alpha = C \sqsubseteq D$ is a TBox axiom, define $\mathsf{datalog}(\alpha) \coloneqq \mathsf{dlg}^{\mathcal{A}}_{H}(A \sqsubseteq \mathsf{NNF}(\neg C \sqcup D)) \cup \{A(x)\}$. If $\alpha = C(a)$ is an ABox axiom, define $\mathsf{datalog}(\alpha) \coloneqq \mathsf{dlg}^{\mathcal{A}}_{a}(C(a), \bot)$. The result follows by Lemma 1 and 2. □

It remains to show that $\mathcal{DLP}_{\mathcal{ALC}}$ is indeed the largest DLP fragment of $\mathcal{ALC}$. We first define auxiliary datalog programs to entail that a concept's extension is empty for arbitrary concepts that are not in $\mathbf{L}^{\mathcal{A}}_{\top}$.

Definition 5. *Given a structural concept $C \notin \mathbf{L}^{\mathcal{A}}_{\top}$, a datalog program $[\![C \sqsubseteq \bot]\!]_{\mathcal{A}}$ is recursively defined as follows:*

- *If $C = \bot$ set $[\![C \sqsubseteq \bot]\!]_{\mathcal{A}} \coloneqq \{\}$.*
- *If $C \in \mathbf{A}$ set $[\![C \sqsubseteq \bot]\!]_{\mathcal{A}} \coloneqq \{C(x) \to \bot\}$.*
- *If $C = \neg B \in \neg\mathbf{A}$ set $[\![C \sqsubseteq \bot]\!]_{\mathcal{A}} \coloneqq \{B(x)\}$.*
- *If $C = \forall R.D$ with $D \notin \mathbf{L}^{\mathcal{A}}_{\top}$ set $[\![C \sqsubseteq \bot]\!]_{\mathcal{A}} \coloneqq \{R(x, x)\} \cup [\![D \sqsubseteq \bot]\!]_{\mathcal{A}}$.*
- *If $C = \exists R.D$ set $[\![C \sqsubseteq \bot]\!]_{\mathcal{A}} \coloneqq \{R(x, y) \to \bot\}$.*
- *If $C = D_1 \sqcap D_2$ with $D_1 \notin \mathbf{L}^{\mathcal{A}}_{\top}$ set $[\![C \sqsubseteq \bot]\!]_{\mathcal{A}} \coloneqq [\![D_1 \sqsubseteq \bot]\!]_{\mathcal{A}}$.*
- *If $C = D_1 \sqcup D_2$ with $D_1, D_2 \notin \mathbf{L}^{\mathcal{A}}_{\top}$ set $[\![C \sqsubseteq \bot]\!]_{\mathcal{A}} \coloneqq [\![D_1 \sqsubseteq \bot]\!]_{\mathcal{A}} \cup [\![D_2 \sqsubseteq \bot]\!]_{\mathcal{A}}$.*

Given a structural concept $C \notin \mathbf{L}^{\mathcal{A}}_{\bot}$, a datalog program $[\![\top \sqsubseteq C]\!]_{\mathcal{A}}$ is defined as $[\![\top \sqsubseteq C]\!]_{\mathcal{A}} \coloneqq [\![\mathsf{NNF}(\neg C) \sqsubseteq \bot]\!]_{\mathcal{A}}$.

Note that this definition is well, and in particular that $C \notin \mathbf{L}^{\mathcal{A}}_{\bot}$ implies $\mathsf{NNF}(\neg C) \notin \mathbf{L}^{\mathcal{A}}_{\top}$. Moreover, it is easy to see that $[\![C \sqsubseteq \bot]\!]_{\mathcal{A}}$ ($[\![\top \sqsubseteq C]\!]_{\mathcal{A}}$) is satisfiable and entails $C \sqsubseteq \bot$ ($\top \sqsubseteq C$).

The next lemma shows that concepts that are not in $\mathbf{L}^{\mathcal{A}}_{B}$ can be forced to require certain positive entailments to hold in any model in which they have a non-empty extension.

Lemma 3. *If $C \notin \mathbf{L}^{\mathcal{A}}_{B}$ is structural then there is a datalog program $[\![C \sqsubseteq A]\!]_{\mathcal{A}}$ for a fresh concept name A such that*

- *$[\![C \sqsubseteq A]\!]_{\mathcal{A}} \cup \{C(a)\}$ is satisfiable for any individual name a, and*
- *$[\![C \sqsubseteq A]\!]_{\mathcal{A}} \models C \sqsubseteq A$.*

Proof. The result is shown by induction over the structure of C. If $C \in \mathbf{A}$ is a concept name, then $[\![C \sqsubseteq A]\!]_{\mathcal{A}} := \{C(x) \to A(x)\}$ clearly satisfies the claim. If $C = \forall R.D$ with $D \notin \mathbf{L}_B^{\mathcal{A}}$ set $[\![C \sqsubseteq A]\!]_{\mathcal{A}} := [\![D \sqsubseteq A]\!]_{\mathcal{A}} \cup \{R(x,x)\}$. The claim follows by induction. If $C = \exists R.D$ with $D \neq \bot$ then $[\![C \sqsubseteq A]\!]_{\mathcal{A}} := \{R(x,y) \to A(x)\}$ clearly satisfies the claim. If $C = D_1 \sqcap D_2$ with $D_1 \notin \mathbf{L}_B^{\mathcal{A}}$, $D_1, D_2 \notin \mathbf{L}_\bot^{\mathcal{A}}$ then $[\![C \sqsubseteq A]\!]_{\mathcal{A}} := [\![D_1 \sqsubseteq A]\!]_{\mathcal{A}}$ satisfies the claim by the induction hypothesis. For the case $C = D_1 \sqcup D_2$ with $D_1 \notin \mathbf{L}_B^{\mathcal{A}}$ and $D_1, D_2 \notin \mathbf{L}_\top^{\mathcal{A}}$, we can define $[\![C \sqsubseteq A]\!]_{\mathcal{A}} := [\![D_1 \sqsubseteq A]\!]_{\mathcal{A}} \cup [\![D_2 \sqsubseteq \bot]\!]_{\mathcal{A}}$. The claim follows by induction. □

Note that the program $[\![C \sqsubseteq A]\!]_{\mathcal{A}}$ does not $\mathbf{FOL}_=$-emulate $C \sqsubseteq A$ since the subprogram $[\![D_2 \sqsubseteq \bot]\!]_{\mathcal{A}}$ that is used for the $\sqcup$ case excludes a number of interpretations that satisfy C. But the previous result suffices for our subsequent arguments.

Theorem 2. *Consider a structural concept C, an individual name a, and a concept name A not occuring in C.*

(1) If $C \notin \mathbf{L}_a^{\mathcal{A}}$ then $C(a)$ cannot be $\mathbf{FOL}_=$-emulated by any datalog program.
(2) If $C \notin \mathbf{L}_H^{\mathcal{A}}$ then $A \sqsubseteq C$ and $\top \sqsubseteq C$ cannot be $\mathbf{FOL}_=$-emulated by any datalog program, unless P = PSpace.

In particular, no fragment of $\mathcal{ALC}$ that is larger than $\mathcal{DLP}_{\mathcal{ALC}}$ can be $\mathbf{FOL}_=$-emulated in datalog, unless P = PSpace.

Proof. The proof for both claims proceeds by an interleaved induction over the structure of C. Note that C cannot be atomic in either case. We begin with the induction steps for claim (1), assuming that the claims hold for all subformulae of C. Suppose for a contradiction that there is a datalog program $P_{C(a)}$ that $\mathbf{FOL}_=$-emulates $C(a)$.

If $C = \exists R.D$ with $D \notin \mathbf{L}_a^{\mathcal{A}}$ then $P_{C(a)} \cup \{R(a,y) \to y \approx b\}$ $\mathbf{FOL}_=$-emulates $D(b)$ for a fresh individual b, contradicting the induction hypothesis (1) for D. If $C = \forall R.D$ with $D \notin \mathbf{L}_H^{\mathcal{A}}$ then $P_{C(a)} \cup \{A(x) \to R(a,x)\}$ $\mathbf{FOL}_=$-emulates $A \sqsubseteq D$, contradicting the induction hypothesis (2) for D. If $C = C_1 \sqcap C_2$ with $C_1 \notin \mathbf{L}_a^{\mathcal{A}}$ and $C_1, C_2 \notin \mathbf{L}_\bot^{\mathcal{A}}$ then $P_{C(a)} \cup [\![\top \sqsubseteq C_2]\!]_{\mathcal{A}}$ $\mathbf{FOL}_=$-emulates $C_1(a)$, contradicting the induction hypothesis (1) for C_1.

Consider the case $C = C_1 \sqcup C_2$ where $C_1, C_2 \notin \mathbf{L}_\top^{\mathcal{A}}$. If $C_1 \notin \mathbf{L}_a^{\mathcal{A}}$ then $P_{C(a)} \cup [\![C_2 \sqsubseteq \bot]\!]_{\mathcal{A}}$ $\mathbf{FOL}_=$-emulates $C_1(a)$, again contradicting the induction hypothesis (1) for C_1. Otherwise, if $C_1, C_2 \in \mathbf{L}_a^{\mathcal{A}}$ then $C_1, C_2 \notin \mathbf{L}_B^{\mathcal{A}}$. Using fresh concept names A_1 and A_2, consider datalog programs $P_i := \{A_i(x) \to \bot\} \cup [\![C_1 \sqsubseteq A_1]\!]_{\mathcal{A}} \cup [\![C_2 \sqsubseteq A_2]\!]_{\mathcal{A}}$ $(i = 1, 2)$. It is not hard to see that $\{C(a)\} \cup P_i$ is satisfiable, so the same is true for $P_{C(a)} \cup P_i$ by $\mathbf{FOL}_=$-emulation. Thus, $P_{C(a)} \cup [\![C_1 \sqsubseteq A_1]\!]_{\mathcal{A}} \cup [\![C_2 \sqsubseteq A_2]\!]_{\mathcal{A}}$ must have a model $\mathcal{I}_i$ such that $A_i^{\mathcal{I}_i} = \emptyset$ for $i = 1, 2$. By the least model property of datalog (see, e.g., [4]), this implies that $P_{C(a)} \cup [\![C_1 \sqsubseteq A_1]\!]_{\mathcal{A}} \cup [\![C_2 \sqsubseteq A_2]\!]_{\mathcal{A}}$ has a model $\mathcal{I}$ such that $A_1^{\mathcal{I}} = A_2^{\mathcal{I}} = \emptyset$. Thus $P_{C(a)} \cup P_1 \cup P_2$ is satisfiable. But clearly $P_i \models C_i \sqsubseteq \bot$ $(i = 1, 2)$ so $\{C(a)\} \cup P_1 \cup P_2$ is unsatisfiable, contradicting the supposed $\mathbf{FOL}_=$-emulation.

This finishes the induction steps for claim (1). For claim (2), suppose for a contradiction that $A \sqsubseteq C$ is $\mathbf{FOL}_=$-emulated by some datalog program $P_{A \sqsubseteq C}$. First consider the case that $C \notin \mathbf{L}_a^{\mathcal{A}}$. Then $P_{A \sqsubseteq C} \cup \{A(a)\}$ $\mathbf{FOL}_=$-emulates $C(a)$ for some fresh individual a, contradicting the induction hypothesis (1) for C. Thus, the remaining induction steps only need to cover the cases of $C \in \mathbf{L}_a^{\mathcal{A}} \setminus \mathbf{L}_H^{\mathcal{A}}$.

The case for $C = C_1 \sqcap C_2$ is similar to step (1). Likewise, the only remaining case of $C = C_1 \sqcup C_2$ is the case where, w.l.o.g., $C_1\mathbf{L}^{\mathcal{A}}_a \setminus \mathbf{L}^{\mathcal{A}}_H$, which can also be treated as before. There are no remaining cases for $C = \forall R.D$.

Consider the case $C = \exists R.D$ with $D \notin \mathbf{L}^{\mathcal{A}}_\bot$. Then $P_{A\sqsubseteq C} \cup [\![\top \sqsubseteq D]\!]_{\mathcal{A}}$ $\mathbf{FOL}_=$-emulates $A \sqsubseteq \exists R.\top$. The logic obtained by extending $\mathcal{DLP}_{\mathcal{ALC}}$ with axioms of the form $A \sqsubseteq \exists R.\top$ is known as Horn-$\mathcal{FL}^-$ [9]. Reasoning in Horn-$\mathcal{FL}^-$ was shown to be PSpace hard op. cit., and this proof can easily be adopted to use only axioms of bounded size. Assuming that $\mathrm{P} \neq \mathrm{PSpace}$ the supposed $\mathbf{FOL}_=$-emulation contradicts Proposition 2. □

Our subsequent results for the maximal DLP fragment of the description logic $\mathcal{SROIQ}^{\text{free}}$ further strengthen the previous theorem so that the assumption $\mathrm{P} \neq \mathrm{PSpace}$ is no longer required. We thus do not invest any more effort to accomplish this for the above case.

5 Defining Description Logic Programs

In this section, we provide a direct definition of DLP. We first summarise the characterisation given in Section 3.

Definition 6. *A description logic $\mathcal{L}$ is a* DLP description logic *if the set of its knowledge bases adheres to the principles DLP 1–DLP 6 of Section 3.*

Our goal in this section thus is to define the maximal DLP description logic. Some practical considerations are needed for this to become practically feasible. Namely, it turns out that the characterisation as given in the previous section 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.

A natural approach is to restrict attention to axioms in some normal form. DLP 4 requires closure under negation normal form, which 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.

A $\mathcal{SROIQ}^{\text{free}}$ concept expression C is in *positive negation normal form* (pNNF) if

- if $\leqslant n\, R.D$ is a subexpression of C, then D has the form $\neg D'$, and
- every other occurrence of $\neg$ in C is part of a subconcept $\neg D$ with $D \in \mathbf{A}$ or $D = \{a\}$ with $a \in \mathbf{I}$.

It is easy to see that any $\mathcal{SROIQ}^{\text{free}}$ concept expression C can be transformed into a semantically equivalent concept expression pNNF(C) in linear time. A DLP description logic thus can be defined by providing its pNNF only.

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

Concepts containing at most *n* elements in any interpretation, and their complements

$$\mathbf{L}_\bot = \mathbf{L}_{\leq 0} ::= \bot \mid \mathbf{L}_\bot \sqcap \mathbf{C} \mid \mathbf{L}_\bot \sqcup \mathbf{L}_\bot \mid \geqslant n\, \mathbf{R}.\mathbf{L}_{\leq n-1}\ (n \geq 1)$$

$$\mathbf{L}_{\leq m+1} ::= \{\mathbf{I}\} \mid \mathbf{L}_{\leq m} \mid \mathbf{L}_{\leq m+1} \sqcap \mathbf{C} \mid \mathbf{L}_{\leq m'} \sqcup \mathbf{L}_{\leq m''}\ (m' + m'' = m + 1)$$

$$\begin{aligned}\overline{\mathbf{L}}_\bot = \overline{\mathbf{L}}_{\leq 0} ::= {} & \top \mid \mathbf{A} \mid \{\mathbf{I}\} \mid \exists \mathbf{R}.\mathsf{Self} \mid \neg\mathbf{A} \mid \neg\{\mathbf{I}\} \mid \neg\exists\mathbf{R}.\mathsf{Self} \mid \overline{\mathbf{L}}_\bot \sqcap \overline{\mathbf{L}}_\bot \mid \overline{\mathbf{L}}_\bot \sqcup \mathbf{C} \mid \\ & \leqslant n\, \mathbf{R}.\neg\mathbf{C}\ (n \geq 0) \mid \geqslant 0\, \mathbf{R}.\mathbf{C} \mid \geqslant n\, \mathbf{R}.\overline{\mathbf{L}}_{\leq n-1}\ (n \geq 1)\end{aligned}$$

$$\begin{aligned}\overline{\mathbf{L}}_{\leq m+1} ::= {} & \top \mid \mathbf{A} \mid \exists \mathbf{R}.\mathsf{Self} \mid \neg\mathbf{A} \mid \neg\{\mathbf{I}\} \mid \neg\exists\mathbf{R}.\mathsf{Self} \mid \\ & \overline{\mathbf{L}}_{\leq m+1} \sqcap \overline{\mathbf{L}}_{\leq m+1} \mid \overline{\mathbf{L}}_{\leq m+1} \sqcup \mathbf{C} \mid \overline{\mathbf{L}}_{\leq m'} \sqcup \overline{\mathbf{L}}_{\leq m''}\ (m' + m'' = m) \mid \\ & \leqslant n\, \mathbf{R}.\neg\mathbf{C}\ (n \geq 0) \mid \geqslant 0\, \mathbf{R}.\mathbf{C} \mid \geqslant n\, \mathbf{R}.\overline{\mathbf{L}}_{\leq n-1}\ (n \geq 1)\end{aligned}$$

Concepts not containing at most *n* elements in any interpretation, and their complements

$$\mathbf{L}_\top = \mathbf{L}_{\geq\omega-0} ::= \top \mid \mathbf{L}_\top \sqcup \mathbf{C} \mid \mathbf{L}_\top \sqcap \mathbf{L}_\top \mid \geqslant 0\, \mathbf{R}.\mathbf{C} \mid \leqslant n\, \mathbf{R}.\neg\mathbf{L}_{\geq\omega-n}\ (n \geq 0)$$

$$\mathbf{L}_{\geq\omega-m-1} ::= \neg\{\mathbf{I}\} \mid \mathbf{L}_{\geq\omega-m} \mid \mathbf{L}_{\geq\omega-m-1} \sqcup \mathbf{C} \mid \mathbf{L}_{\geq\omega-m'} \sqcap \mathbf{L}_{\geq\omega-m''}\ (m' + m'' = m + 1)$$

$$\begin{aligned}\overline{\mathbf{L}}_\top = \overline{\mathbf{L}}_{\geq\omega-0} ::= {} & \bot \mid \mathbf{A} \mid \{\mathbf{I}\} \mid \exists \mathbf{R}.\mathsf{Self} \mid \neg\mathbf{A} \mid \neg\{\mathbf{I}\} \mid \neg\exists\mathbf{R}.\mathsf{Self} \mid \overline{\mathbf{L}}_\top \sqcup \overline{\mathbf{L}}_\top \mid \overline{\mathbf{L}}_\top \sqcap \mathbf{C} \mid \\ & \geqslant n\, \mathbf{R}.\mathbf{C}\ (n \geq 1) \mid \leqslant n\, \mathbf{R}.\neg\overline{\mathbf{L}}_{\geq\omega-n}\ (n \geq 0)\end{aligned}$$

$$\begin{aligned}\overline{\mathbf{L}}_{\geq\omega-m-1} ::= {} & \bot \mid \mathbf{A} \mid \{\mathbf{I}\} \mid \exists \mathbf{R}.\mathsf{Self} \mid \neg\mathbf{A} \mid \neg\{\mathbf{I}\} \mid \neg\exists\mathbf{R}.\mathsf{Self} \mid \\ & \overline{\mathbf{L}}_{\geq\omega-m-1} \sqcup \overline{\mathbf{L}}_{\geq\omega-m-1} \mid \overline{\mathbf{L}}_{\geq\omega-m-1} \sqcap \mathbf{C} \mid \overline{\mathbf{L}}_{\geq\omega-m'} \sqcap \overline{\mathbf{L}}_{\geq\omega-m''}\ (m' + m'' = m) \mid \\ & \geqslant n\, \mathbf{R}.\mathbf{C}\ (n \geq 1) \mid \leqslant n\, \mathbf{R}.\neg\overline{\mathbf{L}}_{\geq\omega-n}\ (n \geq 0)\end{aligned}$$

C: any $\mathcal{SROIQ}^{\text{free}}$ concept

Fig. 4. Grammars for structurally valid, unsatisfiable, refutable, and satisfiable concepts

of complexity is the fact that $\mathcal{SROIQ}$ features many concept expressions for which all possible renamings are necessarily equivalent to $\top$ or $\bot$. Simple examples such as $\top \sqcup C$ were already encountered in the definitions of $\mathbf{L}_\top^{\mathcal{A}}$ and $\mathbf{L}_\bot^{\mathcal{A}}$ in Section 4, but $\mathcal{SROIQ}$ also includes expressions like $\geqslant 0\, R.C$ or $\leqslant 3\, R.\{a\} \sqcup \{b\}$.

Definition 7. *Let C be a $\mathcal{SROIQ}$ concept expression.*

- *C is* structurally valid *if $\top \sqsubseteq C'$ is valid for every renaming C' of C.*
- *C is* structurally unsatisfiable *if $C' \sqsubseteq \bot$ is valid for every renaming C' of C.*
- *C is* structurally refutable *if it is not structurally valid, i.e. if there is a renaming C' of C such that $\top \sqsubseteq C'$ is refutable.*
- *C is* structurally satisfiable *if it is not structurally unsatisfiable, i.e. if there is a renaming C' of C such that $C' \sqsubseteq \bot$ is refutable.*

The renamings C' considered here refer to renamings over arbitrary signatures, and are not restricted to the signature of C.

Many non-trivial examples for such concepts are based on the fact that some DL concepts do not allow for arbitrary interpretations but are in fact constrained to certain extensions. It is possible to provide a complete syntactic characterisation of these $\mathcal{SROIQ}$ concepts.

Lemma 4. *The grammars given in Fig. 4 characterise sets of $\mathcal{SROIQ}$ concept expressions as follows:*

- *$C \in \mathbf{L}_{\leq n}$ iff $C^{\mathcal{I}}$ contains at most n elements for any interpretation $\mathcal{I}$,*
- *$C \in \overline{\mathbf{L}}_{\leq n}$ iff $C^{\mathcal{I}}$ contains more than n elements for some interpretation $\mathcal{I}$,*
- *$C \in \mathbf{L}_{\geq \omega - n}$ iff $\Delta^{\mathcal{I}} \setminus C^{\mathcal{I}}$ contains at most n elements for any interpretation $\mathcal{I}$,*
- *$C \in \overline{\mathbf{L}}_{\geq \omega - n}$ iff $\Delta^{\mathcal{I}} \setminus C^{\mathcal{I}}$ contains more than n elements for some interpretation $\mathcal{I}$.*

In particular, $\mathbf{L}_{\top}$, $\mathbf{L}_{\bot}$, $\overline{\mathbf{L}}_{\top}$, and $\overline{\mathbf{L}}_{\bot}$ characterise the sets of structurally valid, unsatisfiable, refutable, or satisfiable concept expressions.

Proof. We first show the "only if" direction of $\mathbf{L}_{\leq n}$ by induction over the structure of the grammars. The base cases $\bot$ and $\{\mathbf{I}\}$ (where $n \geq 1$ is required) are obvious. The case $\mathbf{L}_{\leq n-1}$ (where $n \geq 1$) is immediate from the induction hypothesis. Note that the cases of $\sqcup$ and $\sqcap$ for $n = 0$ are simply special instances of the respective cases for $n \geq 1$. The cases for $\mathbf{L}_{\leq n} \sqcap \mathbf{C}$ and $\mathbf{L}_{\leq m'} \sqcap \mathbf{L}_{\leq m''}$ are again obvious from the induction hypothesis.

Considering the grammar for each operator, it can be seen that $\overline{\mathbf{L}}_{\leq n}$ is indeed the set complement of $\mathbf{L}_{\leq n}$ for each n. An easy induction over n is used to show this formally, where it suffices to compare the cases for each constructor to see that they are exhaustive and non-overlapping. Thus, to show the "if" direction of the claim for $\mathbf{L}_{\leq n}$, it suffices to show the "only if" direction of the claim for $\overline{\mathbf{L}}_{\leq n}$.

The "only if" direction of the claim for $\overline{\mathbf{L}}_{\leq n}$ if again established induction over the structure of concepts in $\mathbf{L}_{\leq n}$. Most cases are obvious. For the case of $C \sqcap D$, it is necessary to note that the extensions of C and D, in addition to containing more than n elements, can always be selected freely to ensure that the intersection of both extensions contains enough elements.

The proofs for the claims about $\mathbf{L}_{\geq \omega - n}$ and $\overline{\mathbf{L}}_{\geq \omega - n}$ are similar. □

The previous result shows that structural validity, satisfiability, unsatisfiability, and refutability of a concept expression can be recognised in polynomial time by using the given grammars.[4] For another simplification of our characterisation, we may thus assume that almost all occurrences of such concepts have been eliminated in the concepts that we consider. This completes the ingredients we need for defining the normal form that is used below.

Definition 8. *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 $\leqslant n\, R.\neg D$.*

The unique DLP normal form of a concept D is denoted by $\mathsf{DLPNF}(C)$.

It is easy to see that $\mathsf{DLPNF}(C)$ can be computed in polynomial time. In particular, structurally valid and unsatisfiable subconcepts can be replaced by $\top$ and $\bot$, respectively, and expressions of the form $C \sqcup \bot$ and $C \sqcap \top$ can be reduced to C. Also note

[4] Note that the omission of the universal role allows us to ignore concepts such as $\leqslant 0\, U.\{a\}$ which would otherwise be structurally unsatisfiable; similar simplifications occur throughout this section.

that the order of applying the single normalisation steps does not affect the DLP normal form. It therefore suffices to characterise concepts in DLP normal form that are a DLP description logic. When convenient, we continue to use GCIs $C \sqsubseteq D$ to represent the unique DLP normal form of $\neg C \sqcup D$. Exploiting associativity and commutativity of $\sqcap$ and of $\sqcup$, we furthermore disregard order and nesting of multiple conjunctions or disjunctions.

Whereas structurally valid and invalid subconcepts are ignored in DLP normal forms, we still have reason to consider concepts with restricted extensions. We thus use $\mathbf{D}_{\leq n}$ ($\mathbf{D}_{\geq\omega-n}$) to denote the sublanguage of concepts of $\mathbf{L}_{\leq n}$ ($\mathbf{L}_{\geq\omega-n}$) that are in DLP normal form.

Before giving the full definition of a large – actually, as we will show below, *the largest* – DLP description logic, we provide some 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 $A \sqsubseteq \exists R.(\{c\} \sqcap \exists S.\top)$ ($A(x) \rightarrow R(x,c)$, $A(x) \rightarrow S(c,s)$ 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 9. *We define the description logic $\mathcal{DLP}$ to contain all knowledge bases consisting only of $\mathcal{SROIQ}^{\text{free}}$ axioms which are*

- *RBox axioms, or*
- *GCIs $C \sqsubseteq D$ such that the DLP normal form of $\neg C \sqcup D$ is a $\mathbf{D}_{DLP}$ concept as defined in the following grammar:*

$$\mathbf{D}_{DLP} ::= \top \mid \bot \mid \mathbf{C}_H \mid \mathbf{D}^{=n}\ (n \geq 1) \mid \mathbf{C}_{\neq\top}$$

  where $\mathbf{C}_H$ is defined as in Fig. 5, and $\mathbf{D}^{=n}$ and $\mathbf{C}_{\neq\top}$ are defined as in Fig. 6, or
- *Abox axioms $C(a)$ where the DLP normal form of C is $\top$, $\bot$, or a $\mathbf{D}_a$ concept as defined in Fig. 5.*

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. 5. Since all concepts are in DNF, each sublanguage consists of a conjunctive part $\mathbf{C}$ and a disjunctive part $\mathbf{D}$. 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 expressive 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

Body concepts: for C in normal form, $C \in \mathbf{D}_B$ iff $C \sqcup A$ (or $\neg C \sqsubseteq A$) is in DLP

$$\begin{aligned}
\mathbf{C}_B &::= \neg\mathbf{A} \mid \neg\{\mathbf{I}\} \mid \neg\exists\mathbf{R}.\mathsf{Self} \mid \leqslant 0\,\mathbf{R}.\neg(\mathbf{D}_B \cup \{\bot\}) \mid \mathbf{C}_B \sqcap \mathbf{C}_B \\
\mathbf{D}_B &::= \mathbf{C}_B \mid \mathbf{D}_B \sqcup \mathbf{D}_B
\end{aligned}$$

Head concepts: for C in normal form, $C \in \mathbf{D}_H$ iff $A \sqsubseteq C$ is in DLP

$$\begin{aligned}
\mathbf{C}_H &::= \mathbf{C}_B \mid \mathbf{A} \mid \{\mathbf{I}\} \mid \exists\mathbf{R}.\mathsf{Self} \mid \geqslant n\,\mathbf{R}.\mathbf{D}_{n!} \mid \leqslant 0\,\mathbf{R}.\neg\mathbf{D}_H \mid \leqslant 1\,\mathbf{R}.\neg(\mathbf{D}_B \cup \{\bot\}) \mid \mathbf{C}_H \sqcap \mathbf{C}_H \mid \mathbf{D}_{1!} \\
\mathbf{D}_H &::= \mathbf{C}_H \mid \mathbf{D}_H \sqcup \mathbf{D}_B \mid \mathbf{D}_a \sqcup \mathbf{C}_{\geq}
\end{aligned}$$

Assertional concepts: for C in normal form, $C \in \mathbf{D}_a$ iff $\{a\} \sqsubseteq C$ is in DLP

$$\begin{aligned}
\mathbf{C}_a &::= \mathbf{C}_H \mid \geqslant n\,\mathbf{R}.\mathbf{D}^{\geqslant n} \mid \mathbf{C}_a \sqcap \mathbf{C}_a \\
\mathbf{D}_a &::= \mathbf{C}_a \mid \mathbf{D}_a \sqcup \mathbf{D}_B
\end{aligned}$$

Disjunctions of nominal assertions of the form $\{\mathbf{I}\} \sqcap \mathbf{C}_a$

$$\begin{aligned}
\mathbf{D}_{1!} &::= \{\mathbf{I}\} \mid \{\mathbf{I}\} \sqcap \mathbf{C}_a \\
\mathbf{D}_{m+1!} &::= \mathbf{D}_{m!} \sqcup \mathbf{D}_{1!}
\end{aligned}$$

Conjunction of negated nominals, i.e. complements of some nominal disjunction

$$\begin{aligned}
\mathbf{C}_{\neg 1} &::= \neg\{\mathbf{I}\} \\
\mathbf{C}_{\neg(m+1)} &::= \mathbf{C}_{\neg m} \sqcap \neg\{\mathbf{I}\} \\
\mathbf{C}_{\geq} &::= \neg\{\mathbf{I}\} \mid \mathbf{C}_{\geq} \sqcap \mathbf{C}_{\geq}
\end{aligned}$$

Filler concepts for $\geqslant n$ in $\mathbf{D}_a$

$$\begin{aligned}
\mathbf{D}^{\geqslant n} &::= \top \mid \mathbf{C}_{\neg m} \sqcup \mathbf{D}^+_a \;\; (1 \leq m \leq n^2 - n) \mid \mathbf{D}_B \sqcup \mathbf{D}^+_{m!} \;\; (m < n) \mid \\
&\quad\;\; \mathbf{D}_a \sqcup \mathbf{D}^+_{m!} \sqcup \mathbf{D}_{l!} \quad \text{(for } r := n - (m + l) \text{ we have } r > 0 \text{ and } r(r-1) \geq m\text{)} \\
&\quad\;\; \text{where no disjuncts are added for expressions } \mathbf{D}^+_{0!} \text{ and } \mathbf{D}_{0!}
\end{aligned}$$

Extended concepts with restricted forms of ("local") disjunctions, used in $\mathbf{D}^{\geqslant n}$ only

$$\begin{aligned}
\mathbf{C}^+_B &::= \mathbf{C}_B \mid \leqslant 0\,\mathbf{R}.\neg\mathbf{D}^+_B \mid \leqslant n\,\mathbf{R}.\neg(\mathbf{D}^+_a \cap \mathbf{D}_{\geq\omega-m}) \mid \mathbf{C}^+_B \sqcap \mathbf{C}^+_B \\
\mathbf{D}^+_B &::= \mathbf{C}^+_B \mid \mathbf{D}^+_B \sqcup \mathbf{D}^+_B \mid \mathbf{D}^+_a \sqcup \mathbf{C}_{\geq} \\
\mathbf{C}^+_H &::= \mathbf{C}_H \mid \geqslant n\,\mathbf{R}.\mathbf{D}^+_{n!} \mid \leqslant 0\,\mathbf{R}.\neg\mathbf{D}^+_H \mid \leqslant 1\,\mathbf{R}.\neg\mathbf{D}^+_B \mid \leqslant n\,\mathbf{R}.\neg(\mathbf{D}^+_a \cap \mathbf{D}_{\geq\omega-m}) \mid \mathbf{C}^+_H \sqcap \mathbf{C}^+_H \mid \mathbf{D}^+_{1!} \\
\mathbf{D}^+_H &::= \mathbf{C}^+_H \mid \mathbf{D}^+_H \sqcup \mathbf{D}^+_B \mid \mathbf{D}^+_a \sqcup \mathbf{C}_{\geq} \\
\mathbf{C}^+_a &::= \mathbf{C}^+_H \mid \geqslant n\,\mathbf{R}.(\mathbf{D}^+_a \cup \{\top\}) \mid \mathbf{C}^+_a \sqcap \mathbf{C}^+_a \\
\mathbf{D}^+_a &::= \mathbf{C}^+_a \mid \mathbf{D}^+_a \sqcup \mathbf{D}^+_a \\
\mathbf{D}^+_{1!} &::= \{\mathbf{I}\} \sqcap \mathbf{C}^+_a \\
\mathbf{D}^+_{m+1!} &::= \mathbf{D}^+_{m!} \sqcup \mathbf{D}^+_{1!}
\end{aligned}$$

Fig. 5. Grammars for defining DLP concepts in DLP normal form

Additional concepts based on global domain size restrictions

$$\begin{aligned}\mathbf{D}^{=1} &::= \{\mathbf{I}\} \sqcap \mathbf{C}_H^p \\ \mathbf{D}^{=m+1} &::= \mathbf{D}^{=m} \sqcup (\{\mathbf{I}\} \sqcap \mathbf{C}_\bot^{=m+1})\end{aligned}$$

Additional concepts expressing ⊤ for unary domains (“propositional” case)

$$\begin{aligned}\mathbf{C}_\top^p &::= \{\mathbf{I}\} \mid \mathbf{C}_\top^p \sqcap \mathbf{C}_\top^p \mid \leqslant 0\,\mathbf{R}.\neg(\mathbf{D}_\top^p) \mid \leqslant n\,\mathbf{R}.\neg\mathbf{D}\ (n \geq 1) \\ \mathbf{D}_\top^p &::= \mathbf{C}_\top^p \mid \mathbf{D}_\top^p \sqcup \mathbf{D}\end{aligned}$$

Additional head and body concept expressions for unary domains (“propositional” case)

$$\begin{aligned}\mathbf{C}_B^p &::= \mathbf{C}_\bot^{=1} \mid \mathbf{C}_\top^p \mid \neg\mathbf{A} \mid \neg\exists\mathbf{R}.\mathsf{Self} \mid \mathbf{C}_B^p \sqcap \mathbf{C}_B^p \mid \leqslant 0\,\mathbf{R}.\neg(\mathbf{D}_B^p \cup \{\bot\}) \\ \mathbf{D}_B^p &::= \mathbf{D}_\top^p \mid \mathbf{D}_B^p \mid \mathbf{D}_B^p \sqcup \mathbf{D}_B^p \\ \mathbf{C}_H^p &::= \mathbf{C}_B^p \mid \mathbf{A} \mid \exists\mathbf{R}.\mathsf{Self} \mid \mathbf{C}_H^p \sqcap \mathbf{C}_H^p \mid \geqslant 1\,\mathbf{R}.\mathbf{D}_H^p \mid \leqslant 0\,\mathbf{R}.\neg\mathbf{D}_H^p \\ \mathbf{D}_H^p &::= \mathbf{D}_\top^p \mid \mathbf{C}_H^p \mid \mathbf{D}_H^p \sqcup \mathbf{D}_B^p\end{aligned}$$

Additional structurally unsatisfiable concepts for domains of restricted size

$$\begin{aligned}\mathbf{C}_\bot^{=1} &::= \neg\{\mathbf{I}\} \mid \mathbf{C}_\bot^{=1} \sqcap \mathbf{C} \mid \geqslant 1\,\mathbf{R}.\mathbf{D}_\bot^{=1} \mid \geqslant n\,\mathbf{R}.\mathbf{D}\,(n \geq 2) \\ \mathbf{C}_\bot^{=m+1} &::= \mathbf{C}_\bot^{=m+1} \sqcap \mathbf{C} \mid \geqslant n\,\mathbf{R}.\mathbf{D}_\bot^{=m+1}\,(n \geq 1) \mid \geqslant n\,\mathbf{R}.\mathbf{D}\,(n \geq m+2) \\ \mathbf{D}_\bot^{=m} &::= \mathbf{C}_\bot^{=m} \mid \mathbf{D}_\bot^{=m} \sqcup \mathbf{D}_\bot^{=m}\end{aligned}$$

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

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 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 semantically emulated by the following set of datalog rules, where c_i are auxiliary constants:

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

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 semantic 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, as an example, 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. 6 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. The auxiliary classes $\mathbf{C}_\bot^{=m}$ describe concepts that cannot be satisfied by an interpretation with at most m elements in their domain, as described in the following lemma.

Lemma 5. *A structural concept $C \neq \bot$ in DLP normal form is in $\mathbf{C}_\bot^{=m}$ as defined in Fig. 6 for some $m \geq 1$ iff, for all interpretations $\mathcal{I}$ with domain size $\#(\Delta^\mathcal{I}) \leq m$, we find $\mathcal{I} \models C \sqsubseteq \bot$.*

Proof. The "only if" direction can be shown by an easy induction, where the base cases are given by concepts $\geqslant n\, R.D$ with $n > m$, and – in the case $n = 1$ – negated nominals $\neg\{a\}$. The proof is straightforward and we omit further details.

For the "if" direction, assume that $C \notin \mathbf{C}_\bot^{=m} \cup \{\bot\}$, and let Δ be a domain of size m, i.e. $\#(\Delta) = m$. Then, for any $\delta \in \Delta$, we can find an interpretation $\mathcal{I}(\delta, C)$ such that $\Delta^{\mathcal{I}(\delta,C)} = \Delta$ and $\delta \in C^{\mathcal{I}(\delta,C)}$. The base cases with C of the form $\mathbf{C}$, $\exists\mathbf{R}.\mathsf{Self}$, $\{\mathbf{I}\}$, $\neg\mathbf{C}$, $\neg\exists\mathbf{R}.\mathsf{Self}$, and – if $n = 1$ – $\neg\{\mathbf{I}\}$ are obvious. If $C = D_1 \sqcup D_2$, then, without loss of generality, $D_1 \notin \mathbf{C}_\bot^{=m}$ and $\mathcal{I}(\delta, C) \coloneqq \mathcal{I}(\delta, D_1)$ satisfies the claim.

Now assume that C is of the form $D_1 \sqcap D_2$. Then $D_1, D_2 \notin \mathbf{C}_\bot^{=m} \cup \{\bot\}$, and we find interpretations $\mathcal{I}(\delta, D_1)$ and $\mathcal{I}(\delta, D_2)$ as in the hypothesis. Since C is structural, the hypothesis for D_1 is also satisfied by any variant $\mathcal{I}'(\delta, D_1)$ of $\mathcal{I}(\delta, D_1)$ which is obtained by changing the interpretation of symbols that occur in D_2. Thus we can assume without loss of generality that $\mathcal{I}(\delta, D_1)$ has been chosen such that it agrees with $\mathcal{I}(\delta, D_2)$ on all signature symbols that occur in D_2. By a symmetric argumentation for $\mathcal{I}(\delta, D_2)$, we find that such an $\mathcal{I}(\delta, D_1)$ would also satisfy the hypothesis for D_2, and hence we can set $\mathcal{I}(\delta, C) \coloneqq \mathcal{I}(\delta, D_1)$.

If $C = \leqslant n\, R.D$, then any interpretation $\mathcal{I}(\delta, C)$ with $R^\mathcal{I}(\delta, C) = \emptyset$ satisfies the claim. If $C = \geqslant n\, R.D$ with $n \leq m$, then consider distinct elements $\delta_1, \ldots, \delta_n \in \Delta$. Using structurality and the induction hypothesis again, we find a model $\mathcal{I}(\delta, C) = \mathcal{I}(\delta_1, D) = \ldots = \mathcal{I}(\delta_n, D)$ such that $R^{\mathcal{I}(\delta,C)} = \{\langle \delta, \delta_i \mid 1 \leq i \leq n\rangle\}$. □

C	$\mathsf{dlg}_B(\neg A \sqsubseteq C)$
$\bot$	$\{A(x)\} \cup P_{\mathrm{Inv}}$
$\neg B$	$\{B(x) \to A(x)\} \cup P_{\mathrm{Inv}}$
$\neg\{c\}$	$\{A(c)\} \cup P_{\mathrm{Inv}}$
$\neg\exists R.\mathsf{Self}$	$\{R(x,x) \to A(x)\} \cup P_{\mathrm{Inv}}$
$D_1 \sqcap D_2$	$\mathsf{dlg}_B(\neg A \sqsubseteq D_1) \cup \mathsf{dlg}_B(\neg A \sqsubseteq D_2)$
$D_1 \sqcup D_2$	$\mathsf{dlg}_B(\neg X_1 \sqsubseteq D_1) \cup \mathsf{dlg}_B(\neg X_2 \sqsubseteq D_2) \cup \{X_1(x) \wedge X_2(x) \to A(x)\}$
$\leqslant 0\, R.\neg D$	$\mathsf{dlg}_B(\neg X \sqsubseteq D) \cup \{R(x,y) \wedge X(y) \to A(x)\}$
A, B concept names, c an individual name, R a role, $X_{(i)}$ fresh concept names	

Fig. 7. Transforming axioms $\neg\mathbf{A} \sqsubseteq (\mathbf{D}_B \cup \{\bot\})$ to datalog

6 Emulating $\mathcal{DLP}$ in Datalog

In this section, we show that knowledge bases of $\mathcal{DLP}$ as given in Definition 9 can indeed be emulated in datalog.

Emulations are generally established by means of recursively defined functions that translate $\mathcal{DLP}$ axioms to datalog. Relevant (auxiliary) transformations are required for each of the languages defined in Fig. 5 and 6. In all cases, the built-in semantics of inverse roles is explicitly needed in datalog. For this purpose, an auxiliary datalog program P_{Inv} is defined as $P_{\mathrm{Inv}} \coloneqq \{R(x,y) \to \mathrm{Inv}(R)(y,x) \mid R \in \mathbf{R}\}$, where $\mathbf{R}$ is the set of roles of the given signature. We begin with the rather simple case of $\mathbf{D}_B$.

Lemma 6. *Every $\mathcal{DLP}$ axiom $\neg A \sqsubseteq C$ with A a concept name and $C \in \mathbf{D}_B \cup \{\bot\}$ is semantically emulated by the datalog program $\mathsf{dlg}_B(\neg A \sqsubseteq C)$ as defined in Fig. 7.*

Proof. Note that the definition in Fig. 7 is well – especially all recursive uses of dlg_B refer to arguments in the domain of this function. The proof proceeds by induction over the structure of C, showing that the conditions of Definition 1 are satisfied. We show a single induction step to illustrate the easy argumentation.

Consider the case $C = D_1 \sqcup D_2$. For one direction of the claim, consider any model $\mathcal{I}$ of $\neg A \sqsubseteq C$. An interpretation $\mathcal{I}'$ over the extended signature is defined by setting $X_i^{\mathcal{I}'} \coloneqq \Delta^{\mathcal{I}} \setminus D_i^{\mathcal{I}}$ for $i = 1, 2$. It is easy to see that $\mathcal{I}' \models \{\neg X_i \sqsubseteq D_i \mid i = 1, 2\} \cup \{X_1(x) \wedge X_2(x) \to A(x)\}$. 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. The other direction requires us to show that every model of $\mathsf{dlg}_B(\neg A \sqsubseteq C)$ is also a model of $\neg A \sqsubseteq C$, which is obvious when applying the induction hypothesis. □

Now define, for a datalog program P and a ground literal $A(c)$, a datalog program $P|_{A(c)} \coloneqq \{A(c) \wedge F \to H \mid F \to H \in P\}$. This way of manipulating datalog programs is convenient for our following definitions. Clearly, if P semantically emulates a formula φ, then $P|_{A(c)}$ semantically emulates $\varphi \vee \neg A(c)$.

C	$\mathsf{dlg}_H(A \sqsubseteq C)$
$D \in \mathbf{D}_B$	$\mathsf{dlg}_B(\neg X \sqsubseteq D) \cup \{A(x) \wedge X(x) \to \bot\}$
B	$\{A(x) \to B(x)\} \cup P_{\text{Inv}}$
$\{c\}$	$\{A(x) \to c \approx x\} \cup P_{\text{Inv}}$
$\exists R.\mathsf{Self}$	$\{A(x) \to R(x,x)\} \cup P_{\text{Inv}}$
$D_1 \sqcap D_2 \ \in (\mathbf{D}_H \sqcap \mathbf{D}_H)$	$\mathsf{dlg}_H(A \sqsubseteq D_1) \cup \mathsf{dlg}_H(A \sqsubseteq D_2)$
$\{c\} \sqcap D \quad \in \mathbf{D}_{1!}$	$\mathsf{dlg}_a(\{c\} \sqsubseteq D)\vert_{A(c)} \cup \mathsf{dlg}_H(A \sqsubseteq \{c\})$
$D_1 \sqcup D_2 \ \in (\mathbf{D}_H \sqcup \mathbf{D}_B)$	$\mathsf{dlg}_H(X_2 \sqsubseteq D_1) \cup \mathsf{dlg}_B(\neg X_1 \sqsubseteq D_2) \cup \{A(x) \wedge X_1(x) \to X_2(x)\}$
$D_1 \sqcup D_2 \ \in (\mathbf{D}_a \sqcup \mathbf{C}_{\geq})$	$\bigcup_{c \in \mathsf{ind}(D_2)} \mathsf{dlg}_a(\{c\} \sqsubseteq D_1)\vert_{A(c)}$
$\geqslant n\, R.D \quad \in (\geqslant n\, \mathbf{R}.\mathbf{D}_{n!})$	$\bigcup_{c \in I} \Big(\{A(x) \to R(x,c)\} \cup \bigcup_{d \in I \setminus \{c\}} \{A(x) \wedge c \approx d \to \bot\} \cup \mathsf{dlg}_a(\{c\} \sqsubseteq D_c)\Big)$ $I = \mathsf{ind}(D)$, and D_c such that $D = D'_c \sqcup (D_c \sqcap \{c\})$ for some $D'_c \in \mathbf{D}_{n-1!}$
$\leqslant 0\, R.\neg D$	$\mathsf{dlg}_H(X \sqsubseteq D) \cup \{A(x) \wedge R(x,y) \to X(y)\}$
$\leqslant 1\, R.\neg D$	$\mathsf{dlg}_B(\neg X \sqsubseteq D) \cup \{A(x) \wedge R(x,y) \wedge X(y) \wedge R(x,z) \wedge X(z) \to y \approx z\}$

A, B concept names, c, d individual names, R a role, $X_{(i)}$ fresh concept names,
$\mathsf{dlg}_a(\{c\} \sqsubseteq C)$ as defined in Fig. 9 below

Fig. 8. Transforming axioms $\mathbf{A} \sqsubseteq \mathbf{D}_H$ to datalog

The remaining language definitions of Fig. 5 are interdependent, so the corresponding translation needs to be established in a single recursion for which semantic emulation is shown in a single structural induction. We still separate the relevant claims for clarity, so the following lemmata can be considered as induction steps in the overall proof. The following lemma illustrates a first, simple induction step:

Lemma 7. *Consider a concept $C \in \mathbf{D}_H$ such that, for every proper subconcept $D \in \mathbf{D}_a$ of C and individual symbol d, the program $\mathsf{dlg}_a(\{d\} \sqsubseteq D)$ semantically emulates $\{d\} \sqsubseteq D$. Then, given a concept name A, the datalog program $\mathsf{dlg}_H(A \sqsubseteq C)$ as defined in Fig. 8 semantically emulates $A \sqsubseteq C$.*

Proof. Note that the definition is well, and especially that all uses of programs $\mathsf{dlg}_a(\{d\} \sqsubseteq D)$ do indeed refer to proper subconcepts D of C. The proof proceeds by induction, using similar arguments as in Lemma 6. We illustrate a single case which uses some features that did not occur before.

Consider the case $C = \{c\} \sqcap D \in \mathbf{D}_{1!}$. For the one direction, let $\mathcal{I}$ be a model of $A \sqsubseteq C$. If $\pi(\{c\} \sqsubseteq D)$ is a first-order formula that corresponds to $\{c\} \sqsubseteq D$, then $\mathcal{I} \models \neg A(c) \vee \pi(\{c\} \sqsubseteq D)$. Moreover, $\mathcal{I} \models A \sqsubseteq \{c\}$. By our assumptions and the induction hypothesis, $\mathsf{dlg}_a(\{c\} \sqsubseteq D)$ semantically emulates $\{c\} \sqsubseteq D$ – hence $\mathsf{dlg}_a(\{c\} \sqsubseteq D)|_{A(c)}$ semantically emulates $\neg A(c) \vee \pi(\{c\} \sqsubseteq D)$ –, and $\mathsf{dlg}_H(A \sqsubseteq \{c\})$ semantically emulates $A \sqsubseteq \{c\}$. Since the auxiliary symbols that may occur in both datalog programs are distinct, semantic emulation yields a single extended interpretation $\mathcal{I}'$ such that $\mathcal{I}' \models \mathsf{dlg}_a(\{c\} \sqsubseteq D)|_{A(c)}$ and $\mathcal{I}' \models \mathsf{dlg}_H(A \sqsubseteq \{c\})$, as required. The other direction is shown in a similar fashion by applying the induction hypothesis and assumptions of the lemma. □

The induction steps for defining $\mathsf{dlg}_a(\{c\} \sqsubseteq C)$ are rather more complex, and some preparation is needed first. Concepts of the forms $\mathbf{D}^+_a$, $\mathbf{D}^+_H$, and $\mathbf{D}^+_B$ allow for restricted forms of "local" disjunction. To make this notion explicit, we first elaborate how such concepts can be expressed as disjunctions of finitely many $\mathcal{DLP}$ knowledge bases.

Definition 10. *Consider concept expressions C and D such that:*

- $C \in \neg\mathbf{C}$ *and* $D \in \mathbf{D}^+_B$, *or*
- $C \in \mathbf{C}$ *and* $D \in \mathbf{D}^+_H$, *or*
- $C \in \{\mathbf{I}\}$ *and* $D \in \mathbf{D}^+_a$.

A set of knowledge bases $\mathcal{K}_{C \sqsubseteq D}$ is defined recursively as follows:

(1) *If* $D \in \mathbf{D}_a$ *then* $\mathcal{K}_{C\sqsubseteq D} \coloneqq \{\{C \sqsubseteq D\}\}$.
Assume $D \notin \mathbf{D}_a$ *for the remaining cases.*

(2) *If* $D = D_1 \sqcap D_2$ *then* $\mathcal{K}_{C\sqsubseteq D} \coloneqq \{\mathrm{KB}_1 \cup \mathrm{KB}_2 \mid \mathrm{KB}_1 \in \mathcal{K}_{C\sqsubseteq D_1}, \mathrm{KB}_2 \in \mathcal{K}_{C\sqsubseteq D_2}\}$.

(3) *If* $D = D_1 \sqcup D_2$ *then:*

(3a) *If* $D_1 \in \mathbf{C}_\geq$, *define auxiliary sets of knowledge bases* $\mathcal{K}_M$ *for* $M \subseteq \mathsf{ind}(D_1)$ *as follows:* $\mathcal{K}_M \coloneqq \left\{\{C \sqsubseteq \bigsqcap_{d\in M} \neg\{d\}\} \cup \bigcup_{d\in \mathsf{ind}(D_1)\setminus M} \mathrm{KB}_d \mid \mathrm{KB}_d \in \mathcal{K}_{\{d\}\sqsubseteq D_2}\right\}$. *Then set* $\mathcal{K}_{C\sqsubseteq D} \coloneqq \bigcup_{M\subseteq \mathsf{ind}(D_1)} \mathcal{K}_M$.

(3b) *If* $D_1 \in \mathbf{D}^+_B \setminus \mathbf{C}_\geq$, *then consider fresh concept names* B_1 *and* B_2, *and define* $\mathcal{K}_{C\sqsubseteq D} \coloneqq \left\{\{C \sqsubseteq \neg B_1 \sqcup B_2\} \cup \mathrm{KB}_1 \cup \mathrm{KB}_2 \mid \mathrm{KB}_1 \in \mathcal{K}_{\neg B_1 \sqsubseteq D_1}, \mathrm{KB}_2 \in \mathcal{K}_{B_2\sqsubseteq D_2}\right\}$.

(3c) *If* $D_1, D_2 \notin \mathbf{D}^+_B$, *then* $\mathcal{K}_{C\sqsubseteq D} \coloneqq \mathcal{K}_{C\sqsubseteq D_1} \cup \mathcal{K}_{C\sqsubseteq D_2}$.

(4) *If* $D = \geqslant n\, R.D'$ *then:*

(4a) *If* $D' \in \mathbf{D}^+_{n!}$ *then w.l.o.g.* $D' = D_1 \sqcup \ldots \sqcup D_n$ *with* $D_i = \{d_i\} \sqcap D'_i$ *and* $D'_i \in \mathbf{C}^+_a$. *Define* $\mathcal{K}_{C\sqsubseteq D} \coloneqq \left\{\{C \sqsubseteq \geqslant n\, R. \bigsqcup^n_{i=1}\{d_i\}\} \cup \bigcup^n_{i=1} \mathrm{KB}_i \mid \mathrm{KB}_i \in \mathcal{K}_{\{d_i\}\sqsubseteq D'_i}\right\}$.

(4b) *If* $D' \notin \mathbf{D}^+_{n!}$ *then consider a fresh individual name d and assume that* $\mathcal{K}_{\{d\}\sqsubseteq D'} = \{\mathrm{KB}_1, \ldots, \mathrm{KB}_s\}$. *Let* d_i *(*$i = 1, \ldots, n$*) be fresh individuals, and let* KB^i_j *denote the knowledge base* KB_j *with all occurrences of d replaced by* d_i. *Then define* $\mathcal{K}_{C\sqsubseteq D} \coloneqq \left\{\{\{d_i\} \sqcap \{d_j\} \sqsubseteq \bot \mid 1 \le i < j \le n\} \cup \{C \sqsubseteq \geqslant 1\, R.\{d_i\} \mid 1 \le i \le n\} \cup \bigcup_{1\le i\le n} \mathrm{KB}^i_{k_i} \mid k_1, \ldots, k_n \in \{1, \ldots, s\}\right\}$.

(5) *If* $D = \leqslant n\, R.\neg D'$ *then:*

(5a) *If* $D' \in \mathbf{C}_\geq$ *then a* $\geq n$*-partitioning* $\mathcal{M}$ *of* $\mathsf{ind}(D')$ *is a set* $\mathcal{M} = \{M_1, \ldots, M_m\}$ *of* $m \geq n$ *mutually disjoint non-empty sets* $M_i \subseteq \mathsf{ind}(D')$. *Given such a* $\geq n$*-partitioning, define* $\mathrm{KB}_\mathcal{M} \coloneqq \{\{c\} \sqsubseteq \{d\} \mid c, d \in M_i$ *for some* $i \in \{1, \ldots, m\}\} \cup \{C \sqcap \bigsqcap_{c\in S} \geqslant 1\, R.\{c\} \sqsubseteq \bot \mid S \subseteq \mathsf{ind}(D'), \#\{M_i \mid M_i \cap S \neq \emptyset\} > n\}$. *Then define* $\mathcal{K}_{C\sqsubseteq D} \coloneqq \{\mathrm{KB}_\mathcal{M} \mid \mathcal{M}$ *a* $\geq n$*-partitioning of* $\mathsf{ind}(D')\}$.

(5b) *If* $D' = D_1 \sqcup D_2$ *where* $D_1 \in \mathbf{C}_\geq$ *and* $D_2 \in \mathbf{D}^+_a$, *then define a set of knowledge bases* $\mathcal{K}_M$ *for a set* $M \subseteq \mathsf{ind}(D_1)$ *as follows:* $\mathcal{K}_M \coloneqq \left\{\mathrm{KB} \cup \bigcup_{d\in M} \mathrm{KB}_d \mid \mathrm{KB} \in \mathcal{K}_{C\sqsubseteq D''} \text{ with } D'' = \leqslant n\, R.\neg \bigsqcap_{d\in \mathsf{ind}(D_1)\setminus M} \neg\{d\}, \mathrm{KB}_d \in \mathcal{K}_{\{d\}\sqsubseteq D_2}\right\}$. *Then define* $\mathcal{K}_{C\sqsubseteq D} \coloneqq \bigcup_{M\subseteq \mathsf{ind}(D_1)} \mathcal{K}_M$.

(5c) *If* $n \le 1$ *and* $D' \in \mathbf{D}^+_H$ *then consider a fresh concept name B, and set* $C' \coloneqq \neg B$ *if* $D' \in \mathbf{D}^+_B$ *and* $C' \coloneqq B$ *otherwise. Define* $\mathcal{K}_{C\sqsubseteq D} \coloneqq \left\{\{C \sqsubseteq \leqslant n\, R.\neg C'\} \cup \mathrm{KB} \mid \mathrm{KB} \in \mathcal{K}_{C'\sqsubseteq D'}\right\}$.

As usual, empty conjunctions are treated as $\top$. In cases (3a) and (5b), the construction may lead to axioms in $\mathbf{L}_\top$; these axioms are omitted from $\mathcal{K}_{C\sqsubseteq D}$.

Observe that, without loss of generality, the cases in the previous definition are indeed exhaustive and mutually exclusive for $D \in \mathbf{D}_a^+$. In particular, cases (5a) and (5b) cover all situations where $D \in (\mathbf{D}_{\geq\omega-m} \cap \mathbf{D}_a^+)$, where we find $\#\mathsf{ind}(D') > n$ and $\#\mathsf{ind}(D_1) > n$, respectively, since we assume that $D \notin \mathbf{D}_a$. It is easy to verify that all recursive uses of $\mathcal{K}_{C\sqsubseteq D}$ satisfy the definition's conditions on C and D, and that all axioms in knowledge bases of $\mathcal{K}_{C\sqsubseteq D}$ are in DLP normal form. Note that case (4b) can only occur if $D \in \mathbf{D}_a^+ \setminus \mathbf{D}_H^+$, so C must be a nominal in these cases. Similar observations for the other cases allow us to state the following lemma.

Lemma 8. *Consider concept expressions C and D as in Definition 10. If D is in $\mathbf{D}_a^+$ ($\mathbf{D}_H^+$, $\mathbf{D}_B^+$) then all axioms of the form $C \sqsubseteq E$ in knowledge bases of $\mathcal{K}_{C\sqsubseteq D}$ are such that E is in $\mathbf{D}_a$ ($\mathbf{D}_H$, $\mathbf{D}_B$).*

In particular, the knowledge bases in $\mathcal{K}_{C\sqsubseteq D}$ are in $\mathcal{DLP}$.

Proof. The claim can be verified by considering all axioms that are created in the cases of Definition 10. The claims for $\mathbf{D}_a^+$, $\mathbf{D}_H^+$, and $\mathbf{D}_B^+$ are interdependent and must be proven together.

The claim clearly holds for the base case (1). Case (2) immediately follows from the induction hypothesis. Case (3a) is trivial since additional axioms of the form $C \sqsubseteq E$ do not occur in knowledge bases of $\mathcal{K}_{\{d\}\sqsubseteq D_2}$. Case (3b) and (3c) are again immediate from the induction hypothesis, where we note for (3b) that $D_1 \sqcup D_2$ is in $\mathbf{D}_a$ ($\mathbf{D}_H$, $\mathbf{D}_B$) for $D_1 \in \mathbf{D}_B \setminus \mathbf{C}_\geq$ whenever D_2 is in $\mathbf{D}_a$ ($\mathbf{D}_H$, $\mathbf{D}_B$).

Case (4a) can only occur if $D \in \mathbf{D}_H^+ \setminus \mathbf{D}_B^+$ so it suffices to note that the concept $\geqslant n\, R.\bigsqcup_{i=1}^n \{d_i\}$ is in $\mathbf{D}_H$. Case (4b) in turn requires that $D \in \mathbf{D}_a^+ \setminus \mathbf{D}_H^+$, and clearly $\geqslant 1\, R.\{d_i\} \in \mathbf{D}_a$.

Cases (5a) is immediate, since $C \sqcap \bigsqcap_{c\in S} \geqslant 1\, R.\{c\} \sqsubseteq \bot$ is equivalently expressed as $C \sqsubseteq \bigsqcup_{c\in S} \leqslant 0\, R.\neg\neg\{c\}$, the conclusion of which is in $\mathbf{D}_B$. Case (5b) follows directly by induction. Case (5c) comprises three relevant cases: $n = 0$ and $D' \in \mathbf{D}_B^+$ ($D \in \mathbf{D}_B^+$), $n = 0$ and $D' \in \mathbf{D}_H^+$ ($D \in \mathbf{D}_H^+$), $n = 1$ and $D' \in \mathbf{D}_B^+$ ($D \in \mathbf{D}_H^+$). We find that C' is in $\mathbf{D}_B$ ($\mathbf{D}_H$) whenever D' is in $\mathbf{D}_B^+$ ($\mathbf{D}_H^+$), so that the claim holds in each case.

It remains to show the second part of the claim. Using the first part of the claim, the preconditions on C and D imply that all axioms $C \sqsubseteq E$ that are constructed for $\mathcal{K}_{C\sqsubseteq D}$ are in $\mathcal{DLP}$. Axioms $C' \sqsubseteq E$ in $\mathcal{K}_{C\sqsubseteq D}$ with $C' \neq C$ must be obtained from some $\mathcal{K}_{C'\sqsubseteq D'}$ that was used in the construction of $\mathcal{K}_{C\sqsubseteq D}$. But such recursive constructions only occur in cases where the preconditions of the definition are satisfied, so the claim follows by induction. □

The next proposition shows that $C \sqsubseteq D$ is emulated by the disjunction of the knowledge bases in $\mathcal{K}_{C\sqsubseteq D}$, thus establishing the correctness of the decomposition. DL does not provide a syntax for knowledge base disjunctions, and we do not want to move to first-order logic here, so we use a slightly different formulation that follows Definition 1.

Proposition 4. *Consider concept expressions C and D as in Definition 10, both based on some signature $\mathscr{S}$. Let $\mathscr{S}'$ be the extended signature of $\mathcal{K}_{C\sqsubseteq D}$.*

- *Every interpretation $\mathcal{I}$ over $\mathscr{S}$ with $\mathcal{I} \models C \sqsubseteq D$ can be extended to an interpretation $\mathcal{I}'$ over $\mathscr{S}'$ such that $\mathcal{I}' \models \mathrm{KB}$ for some $\mathrm{KB} \in \mathcal{K}_{C \sqsubseteq D}$.*
- *For every interpretation $\mathcal{I}'$ over $\mathscr{S}'$ such that $\mathcal{I}' \models \mathrm{KB}$ for some $\mathrm{KB} \in \mathcal{K}_{C \sqsubseteq D}$, we find that $\mathcal{I}' \models C \sqsubseteq D$.*

Proof. We proceed by induction. Case (1) is obvious. Cases (2) is immediate from the induction hypothesis. For case (3a), let M be the largest set of individuals such that $\mathcal{I} \models C \sqsubseteq \bigsqcap_{d \in M} \neg\{d\}$. Using the induction hypothesis, it is easy to see that $\mathcal{I} \models C \sqsubseteq D$ implies that there is an extension $\mathcal{I}'$ of $\mathcal{I}$ such that $\mathcal{I}' \models \mathrm{KB}$ for some $\mathrm{KB} \in \mathcal{K}_M$. The converse is similar.

For case (3b), consider an interpretation $\mathcal{I}$ over $\mathscr{S}$ with $\mathcal{I} \models C \sqsubseteq D$. Consider the extended signature $\mathscr{S}'$ with the fresh concept names B_1 and B_2, and define an extension $\mathcal{I}''$ of $\mathcal{I}$ over $\mathscr{S}'$ by setting $B_1^{\mathcal{I}''} := \neg D_1^{\mathcal{I}}$ and $B_2^{\mathcal{I}''} := D_2^{\mathcal{I}}$. Then $\mathcal{I}'' \models \neg B_1 \sqsubseteq D_1$ and $\mathcal{I}'' \models B_2 \sqsubseteq D_2$, and we can apply the induction hypothesis for $\mathcal{K}_{\neg B_1 \sqsubseteq D_1}$ and $\mathcal{K}_{B_2 \sqsubseteq D_2}$ to obtain models $\mathcal{I}_i''$ (over some extended signature $\mathscr{S}'''$) such that $\mathcal{I}_1'' \models \mathrm{KB}_1$ for some $\mathrm{KB}_1 \in \mathcal{K}_{\neg B_1 \sqsubseteq D_1}$ and $\mathcal{I}_2'' \models \mathrm{KB}_2$ for some $\mathrm{KB}_2 \in \mathcal{K}_{B_2 \sqsubseteq D_2}$. Since $\mathcal{I}_1''$ and $\mathcal{I}_2''$ agree on B_1, B_2, and all symbols of $C \sqsubseteq D$, there is an interpretation $\mathcal{I}'$ such that $\mathcal{I}' \models \mathrm{KB}_1 \cup \mathrm{KB}_2$. Since $C^{\mathcal{I}} = \neg B_1^{\mathcal{I}'} \cup B_2^{\mathcal{I}'}$, it is easy to see that $\mathcal{I}'$ satisfies the conditions of the claim. The other direction of the claim for (3b) is an easy consequence of the induction hypothesis.

Case (3c) can only occur if $C \in \{\mathbf{I}\}$, and it is easy to see that the claim holds in this case.

Case (4a) is again not hard to see when using the induction hypothesis. For case (4b), first note that C must be a nominal since D is cannot be in $\mathbf{D}_H$. The required semantic emulation then is an easy consequence of standard Skolemization, where each successor d_i may satisfy any of the sufficient subconditions that are captured by $\mathrm{KB}_1^i, \ldots, \mathrm{KB}_s^i$.

The reasoning for case (5a) is similar to case (3a): given an interpretation $\mathcal{I}$, we find a $\geq n$-partitioning $\mathcal{M}$ such that $c, d \in M_i$ iff $c^{\mathcal{I}} = d^{\mathcal{I}}$. It is easy to see that $\mathcal{I} \models C \sqsubseteq D$ implies $\mathcal{I} \models \mathrm{KB}_M$; no induction is required. The other direction is again obvious.

Case (5b) is a simple extension of case (5a) where a subset M of individuals is selected in each knowledge base to ensure that all individuals of M are instances of D_2, thus reducing the requirement to a maximal number of R-successors that do not belong to M. To express this more formally, we use expressions $\geqslant 1\, U.(\{d\} \sqcap E)$ where U is the universal role that can be semantically emulated in $\mathcal{DLP}$ – this allows us to embed ABox assertions into GCIs. With this notation, we observe that $C \sqsubseteq\, \leqslant n\, R.\neg(D_1 \sqcup D_2)$ is semantically emulated by the disjunction of all the axioms $C \sqsubseteq\, \leqslant n\, R.\neg \bigsqcap_{d \in \mathsf{ind}(D_1) \setminus M} \neg\{d\} \sqcap \bigsqcap_{d \in M} \geqslant 1\, U.(\{d\} \sqcap C_2)$ for all $M \subseteq \mathsf{ind}(D_1)$. It is easy to see that the construction in (5b) corresponds to this disjunction, where conjunction is modelled as in case (2), and individual assertions are encoded using the recursive constructions $\mathcal{K}_{\{d\} \sqsubseteq D_2}$ that are valid by the induction hypothesis. The converse is easily obtained by similar considerations.

Case (5c) uses a similar argument as case (3b). Consider an interpretation $\mathcal{I}$ over $\mathscr{S}$ with $\mathcal{I} \models C \sqsubseteq D$. For the extended signature $\mathscr{S}'$ with fresh concept name B, an extension $\mathcal{I}''$ of $\mathcal{I}$ is defined by setting $C'^{\mathcal{I}''} := D^{\mathcal{I}}$. By the induction hypothesis for $\mathcal{K}_{C' \sqsubseteq D'}$, we find a model $\mathcal{I}'$ (over some extended signature $\mathscr{S}''$) such that $\mathcal{I}' \models \mathrm{KB}$ for some $\mathrm{KB} \in \mathcal{K}_{C' \sqsubseteq D'}$. But then there is a corresponding knowledge base $\mathrm{KB}' = \{C \sqsubseteq\, \leqslant n\, R.\neg C'\} \cup \mathrm{KB}$ in $\mathcal{K}_{C \sqsubseteq D}$ such that $\mathcal{I}' \models \mathrm{KB}'$. Thus $\mathcal{I}'$ satisfies the conditions of the claim when restricted to $\mathscr{S}'$. The other direction is again easy. □

We can now define datalog programs for semantically emulating axioms of the form $\{c\} \sqsubseteq \geqslant n\, R.\mathbf{D}^{\geqslant n}$. We consider all three main cases – $\mathbf{C}_{\neg m} \sqcup \mathbf{D}^{+}_{a}$, $\mathbf{D}_{B} \sqcup \mathbf{D}^{+}_{m!}$, $\mathbf{D}_{a} \sqcup \mathbf{D}^{+}_{m!} \sqcup \mathbf{D}_{l!}$ – individually, before combining these cases with the remaining forms of $\mathbf{D}_a$ to complete the induction.

Lemma 9. *Consider a constant c, and a concept $C = \geqslant n\, R.D_1 \sqcup D_2$ such that $D_1 \in \mathbf{C}_{\neg m}$, $D_2 \in \mathbf{D}^{+}_{a}$, and $(1 \leq m \leq n^2 - n)$.*

Assume that, for every individual symbol d and every knowledge base KB $\in \mathcal{K}_{\{d\} \sqsubseteq D_2}$, *there is a datalog program* $\mathsf{datalog}(\mathrm{KB})$ *that semantically emulates* KB.

Then we can effectively construct a datalog program $\mathsf{dlg}_a(\{c\} \sqsubseteq C)$ that semantically emulates $\{c\} \sqsubseteq C$.

Proof. Let h be the smallest number such that $2^h \geq (\#\mathcal{K}_{\{d\} \sqsubseteq D_2})^n$, where d is an arbitrary constant (clearly, the cardinality $\#\mathcal{K}_{\{d\} \sqsubseteq D_2}$ does not depend on the choice of d). Now let $S := \{c_{ijk} \mid i, j \in \{1, \ldots, n\}, k \in \{1, \ldots, h\}\}$ be a set of $n \times n \times h$ fresh constants. It is convenient to consider the indices of constants in S to be coordinates, so that S consists of the elements of a three dimensional matrix with n rows, n columns, and h layers. Now given any $k = 1, \ldots, h$, we define sets $A^k_i, B^k_i \subseteq S$ for all $i = 1, \ldots, n$ by setting:

$$A^k_i := \{c_{i1k}, c_{i2k}, \ldots, c_{ink}\} \quad \text{and} \quad B^k_i := \{c_{1ik}, c_{2ik}, \ldots, c_{nik}\}.$$

In other words, A^k_i (B^k_i) is the ith row (column) in layer h of S. Now given a set $O \subseteq S$, define $O(k) := \{c_{ijk} \in O \mid i, j \in \{1, \ldots, n\}\}$ – the intersection of O with layer h in S. Now for every h-tuple $v = \langle X_1, \ldots, X_h \rangle$ with $X_k \in \{A, B\}$ for all $k = 1, \ldots, h$, there is a unique partitioning $P_v = \{O_1, \ldots, O_n\}$ of S into n disjoint subsets $O_i \subseteq S$ $(1 \leq i \leq n)$ for which the following holds: for every $i \in \{1, \ldots, n\}$ and $k \in \{1, \ldots, h\}$, we find that $O_i(k) = (X_k)^k_i$. Observe that the 2^h partitions P_v that can be constructed in this way are indeed mutually distinct. Intuitively, the partitions P_v thus encode binary numbers of h digits.

Given partitionings $P = \{O_1, \ldots, O_p\}$ and $P' = \{O'_1, \ldots, O'_{p'}\}$ of S, we say that P is *finer than* P' if, for every $i \in \{1, \ldots, p'\}$, we find that O'_i is a union of parts $O_j \in P$. Note that every part O_j can be contained in at most one part O'_i, and thus $p' \leq p$. Partitions of the form P_v have the following important property: for every partition $P = \{O_1, \ldots, O_p\}$ of S with $p \in \{n, \ldots, n + m - 1\}$, there is at most one partition of the form P_v such that P is finer than P_v. To show this, consider two h-tuples $v, w \subseteq \{A, B\}^h$ that differ in (at least) the kth component ($k \in \{1, \ldots, h\}$), i.e. (w.l.o.g.) the kth component of v is A, and the kth component of w is B. Now for any partition P that is finer than P_v and P_w, for every $i \in \{1, \ldots, n\}$ there are parts $O_1, \ldots, O_j \in P$ such that $A^k_i = O_1(k) \cup \ldots \cup O_j(k)$, and parts $O'_1, \ldots, O'_{j'} \in P$ such that $B^k_i = O'_1(k) \cup \ldots \cup O'_{j'}(k)$. This implies that P cannot contain a part O such that $\#O(k) > 1$ since no two sets A^k_i and $B^k_{i'}$ share more than one constant. Hence P must have at least n^2 parts to cover all elements in layer k. Now the precondition $m \leq n^2 - n$ implies that $n + m - 1 < n^2$, which establishes the claim.

To establish the required datalog program, partitions of constants are considered as equality classes, and rules are created to check for particular equalities. To this end, define a conjunction $[\![O]\!] := c_1 \wedge \ldots \wedge c_j$ for every set $O = \{c_1, \ldots, c_n\} \subseteq S$. This notation is extended to partitions $P = \{O_1, \ldots, O_i\}$ of S by setting $[\![P]\!] := [\![O_1]\!] \wedge \ldots \wedge [\![O_i]\!]$.

Consider a fresh constant d. For every h-tuple $v \in \{A, B\}^h$, let $\phi_v : P_v \to \mathcal{K}_{\{d\} \sqsubseteq D_2}$ be a mapping of parts of P_v to knowledge bases in $\mathcal{K}_{\{d\} \sqsubseteq D_2}$ such that, for every n-tuple

$K = \langle \mathrm{KB}_1, \ldots, \mathrm{KB}_n \rangle \in \mathcal{K}^n_{\{d\}\sqsubseteq D_2}$ of knowledge bases, there is an h-tuple $w \in \{A, B\}^h$ with partition $P_v = \{O_1, \ldots, O_n\}$ as defined above, and $\phi_w(O_i) = \mathrm{KB}_i$ for all $i = 1, \ldots, n$. This choice of the functions ϕ_v is possible due to our initial choice of h, since there are 2^h such functions but only $\#\mathcal{K}^n_{\{d\}\sqsubseteq D_2}$ different n-tuples of knowledge bases from $\mathcal{K}_{\{d\}\sqsubseteq D_2}$.

For every partition P of S into $i \in \{1, \ldots, n + m - 1\}$ parts, datalog rules are constructed as follows. If P is not finer than any partition of the form P_v, then only the rule $[\![P]\!] \rightarrow \bot$ is added (this includes the case of P having less than n parts). Otherwise, let P_v be the unique partition of this form that is finer than P. For every part O of P_v, select one part $\pi(O)$ of P such that $\pi(O) \subseteq O$, so that there are n distinct *selected parts* in P. Now let $d_1, \ldots, d_m$ denote the m constants of D_1. For every $e = d_1, \ldots, d_m$ and for every part $O \in P_v$, let A be a fresh concept name and construct the following datalog:

(i) $[\![P]\!] \wedge e \approx f \rightarrow A(e)$, where $f \in \pi(O)$ is arbitrary,
(ii) $\mathsf{datalog}(\mathrm{KB}')|_{A(e)}$ where KB' is obtained from $\phi_v(O)$ by replacing all occurrences of $\{d\}$ with $\{e\}$.

Now $\mathsf{dlg}_a(\{c\} \sqsubseteq C)$ is defined to be the union of P_{Inv} and all datalog rules constructed above, and the datalog facts $R(c, c_{ijk})$ for all $i, j \in \{1, \ldots, n\}$ and $k \in \{1, \ldots, h\}$.

It remains to show that $\mathsf{dlg}_a(\{c\} \sqsubseteq C)$ semantically emulates $\{c\} \sqsubseteq C$. For the one direction, consider a model $\mathcal{I}$ of $\{c\} \sqsubseteq C$. We need to show that it can be extended to a model of $\mathsf{dlg}_a(\{c\} \sqsubseteq C)$. Select n distinct R-successors $\delta_1, \ldots, \delta_n$ of $c^\mathcal{I}$ such that $\delta_i \in (D_1 \sqcup D_2)^\mathcal{I}$ for all $i = 1, \ldots, n$. By Proposition 4, for all $e \in \{d_1, \ldots, d_m\}$, if $e^\mathcal{I} \in D_2^\mathcal{I}$ then there is an extended interpretation $\mathcal{I}_e$ such that $\mathcal{I}_e \models \mathrm{KB}_e$ for some $\mathrm{KB}_e \in \mathcal{K}_{\{e\}\sqsubseteq D_2}$. Since $\mathcal{I}_e$ extends $\mathcal{I}$ only over fresh symbols that occur in one $\mathcal{K}_{\{e\}\sqsubseteq D_2}$, all interpretations $\mathcal{I}_e$ can be combined into a single extension $\mathcal{I}'$ of $\mathcal{I}$.

Now let $\mathrm{KB}'_e \in \mathcal{K}_{\{d\}\sqsubseteq D_2}$ denote the knowledge base from which KB_e is obtained by replacing all axioms of the form $\{d\} \sqsubseteq F$ by $\{e\} \sqsubseteq F$, where d is the constant used when constructing $\mathsf{dlg}_a(\{c\} \sqsubseteq C)$. By the construction of $\mathsf{dlg}_a(\{c\} \sqsubseteq C)$, there is a tuple $v \in \{A, B\}^h$ and a partition $P_v = \{O_1, \ldots, O_n\}$ such that $\phi_v(O_i) = \mathrm{KB}'_{d_j}$ for all $i = 1, \ldots, n$ for which $d_j^\mathcal{I} = \delta_i$ and $d_l^\mathcal{I} \neq \delta_i$ for all $l < j$.

Consider any $e \in \{d_1, \ldots, d_m\}$ with $e^\mathcal{I} \in D_2^\mathcal{I}$. The model $\mathcal{I}'$ above was constructed such that $\mathcal{I}' \models \mathrm{KB}_e$, and thus, by the assumption of the lemma, there is an extension $\mathcal{J}'$ of $\mathcal{I}'$ such that $\mathcal{J}' \models \mathsf{datalog}(\mathrm{KB}_e)$. We define a model $\mathcal{J}$ of $\mathsf{dlg}_a(\{c\} \sqsubseteq C)$ by further extending $\mathcal{J}'$. For all constants $f \in S$, define $f^\mathcal{J} := \delta_i$ for the unique $i \in \{1, \ldots, n\}$ such that $f \in O_i$. Moreover, for each of the fresh concept name A introduced in (i) above, let $A^\mathcal{J}$ be the smallest extension for which all rules of (i) are satisfied by $\mathcal{J}$.

Now it is easy to see that $\mathcal{J}$ satisfies the facts $R(c, c_{ijk})$ for all $i, j \in \{1, \ldots, n\}$ and $k \in \{1, \ldots, h\}$. To see that it also satisfies the rules constructed in (ii) above, note that the rules (ii) for some particular $e \in \{c_1, \ldots, c_m\}$ are always satisfied if $\mathcal{J} \not\models A(e)$. Assume $\mathcal{J} \models A(e)$. By minimality of $A^\mathcal{J}$, this implies that $\mathcal{J} \models e \approx f$ for some $f \in S$ that belongs to a part O_i of P_v, and thus $e^\mathcal{J} = \delta_i$ for some $i \in \{1, \ldots, n\}$. By construction, $\phi_v(O_i)$ is of the form KB'_{d_j} (where e might be unequal to d_j, but with $e^\mathcal{J} = d_j^\mathcal{J} = \delta_i$). Since $\delta_i \in (D_1 \sqcup D_2)^\mathcal{I}$, we find $\delta_i \in D_2^\mathcal{I}$ and thus $\mathcal{J} \models \mathsf{dlg}_a(\{b'\} \sqsubseteq F)$ for all $\{b\} \sqsubseteq F \in \mathrm{KB}'_{d_j}$, where $b' = e$ if $b = d$ and $b' = b$ otherwise. This shows that the rules (ii) are indeed satisfied by $\mathcal{J}$.

For the other direction, consider a model $\mathcal{I}$ of $\mathsf{dlg}_a(\{c\} \sqsubseteq C)$. We need to show that it is also a model of $\{c\} \sqsubseteq C$. Let P be the partition of S that corresponds to the $\approx$ equivalence classes on S induced by $\mathcal{I}$. By the construction of $\mathsf{dlg}_a(\{c\} \sqsubseteq C)$, the partition P is finer than some partition of the form P_v, and thus has at least n parts. Moreover, n of the parts of P are selected parts of the form $\pi(O)$ for some $O \in P_v$. It is not hard to see that the n domain elements of $\mathcal{I}$ that correspond to the selected parts are R-successors of c that belong to $(D_1 \sqcup D_2)^{\mathcal{I}}$, which is an easy consequence of rules (i) and (ii) together with the assumed model-theoretic correspondences for axioms in $\mathcal{K}_{\{d\} \sqsubseteq D_2}$. □

Lemma 10. *Consider a constant c, and a concept $C = \geqslant n\, R.D_1 \sqcup D_2$ such that $D_1 \in \mathbf{D}_B$, $D_2 \in \mathbf{D}^+_{m!}$ with $m < n$ of the form $D_2 = (\{c_1\} \sqcap C_1) \sqcup \ldots \sqcup (\{c_m\} \sqcap C_m)$.*

Assume that, for every $i \in \{1, \ldots, m\}$ and every knowledge base KB $\in \mathcal{K}_{\{c_i\} \sqsubseteq C_i}$*, there is a datalog program* $\mathsf{datalog}(\mathrm{KB})$ *that semantically emulates* KB.

Then we can effectively construct a datalog program $\mathsf{dlg}_a(\{c\} \sqsubseteq C)$ *that semantically emulates* $\{c\} \sqsubseteq C$.

Proof. For each $i = 1, \ldots, m$, let $l_i \geq 1$ be the least number such that $2^{l_i} \geq \#\mathcal{K}_{\{c_i\} \sqsubseteq C_i}$, and consider a set S_i of fresh constants $S_i \coloneqq \{a_{i1}, b_{i1}, \ldots, a_{il_i}, b_{il_i}\}$. Let V_i denote the set of all sets of the form $\{x_1, x_2, \ldots, x_{l_i}\}$ with $x_h \in \{a_{ih}, b_{ih}\}$ for all $h \in \{1, \ldots, l_i\}$. Let $\phi_i : V_i \to \mathcal{K}_{\{c_i\} \sqsubseteq C_i}$ be an arbitrary surjective function (which exists due to the choice of a sufficiently large l_i).

Consider fresh constants $d_1, \ldots, d_{n-m}$ (note that $n - m \geq 1$) and a fresh concept name B. We construct the following datalog rules and programs:

(i) $\mathsf{dlg}_B(\neg B \sqsubseteq D_1)$

(ii) for every $i \in \{1, \ldots, n-m\}$:
$R(c, d_i)$,
$B(d_i) \to \bot$ for a fresh concept name A,

(iii) for every $i, j \in \{1, \ldots, n-m\}$, $i \neq j$:
$d_i \approx d_j \to \bot$,

(iv) for every $i \in \{1, \ldots, n-m\}$ and $j \in \{1, \ldots, m\}$:
$d_i \approx c_j \to \bot$,

(v) for every $i \in \{1, \ldots, m\}$ and $h \in \{1, \ldots, l_i\}$:
$R(c, a_{ih})$,
$R(c, b_{ih})$,
$a_{ih} \approx b_{ih} \to \bot$,

(vi) for every $i \in \{1, \ldots, m\}$ and $v = \{x_{i1}, x_{i2}, \ldots, x_{il_i}\} \in V_i$:
$B(x_{i1}) \wedge \ldots \wedge B(x_{il_i}) \to A(c_i)$ for a fresh concept name A,
$A(c_i) \to c_i \approx x_{i1}$,
$\mathsf{datalog}(\phi_i(v))|_{A(c_i)}$,

(vii) for every $i, j \in \{1, \ldots, m\}$, $i \neq j$, for every $e \in S_i$ and $f \in S_j \cup \{d_1, \ldots, d_{n-m}\}$:
$e \approx f \to e \approx d_1$.

Now $\mathsf{dlg}_a(\{c\} \sqsubseteq C)$ is defined as the union of P_{Inv} and all rules and programs constructed above.

It remains to show that $\mathsf{dlg}_a(\{c\} \sqsubseteq C)$ semantically emulates $\{c\} \sqsubseteq C$. For the one direction, consider any model $\mathcal{I}$ of $\{c\} \sqsubseteq C$. Select n distinct R-successors $\delta_1, \ldots, \delta_n$ of $c^{\mathcal{I}}$ such that $\delta_i \in (D_1 \sqcup D_2)^{\mathcal{I}}$ for all $i = 1, \ldots, n$. By Proposition 4, for all $i \in \{1, \ldots, m\}$, if $c_i^{\mathcal{I}} \in C_i^{\mathcal{I}}$ then there is an extended interpretation $\mathcal{I}_i$ such that $\mathcal{I}_i \models \mathrm{KB}_i$ for some $\mathrm{KB}_i \in \mathcal{K}_{\{c_i\} \sqsubseteq C_i}$. Since $\mathcal{I}_i$ extends $\mathcal{I}$ only over fresh symbols that occur in one $\mathcal{K}_{\{c_i\} \sqsubseteq C_i}$, all interpretations $\mathcal{I}_i$ can be combined into a single extension $\mathcal{I}'$ of $\mathcal{I}$. By the assumption of the lemma, we find an extension $\mathcal{J}'$ of $\mathcal{I}'$ such that $\mathcal{J}' \models \mathsf{datalog}(\mathrm{KB}_i)$.

A model $\mathcal{J}$ of $\mathsf{dlg}_a(\{c\} \sqsubseteq C)$ is defined by further extending $\mathcal{J}'$. For the auxiliary concept B of (i), define $B^{\mathcal{J}} := (\neg D_1)^{\mathcal{I}}$ and let $\mathcal{J}$ be such that $\mathcal{J} \models \mathsf{dlg}_B(\neg B \sqsubseteq D_1)$ (which is possible by Lemma 6). For each $i \in \{1, \ldots, n - m\}$, select $d_i^{\mathcal{J}} \in \{\delta_1, \ldots, \delta_n\}$ such that rules (ii)–(iv) above are satisfied. This is always possible since at most m elements of $\{\delta_1, \ldots, \delta_n\}$ can be in $(\neg D_1)^{\mathcal{I}}$. Without loss of generality, we assume that $d_i^{\mathcal{J}} = \delta_i$.

Now select an injective function $\psi : \{1, \ldots, m\} \to \{2, \ldots, n\}$ such that $\psi(i) = j$ if $c_i^{\mathcal{I}} = \delta_j$ for some $j \in \{1, \ldots, n\}$ and there is no $i' < i$ such that $c_{i'}^{\mathcal{I}} = \delta_j$; and $\psi(i) \in D_1^{\mathcal{I}}$ otherwise. Again, it is not hard to see that this is always possible. Now for each $i \in \{1, \ldots, m\}$, interpretations for constants in S_i are defined as follows. If $c_i^{\mathcal{I}} \in C_i^{\mathcal{I}}$, then let $v \in V_i$ be such that $\mathrm{KB}_i = \phi_i(v)$. Otherwise, let $v \in V_i$ be arbitrary. For all $h \in \{1, \ldots, l_i\}$ and $x \in \{a_{ih}, b_{ih}\}$, define $x^{\mathcal{J}} := \delta_{\psi(i)}$ if $x \in v$, and $x^{\mathcal{J}} := \delta_1$ otherwise. It is not hard to see that $\mathcal{J}$ satisfies rules (v) and (vii). For the auxiliary concepts A introduced in (vi) for some set $w \in V_i$, set $A^{\mathcal{J}} := \{c_i^{\mathcal{I}}\}$ if $w = v$ and $\delta_{\psi(i)} \in (\neg D_1)^{\mathcal{I}}$ (which also implies $c_i^{\mathcal{I}} = \delta_{\psi(i)}$), and set $A^{\mathcal{J}} := \emptyset$ otherwise. Thus, there is at most one such auxiliary concept for i that is non-empty, corresponding to the set $v \in V_i$ for which $\mathrm{KB}_i = \phi_i(v)$. The construction of $\mathcal{J}'$ ensures that the remaining rules of (vi) are satisfied as required. It should be observed that this construction also works in the case that $c_i^{\mathcal{I}} = c_j^{\mathcal{I}}$ for some $i \neq j$.

For the other direction, consider any model $\mathcal{I}$ of $\mathsf{dlg}_a(\{c\} \sqsubseteq C)$. The rules of (i)–(iv) obviously establish $n - m$ distinct R-successors $d_1, \ldots, d_{n-m}$ of c that are in D_1. According to rules (vii), for every $i \in \{1, \ldots, m\}$ and every $k \in \{1, \ldots, l_i\}$, some $x_{ik} \in \{a_{ik}, b_{ik}\}$ is unequal to all constants in $S_j \cup \{d_1, \ldots, d_{m-n}\}$ for all $j \neq i$ with $j \in \{1, \ldots, m\}$. Hence, if the premise of the first rule of (vi) is false for all $v \in V_i$, then there must be some $k \in \{1, \ldots, l_i\}$ such that $x_{ik}^{\mathcal{I}} \notin B^{\mathcal{I}}$ and hence, by (i), $x_{ik}^{\mathcal{I}} \in D_1^{\mathcal{I}}$, yielding the required distinct R-successor for i. Otherwise, if the premise of the first rule of (vi) is true for some $v \in V_i$, then $c_i^{\mathcal{I}} \approx x_{i1}$ is the required successor, since $c_i^{\mathcal{I}} \in D_2^{\mathcal{I}}$ is ensured by the rules of (vi) together with the assumptions of the lemma. □

Lemma 11. *Consider a constant c, and a concept $C = \geqslant n\, R.D_1 \sqcup D_2 \sqcup D_3$ such that $D_1 \in \mathbf{D}_a$, $D_2 \in \mathbf{D}^{+}_{m!}$ of the form $D_2 = (\{c_1\} \sqcap C_1) \sqcup \ldots \sqcup (\{c_m\} \sqcap C_m)$, $D_3 \in \mathbf{D}^{+}_{l!}$ of the form $D_3 = (\{c_{m+1}\} \sqcap C_{m+1}) \sqcup \ldots \sqcup (\{c_{m+l}\} \sqcap C_{m+l})$, and for $r := n - (m + l)$ we have $r > 0$ and $r(r - 1) \geq m$.*

Assume that, for every constant e, $\mathsf{dlg}_a(\{e\} \sqsubseteq D_1)$ semantically emulates $\{e\} \sqsubseteq D_1$, and that, for every $\mathrm{KB} \in \mathcal{K}_{\{c_i\} \sqsubseteq C_i}$ ($i \in \{1, \ldots, m + l\}$), $\mathsf{datalog}(\mathrm{KB})$ semantically emulates KB.

Then we can effectively construct a datalog program $\mathsf{dlg}_a(\{c\} \sqsubseteq C)$ that semantically emulates $\{c\} \sqsubseteq C$.

Proof. Let $s \geq 1$ be such that $2^s \geq \prod_{i=1}^{m} \#\mathcal{K}_{\{c_i\} \sqsubseteq C_i}$. Consider the following sets of fresh constants:

- $\{d_i \mid i = 1, \ldots, r\}$,
- $\{e_{ij} \mid i = 1, \ldots, m, j = 1, \ldots, s\}$,
- $\{f_i \mid i = 1, \ldots, l\}$.

Now, for each $i = 1, \ldots, m$, let $\phi_i : \{1, 2\}^s \to \mathcal{K}_{\{c_i\} \sqsubseteq C_i}$ be a surjective function from s-ary binary numbers to $\mathcal{K}_{\{c_i\} \sqsubseteq C_i}$, which exists due to our choice of s. Moreover, for each $i = 1, \ldots, m$, let $\psi_i = \langle h, k \rangle$ be a pair of distinct numbers $h, k \in \{1, \ldots, r\}, h \neq k$ such that $\psi_i \neq \psi_j$ whenever $i \neq j$. This choice is possible since there are $r(r-1)$ such pairs and $r(r-1) \geq m$ was assumed. Given any j-ary tuple θ, we use $\theta(k)$ to denote the kth component of θ for $k = 1, \ldots, j$. In particular, we use the notation $\psi_i(\nu(j))$ ($i = 1, \ldots, m$, $j = 1, \ldots, s$) with tuples $\nu \in \{1, 2\}^s$ below.

Let B be a fresh concept name – we will use it to mark certain distinct R-successors that the datalog program must ensure to exist. We construct the following datalog rules and programs:

(i) for all $e, f \in \{d_1, \ldots, d_r, c_1, \ldots, c_{m+l}\}$ with $e \neq f$:
$B(e) \wedge B(f) \wedge e \approx f \to \bot$,
$B(e) \to R(c, e)$,
(ii) for all $i \in \{1, \ldots, r\}$:
$B(d_i)$,
$\mathsf{dlg}_a(\{d_i\} \sqsubseteq D_1)$,
(iii) for all $i \in \{1, \ldots, m\}$, $\nu \in \{1, 2\}^s$, $h \in \{1, \ldots, s\}$:
$R(c, e_{ih})$,
$\mathsf{dlg}_a(\{e_{ih}\} \sqsubseteq D_1)$,
for all $j \in \{1, \ldots, r\}$, $j \neq \psi_i(1)$, $j \neq \psi_i(2)$: $e_{ih} \approx d_j \to \bot$,
$e_{i1} \approx d_{\psi_i(\nu(1))} \wedge \ldots \wedge e_{is} \approx d_{\psi_i(\nu(s))} \to A(c_i)$ for a fresh concept name A,
$A(c_i) \to B(c_i)$,
$\mathsf{datalog}(\phi_i(\nu))|_{A(c_i)}$,
(iv) for all $i, j \in \{1, \ldots, m\}$ with $i \neq j$, for all $h \in \{1, \ldots, s\}$:
if there is $k \in \{1, 2\}$ such that $\psi_i(k) = \psi_j(k)$: $e_{ih} \approx e_{jh} \to e_{ih} \approx d_{\psi_i(k)}$,
otherwise: $e_{ih} \approx e_{jh} \to \bot$,
(v) for all $i \in \{1, \ldots, l\}$, $j \in \{1, \ldots, r\}$:
$R(c, f_i)$,
$\mathsf{dlg}_a(\{f_i\} \sqsubseteq D_1)$,
$f_i \approx d_j \to A(c_{m+i})$ for a fresh concept name A,
$A(c_{m+i}) \to B(c_{m+i})$,
$\mathsf{dlg}_a(\{c_{m+i}\} \sqsubseteq C_{m+i})|_{A(c_{m+i})}$,
(vi) for all $e \in \{f_1, \ldots, f_l, e_{11}, \ldots, e_{1s}, \ldots, e_{m1}, \ldots, e_{ms}\}$:
for all $f \in \{f_1, \ldots, f_l\}$ with $e \neq f$: $f \approx e \to f \approx d_1$,
for all $f \in \{c_1, \ldots, c_{m+l}\}$: $B(f) \wedge f \approx e \to \bot$.

Now $\mathsf{dlg}_a(\{c\} \sqsubseteq C)$ is defined as the union of P_{Inv} and all rules and programs constructed above.

It remains to show that $\mathsf{dlg}_a(\{c\} \sqsubseteq C)$ that semantically emulates $\{c\} \sqsubseteq C$. For the one direction, consider any model $\mathcal{I}$ of $\{c\} \sqsubseteq C$. Select n distinct R-successors $\delta_1, \ldots, \delta_n$

of $c^{\mathcal{I}}$ such that $\delta_i \in (D_1 \sqcup D_2 \sqcup D_3)^{\mathcal{I}}$ for all $i = 1, \ldots, n$. By Proposition 4, for all $i \in \{1, \ldots, m\}$, if $c_i^{\mathcal{I}} \in C_i^{\mathcal{I}}$ then there is an extended interpretation $\mathcal{I}_i$ such that $\mathcal{I}_i \models \mathrm{KB}_i$ for some $\mathrm{KB}_i \in \mathcal{K}_{\{c_i\} \sqsubseteq C_i}$. As in the proof of Lemma 10 above, we can find an extended interpretation $\mathcal{J}'$ such that $\mathcal{J}' \models \mathrm{KB}_i$. Using a similar argument, we can chose $\mathcal{J}'$ such that $\mathcal{J}' \models \mathsf{dlg}_a(\{c_j\} \sqsubseteq C_j)$ for each $j \in \{m+1, \ldots, m+l\}$ for which $c_j^{\mathcal{I}} \in C_j^{\mathcal{I}}$.

A model $\mathcal{J}$ of $\mathsf{dlg}_a(\{c\} \sqsubseteq C)$ is defined by further extending $\mathcal{J}'$. At least r elements $\delta \in \{\delta_1, \ldots, \delta_n\}$ must satisfy $\delta \in D_1^{\mathcal{I}}$ – w.l.o.g. we assume that this is the case for $\delta_1, \ldots, \delta_r$. Then set $d_i^{\mathcal{J}} := \delta_i$ for all $i \in \{1, \ldots, r\}$.

Now select an injective function $\sigma : \{1, \ldots, m+l\} \to \{1, \ldots, n\}$ such that $\sigma(i) = j$ if $c_i^{\mathcal{I}} \in C_i^{\mathcal{I}}$, $c_i^{\mathcal{I}} = \delta_j$ for some $j \in \{1, \ldots, n\}$ and there is no $i' < i$ such that $c_{i'}^{\mathcal{I}} = \delta_j$; and $\sigma(i) \in D_1^{\mathcal{I}}$ otherwise. Such a function clearly exists. Consider some $i \in \{1, \ldots, m\}$. If $\delta_{\sigma(i)}^{\mathcal{I}} \in D_1^{\mathcal{I}}$, then set $e_{ih}^{\mathcal{J}} := \sigma(i)$ for each $h \in \{1, \ldots, s\}$. Otherwise, $\delta_{\sigma(i)} = c_i^{\mathcal{I}}$ and $c_i^{\mathcal{I}} \in C_i^{\mathcal{I}}$. In this case, let $\nu \in \{1, 2\}^s$ be such that $\mathrm{KB}_i = \phi_i(\nu)$, and define $e_{ih}^{\mathcal{J}} := d_{\psi_i(\nu(h))}^{\mathcal{J}}$ for each $h \in \{1, \ldots, s\}$. Finally, for $i \in \{1, \ldots, l\}$, define $f_i^{\mathcal{J}} := \delta_{\sigma(m+i)}$.

By the assumption of the lemma, for each program of the form $\mathsf{dlg}_a(\{e\} \sqsubseteq D)$ that is constructed in rules (ii), (iii), and (v), we can extend $\mathcal{J}$ to symbols of $\mathsf{dlg}_a(\{e\} \sqsubseteq D)$ so that the respective programs are satisfied. For B we select the smallest extensions $B^{\mathcal{J}}$ for which the rules of (ii), (iii), and (v) that use B are satisfied. It is easy to check that the rules of (i) are satisfied. Similarly, we assign minimal extensions to all auxiliary concept names A introduced in (iii) and (v). Now it is not hard to check that $\mathcal{J}$ satisfies all rules of (i)–(vi) as required.

For the other direction, consider any model $\mathcal{I}$ of $\mathsf{dlg}_a(\{c\} \sqsubseteq C)$. The rules of (ii) establish r distinct R-successors $d_1, \ldots, d_r$ of c that are in D_1. For any $i \in \{1, \ldots, l\}$, the rules of (iv) ensure that f_i is not equal to any c_j in B. The rules of (v) leave two possibilities. Either f_i is equal to some constant d_j, in which case c_{m+i} is an R-successor of c that is in C_{m+i}, and that is distinct from all other c_h and d_h by (i). Or f_i is not equal to any constant d_j or f_h $(h \neq i)$, and thus not equal to any e_{hk} either (vi); so f_i constitutes a new R-successor of c that is in D_1.

For any $i \in \{1, \ldots, m\}$, if some e_{ih} is not equal to $d_{\psi_i(1)}$ or $d_{\psi_i(2)}$, then the rules of (iii) and (iv) ensure that e_{ih} is not equal to any other constant of the form d_j or e_{jk}. Rules (iv) ensure that e_{ih} is also not equal to any constant of the form f_j, and thus e_{ih} constitutes an additional R-successor of c that is in D_1. If no such e_{ih} exists, then a rule of (iii) applies for some $\nu \in \{1, 2\}^s$, implying that $c_i^{\mathcal{I}} \in A^{\mathcal{I}}$ for the respective fresh concept name A. But then the rules of (iii) together with the assumptions of the lemma imply that $\mathcal{I} \models \phi_i(\nu) \in \mathcal{K}_{\{c_i\} \sqsubseteq C_i}$. By Proposition 4, we find that $c_i^{\mathcal{I}} \in C_i^{\mathcal{I}}$. Rules (i) and (iv) ensure that c_i is distinct from the remaining R-successors. Overall, we thus obtain $r + m + l = n$ distinct R-successors of c that belong to $D_1 \sqcup D_2 \sqcup D_3$. □

Lemma 12. *Consider a concept $C \in \mathbf{D}_a$ and constant c such that every datalog program $\mathsf{dlg}_a(\{c\} \sqsubseteq D)$ ($\mathsf{dlg}_H(X \sqsubseteq D)$) on the right hand side of Fig. 9 semantically emulates $\{c\} \sqsubseteq D$ ($X \sqsubseteq D$). Then the datalog program $\mathsf{dlg}_a(\{c\} \sqsubseteq C)$ as defined in Fig. 9 semantically emulates $\{c\} \sqsubseteq C$.*

C	$\mathsf{dlg}_a(\{c\} \sqsubseteq C)$
$D \in \mathbf{D}_H$	$\mathsf{dlg}_H(X \sqsubseteq D) \cup \{X(c)\}$
$D_1 \sqcap D_2$	$\mathsf{dlg}_a(\{c\} \sqsubseteq D_1) \cup \mathsf{dlg}_a(\{c\} \sqsubseteq D_2)$
$D_1 \sqcup D_2 \in (\mathbf{D}_a \sqcup \mathbf{D}_B)$	$\mathsf{dlg}_B(\neg X \sqsubseteq D_2) \cup \mathsf{dlg}_a(\{c\} \sqsubseteq D_1)\vert_{X(c)}$
$\geqslant n\, R.\top$	$\{R(c, a_1), \ldots, R(c, a_n)\} \cup P_{\mathrm{Inv}}$
$\geqslant n\, R.D \quad (D \neq \top)$	$\mathsf{dlg}_a(\{c\} \sqsubseteq C)$ as defined in Lemma 9, 10, and 11
X a fresh concept name, a_i fresh constants	

Fig. 9. Transforming axioms $\{\mathbf{I}\} \sqsubseteq \mathbf{D}_a$ to datalog

Proof. The proof proceeds by induction. The complex cases have already been established in Lemma 9, 10, and 11. The remaining induction steps are very similar to the steps in Lemma 6 and 7. □

We can now complete our induction by summarising the previous lemmata.

Proposition 5. *Consider concepts $C \in \mathbf{D}_H$, $D \in \mathbf{D}_a$, a concept name A, and a constant symbol c. Lemma 6, 7, 9, 10, 11, and 12 together define a recursive construction procedure for datalog programs $\mathsf{dlg}_H(A \sqsubseteq C)$ and $\mathsf{dlg}_a(\{c\} \sqsubseteq D)$ that semantically emulate $A \sqsubseteq C$ and $\{c\} \sqsubseteq D$, respectively.*

Proof. The mentioned results are the basis for establishing an inductive argument to proof the claim. Lemma 9, 10 and 11 require the existence of certain datalog programs $\mathsf{datalog}(\mathrm{KB})$. For this proof, we define $\mathsf{datalog}(\mathrm{KB}) \coloneqq \{\mathsf{dlg}_B(\neg A \sqsubseteq E) \mid \neg A \sqsubseteq E \in \mathrm{KB}\} \cup \{\mathsf{dlg}_H(A \sqsubseteq E) \mid A \sqsubseteq E \in \mathrm{KB}\} \cup \{\mathsf{dlg}_a(\{f\} \sqsubseteq E) \mid \{f\} \sqsubseteq E \in \mathrm{KB}\}$ (we provide a more general definition of $\mathsf{datalog}(\mathrm{KB})$ for other forms knowledge bases at the end of this section). According to Lemma 8 this definition is well and covers all axioms that can occur in KB.

It remains to show that the preconditions of each induction step are indeed satisfied by applying the induction hypothesis that the claim hold for proper subconcepts of the considered concepts. This is obvious whenever preconditions require the claim to hold for programs of the form $\mathsf{dlg}_H(A' \sqsubseteq C')$ or $\mathsf{dlg}_a(\{c'\} \sqsubseteq D')$ where C' and D' are proper subconcepts of C and D, respectively.

The induction steps for $\mathsf{dlg}_a(\{c\} \sqsubseteq D)$, however, need to use Lemma 9, 10 which 11 additionally require that, for a proper subconcept D' of D and some $\mathrm{KB} \in \mathcal{K}_{\{c'\} \sqsubseteq D'}$, the claim holds for all programs $\mathsf{dlg}_H(A \sqsubseteq E)$ with $A \sqsubseteq E \in \mathrm{KB}$ and for all programs $\mathsf{dlg}_a(\{f\} \sqsubseteq E)$ with $\{f\} \sqsubseteq E \in \mathrm{KB}$ (the translations $\mathsf{dlg}_B(\neg A \sqsubseteq E)$ are always given by Lemma 6). Inspecting Definition 10, we find that most axioms in knowledge bases of $\mathcal{K}_{\{c'\} \sqsubseteq D'}$ are of the form $C \sqsubseteq D''$ with D'' a proper subconcept of D', so that the induction hypothesis applies. However, all cases other than (1), (2), and (3c) also introduce additional axioms that are not referring to subconcepts. By checking the recursive definitions of these axioms, it is easy to see that the claim holds for all axioms of this form. □

We still need to show that the "propositional" concepts in $\mathbf{D}^{=n}$ can also be emulated in datalog.

α	$\mathsf{dlg}(\alpha) \setminus P_{\mathrm{Inv}}$
$\mathsf{Ref}(R)$	$\{R(x,x)\}$
$\mathsf{Irr}(R)$	$\{R(x,x) \to \bot\}$
$\mathsf{Sym}(R)$	$\{R(x,y) \to R(y,x)\}$
$\mathsf{Asy}(R)$	$\{R(x,y) \wedge R(y,x) \to \bot\}$
$\mathsf{Dis}(R_1, R_2)$	$\{R_1(x,y) \wedge R_2(x,y) \to \bot\}$
$\mathsf{Tra}(R)$	$\{R(x,y) \wedge R(y,z) \to R(x,z)\}$
$R_1 \circ R_2 \circ \ldots \circ R_n \sqsubseteq R$	$\{R_1(x_0,x_1) \wedge \ldots \wedge R_n(x_{n-1},x_n) \to R(x_0,x_n)\}$

Fig. 10. Transforming $\mathcal{SROIQ}$ RBox axioms to datalog

Lemma 13. *For every concept $C \in \mathbf{D}^{=n}$ for some $n \geq 1$, one can construct a datalog program $\mathsf{datalog}(C)$ that semantically emulates C.*

Proof. C is of the form $(\{c_1\} \sqcap C_1) \sqcup \ldots \sqcup (\{c_n\} \sqcap C_n)$ with $C_1 \in \mathbf{C}_H^p$ and $C_i \in \mathbf{C}_\bot^{=i}$ for $i = 2, \ldots, n$. It is not hard to see that C is semantically equivalent to $\{c_1\} \sqcap C_1$. This is shown by induction over n. Clearly, all models of C have domains with at most n elements. By Lemma 5, for all $n > 2$, $(\{c_1\} \sqcap C_1) \sqcup \ldots \sqcup (\{c_n\} \sqcap C_n)$ is semantically equivalent to $(\{c_1\} \sqcap C_1) \sqcup \ldots \sqcup (\{c_{n-1}\} \sqcap C_{n-1})$, as required.

All models of $\{c_1\} \sqcap C_1$ have a unary domain, so that further simplifications are possible. Given any concept D in DLP normal form, let $\phi(D)$ be the concept that is obtained by exhaustively applying the following rules:

- If D has a subconcept $\geqslant 1\, R.E$, replace this subconcept by $E \sqcap \exists R.\mathsf{Self}$.
- If D has a subconcept $\geqslant m\, R.E$ with $m > 1$, replace this subconcept by $\bot$.
- If D has a subconcept $\leqslant m\, R.\neg E$ with $m > 1$, replace this subconcept by $\top$.

It is easy to check that $D \in \mathbf{C}_B^p$ implies $\mathsf{DLPNF}(\phi(D)) \in \mathbf{D}_B$, and that $D \in \mathbf{C}_H^p$ implies $\mathsf{DLPNF}(\phi(D)) \in \mathbf{D}_H$. Clearly, $\{c_1\} \sqcap C_1$ is semantically equivalent to $\{c_1\} \sqcap \phi(C_1)$, which is in turn equivalent to the knowledge base $\{\top \sqsubseteq \{c_1\}, \{c_1\} \sqsubseteq \phi(C_1)\}$. Thus, by Proposition 5, C is semantically emulated by $\mathsf{datalog}(C) := \{x \approx c_1\} \cup \mathsf{dlg}_a(\{c_1\} \sqsubseteq \mathsf{DLPNF}(\phi(C_1)))$ as long as $\mathsf{DLPNF}(\phi(C_1)) \notin \{\top, \bot\}$. If $\mathsf{DLPNF}(\phi(C_1)) = \top$ set $\mathsf{datalog}(C) := \{\}$. If $\mathsf{DLPNF}(\phi(C_1)) = \bot$ set $\mathsf{datalog}(C) := \{\top \to \bot\}$ (the unsatisfiable rule with empty body and head). □

To obtain the main result of this section, it remains to show that RBox and ABox axioms in $\mathcal{DLP}$ can also be emulated in datalog.

Theorem 3. *For every $\mathcal{DLP}$ axiom α as in Definition 9, one can construct a datalog program $\mathsf{datalog}(\alpha)$ that semantically emulates α.*

Proof. If α is a TBox axiom of the form $C \sqsubseteq D$, then set $E := \mathsf{DLPNF}(\neg C \sqcup D)$. If $E = \top$ then $\mathsf{datalog}(\alpha) := \{\}$. If $E = \bot$ of $E \in \mathbf{C}_{\neq\top}$ then $\mathsf{datalog}(\alpha) := \{\top \to \bot\}$ (the unsatisfiable rule with empty body and head). It is easy to see, that concepts of the form $\mathbf{C}_{\neq\top}$ are indeed unsatisfiable when used as axioms. If $E \in \mathbf{D}^{=n}$ for some $n \geq 1$

then set $\mathsf{datalog}(\alpha) \coloneqq \mathsf{datalog}(E)$ as defined in Lemma 13. Finally, if $E \in \mathbf{D}_H$ then set $\mathsf{datalog}(\alpha) \coloneqq \mathsf{dlg}_H(A \sqsubseteq E) \cup \{A(x)\}$ as defined in Proposition 5, where A is a fresh concept name.

If α is an ABox axiom of the form $C(a)$ with $\mathsf{DLPNF}(C) \in \mathbf{D}_a$ then set $\mathsf{datalog}(\alpha) \coloneqq \mathsf{dlg}_a(\{a\} \sqsubseteq \mathsf{DLPNF}(C))$ as given in Proposition 5.

If α is an RBox axiom then $\mathsf{dlg}_R(\alpha)$ is obtained as the union of P_{Inv} and the rules given in Fig. 10. Set $\mathsf{datalog}(\alpha) \coloneqq \mathsf{dlg}_R(\alpha)$. It is easy to see that this datalog program satisfies the claim. □

7 Model Constructions for Datalog

In this section, we introduce constructions on first-order logic interpretations which will be essential for showing that certain formulae cannot be in DLP. The general approach is to find operations that preserve models for datalog programs, i.e. operations under which the set of models of any datalog program must be closed. A well-known model construction in logic programming is the intersection of two Herbrand models, and it is well-known that Horn logic is closed under such intersections. The next definition generalises intersections in two ways: on the one hand, it uses functions to allow for interpretations with different (non-Herbrand) domains; on the other hand, it allows us to construct additional domain elements as feature combinations of existing elements.

Definition 11. *Consider a first-order logic signature $\mathscr{S}$ and two interpretations $\mathcal{I}_1$ and $\mathcal{I}_2$ over that signature. Consider a set Δ and functions $\mu : \Delta \to \Delta^{\mathcal{I}_1}$ and $\nu : \Delta \to \Delta^{\mathcal{I}_2}$ such that, for each constant c in $\mathscr{S}$, there is exactly one element $\delta_c \in \Delta$ for which $\mu(\delta_c) = c^{\mathcal{I}_1}$ and $\nu(\delta_c) = c^{\mathcal{I}_2}$. The* product interpretation *$\mathcal{J} = \mathcal{I}_1 \times_{\mu,\nu} \mathcal{I}_2$ is defined as follows:*

- *$\Delta^{\mathcal{J}} \coloneqq \Delta$,*
- *for each constant c in $\mathscr{S}$, set $c^{\mathcal{J}} \coloneqq \delta_c$,*
- *for each n-ary predicate symbol p and n-tuple $\overline{\delta} \in \Delta^n$, set $\overline{\delta} \in p^{\mathcal{J}}$ iff $\mu(\overline{\delta}) \in p^{\mathcal{I}_1}$ and $\nu(\overline{\delta}) \in p^{\mathcal{I}_2}$, where $\mu(\overline{\delta})$ and $\nu(\overline{\delta})$ denote the tuples obtained by applying μ and ν to each component of $\overline{\delta}$.*

The previous definition does not imply that constants have distinct interpretations: $\delta_c = \delta_d$ if and only if $c^{\mathcal{I}_1} = d^{\mathcal{I}_1}$ and $c^{\mathcal{I}_2} = d^{\mathcal{I}_2}$. As the definition of equality in product models is similar to the definition of predicate extensions, it is convenient to formulate Definition 11 for first-order logic without equality, assuming that $\approx$ is introduced by the well-known axiomatisation of its properties. A direct definition for $\mathbf{FOL}_=$ is straightforward.

The essential property of product interpretations is the following:

Proposition 6. *Consider a signature $\mathscr{S}$, interpretations $\mathcal{I}_1$ and $\mathcal{I}_2$, and functions $\mu : \Delta \to \Delta^{\mathcal{I}_1}$ and $\nu : \Delta \to \Delta^{\mathcal{I}_2}$ as in Definition 11. Then, for every datalog program P over $\mathscr{S}$, we find that $\mathcal{I}_1 \models P$ and $\mathcal{I}_2 \models P$ implies $\mathcal{I}_1 \times_{\mu,\nu} \mathcal{I}_2 \models P$.*

Proof. Let $\mathcal{J} := \mathcal{I}_1 \times_{\mu,\nu} \mathcal{I}_2$. Consider any rule $B \to H$ in P, and a variable assignment $\mathcal{Z}$ for $\mathcal{J}$ such that $\mathcal{J}, \mathcal{Z} \models B$. Define a variable assignment $\mathcal{Z}_1$ for $\mathcal{I}_1$ by setting $\mathcal{Z}_1(x) := \mu(\mathcal{Z}(x))$. By Definition 11, it is easy to see that $\mathcal{I}_1, \mathcal{Z}_1 \models B$, and thus $\mathcal{I}_1, \mathcal{Z}_1 \models H$. Analogously, we construct a variable assignment $\mathcal{Z}_2$ such that $\mathcal{I}_2, \mathcal{Z}_2 \models B$ and $\mathcal{I}_2, \mathcal{Z}_2 \models H$. It is easy to see that this implies $\mathcal{J}, \mathcal{Z} \models H$ as required. □

A well-known special case of the above product construction is obtained for $\Delta = \Delta^{\mathcal{I}_1} \times \Delta^{\mathcal{I}_2}$ with μ and ν being the projections to the first and second component of each pair in Δ. It turns out that this canonical product construction is not sufficient to detect all cases of knowledge bases that cannot be $\mathbf{FOL}_{=}$-emulated in datalog. For example, the set of models of the non-DLP axiom $\{a\} \sqsubseteq \geqslant 2\, R.(\neg\{b\} \sqcup \leqslant 1\, S.\neg A)$ is closed under canonical products. The more general construction above is needed to address such cases.

When using Proposition 6 to show that a knowledge base cannot be $\mathbf{FOL}_{=}$-emulated in datalog, it must be taken into account that $\mathbf{FOL}_{=}$-emulation is not as strong as semantic equivalence. It is not sufficient to show that the models of a knowledge base are not closed under products. For example, the DLP axiom $\{a\} \sqsubseteq \geqslant 1\, R.\top$ has a model $\mathcal{I}$ with domain $\Delta^{\mathcal{I}} := \{a, x\}$, $a^{\mathcal{I}} := a$, and $R^{\mathcal{I}} := \langle a, x \rangle$. Yet, the function $\mu : \{a\} \to \{a, x\}$ with $\mu(a) = a$ can be used to construct an interpretation $\mathcal{I} \times_{\mu,\mu} \mathcal{I}$ that is not a model of the axiom. Note that all preconditions of Definition 11 are satisfied. Proposition 6 allows us to conclude that there is no datalog program that is semantically equivalent to $\{a\} \sqsubseteq \geqslant 1\, R.\top$, but not that there is no such program $\mathbf{FOL}_{=}$-*emulating* the axiom. To show that a knowledge base cannot even be emulated in datalog, we therefore use the following observation.

Lemma 14. *Consider a knowledge base* KB *over some signature $\mathscr{S}$. If there are* $\mathbf{FOL}_{=}$ *theories T_1 and T_2 over $\mathscr{S}$ such that:*

- KB $\cup\, T_1$ *and* KB $\cup\, T_2$ *are satisfiable, and*
- *for every pair of models $\mathcal{I}_1 \models$* KB $\cup\, T_1$ *and $\mathcal{I}_2 \models$* KB $\cup\, T_2$*, possibly based on an extended signature $\mathscr{S}'$, there are functions μ and ν such that $\mathcal{I}_1 \times_{\mu,\nu} \mathcal{I}_2 \not\models$* KB,

then KB *cannot be* $\mathbf{FOL}_{=}$*-emulated in datalog.*

If $T_1 = T_2$ then this conclusion can also be obtained if the precondition only holds for pairs of equal models $\mathcal{I}_1 = \mathcal{I}_2$.

Proof. For a contradiction, suppose that the preconditions of the lemma hold and there is a datalog program P that $\mathbf{FOL}_{=}$-emulates KB. Then $P \cup \text{KB} \cup T_i$ is satisfied by some model $\mathcal{I}_i$ of P for each $i = 1, 2$, where the relevant signature of P may be larger than the signature of KB. Let $\mathcal{J} = \mathcal{I}_1 \times_{\mu,\nu} \mathcal{I}_2$ denote the product interpretation from the second condition. Applying Proposition 6, we find that $\mathcal{J}$ is a model of P that is not a model of KB. But then the union of P with a $\mathbf{FOL}_{=}$ formula of $\mathscr{S}$ that is semantically equivalent to the negation of the conjunction of all axioms in KB is satisfiable, contradicting the supposed emulation. The last part of the claim is obvious. □

The optional extension of the signature in the previous lemma can be important since the preconditions of Definition 11 require that the domain of the constructed model contains elements for all constant symbols.

As a simple example for this approach, we show that $\mathrm{KB} = \{\top \sqsubseteq A \sqcup B\}$ cannot be $\mathbf{FOL}_=$-emulated in datalog. Define auxiliary knowledge bases $\mathrm{KB}_1 = \{A \sqsubseteq \bot\}$ and $\mathrm{KB}_2 = \{B \sqsubseteq \bot\}$. Clearly, $\mathrm{KB} \cup \mathrm{KB}_1$ and $\mathrm{KB} \cup \mathrm{KB}_2$ are satisfied by some models $\mathcal{I}_1$ and $\mathcal{I}_2$, respectively. However, it is easy to see that no product of $\mathcal{I}_1$ and $\mathcal{I}_2$ can be a model of KB – independent of the choice of μ and ν – since the extensions of A and B must always be empty in such a product.

Of course there are other examples for which μ and ν must be chosen more carefully. In particular, it is sometimes necessary to restrict the amount of new elements that are introduced by the product. The following definition provides a useful notation for such a restricted form of products that will be sufficient for most applications:

Definition 12. *Consider interpretations $\mathcal{I}_1$ and $\mathcal{I}_2$ over a signature $\mathscr{S}$, and let $\mathbf{I}$ be the set of constants in $\mathscr{S}$. Given a set $S \subseteq \mathbf{I} \times \mathbf{I}$, functions $\mu : \Delta \to \Delta^{\mathcal{I}_1}$ and $\nu : \Delta \to \Delta^{\mathcal{I}_2}$ are defined as follows:*

- $\Delta \coloneqq S \cup \{\langle c, c\rangle \mid c \in \mathbf{I}\}$,
- $\mu(\langle c, d\rangle) \coloneqq c^{\mathcal{I}_1}$,
- $\nu(\langle c, d\rangle) \coloneqq d^{\mathcal{I}_2}$.

$\mathcal{I}_1 \times_S \mathcal{I}_2$ denotes the product interpretation $\mathcal{I}_1 \times_{\mu,\nu} \mathcal{I}_2$ for these functions.

A special aspect of the previous definition is that it restricts attention to named elements – elements that are represented by some individual name – in the original models. It is an easy corollary of Proposition 6 that all other elements are indeed irrelevant for satisfying a datalog program.

8 Showing Structural Maximality of $\mathcal{DLP}$

In this section, we show that the earlier definition of $\mathcal{DLP}$ is indeed maximal for the underlying principles. The proof mainly uses the principle of structurality (DLP 6) due to which it suffices to show that structural concept expressions that are not in $\mathcal{DLP}$ cannot be $\mathbf{FOL}_=$-emulated in datalog. To this end, we generally use the strategy suggested by Lemma 14. The below discussions often use datalog rules or DL axioms in the context of first-order logic to conveniently denote an arbitrary $\mathbf{FOL}_=$ theory of the same semantics, as obtained by any of the standard translations. Especially, this abbreviated form never refers to the more complex datalog transformation of $\mathcal{DLP}$ concepts, and it is only used when syntactic details are not relevant. Moreover, we assume that $\approx$ always denotes the equality predicate, and do not explicitly provide an axiomatisation for it.

The outline of the proof is as follows. We start by specifying some useful kinds of auxiliary datalog programs in Definition 13 and 14. The first major class of concept expressions is excluded by Proposition 7 which shows that concepts that are not in $\mathbf{D}_a^+$ can usually not be emulated in datalog. This result is prepared by Lemma 15, Lemma 16, and Lemma 17. These lemmata also are of some utility later on, since they can be used to exclude most forms of existential statements from DLP.

The second main ingredient of the maximality proof is Corollary 1. It extends Proposition 7 by establishing that concepts can typically not be emulated in datalog if they are not in $\mathbf{D}_H$. The chief insight that leads to this result is formulated in Lemma 18 which sports the most complex proof of this section. After this, it is comparatively easy to establish Lemma 19 to treat some pathological cases that had been excluded from the earlier considerations. In particular, it includes the “propositional” case where a DL concept enforces a unary interpretation domain.

The outcomes of Proposition 7, Corollary 1, and Lemma 19 are finally summarised in the main Theorem 4.

To pursue the proof strategy outlined by Lemma 14, our main work consists in specifying suitable auxiliary theories T_1 and T_2. To simplify this task, we first define some auxiliary theories that will be used frequently. Many of these constructions have the additional advantage of being in datalog – with the important consequence that they are still satisfied by product interpretations (Proposition 6). Often this is relevant for showing that said product interpretations cannot satisfy a given non-DLP concept.

Whereas many concept expressions C cannot be $\mathbf{FOL}_=$-emulated in datalog, it is usually possible to specify a datalog program that entails $\{c\} \sqsubseteq C$ for a given constant c by specifying sufficient properties that c must satisfy for this to be true. This only fails if C is structurally unsatisfiable. The below construction generalises this idea to any number of constants, and to the dual case where $\{c\} \sqsubseteq \neg C$ is entailed. The constructions in Definition 13 and 14 should be compared to the simpler cases discussed in Definition 5 which serve essentially the same purpose for $\mathcal{ALC}$.

Definition 13. *Consider a structural concept C in positive normal form, and individual names $c_0, \ldots, c_n$ for $n \geq 0$. If $C \notin \mathbf{L}_{\leq n}$ the datalog program $\llbracket c_0, \ldots, c_n \in C \rrbracket$ is defined recursively as follows:*

- *If $C = \top$ or $C = \geqslant 0\, R.D$ then $\llbracket c_0, \ldots, c_n \in C \rrbracket := \emptyset$.*
- *If $C = \{d\}$ then $n = 0$ and $\llbracket c_0 \in C \rrbracket := \{c_0 \approx d\}$.*
- *If C is of the form $\mathbf{A}$, $\neg\mathbf{A}$, $\neg\{\mathbf{I}\}$, $\exists\mathbf{R}.\mathsf{Self}$, or $\neg\exists\mathbf{R}.\mathsf{Self}$, then $\llbracket c_0, \ldots, c_n \in C \rrbracket := \bigcup_{0 \leq i \leq n} \mathsf{datalog}(\{c_i\} \sqsubseteq C)$.*
- *$C = D_1 \sqcap D_2$, then $D_i \notin \mathbf{L}_{\leq n}$ for $i = 1, 2$, and $\llbracket c_0, \ldots, c_n \in C \rrbracket := \llbracket c_0, \ldots, c_n \in D_1 \rrbracket \cup \llbracket c_0, \ldots, c_n \in D_2 \rrbracket$.*
- *If $C = D_1 \sqcup D_2$ with $D_1 \notin \mathbf{L}_{\leq n}$, then $\llbracket c_0, \ldots, c_n \in C \rrbracket := \llbracket c_0, \ldots, c_n \in D_1 \rrbracket$.*
- *If $C = D_1 \sqcup D_2$ with $D_1 \in \overline{\mathbf{L}}_{\leq m'}$ and $D_1 \in \overline{\mathbf{L}}_{\leq m''}$ such that $m' + m'' = n - 1$, then $\llbracket c_0, \ldots, c_n \in C \rrbracket := \llbracket c_0, \ldots, c_{m'} \in D_1 \rrbracket \cup \llbracket c_{m'+1}, \ldots, c_{m'+m''+1} \in D_2 \rrbracket$.*
- *If $C = \geqslant m\, R.D$ with $m \geq 1$, consider fresh constants $d_0, \ldots, d_m$, and set $\llbracket c_0, \ldots, c_n \in C \rrbracket := \llbracket d_0, \ldots, d_m \in D \rrbracket \cup \{R(c_i, d_j) \mid 0 \leq i \leq n,\ 0 \leq j \leq m\} \cup \{d_i \approx d_j \rightarrow \bot \mid 0 \leq i < j \leq m\}$.*
- *If $C = \leqslant m\, R.\neg D$, then $\llbracket c_0, \ldots, c_n \in C \rrbracket := \{x \approx c_i \wedge R(x, y) \rightarrow \bot \mid 1 \leq i \leq n\}$.*

If $C \notin \mathbf{L}_{\geq \omega - n}$, define a datalog program $\llbracket c_0, \ldots, c_n \notin C \rrbracket := \llbracket c_0, \ldots, c_n \in \mathsf{pNNF}(\neg C) \rrbracket$.

Note that the given cases directly follow the definition of $\overline{\mathbf{L}}_{\leq n}$ in Fig. 4. Also note that $\llbracket c_0, \ldots, c_n \in C \rrbracket$ and $\llbracket c_0, \ldots, c_n \notin C \rrbracket$ are satisfiable, even if we additionally require that all constants c_i are mutually unequal (which is not implied by the datalog programs).

Definition 13 can be viewed as a way to entail statements of the form $\{c_0\} \sqcup \ldots \sqcup \{c_n\} \sqsubseteq C$ if $C \notin \mathbf{L}_{\leq n}$. For cases where C is not in $\mathbf{L}_{\leq n}$ for any $n \geq 0$ this approach can be generalised to entail statements of the form $D \sqsubseteq C$ for a more general class of concepts D. The necessary construction is provided by the following definition which is very similar to Definition 13. We provide an alternative perspective and specify the dual case – entailing $C \sqsubseteq D$ in cases where $C \notin \mathbf{L}_{\geq\omega-n}$ for all $m \geq 0$ – which is the only case that is needed in our subsequent arguments.

Definition 14. *Consider a structural concept C in positive normal form, and a concept $D \in \mathbf{D}_H$.*

If $C \notin \mathbf{L}_{\geq\omega-m}$ for any $m \geq 0$, the datalog program $[\![C \sqsubseteq D]\!]_{\leq}$ is defined recursively as follows:

- *If $C = \bot$ then $[\![C \sqsubseteq D]\!]_{\leq} := \emptyset$.*
- *If C is of the form $\mathbf{A}$, $\{\mathbf{I}\}$, or $\exists\mathbf{R}.\mathsf{Self}$, then $[\![C \sqsubseteq D]\!]_{\leq} := \mathsf{datalog}(C \sqsubseteq D)$.*
- *If C is of the form $\neg\mathbf{A}$, or $\neg\exists\mathbf{R}.\mathsf{Self}$, then $[\![C \sqsubseteq D]\!]_{\leq} := \mathsf{datalog}(C \sqsubseteq \bot)$.*
- *If $C = D_1 \sqcup D_2$, then $D_i \notin \mathbf{L}_{\geq\omega-m}$ for any $m \geq 0$ ($i = 1, 2$), and $[\![C \sqsubseteq D]\!]_{\leq} := [\![D_1 \sqsubseteq D]\!]_{\leq} \cup [\![D_2 \sqsubseteq D]\!]_{\leq}$.*
- *If $C = D_1 \sqcap D_2$ with $D_1 \notin \mathbf{L}_{\geq\omega-m}$ for any $m \geq 0$, then $[\![C \sqsubseteq D]\!]_{\leq} := [\![D_1 \sqsubseteq D]\!]_{\leq}$.*
- *If $C = \leqslant m\,R.\neg E$, then consider fresh constants $d_0, \ldots, d_m$, and define $[\![C \sqsubseteq D]\!]_{\leq} := [\![d_0, \ldots, d_m \notin E]\!] \cup \{d_i \approx d_j \to \bot \mid 0 \leq i < j \leq m\} \cup \bigcup_{0\leq i\leq m} \mathsf{datalog}(\geqslant 1\,R.\{d_i\})$.*
- *If $C = \geqslant m\,R.E$, then $[\![C \sqsubseteq D]\!]_{\leq} := \mathsf{datalog}(\geqslant 1\,R.\top \sqsubseteq D)$.*

It should be noted that the cases of the definition are indeed exhaustive. Also observe that $[\![C \sqsubseteq D]\!]_{\leq}$ is always satisfiable, where $D \neq \bot$ is important to ensure that this is actually true for cases like $[\![\{c\} \sqsubseteq D]\!]_{\leq}$. This also shows that $[\![C \sqsubseteq A]\!]_{\leq} \cup \{A \sqsubseteq \bot\}$ cannot be assumed to be satisfiable in general.

Some further observations should be made in order to understand how Definitions 13 and 14 can be used when discussing datalog emulation. The constructions in both cases do certainly not $\mathbf{FOL}_{=}$-emulate the statement that they entail. For example, $[\![c \in C]\!]$ enforces one particular case for which $\{c\} \sqsubseteq C$; it does in general not describe all such cases. Moreover, the program $[\![C \sqsubseteq D]\!]_{\leq}$ may enforce a much stronger condition such as $C \sqsubseteq \bot$ as in the case of $C = \leqslant m\,R.\neg E$. This illustrates that the extension of C can be constrained by $[\![C \sqsubseteq D]\!]_{\leq}$. Conversely, a knowledge base $[\![A \sqcup B \sqsubseteq D]\!]_{\leq}$ might entail the stronger statement $A \sqsubseteq D$.

Luckily, as long as structurality is assumed, the knowledge bases of Definition 13 and 14 hardly semantically interact with concept expressions other than those that they are constructed from. Yet, it must be noted that $[\![c_0, \ldots, c_n \in C]\!]$ may introduce mutually unequal individuals d_i for the case $C = \geqslant m\,R.D$, and that two distinct individuals are already required if $C = \neg\{d\}$. This effect can occur for all of the above constructions. Logical theories in $\mathbf{FOL}_{=}$ can restrict the maximum size of the domain, and the same is accomplished by DL axioms that correspond to concept expressions in $\mathbf{L}_{\leq m}$ for some $m \geq 0$. We need to exclude this possibility when using the above definitions.

The previous discussion shows that it is important to carefully check all uses of Definitions 13 and 14 to avoid undesired semantic ramifications. A useful intuition is that the constructed theories enforce a simplification upon C that allows us to disregard the concept's internal structure. As an example of a typical usage of these constructions, consider the axiom $\alpha = \{a\} \sqsubseteq C_1 \sqcup C_2$ with $C_2 \notin \mathbf{L}_{\top}$. Then $\alpha \cup [\![a \notin C_2]\!]$ implies

$\{\{a\} \sqsubseteq C_1\}$.[5] So $[\![a \notin C_2]\!]$ allowed us to dismiss an "uninteresting" C_2 to focus on the impact of C_1.

The following lemmata use the product construction to create elements that are not in a given concept's extension, where we usually use the abbreviated product construction of Definition 12. In the weakest case, elements outside the extension must be provided to achieve this (Lemma 15). With stronger side conditions, some or even all of the elements can be part of the concept extension (Lemma 16 and 17). The lemmata are essential ingredients for showing that subconcepts that are not in $\mathbf{D}_a^+$ cannot occur in any DLP concept that is in normal form, and the assumptions of the lemma are therefore motivated by the definition of $\mathbf{D}_a^+$.

Lemma 15. *Consider a structural concept C in DLP normal form such that $C \neq \bot$ and $C \notin \mathbf{D}_{\geq\omega-n}$ for all $n \geq 0$ (in particular $C \neq \top$). Let $c_0, \ldots, c_n$ be fresh constants. There is a consistent datalog program $[\![c_0, \ldots, c_n \notin C]\!]_\times$ such that*

- *$[\![c_0, \ldots, c_n \notin C]\!]_\times \models \neg(c_i \approx c_j)$ for all $i, j \in \{0, \ldots, n\}$ with $i \neq j$,*
- *$[\![c_0, \ldots, c_n \notin C]\!]_\times \models \{c_i\} \sqsubseteq \neg C$ for all $i = 0, \ldots, n$,*
- *for all models $\mathcal{I}_1$, $\mathcal{I}_2$ of $[\![c_0, \ldots, c_n \notin C]\!]_\times$, and any set of constants $N \subseteq \mathbf{I}$ with $\{c_0, \ldots, c_n\} \subseteq N$, the product $\mathcal{J} = \mathcal{I}_1 \times_{(N\times N)} \mathcal{I}_2$ is such that $\langle c_i, c_j\rangle \notin C^{\mathcal{J}}$ for all $i, j \in \{0, \ldots, n\}$.*

Proof. Using a fresh concept name A, we define $[\![c_0, \ldots, c_n \notin C]\!]_\times := [\![C \sqsubseteq \neg A]\!]_\leq \cup \{A(c_i) \mid 0 \leq i \leq n\} \cup \{c_i \approx c_j \to \bot \mid 0 \leq i < j \leq n\}$. Given models $\mathcal{I}_1$ and $\mathcal{I}_2$ of $[\![c_0, \ldots, c_n \notin C]\!]_\times$, and $\mathcal{J} = \mathcal{I}_1 \times_{(N\times N)} \mathcal{I}_2$, we find that $\langle c_i, c_j\rangle \in A^{\mathcal{J}}$ for all $i, j \in \{0, \ldots, n\}$. Since $[\![c_0, \ldots, c_n \notin C]\!]_\times$ is in datalog, it is satisfied by $\mathcal{J}$, and thus we conclude $\langle c_i, c_j\rangle \notin C^{\mathcal{J}}$ for all $i, j \in \{0, \ldots, n\}$ as required. □

The next lemma considers concepts $C \notin \mathbf{D}_B^+$. The lemma is also stated for sets of individuals, and additional care is now needed to ensure that it is possible for C to have a set of (distinct) instances. It is not enough to assume $C \notin \mathbf{D}_{\leq n}$ for some or all $n \geq 0$ since this pre-condition cannot be preserved by all recursive constructions. Namely, the recursion in the case $C = D_1 \sqcup D_2$ must be based on the one subconcept D_i for which we have $D_i \notin \mathbf{D}_B^+$, but there is no reason for $D_i \notin \mathbf{D}_{\leq n}$ to hold for any $n \geq 1$ (only $n = 0$ is excluded since C is in DLP normal form). This explains why the lemma considers multiple individuals $c_0, \ldots, c_n$ only in cases where this problem can be avoided.

Lemma 16. *Consider a structural concept C in DLP normal form such that $C \notin \mathbf{D}_B^+$, and C does not have a subconcept $D \notin \mathbf{D}_a^+$. Let $n \geq 0$ be such that $n = 0$ if C is a disjunction or $C \in \mathbf{D}_{\leq k}$ for some $k \geq 0$, and consider fresh constants $c_0, \ldots, c_n, d_0, \ldots, d_m$. There is a consistent datalog program $[\![c_0, \ldots, c_n \in C, d_0, \ldots, d_m \notin C]\!]_\times$ and according set $M := \{c_0, \ldots, c_n, d_0, \ldots, d_m\} \cup \{c \in \mathbf{I} \mid c \text{ occurs in } [\![c_0, \ldots, c_n \in C, d_0, \ldots, d_m \notin C]\!]_\times\}$ such that*

- *$[\![c_0, \ldots, c_n \in C, d_0, \ldots, d_m \notin C]\!]_\times \models \neg(e \approx f)$ for all $e, f \in \{c_0, \ldots, c_n, d_0, \ldots, d_m\}$ with $e \neq f$,*

[5] This implication is not quite a $\mathbf{FOL}_=$-emulation since $[\![a \notin C_2]\!]$ can require a minimal domain cardinality, as discussed above.

- *$[\![c_0, \dots, c_n \in C, d_0, \dots, d_m \notin C]\!]_\times \models \{c_i\} \sqsubseteq C$ for all $i = 0, \dots, n$,*
- *$[\![c_0, \dots, c_n \in C, d_0, \dots, d_m \notin C]\!]_\times \models \{d_i\} \sqsubseteq \neg C$ for all $i = 0, \dots, m$,*
- *for all models $\mathcal{I}_1, \mathcal{I}_2$ of $[\![c_0, \dots, c_n \in C, d_0, \dots, d_m \notin C]\!]_\times$, and any set of constants $N \subseteq \mathbf{I}$ with $M \subseteq N$, the product $\mathcal{J} = \mathcal{I}_1 \times_{(N\times N)} \mathcal{I}_2$ is such that $\langle c_i, d_j \rangle \notin C^{\mathcal{J}}$ for all $i \in \{0, \dots, n\}$ and $j \in \{0, \dots, m\}$.*

Proof. Note that the conditions imply that $C \in \mathbf{D}_a^+$, and hence $C \notin \{\top, \bot\}$. Set $P \coloneqq \{e \approx f \to \bot \mid e, f \in \{c_0, \dots, c_n, d_0, \dots, d_m\}, e \neq f\}$. We define $[\![c_0, \dots, c_n \in C, d_0, \dots, d_m \notin C]\!]_\times$ recursively based on the structure of C, and we inductively show that it has the required properties. Both parts can conveniently be interleaved. Thus, in each of the following cases, let $\mathcal{I}_1$ and $\mathcal{I}_2$ be models of the $[\![c_0, \dots, c_n \in C, d_0, \dots, d_m \notin C]\!]_\times$ just defined, and let $\mathcal{J}$ be the product interpretation as in the claim:

- If C has the form $\mathbf{A}$, $\{\mathbf{I}\}$ or $\exists\mathbf{R}.\mathsf{Self}$, then $[\![c_0, \dots, c_n \in C, d_0, \dots, d_m \notin C]\!]_\times \coloneqq P \cup [\![c_0, \dots, c_n \in C]\!] \cup [\![d_0, \dots, d_m \notin C]\!]$.
  It is easy to see that $\mathcal{J}$ satisfies the claim. Note that the pre-conditions of the lemma imply $n = 0$ whenever $C \in \{\mathbf{I}\}$.
- If $C = D_1 \sqcap D_2$ with $D_1 \notin \mathbf{D}_B^+$, then $[\![c_0, \dots, c_n \in C, d_0, \dots, d_m \notin C]\!]_\times \coloneqq [\![c_0, \dots, c_n \in D_1, d_0, \dots, d_m \notin D_1]\!]_\times$.
  Since $\mathcal{I}_1$ and $\mathcal{I}_2$ are models of $[\![c_0, \dots, c_n \in D_1, d_0, \dots, d_m \notin D_1]\!]_\times$, the claim follows immediately by induction.
- If $C = D_1 \sqcup D_2$ with $D_1 \notin \mathbf{C}_B^+$ and $D_2 \notin \mathbf{D}_{\geq\omega-k}$ for all $k \geq 0$, then $n = 0$ is required. Define $[\![c_0 \in C, d_0, \dots, d_m \notin C]\!]_\times \coloneqq [\![c_0 \in D_1, d_0, \dots, d_m \notin D_1]\!]_\times \cup [\![D_2 \sqsubseteq \{c_0\}]\!]_\leq$.
  $\mathcal{I}_1$ and $\mathcal{I}_2$ are models of $[\![c_0, \dots, c_n \in D_1, d_0, \dots, d_m \notin D_1]\!]_\times$ and we can apply the induction hypothesis. The desired result follows since the product $\mathcal{J}$ also satisfies the datalog program $[\![D_2 \sqsubseteq \{c_0\}]\!]_\leq$.
- If $C = \geqslant k\, R.D$ with $k \geq 1$, then $[\![c_0, \dots, c_n \in C, d_0, \dots, d_m \notin C]\!]_\times \coloneqq P \cup \{R(c_i, e_j) \mid 0 \leq i \leq n, 1 \leq j \leq k\} \cup [\![e_1, \dots, e_k \in D]\!] \cup \{R(d_i, x) \to \bot \mid 0 \leq j \leq m\}$ for fresh individual names $e_1, \dots, e_k$.
  It is again easy to see that $\mathcal{J}$ satisfies the claim.
- If $C = \leqslant 0\, R.\neg D$ with $D \notin \mathbf{D}_B^+$, then, for a fresh constant e, define $[\![c_0, \dots, c_n \in C, d_0, \dots, d_m \notin C]\!]_\times \coloneqq P \cup \{R(c_i, x) \to x \approx e, R(c_i, e) \mid 0 \leq i \leq n\} \cup \{R(d_i, f) \mid 0 \leq i \leq m\} \cup [\![e \in D, f \notin D]\!]_\times$.
  We find that $\langle\langle c_i, d_j\rangle, \langle e, f\rangle\rangle \in R^{\mathcal{J}}$ for all $i \in \{0, \dots, n\}$ and $j \in \{0, \dots, m\}$. The claim follows from the induction hypothesis.
- If $C = \leqslant 1\, R.\neg D$ with $D \notin \mathbf{D}_{\geq\omega-k}$ for all $k \geq 0$, then consider fresh individuals e, f, g. Define $[\![c_0, \dots, c_n \in C, d_0, \dots, d_m \notin C]\!]_\times \coloneqq P \cup \{R(c_i, x) \to x \approx e, R(c_i, e) \mid 0 \leq i \leq n\} \cup \{R(d_i, f), R(d_i, g) \mid 0 \leq i \leq m\} \cup [\![e, f, g \notin D]\!]_\times$. Note that the last component of this union also requires that the individuals denoted by e, f, g are mutually distinct.
  We find that $\langle\langle c_i, d_j\rangle, \langle e, f\rangle\rangle \in R^{\mathcal{J}}$ and $\langle\langle c_i, d_j\rangle, \langle e, g\rangle\rangle \in R^{\mathcal{J}}$ for all $i \in \{0, \dots, n\}$ and $j \in \{0, \dots, m\}$. The claim follows from Lemma 15.

It should be verified that the given cases are exhaustive. In particular, $C = \leqslant 1\, R.\neg D$ with $D \notin \mathbf{D}_{\geq\omega-k}$ for all $k \geq 0$ is the only case where $C = \leqslant k\, R.\neg D$ for some $k \geq 1$ – all other forms are either in $\mathbf{D}_B^+$ or not in $\mathbf{D}_a^+$. Moreover, all recursive applications of the construction satisfy the necessary pre-conditions, especially the requirements for $n \geq 1$ are preserved. □

The third and final lemma in this series is only needed for two individuals so that we can simplify our presentation slightly. However, the construction now becomes more complex since we can no longer use an auxiliary datalog theory, and since more care is needed in selecting a suitable product interpretation.

Lemma 17. *Consider a structural concept C in DLP normal form such that $C \notin \mathbf{D}^+_H$, and C does not have a subconcept $D \notin \mathbf{D}^+_a$. Let c_0, c_1 be fresh constants. There is a consistent first-order theory $\llbracket c_0, c_1 \in C \rrbracket_\times$ and a set of constants $N \subseteq \mathbf{I}$ such that*

- *$\llbracket c_0, c_1 \in C \rrbracket_\times \models \neg(c_0 \approx c_1)$,*
- *$\llbracket c_0, c_1 \in C \rrbracket_\times \models \{c_i\} \sqsubseteq C$ for $i = 0, 1$,*
- *for all models $\mathcal{I}$ of $\llbracket c_0, c_1 \in C \rrbracket_\times$, the product $\mathcal{J} = \mathcal{I} \times_{(N \times N)} \mathcal{I}$ is such that $\langle c_0, c_1 \rangle \notin C^{\mathcal{J}}$.*

Proof. The conditions again imply that $C \in \mathbf{D}^+_a$, and hence $C \notin \{\top, \bot\}$. Moreover, $C \notin \mathbf{D}^+_H$ and $C \in \mathbf{D}^+_a$ implies that $C \notin \mathbf{D}_{\leq 1}$. Indeed, $C \notin \mathbf{D}_{\leq 0}$ since C is in DLP normal form, and thus $C \in \mathbf{D}_{\leq 1}$ would imply that C is of the form $\{\mathbf{I}\} \sqcap \mathbf{C}^+_a \subseteq \mathbf{D}^+_{1!} \subseteq \mathbf{D}^+_H$. This property is inherited by subconcepts D of C as long as $D \notin \mathbf{D}^+_H$.

We define $\llbracket c_0, c_1 \in C \rrbracket_\times$ recursively based on the structure of C, and we inductively show that it has the required properties. Both parts can conveniently be interleaved. In addition, we also specify a suitable set N of constant symbols to use in the product construction in the recursion. Thus, in each of the following cases, let $\mathcal{I}$ be a model of the $\llbracket c_0, \ldots, c_n \in C \rrbracket_\times$ just defined, and let $\mathcal{J}$ be the product interpretation as in the claim.

- If $C = D_1 \sqcup D_2$ with $D_1, D_2 \notin \mathbf{D}^+_B$ then $\llbracket c_0, c_1 \in C \rrbracket_\times \coloneqq \llbracket c_0 \in D_1, c_1 \notin D_1 \rrbracket_\times \cup \llbracket c_1 \in D_2, c_0 \notin D_2 \rrbracket_\times$ and the set N is defined as in Lemma 16.
  Using Lemma 16, it is easy to see that $\mathcal{J}$ satisfies the claim.
- If $C = D_1 \sqcup D_2$ with $D_1 \notin \mathbf{D}^+_H$ and $D_2 \in \mathbf{D}^+_B$ then consider a fresh concept name A. Since $C \notin \mathbf{D}_{\geq \omega - n}$ for all $n \geq 0$, the same holds for D_1 and D_2. Moreover, $D_1 \notin \mathbf{D}_{\leq 1}$ as discussed initially. We thus can define $\llbracket c_0, c_1 \in C \rrbracket_\times \coloneqq \llbracket c_0, c_1 \in D_1 \rrbracket_\times \cup \llbracket D_2 \sqsubseteq \neg A \rrbracket_\leq \cup \{A(c_0), A(c_1)\}$. The set N is defined to be the same as for $\llbracket c_0, c_1 \in D_1 \rrbracket_\times$.
  $\mathcal{I}$ is a model of $\llbracket c_0, c_1 \in D_1 \rrbracket_\times$ and we can apply the induction hypothesis. The desired result follows since the product $\mathcal{J}$ also satisfies the datalog program $\llbracket D_2 \sqsubseteq \neg A \rrbracket_\leq \cup \{A(c_0), A(c_1)\}$ (Proposition 6).
- If $C = D_1 \sqcap D_2$ then we can assume $D_1 \notin \mathbf{D}^+_H$. Clearly, $C \notin \mathbf{D}_{\leq 1}$ implies $D_1, D_2 \notin \mathbf{D}_{\leq 1}$. Thus we can set $\llbracket c_0, c_1 \in C \rrbracket_\times \coloneqq \llbracket c_0, c_1 \in D_1 \rrbracket_\times \cup \llbracket c_0, c_1 \in D_2 \rrbracket$, where N is again taken to be the set of constants as defined for $\llbracket c_0, c_1 \in D_1 \rrbracket_\times$.
  We can again apply the induction hypothesis since $\mathcal{I} \models \llbracket c_0, c_1 \in D_1 \rrbracket_\times$, and use the fact that $\mathcal{J} \models \llbracket c_0, c_1 \in D_2 \rrbracket$.
- If $C = \geqslant n\, R.D$ then $D \notin \mathbf{D}^+_{n!} \cup \mathbf{D}_{\leq n-1} \cup \{\bot\}$. Since all subconcepts of C are assumed to be in $\mathbf{D}^+_a$, we conclude that $D \notin \mathbf{D}_{\leq n}$. Thus we can introduce fresh individual symbols $d_0, \ldots, d_n$ and set $\llbracket c_0, c_1 \in C \rrbracket_\times \coloneqq \llbracket d_0, \ldots, d_n \in D \rrbracket \cup \{\neg(e \approx f) \mid e, f \in \{c_0, c_1, d_0, \ldots, d_n\}, e \neq f\} \cup \{\forall x. R(c_0, x) \leftrightarrow \bigvee_{0 \leq i < n} x \approx d_i\} \cup \{\forall x. R(c_1, x) \leftrightarrow \bigvee_{0 < i \leq n} x \approx d_i\}$. Define $N \coloneqq \{c_0, c_1\}$.
  We claim that $\langle c_0, c_1 \rangle \in \Delta^{\mathcal{J}}$ is such that $\langle c_0, c_1 \rangle \notin C^{\mathcal{J}}$. Consider any element $\langle e, f \rangle \in \Delta^{\mathcal{J}}$ such that $\langle \langle c_0, c_1 \rangle, \langle e, f \rangle \rangle \in R^{\mathcal{J}}$. By the construction of $\mathcal{J}$, we have that

$\langle c_0^{\mathcal{I}}, e^{\mathcal{I}}\rangle, \langle c_1^{\mathcal{I}}, f^{\mathcal{I}}\rangle \in R^{\mathcal{I}}$, and thus $e^{\mathcal{I}} = d_i^{\mathcal{I}}$ and $f^{\mathcal{I}} = d_j^{\mathcal{I}}$ for some $i \in \{0, \ldots, n-1\}$, $j \in \{1, \ldots, n\}$. Since the constants d_i are unequal to c_0, c_1, this implies that $e, f \notin N$, and thus $e = f = d_i = d_j$. Therefore, $\langle e, f\rangle$ is equal to $d_i^{\mathcal{J}}$ for some $i \in \{1, \ldots, n-1\}$ whenever $\langle\langle c_0, c_1\rangle, \langle e, f\rangle\rangle \in R^{\mathcal{J}}$, as required for $\langle c_0, c_1\rangle \notin C^{\mathcal{J}}$.

- If $C = \leqslant 0\, R.\neg D$ with $D \notin \mathbf{D}_H^+$ then define $[\![c_0, c_1 \in C]\!]_\times := [\![c_0, c_1 \in D]\!]_\times \cup \{R(c_0, c_0), R(c_1, c_1)\}$, where N is defined as for $[\![c_0, c_1 \in D]\!]_\times$.
  The claim follows by induction as before.
- If $C = \leqslant 1\, R.\neg D$ with $D \notin \mathbf{D}_B \cup \{\bot\}$ then $[\![c_1, c_0 \in C]\!]_\times := [\![c_0 \in D, c_1 \notin D]\!]_\times \cup \{R(c_0, c_0), R(c_0, c_1), R(c_1, c_0), R(c_1, c_1)\}$, where N is defined to be the set M as given in Lemma 16.
  The claim is a consequence of Lemma 16.
- If $C = \leqslant n\, R.\neg D$ with $n \geq 2$ then consider fresh individual symbols $c_2, \ldots, c_n$ and define $[\![c_0, c_1 \in C]\!]_\times := [\![c_0, c_1 \notin D]\!]_\times \cup [\![c_2, \ldots, c_n \notin D]\!] \cup \{R(c_i, c_j) \mid i \in \{0, 1\}, j \in \{0, \ldots, n\}, i \neq j\} \cup \{\neg(c_i \approx c_j) \mid 0 \leq i < j \leq n\}$, where N is defined to be the set M as given in Lemma 15.
  It is easy to see that $\langle c_0, c_1\rangle$ in $\mathcal{J}$ has at least n distinct R-successors $\langle c_i, c_i\rangle$ ($i = 2, \ldots, n$) and $\langle c_1, c_0\rangle$. The former are not in D since $\mathcal{J}$ satisfies the datalog program $[\![c_2, \ldots, c_n \notin D]\!]$. The latter are not in D by Lemma 15.

Atomic concepts, nominals, Self restrictions, and their negations do not occur since $C \notin \mathbf{D}_H^+$. □

The previous result is used in the following proposition to show that certain kinds of atmost-concepts are generally excluded from DLP, even if they occur as subconcepts only.

Proposition 7. *Given a structural concept $C \notin \{\top, \bot\}$ in DLP normal form, the following three statements are equivalent:*

- *$C \notin \mathbf{D}_a^+$,*
- *C has a subconcept $D \notin \mathbf{D}_a^+$,*
- *C contains a subconcept $\leqslant k\, S.\neg F$ such that $F \in \mathbf{D}_a^+$ and $F \notin \mathbf{D}_{\geq\omega-l}$ for all $l \geq 0$ and:*
  (a) $k = 0$ and $F \notin \mathbf{D}_H^+ \cup \{\bot\}$, or
  (b) $k = 1$ and $F \notin \mathbf{D}_B^+ \cup \{\bot\}$, or
  (c) $k \geq 2$.

If these statements hold and, in addition, $C \notin \mathbf{D}_{\leq n}$ for all $n \geq 0$, and $C \notin \mathbf{C}_{\neq\top}$, then C cannot be $\mathbf{FOL}_=$-emulated in datalog.

Proof. Note that the preconditions on C imply that $\{C\}$ is satisfiable. The claimed equivalence is easily verified by considering the grammar for $\mathbf{D}_a^+$ given in Fig. 5, where it should be noted that some cases are inherited from $\mathbf{D}_H^+$ and $\mathbf{D}_B^+$. Also observe that $F \in \mathbf{D}_a^+$ is thus equivalent to saying that F has no subconcept $E \notin \mathbf{D}_a^+$.

First, we define an auxiliary theory that requires $\leqslant k\, S.\neg F$ to be non-empty in order for C to be satisfied. As before, we sometimes mix first-order logic and DL to denote an arbitrary $\mathbf{FOL}_=$ theory that represents the first-order semantics of this combination. Given a constant symbol c, and a subconcept D of C such that $\leqslant k\, S.\neg F$ is a subconcept of D, we recursively construct a $\mathbf{FOL}_=$ theory $T(c, D)$:

- If $D = {\leqslant}k\,S.\neg F$, then $T(c, D) \coloneqq \emptyset$.
- If $D = D_1 \sqcap D_2$ with ${\leqslant}k\,S.\neg F$ a subconcept of D_1, then $T(c, D) \coloneqq T(c, D_1)$.
- If $D = D_1 \sqcup D_2$ with ${\leqslant}k\,S.\neg F$ a subconcept of D_1, then $T(c, D) \coloneqq T(c, D_1) \cup [\![c \notin D_2]\!]$.
- If $D = {\geqslant}n\,R.D'$, then consider fresh constants $c_1, \ldots, c_n$ and define $T(c, D) \coloneqq \{\forall x.R(c, x) \to \bigvee_{1 \leq i \leq n} c_i \approx x\} \cup T(c_0, D')$.
- If $D = {\leqslant}n\,R.\neg D'$ (with $R \neq S$), then consider fresh constants $c_0, \ldots, c_n$ and set $T(c, D) \coloneqq \{\bigwedge_{0 \leq i \leq n} R(c, c_i) \wedge \bigwedge_{0 \leq i < j \leq n} \neg(c_i \approx c_j)\} \cup [\![c_1, \ldots, c_n \notin D]\!] \cup T(c_0, D')$.

Note that $T(c, D)$ is satisfiable, due to structurality of C and the fact that the subconcept ${\leqslant}k\,S.\neg F$ cannot be part of a subconcept of the form $\mathbf{L}_\top$ or $\mathbf{L}_\bot$ since C is in DLP normal form. Now the theory T is defined as $T \coloneqq T(c, C)$ for some fresh constant c. It is easy to see that $T \cup \{C\}$ is satisfiable, and that $T \cup \{C\} \cup \{{\leqslant}k\,S.\neg F \sqsubseteq \bot\}$ is unsatisfiable.

Consider the case $k = 0$. Let a and b be fresh constants. We use the construction of Lemma 17 to ensure that every element in the respective product interpretations has an S-successor $\langle a, b\rangle$ in $\neg F$, and N denotes the according set of constant symbols as in the definition of $[\![a, b \in F]\!]_\times$. Some care is needed to ensure that the auxiliary theory T remains true in any such product interpretation. Thus define $T' \coloneqq T \cup \{\neg(c \approx d) \mid c \in N, d \text{ occurs in } T\} \cup \{\forall x.S(x, a) \wedge S(x, b)\} \cup [\![a, b \in F]\!]_\times$. It is not hard to see that $T' \cup \{C\}$ is satisfiable. For an arbitrary model $\mathcal{I}$ of $T' \cup \{C\}$, consider the product interpretation $\mathcal{J} \coloneqq \mathcal{I} \times_{(N \times N)} \mathcal{I}$. Since $\mathcal{J}$ satisfies $\forall x.S(x, a) \wedge S(x, b)$ (by Proposition 6), we find $\langle \delta, \langle a, b\rangle\rangle \in S^{\mathcal{J}}$ for all $\delta \in \Delta^{\mathcal{J}}$. Thus Lemma 17 entails $\mathcal{J} \models {\leqslant}0\,S.\neg F \sqsubseteq \bot$.

Moreover, $\mathcal{J}$ satisfies T. This is a consequence of Proposition 6 for all axioms of T that are in datalog. The only axioms for which this is not the case are of the form $\forall x.R(c, x) \to \bigvee_{1 \leq i \leq n} c_i \approx x$. Consider any element $\langle e, f\rangle \in \Delta^{\mathcal{J}}$ such that $\langle c^{\mathcal{J}}, \langle e, f\rangle\rangle \in R^{\mathcal{J}}$. By the construction of $\mathcal{J}$, we have that $\langle c^{\mathcal{I}}, e^{\mathcal{I}}\rangle, \langle c^{\mathcal{I}}, f^{\mathcal{I}}\rangle \in R^{\mathcal{I}}$, and thus $e^{\mathcal{I}} = c_i^{\mathcal{I}}$ and $f^{\mathcal{I}} = c_j^{\mathcal{I}}$ for some $i, j \in \{1, \ldots, n\}$. Since all constants in N must be unequal to constants c_i, this implies that $e, f \notin N$, and thus $e = f = c_i = c_j$. Therefore, $\langle e, f\rangle$ is equal to $c_i^{\mathcal{J}}$ for some $i \in \{1, \ldots, n\}$ whenever $\langle c^{\mathcal{J}}, \langle e, f\rangle\rangle \in R^{\mathcal{J}}$, so that the considered axiom of T is indeed satisfied.

Since $T \cup \{C\} \cup \{{\leqslant}k\,S.\neg F \sqsubseteq \bot\}$ is unsatisfiable, this implies $\mathcal{J} \not\models \{C\}$. This establishes the preconditions for Lemma 14 (for the case $T_1 = T_2$) and thus shows the claim.

The other cases $k = 1$ and $k \geq 2$ are very similar, using constructions $[\![a \in F, b \notin F]\!]_\times$ and $[\![c_1, \ldots, c_k \notin F]\!]_\times$ of Lemma 16 and 15. For $k = 1$, it is admissible that $a^{\mathcal{I}} \notin F^{\mathcal{I}}$ is an S-successor of all elements. For $k \geq 2$, k such S-successors $c_1^{\mathcal{I}}, \ldots, c_k^{\mathcal{I}} \notin F^{\mathcal{I}}$ are allowed. In either case, the product construction generates further S-successors that require ${\leqslant}k\,S.\neg F$ to be empty. □

Observe how the previous proof depends on using the second pre-condition of Lemma 14 where a single model is multiplied with itself. This is essential to ensure that the auxiliary theory T is satisfied in the product, even though it contains non-datalog axioms. The above result also marks a case where we really need product constructions that are different from the canonical product that uses all pairs of (named) individuals as the new interpretation domain. The auxiliary theory T in the above case would not generally be satisfied in a canonical product: the non-datalog axioms introduced for atleast-restrictions require a fixed set of successor individuals, whereas a canonical product contains additional successors that correspond to pairs of the original individuals.

For the remaining steps of the proof, we use some additional auxiliary constructions. The datalog programs of Definitions 13 and 14 are not suitable to isolate properties that exclude a concept from DLP: to the contrary, they simply enforce certain entailments to override any complex semantic effects. The following definition therefore provides us with knowledge bases that can be used to "measure" information about the extension of a concept C without enforcing $C \sqsubseteq \bot$. The underlying intuition is that non-emptiness of some concepts can be ensured to entail *positive* information. The construction thus can be viewed as a generalisation of the construction in Lemma 3 to the more complex case of $\mathcal{SROIQ}$.

We provide two cases: $\llbracket c \in C \rightsquigarrow A\rrbracket_B$ is used to detect whether a constant c is in C, while $\llbracket C \rightsquigarrow A\rrbracket_{B\leq}$ is used to detect if C is generally non-empty. Both constructions can only work (in $\mathcal{DLP}$) if C "contains" positive information, i.e. if it is not in $\mathbf{D}_B$. Note that the constructions can be considered as specialisations of $\llbracket a \notin C\rrbracket$ and $\llbracket C \sqsubseteq A\rrbracket_{\leq}$.

Definition 15. *Consider a structural concept C in DLP normal form such that $C \notin \mathbf{D}_B \cup \{\bot, \top\} \cup \mathbf{D}_{\geq\omega-k}$ for some $k \geq 0$. For individual names $c_0, \dots, c_k$ and concepts $A_0, \dots, A_k \in \mathbf{D}_H$, a datalog program $\llbracket c_0, \dots, c_k \in C \rightsquigarrow A_0, \dots, A_k\rrbracket_B$ is defined recursively as follows:*

- *If C has the form $\mathbf{A}$, $\{\mathbf{I}\}$ or $\exists\mathbf{R}.\mathsf{Self}$, then $\llbracket c_0, \dots, c_k \in C \rightsquigarrow A_0, \dots, A_k\rrbracket_B := \bigcup_{0\leq i\leq k} \mathsf{datalog}(\{c_i\} \sqcap C \sqsubseteq A_i)$.*
- *If $C = D_1 \sqcap D_2$ with $D_1 \notin \mathbf{D}_B$, then w.l.o.g. D_1 is not a conjunction and thus $D_1 \notin \mathbf{D}_{\geq\omega-m}$ for all $m \geq 0$. Define $\llbracket c_0, \dots, c_k \in C \rightsquigarrow A_0, \dots, A_k\rrbracket_B := \llbracket c_0, \dots, c_k \in D_1 \rightsquigarrow A_0, \dots, A_k\rrbracket_B$.*
- *If $C = D_1 \sqcup D_2$ with $D_1 \notin \mathbf{D}_B$, then $D_1, D_2 \notin \mathbf{D}_{\geq\omega-k}$. Set $\llbracket c_0, \dots, c_k \in C \rightsquigarrow A_0, \dots, A_k\rrbracket_B := \llbracket c_0, \dots, c_k \in D_1 \rightsquigarrow A_0, \dots, A_k\rrbracket_B \cup \llbracket c_0, \dots, c_k \notin D_2\rrbracket$.*
- *If $C = \geqslant n\, R.D$ with $n \geq 1$, then $\llbracket c_0, \dots, c_k \in C \rightsquigarrow A_0, \dots, A_k\rrbracket_B := \{R(c_i, x) \to A_i(c_i) \mid 0 \leq i \leq k\}$.*
- *If $C = \leqslant 0\, R.\neg D$, then, for a fresh constant d and fresh concept name B, define $\llbracket c_0, \dots, c_k \in C \rightsquigarrow A_0, \dots, A_k\rrbracket_B := \llbracket d \in D \rightsquigarrow B\rrbracket_B \cup \{R(c_i, d), B(d) \to A_i(c_i) \mid 0 \leq i \leq k\}$.*
- *If $C = \leqslant n\, R.\neg D$ with $n \geq 1$, then consider fresh constants d_i ($i = 0, \dots, n$). Define $\llbracket c_0, \dots, c_k \in C \rightsquigarrow A_0, \dots, A_k\rrbracket_B := \{R(c_i, d_j) \mid 0 \leq i \leq k, 0 \leq j \leq n\} \cup \{d_j \approx d_l \to A_i(c_i) \mid 0 \leq j < l \leq n, 0 \leq i \leq k\} \cup \llbracket d_0, \dots, d_n \notin D\rrbracket$.*

Moreover, if $C \notin \mathbf{D}_{\geq\omega-k}$ for all $k \geq 0$, then a datalog program $\llbracket C \rightsquigarrow A\rrbracket_{B\leq}$ is defined recursively as follows:

- *If C has the form $\mathbf{A}$, $\{\mathbf{I}\}$ or $\exists\mathbf{R}.\mathsf{Self}$, then $\llbracket C \rightsquigarrow A\rrbracket_{B\leq} := \mathsf{datalog}(C \sqsubseteq A)$.*
- *If $C = D_1 \sqcap D_2$ with $D_1 \notin \mathbf{D}_B$ and $D_1 \notin \mathbf{D}_{\geq\omega-n}$ for all $n \geq 0$, then $\llbracket C \rightsquigarrow A\rrbracket_{B\leq} := \llbracket D_1 \rightsquigarrow A\rrbracket_{B\leq}$.*
- *If $C = D_1 \sqcup D_2$ with $D_1 \notin \mathbf{D}_B$, then $\llbracket C \rightsquigarrow A\rrbracket_{B\leq} := \llbracket D_1 \rightsquigarrow A\rrbracket_{B\leq} \cup \llbracket D_2 \sqsubseteq A\rrbracket_{\leq}$.*
- *If $C = \geqslant n\, R.D$ with $n \geq 1$, then $\llbracket C \rightsquigarrow A\rrbracket_{B\leq} := \{R(x, y) \to A(x)\}$.*
- *If $C = \leqslant 0\, R.\neg D$, then, for a fresh constant c and fresh concept name B, define $\llbracket C \rightsquigarrow A\rrbracket_{B\leq} := \llbracket c \in D \rightsquigarrow B\rrbracket_B \cup \{R(x, c), B(c) \to A(x)\}$.*

- *If* $C = \leqslant n\,R.\neg D$ *with* $n \geq 1$, *then consider fresh constants* c_i *(*$i = 0, \ldots, n$*). Define* $[\![C \rightsquigarrow A]\!]_{B\leq} \coloneqq \{R(x, c_i) \mid 0 \leq i \leq n\} \cup \{c_i \approx c_j \rightarrow A(x) \mid 0 \leq i < j \leq n\} \cup [\![c_0, \ldots, c_n \notin D]\!]$.

It should be noted that the cases in the previous definition are indeed exhaustive: side conditions usually are only provided to specify a particular situation that can be assumed without loss of generality. Conditions that follow from the assumptions are omitted. Observe that the necessary conditions for recursion are satisfied in all cases of the definition. The choice of D_1 in the cases for $C = D_1 \sqcap D_2$ is possible since we disregard the nesting order of $\sqcup$: if there is some $D_1 \notin \mathbf{D}_B$, then there is some such D_1 that does not have a $\mathbf{C}_{\geq}$ disjunct (which is in $\mathbf{D}_B$) while still $D_1 \notin \mathbf{D}_B$. But then this $D_1 \notin \mathbf{D}_{\geq\omega-m}$ for all $m \geq 0$ as required.

It is not hard to see that, given the preconditions of Definition 15, we find that $[\![c_0, \ldots, c_k \in C \rightsquigarrow A_0, \ldots, A_k]\!]_B \models \bigcup_{0\leq i\leq k}\{C \sqcap \{c_i\} \sqsubseteq A_i\}$ and $[\![C \rightsquigarrow A]\!]_{B\leq} \models C \sqsubseteq A$. Notably, the case $C = \leqslant n\,R.\neg D$ uses a different approach than the other cases: the positive information used to entail non-emptiness of A is found in the equality relations that are implied between auxiliary constants d_i.

Observe that the datalog programs of Definition 15 again may significantly constrain the extension of C. For example, if $C = \leqslant 1\,R.\neg\bot$ then $[\![C \rightsquigarrow A]\!]_{B\leq}$ is only satisfied by interpretations that entail either $C \sqsubseteq \bot$ or $\top \sqsubseteq C$. This may entail $\top \sqsubseteq A$, so we will only use $[\![C \rightsquigarrow A]\!]_{B\leq}$ if $\top \sqsubseteq A$ or $C \sqsubseteq \bot$ is satisfiable. Non-emptiness of C might also be unavoidable, so one cannot assume that $[\![C \rightsquigarrow A]\!]_{B\leq} \cup \{A \sqsubseteq \bot\}$ is satisfiable. Yet, the remaining freedom will generally suffice for our purposes.

Another noteworthy fact is that $[\![c_0, c_1 \in C \rightsquigarrow A_0, A_1]\!]_B$ is not the same as $[\![c_0 \in C \rightsquigarrow A_0]\!]_B \cup [\![c_1 \in C \rightsquigarrow A_1]\!]_B$, which is the reason why the definition must explicitly include cases with $k > 0$. To see this, consider $C = (\neg\{a\}\sqcap\neg\{b\})\sqcup B$. Then $[\![c_0, c_1 \in C \rightsquigarrow A_0, A_1]\!]_B = \{B(c_0) \rightarrow A_0(c_0), B(c_1) \rightarrow A_0(c_1), c_0 \approx a, c_1 \approx b\}$ but $[\![c_0 \in C \rightsquigarrow A_0]\!]_B \cup [\![c_1 \in C \rightsquigarrow A_1]\!]_B = \{B(c_0) \rightarrow A_0(c_0), B(c_1) \rightarrow A_0(c_1), c_0 \approx a, c_1 \approx a\}$. The latter entails the unwanted consequence $c_0 \approx c_1$ since the auxiliary programs $[\![c_i \notin \neg\{a\} \sqcap \neg\{b\}]\!]$ are constructed independently for $i = 0, 1$ instead of using $[\![c_0, c_1 \notin \neg\{a\} \sqcap \neg\{b\}]\!]$.

The following lemma provides some important ingredients for showing maximality of $\mathcal{DLP}$, since it establishes the pre-conditions of Lemma 14 for broad classes of concepts.

Lemma 18. *Let* $C \in \mathbf{D}_a^+$ *be a structural concept expression in DLP normal form, let* $\mathbf{I}$ *be the set of constants of the given signature, and let* $a, b, c \in \mathbf{I}$ *be arbitrary constants not occurring in* C.

(1) If $C \notin \mathbf{D}_H$, *then one of the following is true:*
  - *There is a theory* T *and a set of constants* $N \subseteq \mathbf{I}$ *with* $a, b \in N$ *such that: given an arbitrary model* $\mathcal{I}$ *of* $\{\{a\} \sqcup \{b\} \sqsubseteq C\} \cup T$, *we find that* $\mathcal{J} = \mathcal{I} \times_{(N\times N)} \mathcal{I}$ *is such that* $\langle a, b\rangle \notin C^{\mathcal{J}}$.
  - *There are theories* T_1, T_2 *such that: given arbitrary models* $\mathcal{I}_i$ *of* $\{\{a\} \sqcup \{b\} \sqsubseteq C\} \cup T_i$ *(*$i = 1, 2$*), we find that* $\mathcal{J} = \mathcal{I}_1 \times_{(\mathbf{I}\times\mathbf{I})} \mathcal{I}_2$ *is such that* $\langle a, b\rangle \notin C^{\mathcal{J}}$.

(2) If $C \notin \mathbf{D}_a$, *then there are theories* T_1, T_2 *such that: given arbitrary models* $\mathcal{I}_i$ *of* $\{\{c\} \sqsubseteq C\} \cup T_i$ *(*$i = 1, 2$*), we find that* $\mathcal{J} = \mathcal{I}_1 \times_{(\mathbf{I}\times\mathbf{I})} \mathcal{I}_2$ *is such that* $c^{\mathcal{J}} = \langle c, c\rangle \notin C^{\mathcal{J}}$.

In all cases, models $\mathcal{I}$, $\mathcal{I}_1$ and $\mathcal{I}_2$ as described in the claims exist.

Proof. By Proposition 7, $C \in \mathbf{D}_a^+$ implies $D \in \mathbf{D}_a^+$ for all subconcepts D of C.

We start by considering claim (1). Claim (2) is shown independently below, so if $C \notin \mathbf{D}_a$ then we obtain theories T_1 and T_2 as in claim (2) for some fresh constant c. It is easy to see that the theories $T_i' \coloneqq T_i \cup \{a \approx c, b \approx c\}$ $(i = 1, 2)$ suffice for establishing claim (1). It remains to show claim (1) for cases where $C \in \mathbf{D}_a$. An easy induction can be used to show that $\mathbf{D}_H^+ \cap \mathbf{D}_a \subseteq \mathbf{D}_H$. Hence, using our assumption that $C \notin \mathbf{D}_H$, we can also conclude $C \notin \mathbf{D}_H^+$.

The only remaining cases for claim (1) therefore are such that $C \notin \mathbf{D}_H^+$, so that Lemma 17 can be applied. Define $T_1 \coloneqq T_2 \coloneqq [\![a, b \in C]\!]_\times$, and define N as in the lemma. The claim follows from Lemma 17.

For claim (2), we construct theories $T_1 = T_1(c, C)$ and $T_2 = T_2(c, C)$ for a fresh constant c as in the claim. The proof proceeds by induction over the structure of C. Note that C cannot be an atomic class, nominal, Self restriction, or the negation thereof.

Consider the case $C = D_1 \sqcap D_2$. Without loss of generality, we find that $D_1 \notin \mathbf{D}_a$. Applying the induction hypothesis, we obtain theories $T_i(c, C) \coloneqq T_i(c, D_1)$ $(i = 1, 2)$ that satisfy the claim.

Consider the case $C = D_1 \sqcup D_2$. As a first case, assume that $D_1 \notin \mathbf{D}_a$. Then we can define theories $T_i(c, C) \coloneqq T_i(c, D_1) \cup [\![c \notin D_2]\!]$ $(i = 1, 2)$. The claim then follows from the induction hypothesis together with the fact that every product interpretation constructed from models of $T_i(c, C)$ $(i = 1, 2)$ must also satisfy $[\![c \notin D_2]\!]$ by Proposition 6. The case $D_2 \notin \mathbf{D}_a$ is similar.

Now assume that $C = D_1 \sqcup D_2$ with $D_1, D_2 \notin \mathbf{D}_B$. Using fresh concept names A_1, A_2 and the construction of Definition 15, define $T_i(c, C) \coloneqq \{A_i(c) \to \bot\} \cup \bigcup_{j=1,2} [\![c \in D_j \rightsquigarrow A_j]\!]_B$ for $i = 1, 2$. Then any product interpretation $\mathcal{J}$ of any two models of $T_i(c, C)$ $(i = 1, 2)$ satisfies $\bigcup_{j=1,2} [\![c \in D_j \rightsquigarrow A_j]\!]_B \cup \{A_j(c) \to \bot\}$, and hence $\mathcal{J} \not\models \{c\} \sqcup D_i$ $(i = 1, 2)$ as required.

Consider the case $C = \leqslant 0\, R.\neg D$ with $D \notin \mathbf{D}_H$. Since $C \in \mathbf{D}_a^+$ we find $D \in \mathbf{D}_H^+$. Using $\mathbf{D}_H^+ \cap \mathbf{D}_a \subseteq \mathbf{D}_H$ as above, we conclude that $D \notin \mathbf{D}_a$, which allows us to apply the induction hypothesis. Consider a fresh individual name d and define $T_i(c, C) \coloneqq T_i(d, D) \cup \{R(c, d)\}$ $(i = 1, 2)$. Given models $\mathcal{I}_i$ of $T_i(c, C)$ $(i = 1, 2)$, the induction hypothesis implies that $\mathcal{J} \coloneqq \mathcal{I}_1 \times_{(\mathbf{I} \times \mathbf{I})} \mathcal{I}_2$ does not satisfy $\{d\} \sqsubseteq D$. Since $\mathcal{J} \models R(c, d)$ we conclude $\mathcal{J} \not\models \{c\} \sqsubseteq C$.

Consider the case $C = \leqslant 1\, R.\neg D$ with $D \notin \mathbf{D}_B$ and $D \notin \mathbf{D}_{\geq\omega-1}$. Using fresh symbols c_1, c_2, A_1, A_2, we define $T_i(c, C) \coloneqq \{A_i(c_i) \to \bot\} \cup [\![c_1, c_2 \in D \rightsquigarrow A_1, A_2]\!]_B \cup \{R(c, c_1), R(c, c_2)\}$ for $i = 1, 2$. Using similar arguments as in the last case of $C = D_1 \sqcup D_2$, we find that no product interpretation of models of $T_i(c, C)$ $(i = 1, 2)$ can satisfy $\{c\} \sqsubseteq C$.

Consider the case $C = \leqslant n\, R.\neg D$ with $n \geq 2$ and $D \notin \mathbf{D}_{\geq\omega-n}$. Using fresh individuals symbols $c_0, \ldots, c_n$, set $T \coloneqq [\![c_0, \ldots, c_n \notin D]\!] \cup \{R(c, c_i \mid 0 \leq i \leq n\}$. We define $T_1(c, C) \coloneqq T \cup \{c_i \approx c_j \to \bot \mid 1 \leq i < j \leq n\}$ and $T_2(c, C) \coloneqq T \cup \{c_i \approx c_j \to \bot \mid 0 \leq i < j \leq n - 1\}$. Thus, any model of $\{\{c\} \sqsubseteq C\} \cup T_1(c, C)$ $(\{\{c\} \sqsubseteq C\} \cup T_2(c, C))$ entails $c_0 \approx c_1$ $(c_{n-1} \approx c_n)$, but this entailment is lost in every product interpretation. This shows the desired result since product interpretations satisfy T by Proposition 6.

Consider the case $C = \geqslant 1\, R.D$ with $D \notin \mathbf{D}^{\geqslant 1}$. Then $D \in \mathbf{D}_a^+$ and $D \notin \mathbf{D}_a$. For a fresh constant d, define $T_i(c, C) \coloneqq T_i(d, D) \cup \{R(c, x) \rightarrow d \approx x\}$ for $i = 1, 2$. The claim follows from the induction hypothesis and the fact that every considered product interpretation also satisfies $\{R(c, x) \rightarrow d \approx x\}$.

Consider the case $C = \geqslant n\, R.D$ with $n \geq 2$ and $D \notin \mathbf{D}^{\geqslant n}$. Without loss of generality, we can assume that D is of the form $C_1 \sqcup \ldots \sqcup C_p \sqcup E$ ($p \geq 1$) where no C_i is a disjunction, $C_i \notin \mathbf{C}_B$ for $i = 1, \ldots, p$, and $E \in \mathbf{D}_B \cup \{\bot\}$. For the following argument, we use $E = \bot$ to cover the case where no such E is given in the original DLP normal form. Note that E might be a disjunction but cannot be $\top$.

First assume that there is some $F \in \{E, C_1, \ldots, C_p\}$ such that $F \in \mathbf{D}_{\geq\omega-k}$ for some $k \geq 0$. Since F is in DLP normal form, it is a disjunction that contains some disjunct in $\mathbf{C}_{\neg m}$ ($m \geq 1$). All subconcepts of D are assumed to be in $\mathbf{D}_a^+$, so if $m \leq n^2 - n$ then $D \in \mathbf{D}^{\geqslant n}$; a contradiction. Thus D is of the form $D_1 \sqcup D_2$ with $D_1 \in \mathbf{C}_{\neg m}$ and $m > n^2 - n$. Moreover, $D_2 \notin \mathbf{D}_a$ since otherwise we would find $D \in \mathbf{D}_a \subset \mathbf{D}^{\geqslant n}$.

The set of constants in D_1 is denoted as $\mathsf{ind}(D_1) = \{c_1, \ldots, c_m\}$. Let $p_1, p_2, \ldots, p_{n^2-n}$ denote a sequence of all pairs $p_i = \langle d_1, d_2 \rangle$ of constants $d_1, d_2 \in \{c_1, \ldots, c_n\}$ with $d_1 \neq d_2$. The order is inessential, but some order is needed for notational purposes. Define auxiliary theories $T_i(c, C) \coloneqq \left\{\forall x.R(c, x) \rightarrow \bigvee_{1 \leq j \leq n} c_j \approx x\right\} \cup \bigcup_{1 \leq j \leq m} T_i(c_j, D_2) \cup \{c_j \approx d_i \mid n < j \leq m, p_{j-n} = \langle d_1, d_2 \rangle\}$. Observe that the first component in this definition refers only to the first n constants $c_1, \ldots, c_n$, the second part is specified for all m constants, and the third component refers to the last $m - n$ constants $c_{n+1}, \ldots, c_m$ only.

To see that these theories satisfy the claim, consider models $\mathcal{I}_i$ of $\{\{c\} \sqsubseteq C\} \cup T_i(c, C)$ ($i = 1, 2$), and let $\mathcal{J} = \mathcal{I}_1 \times_{(\mathbf{I}\times\mathbf{I})} \mathcal{I}_2$ denote their product. Observe that, by the construction of $T_i(c, C)$, the constants c_j ($1 \leq j \leq m$) are mutually unequal in $\mathcal{J}$. Now consider an arbitrary element $\delta \in \Delta^{\mathcal{J}}$ such that $\langle c^{\mathcal{J}}, \delta \rangle \in R^{\mathcal{J}}$. By definition of the product, there must be a constant symbol d – possibly an auxiliary constant that did not occur in C – such that $\delta = \langle d, d \rangle$ and $\langle c^{\mathcal{I}_i}, d^{\mathcal{I}_i} \rangle \in R^{\mathcal{I}_i}$ for $i = 1, 2$. Since the models $\mathcal{I}_i$ satisfy $\forall x.R(c, x) \rightarrow \bigvee_{1 \leq j \leq n} c_j \approx x$, we conclude that $\mathcal{I}_1 \models d \approx c_j$ and $\mathcal{I}_2 \models d \approx c_k$ for some (possibly distinct!) $j, k \in \{1, \ldots, n\}$. Thus, there are at most n^2 elements $\delta \in \Delta^{\mathcal{J}}$ such that $\langle c^{\mathcal{J}}, \delta \rangle \in R^{\mathcal{J}}$, since there are at most n^2 distinct ways of selecting j, k. Now m of those n^2 elements are of the for $c_j^{\mathcal{J}}$ for some $j = 1, \ldots, m$, and by the induction hypothesis we find that $c_j^{\mathcal{J}} \notin D_2^{\mathcal{J}}$. Since $c_j^{\mathcal{J}} \notin D_1^{\mathcal{J}}$ is immediate, we thus find that $c_j^{\mathcal{J}} \notin D^{\mathcal{J}}$ for all $j = 1, \ldots, m$. Summing up, we conclude that $\mathcal{J}$ can have most $n^2 - m$ distinct R-successors for c which are in D. Since $n^2 - m < n^2 - (n^2 - n) = n$, we find that $\mathcal{J} \not\models \{c\} \sqsubseteq \geqslant n\, R.D$, as required.

For the remainder of the proof, assume that $F \notin \mathbf{D}_{\geq\omega-k}$ for all $F \in \{E, C_1, \ldots, C_p\}$ and $k \geq 0$. In particular, we can use the constructions of Definition 14 and 15. Now if $\{\{c\} \sqsubseteq C\} \cup [\![E \sqsubseteq \leqslant 0\, R^-.\neg\neg\{c\}]\!]_{\leq}$ is unsatisfiable, then $C_1 \sqcup \ldots \sqcup C_p \in \mathbf{D}_{\leq n-1}$. Since we assumed that $C_1 \sqcup \ldots \sqcup C_p \in \mathbf{D}_a^+$, this again implies $D \in \mathbf{D}^{\geqslant n}$. Hence, $\{\{c\} \sqsubseteq C\} \cup [\![E \sqsubseteq \leqslant 0\, R^-.\neg\neg\{c\}]\!]_{\leq}$ must be satisfiable (note that this includes the case $E = \bot$). It is easy to see that $\{\{c\} \sqsubseteq C\} \cup [\![E \sqsubseteq \leqslant 0\, R^-.\neg\neg\{c\}]\!]_{\leq}$ semantically emulates $\{\{c\} \sqsubseteq \geqslant n\, R.C_1 \sqcup \ldots \sqcup C_p\}$, and that the claim can thus be established by induction. So for the remaining considerations we can assume that E is not present at all, i.e. that $C = \geqslant n\, R.C_1 \sqcup \ldots \sqcup C_p$.

By the assumptions on C_i, we can apply Definition 15 and set $T \coloneqq \bigcup_{1 \leq i \leq p} ([\![C_i \rightsquigarrow A_i]\!]_{B\leq} \cup \{R(x, y) \wedge A_i(y) \rightarrow B_i(x)\})$ for fresh concept names $A_1, \ldots, A_p, B_1, \ldots, B_p$. It is easy to verify that $\{\{c\} \sqsubseteq C\} \cup T$ is consistent. Now consider the theory $T' := T \cup \{B_i(x) \rightarrow \bot \mid T \cup \{\{c\} \sqsubseteq C\} \cup \{B_i \sqsubseteq \bot\}$ is consistent$\}$, where it should be noted how the B_i are used to avoid inconsistencies that could arise immediately when requiring $A_i \sqsubseteq \bot$. Consider the case (A) that $T' \cup \{\{c\} \sqsubseteq C\}$ is inconsistent. Then there are two disjoint subsets $I_1, I_2 \subseteq \{1, \ldots, p\}$ for which $T_k(c, C) := T \cup \{B_i \sqsubseteq \bot \mid i \in I_k\}$ is such that $T_k(c, C) \cup \{\{c\} \sqsubseteq C\}$ is consistent for $k = 1, 2$, while $T_1(c, C) \cup T_2(c, C) \cup \{\{c\} \sqsubseteq C\}$ is inconsistent. Every product interpretation of models of $T_k(c, C)$ ($k = 1, 2$) entails T (by Proposition 6) and $B_i \sqsubseteq \bot$ (by Definition 11), and thus cannot be a model of $\{\{c\} \sqsubseteq C\}$, as required.

Now consider the case (B) where $T' \cup \{\{c\} \sqsubseteq C\}$ is consistent. Then there is B_h such that $T \cup \{\{c\} \sqsubseteq C\} \cup \{B_h \sqsubseteq \bot\}$ is inconsistent. This implies that $\{\{c\} \sqsubseteq C\} \cup \{\geqslant 1\, R.C_h \sqsubseteq \bot\}$ is inconsistent. Since $C_h \notin \mathbf{C}_{\geq} \subseteq \mathbf{D}_B$, we conclude that either $\bigsqcup_{1 \leq i \leq p, i \neq h} C_i \in \mathbf{D}_{\leq n-1}$ or this concept is empty, i.e. $p = h = 1$.

First consider the case (B.1) where $C_h \in \mathbf{D}_{\leq 1}$. Then $C_1 \sqcup \ldots \sqcup C_p \notin \mathbf{D}_{\leq n-1}$ implies $p = n$ and $C_i \in \mathbf{D}_{\leq 1}$ for all $i \neq h$, $1 \leq i \leq p$. Since C is not of the $\mathbf{D}_H$-form $\geqslant n\, R.\mathbf{D}_{n!}$, there is k such that $C_k \notin \mathbf{D}_a$. Now $C_k \in \mathbf{D}_{\leq 1}$ implies that $C_k = \{a\} \sqcap C'_k$ for some individual a and concept $C'_k \notin \mathbf{D}_a$. As each model of C requires one R-successor of c in each concept of the form C_i, we find that $\{\{c\} \sqsubseteq C\}$ semantically emulates $\{\{a\} \sqsubseteq C_k\}$. The claim follows by induction.

As a second case (B.2), assume that $C_h \notin \mathbf{D}_{\leq 1}$. Then $C_h \notin \mathbf{D}_{\leq k}$ for all $k \geq 0$ since C_h is not a disjunction. Since this implies that $T \cup \{\{c\} \sqsubseteq C\} \cup \{B_i \sqsubseteq \bot \mid i \neq h\}$ is consistent, this theory must be equal to $T' \cup \{\{c\} \sqsubseteq C\}$.

Consider the case (B.2.1) where $C_h \notin \mathbf{D}_a$. For fresh individuals $c_1, \ldots, c_n$ define $T'' := T' \cup \{\forall R(c, x) \rightarrow \bigvee_{1 \leq i \leq n} c_i \approx x\}$. Note that $T'' \cup \{\{c\} \sqsubseteq C\}$ is satisfiable by interpretations $\mathcal{I}$ that have $c_i^{\mathcal{I}} \in C_h^{\mathcal{I}}$ as the n distinct R-successors of c. Define $T_i(c, C) := \bigcup_{1 \leq j \leq n} T_i(c_j, C_h) \cup T''$ ($i = 1, 2$).

To show that this satisfies the claim, consider models $\mathcal{I}_i$ of $\{\{c\} \sqsubseteq C\} \cup T_i(c, C)$ ($i = 1, 2$). Since the induction hypothesis only applies to named individuals, we introduce n^2 fresh constants $\langle c_j, c_k \rangle$ for $j, k \in \{1, \ldots, n\}$. $\mathcal{I}_1$ is extended to $\mathcal{I}'_1$ over this extended signature by setting $\langle c_j, c_k \rangle^{\mathcal{I}'_1} := c_j^{\mathcal{I}_1}$, so that $\mathcal{I}'_1 \models \langle c_j, c_k \rangle \approx c_j$. The extended interpretation $\mathcal{I}'_2$ is defined analogously for the second components. Due to the constructions in this proof, for any constants e, f, we find that $T_i(e, C_h)$ is the same as $T_i(f, C_h)$ with e uniformly replaced by f ($i = 1, 2$). Thus, we find that $\mathcal{I}'_i \models T_i(\langle c_j, c_k \rangle, C_h)$ for $i = 1, 2$ and all $j, k \in \{1, \ldots, n\}$. Moreover, $\mathcal{I}'_i \models \{\{\langle c_j, c_k \rangle\} \sqsubseteq C_h\}$ so the induction hypothesis can be applied to obtain $\mathcal{I}'_1 \times_{(\mathbf{I}' \times \mathbf{I}')} \mathcal{I}'_2 \not\models \{\langle c_j, c_k \rangle\} \sqsubseteq C_h$ where $\mathbf{I}'$ denotes the extended set of constants.

It is not hard to see that the interpretations $\mathcal{J}' = \mathcal{I}'_1 \times_{(\mathbf{I}' \times \mathbf{I}')} \mathcal{I}'_2$ and $\mathcal{J} = \mathcal{I}_1 \times_{(\mathbf{I} \times \mathbf{I})} \mathcal{I}_2$ are equal (possibly up to renaming of domain elements). In particular, $\mathcal{J}'$ entails $\langle c_j, c_k \rangle \approx \langle \langle c_j, c_{j'} \rangle, \langle c_{k'}, c_k \rangle \rangle$. Hence we find that $\mathcal{J} \not\models \{\langle c_j, c_k \rangle\} \sqsubseteq C_h$. Moreover, since $\mathcal{I}_1$ and $\mathcal{I}_2$ satisfy T'', we find that $\langle c^{\mathcal{J}}, \delta \rangle \in R^{\mathcal{J}}$ implies $\delta = \langle c_j, c_k \rangle^{\mathcal{J}}$ for some $j, k \in \{1, \ldots, n\}$. Thus we obtain $\mathcal{J} \not\models \{\{c\} \sqsubseteq C\}$ as required.

As the final case (B.2.2), assume that $C_h \in \mathbf{D}_a$. Since $D \notin \mathbf{D}^{\geqslant n}$, we find $D \neq C_h$, i.e. $p > 1$. We concluded $\bigsqcup_{1 \leq i \leq p, i \neq h} C_i \in \mathbf{D}_{\leq n-1}$ above for all sub-cases of (B). Hence D is of the form $\mathbf{D}_a \sqcup \mathbf{D}^+_{m!} \sqcup \mathbf{D}_{l!}$ – where we assume that m is the least natural number for which D has this form – and m and l do not satisfy the relevant conditions in the definition of $\mathbf{D}^{\geqslant n}$. Accordingly, we denote D as $C_h \sqcup M_1 \sqcup \ldots \sqcup M_m \sqcup L_1 \sqcup \ldots \sqcup L_l$. Since $M_1, \ldots, M_m, L_1, \ldots, L_l \in \mathbf{D}_{\leq 1}$, they are each of the form $\{d\} \sqcap C$ for some individual name d: let $e_1, \ldots, e_m, f_1, \ldots, f_l$ denote these individual names. Set $r := n - (m + l)$, and consider fresh individual names $c_1, \ldots, c_r$. Define a set $X := \{c_1, \ldots, c_r, e_1, \ldots, e_m, f_1, \ldots, f_l\}$ of all constants considered as R-successors of c. Using the induction hypothesis, define

$$\begin{aligned} T_i(c, C) := {} & [\![e_1, \ldots, e_m, f_1, \ldots, f_l \notin C_h]\!]_\times \cup \\ & [\![c_1, \ldots, c_r \in C_h, e_1, \ldots, e_m, f_1, \ldots, f_l \notin C_h]\!]_\times \cup \\ & \textstyle\bigcup_{1 \leq j \leq m} T_i(e_j, M_j) \cup \{\forall x.R(c, x) \to \bigvee_{d \in X} d \approx x\} \end{aligned}$$

for $i = 1, 2$. Note that the construction of Lemma 15 is possible: if C_h would be in $\mathbf{D}_B^+$, then $C \in \mathbf{D}_a$ would imply $C \in \mathbf{D}_B$, which cannot be.

To show that this satisfies the claim, consider models $\mathcal{I}_i$ of $\{\{c\} \sqsubseteq C\} \cup T_i(c, C)$ ($i = 1, 2$), and let $\mathcal{J} = \mathcal{I}_1 \times_{(\mathbf{I} \times \mathbf{I})} \mathcal{I}_2$ be the corresponding product interpretation. By the constructions of $T_i(c, C)$, we obtain that $\langle c^{\mathcal{J}}, \delta\rangle \in R^{\mathcal{J}}$ implies $\delta = \langle a, b\rangle^{\mathcal{J}}$ for some $a, b \in X$. We distinguish various cases:

- If $a, b \in \{e_1, \ldots, e_m, f_1, \ldots, f_l\}$ and $a \neq b$, then $\langle a, b\rangle^{\mathcal{J}} \notin E^{\mathcal{J}}$ for all $E = M_1, \ldots, M_m, L_1, \ldots, L_l$ can be concluded from $\langle a, b\rangle^{\mathcal{J}} \neq d^{\mathcal{J}}$ for all $d = e_1, \ldots, e_m, f_1, \ldots, f_l$. Moreover, $\langle a, b\rangle^{\mathcal{J}} \notin C_h$ by Lemma 15.
- If $a = b = e_j$ for some $j = 1, \ldots, m$, then $\langle a, b\rangle^{\mathcal{J}} \notin C_h$ again by Lemma 15. As above, $\langle a, b\rangle^{\mathcal{J}} \notin L_i^{\mathcal{J}}$ for all $i = 1, \ldots, l$. A similar argument shows $\langle a, b\rangle^{\mathcal{J}} \notin M_i^{\mathcal{J}}$ for all $i = 1, \ldots, m$ with $i \neq j$, whereas $\langle a, b\rangle^{\mathcal{J}} \notin M_j^{\mathcal{J}}$ follows by the induction hypothesis.
- If $a \in \{e_1, \ldots, e_m, f_1, \ldots, f_l\}$ and $b \in \{c_1, \ldots, c_r\}$, then $\langle a, b\rangle^{\mathcal{J}} \notin C_h$ follows from Lemma 16. The conclusion $\langle a, b\rangle^{\mathcal{J}} \notin E^{\mathcal{J}}$ for all $E = M_1, \ldots, M_m, L_1, \ldots, L_l$ follows as before.

In each of these cases, we thus find that $\langle a, b\rangle^{\mathcal{J}} \notin D^{\mathcal{J}}$. Therefore, the only elements $\langle a, b\rangle^{\mathcal{J}}$ that might be in $D^{\mathcal{J}}$ are such that either $a = b \in \{f_i, \ldots, f_l\}$ or $a, b \in \{c_1, \ldots, c_r\}$. This yields a maximum of $l + r^2$ R-successors for $c^{\mathcal{J}}$. Since $D \notin \mathbf{D}^{\geqslant n}$, we find that $r(r - 1) < m$ (the case $r \leq 0$ cannot occur for any case under (B)). Equivalently, $r^2 - r < m$ which in turn is equivalent to $r^2 - n + m + l < m$. But then $r^2 + l < n$, and we find $\mathcal{J} \not\models \{c\} \sqsubseteq C$, as required. □

The previous lemma already suffices to exclude a significant amount of axioms from DLP:

Corollary 1. *Let C be a structural concept expression in DLP normal form, let A be a fresh concept name, and let c be a fresh constant symbol.*

(1) *If $C \notin \mathbf{D}_H \cup \{\top, \bot\}$, then $A \sqsubseteq C$ cannot be $\mathbf{FOL}_=$-emulated by any datalog program.*
(2) *If $C \notin \mathbf{D}_a \cup \{\top, \bot\}$, then $\{c\} \sqsubseteq C$ cannot be $\mathbf{FOL}_=$-emulated by any datalog program.*
(3) *If $C \notin \mathbf{D}_H \cup \{\top, \bot\}$, and $C \notin \mathbf{D}_{\leq n}$ for all $n \geq 0$, and $C \notin \mathbf{C}_{\neq\top}$, then C cannot be $\mathbf{FOL}_=$-emulated by any datalog program.*

Proof. If $C \notin \mathbf{D}_a^+$, then the result follows from Proposition 7 in all cases. Thus assume that $C \in \mathbf{D}_a^+$ for the remainder of the proof.

For claim (1), consider fresh individual symbols a and b, and construct T_1 and T_2 as in Lemma 18 (1). Define $T_i' \coloneqq T_i \cup \{A(a), A(b)\}$ for $i = 1, 2$. Then T_1 and T_2 satisfy the preconditions of Lemma 14 for the knowledge base $\mathrm{KB} = \{\{a\} \sqsubseteq A, \{b\} \sqsubseteq A, A \sqsubseteq C\}$. In particular, $T_i \cup \{A \sqsubseteq C\}$ is satisfiable since C is in DLP normal form and $C \neq \bot$. This suffices to establish the claim.

For claim (2) and (3), we can directly use the theories T_1 and T_2 of Lemma 18 (2) and (1), respectively. To ensure that the preconditions of Lemma 14 hold for claim (3), we need to ensure that $\{C\} \cup T_i$ is satisfiable for $i = 1, 2$. To this end, $C \notin \mathbf{C}_{\neq\top} \cup \{\bot\}$ ensures that $\{C\}$ is satisfiable. $C \notin \mathbf{D}_{\leq n}$ for all $n \geq 0$ ensures that C is satisfiable by interpretations of arbitrary domain sizes, and it is not hard to see that $\{C\} \cup T_i$ is consistent when considering the construction in Lemma 18. □

The previous result already covers a significant amount of concept expressions that are not in $\{\top, \bot\} \cup \mathbf{D}_H \cup \mathbf{D}^{=n} \cup \mathbf{C}_{\neq\top}$. It remains to show that concepts in $\mathbf{D}_{\leq n} \setminus (\mathbf{D}^{=n} \cup \mathbf{C}_{\neq\top})$ for some $n \geq 1$ cannot belong to DLP.

Lemma 19. *Let C be a structural concept expression in DLP normal form such that $C \notin \{\top, \bot\}$, and $C \in \mathbf{D}_{\leq n} \setminus (\mathbf{D}^{=n} \cup \mathbf{C}_{\neq\top} \cup \mathbf{D}_H)$ for some $n \geq 1$. Then C cannot be $\mathbf{FOL}_=$-emulated by any datalog program.*

Proof. Observe that, for any $m \geq 1$, we find $\mathbf{C}_H^p \subset \mathbf{D}_H^p \subset \mathbf{C}_\bot^{=m} \subset \mathbf{C}_\bot^{=m+1}$. We define the *degree* $d(D)$ of a concept expression D as follows. If $D \in \mathbf{C}_\bot^{=m}$ for some $m \geq 1$, then let $d(D)$ be the largest such m. Otherwise, if $D \in \mathbf{D}_H^p$, then define $d(D) \coloneqq 1$. Otherwise set $d(D) \coloneqq 0$. Now since $C \in \mathbf{D}^{=n}$ it is of the form $C = (\{c_1\} \sqcap C_1) \sqcup \ldots \sqcup (\{c_n\} \sqcap C_n)$, and we can assume that $d(C_i) \leq d(C_{i+1})$ for all $i = 1, \ldots, n-1$. Using this notation, it is not hard to see that $C \notin \mathbf{D}^{=n}$ is equivalent to saying that $d(C_i) < i$ for some $i = 1, \ldots, n$.

First consider the case that $i > 1$. We find that C is semantically equivalent to $(\{c_1\} \sqcap C_1) \sqcup \ldots \sqcup (\{c_i\} \sqcap C_i)$. To see this, assume that $n \geq i$. Every model of C has at most n elements in its domain. Since $d(C_n) \geq n$ by construction, $C_n \in \mathbf{C}_\bot^{=n}$. By Lemma 5, we thus obtain $C_n \sqsubseteq C$ as a consequence of C, showing that C is equivalent to $(\{c_1\} \sqcap C_1) \sqcup \ldots \sqcup (\{c_{n-1}\} \sqcap C_{n-1})$. The claim thus follows by induction.

Now $C_j \notin \mathbf{C}_\bot^{=i}$ holds for all $j \leq i$. Using Lemma 5, we thus find that $\{c_j\} \sqsubseteq C_j$ is satisfiable by models of at most i elements in their domain. By structurality of C, we find that C is satisfiable, and clearly C is only satisfied by models with exactly $i > 1$ domain elements. Finite domain sizes can be enforced by $\mathbf{FOL}_=$ theories, and hence must be preserved by $\mathbf{FOL}_=$-emulation. But domain sizes greater than 1 are not preserved by the product construction of Definition 11, so the fact that C cannot be $\mathbf{FOL}_=$-emulated in datalog is a consequence of Proposition 6.

Consider the case $i = 1$. Using the same argument as above, we find that C is semantically equivalent to $\{c_1\} \sqcap C_1$. By construction, $C_1 \notin \mathbf{D}_H^p$. The claim is now shown by a miniature version of the proof steps that were used to establish Corollary 1, where relevant constructions and arguments largely collapse due to the requirement that the domain of interpretation is unary. We first provide two auxiliary constructions for the "propositional" variants of Definition 13 and 15. Given a structural concept $D \notin \mathbf{D}_\top^p$ and a constant d, recursively construct a datalog program $[\![d \notin D]\!]^p$ as follows:

- If $D \in \mathbf{C}_{\perp}^{=1}$ then $\llbracket d \notin D \rrbracket^p \coloneqq \emptyset$.
- If D is of the form $\mathbf{A}$, $\neg\mathbf{A}$, $\neg\{\mathbf{I}\}$, $\exists\mathbf{R}.\mathsf{Self}$, or $\neg\exists\mathbf{R}.\mathsf{Self}$, then we define $\llbracket d \notin D \rrbracket^p \coloneqq \mathsf{datalog}(\{d\} \sqsubseteq \neg D)$.
- If $D = D_1 \sqcap D_2$ with $D_1 \notin \mathbf{D}_{\top}^{p}$, then $\llbracket d \notin D \rrbracket^p \coloneqq \llbracket d \notin D_1 \rrbracket^p$.
- If $D = D_1 \sqcup D_2$ with $D_1, D_2 \notin \mathbf{D}_{\top}^{p}$, then $\llbracket d \notin D \rrbracket^p \coloneqq \llbracket d \notin D_1 \rrbracket^p \cup \llbracket d \notin D_2 \rrbracket^p$.
- If $D = \leqslant 0\, R.\neg D'$ with $D' \notin \mathbf{D}_{\top}^{p}$, then $\llbracket d \notin D \rrbracket^p \coloneqq \{R(d,d)\} \cup \llbracket d \notin D' \rrbracket^p$.
- If $D = \geqslant n\, R.D'$ with $n > 0$, then $\llbracket d \notin D \rrbracket^p \coloneqq \{\neg R(d,d)\}$.

If $D \notin \mathbf{D}_{B}^{p}$ then, for a concept name A, we recursively construct a datalog program $\llbracket \{d\} \sqcap D \sqsubseteq A \rrbracket_{B}^{p}$ as follows:

- If D has the form $\mathbf{A}$ or $\exists\mathbf{R}.\mathsf{Self}$, then $\llbracket \{d\} \sqcap D \sqsubseteq A \rrbracket_{B}^{p} \coloneqq \mathsf{datalog}(\{d\} \sqcap D \sqsubseteq A)$.
- If $D = D_1 \sqcap D_2$ with $D_1 \notin \mathbf{D}_{B}^{p}$, then $\llbracket \{d\} \sqcap D \sqsubseteq A \rrbracket_{B}^{p} \coloneqq \llbracket \{d\} \sqcap D_1 \sqsubseteq A \rrbracket_{B}^{p}$.
- If $D = D_1 \sqcup D_2$ with $D_1 \notin \mathbf{D}_{B}^{p}$ and $D_2 \notin \mathbf{D}_{\top}^{p}$, then $\llbracket \{d\} \sqcap D \sqsubseteq A \rrbracket_{B}^{p} \coloneqq \llbracket \{d\} \sqcap D_1 \sqsubseteq A \rrbracket_{B}^{p} \cup \llbracket d \notin D_2 \rrbracket^p$.
- If $D = \leqslant 0\, R.\neg D'$ with $D' \notin \mathbf{D}_{B}^{p}$, then $\llbracket \{d\} \sqcap D \sqsubseteq A \rrbracket_{B}^{p} \coloneqq \llbracket \{d\} \sqcap D' \sqsubseteq A \rrbracket_{B}^{p} \cup \{R(d,d)\}$.
- If $D = \geqslant 1\, R.D'$ with $D' \notin \mathbf{C}_{\perp}^{=1}$, then $\llbracket \{d\} \sqcap D \sqsubseteq A \rrbracket_{B}^{p} \coloneqq \{R(d,x) \rightarrow A(x)\}$.

To establish the claim, we recursively construct theories $T_1 \coloneqq T_1(c_1, C_1)$ and $T_2 \coloneqq T_2(c_1, C_1)$ that satisfy the preconditions of Lemma 14. Note that C cannot be an atomic class, nominal, Self restriction, or the negation thereof.

Consider the case $C = D_1 \sqcup D_2$ with $D_1, D_2 \notin \mathbf{D}_{B}^{p}$. It is easy to see that $T_i(c_1, C) \coloneqq T_i(c_1, D_1) \cup \{\neg A_i(x)\} \cup \bigcup_{j=1,2} \llbracket \{d\} \sqcap D_j \sqsubseteq A_j \rrbracket_{B}^{p}$ $(i = 1, 2)$ satisfy the claim for fresh concept names A_1, A_2. Furthermore, if $C = D_1 \sqcup D_2$ with $D_1 \notin \mathbf{D}_{H}^{p}$ and $D_2 \in \mathbf{D}_{B}^{p}$ then the claim is satisfied by $T_i(c_1, C) \coloneqq T_i(c_1, D_1) \cup \llbracket d \notin D_2 \rrbracket^p$ $(i = 1, 2)$. Similarly, for the case $C = D_1 \sqcap D_2$ with $D_1 \notin \mathbf{D}_{H}^{p}$, the theories $T_i(c_1, D_1)$ $(i = 1, 2)$ satisfy the claim.

Now consider the case $C = \geqslant n\, R.D$. Then $n = 1$ and $D \notin \mathbf{D}_{H}^{p}$. Since C is semantically equivalent to D on singleton domains, the claim follows again by induction. A similar reasoning is possible for the case $C = \leqslant n\, R.\neg D$ with $n = 0$ and $D \notin \mathbf{D}_{H}^{p}$. □

We are now, finally, in a position to state the main theorem of this section.

Theorem 4. *If C is a concept expression in DLP normal form such that $C \notin \mathcal{DLP}$, then C cannot be contained in any DLP description logic in the sense of Definition 6.*

Proof. By Definition 9, $C \notin \{\top, \perp\} \cup \mathbf{C}_H \cup \mathbf{D}^{=n} \cup \mathbf{C}_{\neq\top}$ for all $n \geq 1$. If $C \notin \mathbf{D}_{\leq n}$ for all $n \geq 0$ and $C \notin \mathbf{D}_{a}^{+}$, then the result follows by Proposition 7. If $C \notin \mathbf{D}_{\leq n}$ for all $n \geq 0$ and $C \in \mathbf{D}_{a}^{+}$, then the result follows by Corollary 1. If $C \in \mathbf{D}_{\leq n}$ for some $n \geq 0$, then the result follows by Lemma 19. □

9 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), and showed that the presented DLP description logic is the largest one possible. 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 work 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. First of all, neither $\mathcal{EL}$ nor Horn-$\mathcal{SHIQ}$ can be $\mathbf{FOL}_{=}$-emulated in datalog (DLP 2). The datalog obtained in these cases still preserves satisfiability even when arbitrary ABox facts (without complex concepts) are added. In other words, $\mathcal{EL}$ and Horn-$\mathcal{SHIQ}$ satisfy a weaker version of DLP 2 based on $\mathbf{FOL}_{\approx}^{\text{ground}}$-emulation of Definition 2, where $\mathbf{FOL}_{\approx}^{\text{ground}}$ is the variable-free fragment of $\mathbf{FOL}_{=}$. Under those weakened principles, a larger space of possible DL fragments is allowed, 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.

Even when weakening the principles of DLP like this, Horn-$\mathcal{SHIQ}$ is still excluded since it cannot be modular (DLP 5) by Proposition 2. In the presence of transitivity, Horn-$\mathcal{SHIQ}$ also is not strictly structural (DLP 6), 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 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.

Finally, the approach of this paper – seeking a structural 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 $\cup$ $\mathcal{SROIQ}$") that can be expressed in $\mathcal{SROIQ}$? Clearly, this fragment would contain DL Rules [10] and maybe some form of DL-safe rules [12]. 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 by a careful articulation of basic design principles. At the same time, our study 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 structurally closed.

Acknowledgements 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.

References

1. Serge Abiteboul, Richard Hull, and Victor Vianu. *Foundations of Databases*. Addison Wesley, 1994.
2. Hajnal Andréka, Johan F. A. K. van Benthem, and István Németi. Modal languages and bounded fragments of predicate logic. *Journal of Philosophical Logic*, 27(3):217–274, 1998.
3. Patrick Blackburn, Johan F. A. K. van Benthem, and Frank Wolter, editors. *Handbook of Modal Logic*, volume 3 of *Studies in Logic and Practical Reasoning*. Elsevier Science, 2006.
4. Evgeny Dantsin, Thomas Eiter, Georg Gottlob, and Andrei Voronkov. Complexity and expressive power of logic programming. *ACM Computing Surveys*, 33(3):374–425, 2001.
5. Benjamin N. Grosof, Ian Horrocks, Raphael Volz, and Stefan Decker. Description logic programs: combining logic programs with description logic. In *Proceedings of the 12th International Conference on World Wide Web (WWW'03)*, pages 48–57. ACM, 2003.
6. Ian Horrocks, Oliver Kutz, and Ulrike Sattler. The even more irresistible $\mathcal{SROIQ}$. In Patrick Doherty, John Mylopoulos, and Christopher A. Welty, editors, *Proceedings of the 10th International Conference on Principles of Knowledge Representation and Reasoning (KR'06)*, pages 57–67. AAAI Press, 2006.
7. Ulrich Hustadt, Boris Motik, and Ulrike Sattler. Data complexity of reasoning in very expressive description logics. In Leslie Pack Kaelbling and Alessandro Saffiotti, editors, *Proceedings of the 19th International Joint Conference on Artificial Intelligence (IJCAI'05)*, pages 466–471. Professional Book Center, 2005.
8. Yevgeny Kazakov. *Saturation-Based Decision Procedures for Extensions of the Guarded Fragment*. PhD thesis, Universität des Saarlandes, Saarbrücken, Germany, 2006.
9. Markus Krötzsch, Sebastian Rudolph, and Pascal Hitzler. Complexity boundaries for Horn description logics. In *Proceedings of the 22nd AAAI Conference on Artificial Intelligence (AAAI'07)*, pages 452–457. AAAI Press, 2007.
10. Markus Krötzsch, Sebastian Rudolph, and Pascal Hitzler. Description logic rules. In Malik Ghallab, Constantine D. Spyropoulos, Nikos Fakotakis, and Nikos Avouris, editors, *Proceedings of the 18th European Conference on Artificial Intelligence (ECAI'08)*, pages 80–84. IOS Press, 2008.
11. Markus Krötzsch, Sebastian Rudolph, and Pascal Hitzler. ELP: Tractable rules for OWL 2. In Amit Sheth, Steffen Staab, Mike Dean, Massimo Paolucci, Diana Maynard, Timothy Finin, and Krishnaprasad Thirunarayan, editors, *Proceedings of the 7th International Semantic Web Conference (ISWC'08)*, volume 5318 of *LNCS*, pages 649–664. Springer, 2008.
12. Boris Motik, Ulrike Sattler, and Rudi Studer. Query answering for OWL DL with rules. *Journal of Web Semantics*, 3(1):41–60, 2005.
13. Andrea Schaerf. Reasoning with individuals in concept languages. *Data Knowledge Engineering*, 13(2):141–176, 1994.
14. Stephan Tobies. *Complexity Results and Practical Algorithms for Logics in Knowledge Representation*. PhD thesis, RWTH Aachen, Germany, 2001.
15. Raphael Volz. *Web Ontology Reasoning with Logic Databases*. PhD thesis, Universität Karlsruhe (TH), Germany, 2004.