A Closer Look at the Semantic Relationship between Datalog and Description Logics

Markus Krötzsch a, Sebastian Rudolph b, Peter H. Schmitt c,∗

a Department of Computer Science, University of Oxford, UK, e-mail: markus.kroetzsch@cs.ox.ac.uk
b Institute AIFB, Karlsruhe Institute of Technology, DE, e-mail: sebastian.rudolph@kit.edu
c Institute for Theoretical Computer Science, Karlsruhe Institute of Technology, DE, e-mail: pschmitt@ira.uka.de

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

Keywords: Description Logic Programs, OWL RL, conservative extension, knowledge representation and reasoning

1. Introduction

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

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

\[
\begin{align*}
\text{Student} \sqsubseteq & \forall \text{supervisor}.\text{Prof}, \\
\text{Prof} \sqsubseteq & \exists \text{hasDegree}.\text{PhD}, \\
\text{Prof} \sqsubseteq & \text{FullProf} \sqcup \text{AssociateProf}.
\end{align*}
\]

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

\[
\begin{align*}
\forall x. \text{Student}(x) & \rightarrow (\exists y. \text{supervisor}(x,y) \rightarrow \text{Prof}(y)), \\
\forall x. \text{Prof}(x) & \rightarrow \exists y. \text{hasDegree}(x,y) \land \text{PhD}(y), \\
\forall x. \text{Prof}(x) & \rightarrow \text{FullProf}(x) \lor \text{AssociateProf}(x).
\end{align*}
\]

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

\[
\text{Student}(x) \land \text{supervisor}(x, y) \rightarrow \text{Prof}(y) \tag{7}
\]
\[
\text{hasAuthor}(x,y) \land \text{hasReviewer}(x,z) \land \text{supervisor}(y, z) \rightarrow \text{conflict}(z). \tag{8}
\]

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

A natural question to ask is how DLs and Datalog – viewed as decidable fragments of first-order logic – relate to each other. One direction of research explores how either formalism could be extended with features of the other. Approaches to extending the expressivity of DLs with first-order rules include \(\mathcal{ALP}\)-log [9], CARIN [26], SWRL [15,16], \(\mathcal{DL}+\log\) [35], DL-safe rules [31], and DL Rules [24,12].

A dual approach is to extended Datalog with typical DL features, in particular with existential quantification, which results in formalisms such as Datalog\(^+\) [6], \(\forall \exists\)-rules [4], and various related fragments of existential rules; see [22,32] for recent overviews.

These manifold research activities are based on the observation that DL and Datalog have distinct modelling capabilities that are not easily reconciled in a single formalism without sacrificing useful computational properties. For example, DLs feature existential quantification (2) and disjunction (3), while Datalog can capture dependency structures that are not expressible in DL axioms (8). However, DL and Datalog also have some overlapping expressivity. Formulae (1) and (7), e.g., are semantically equivalent. DLP (“Description Logic Programs”) has been proposed as a family of DLs that can be faithfully expressed in first-order Horn-logic, and in particular in Datalog [13,37]. This bears computational advantages since rule-based reasoning methods can be applied, and indeed DLP in its simplest form became the basis of the W3C standard OWL RL [28].

This raises the core question of this paper:

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

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

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

On the other hand, restricting to DL knowledge bases that are equisatisfiable to some set of Datalog rules may still be insufficient to characterise the “intersection” of DL and Datalog. For example, it is well-known that other tractable DLs such as \(\mathcal{EL}++\) can also be translated into equisatisfiable sets of Datalog rules [25,19,20]. The union of DLP and \(\mathcal{EL}++\) is not a DL for which standard reasoning tasks are tractable (see [25] for some discussion), so DLP and \(\mathcal{EL}++\) may merely be two among several tractable subsets of the “expressive intersection” of DL and Datalog, without actually capturing the essence of this slogan. Indeed, tractability was not among the original design goals of DLP, although it is now considered a major practical advantage that motivated the use in OWL RL.

Could the union of DLP and \(\mathcal{EL}++\) then be considered as an extended version of DLP? Possibly yes, since it is contained in the DL Horn-\(\mathcal{SHIQ}\) and the even more expressive Horn-\(\mathcal{SHIQ}\) for which satisfiability-preserving Datalog transformations are known [17,33]. However, for \(\mathcal{EL}++\) and DLP there

---

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

\(^2\)We generally allow rules with empty heads, interpreted as \textit{false}. Thus Datalog rules can be inconsistent.
exist modular (i.e., axiom-by-axiom) translations into Datalog. Opposed to this, the known Datalog transformation for Horn-SHOTIQ from [17] needs to process the whole knowledge base in an ExpTime compilation process to obtain the Datalog output.

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

But how can we be sure that there is no simpler transformation given that both data-complexity and combined complexity of Datalog, Horn-SHOTIQ, and Horn-SHOTIQ agree? The answer is given in Proposition 4.4 below. In any case, it is obvious from this discussion that the design principles for DLP — but also for EL and Horn-DLs — are not sufficiently well articulated to clarify the conceptual distinction between those formalisms.

This paper thus approaches an explicit characterisation of a maximal DLP-type logic. To do this, we first develop concrete requirements for such a language, that capture the specifics of the original DLP proposal, in Section 4. The above discussion indicates that some care is needed to define such principles. Thereafter, we ask whether DLP could be defined as a larger, or even as the largest, DL language that satisfies our design principles. A positive answer to this question is given by defining such a “largest possible Datalog fragment” DLP_{ALC} for the DL ALC in Section 5. This approach can be extended to more complex DLs such as SHOTIQ but the necessary canonical syntactic descriptions are significantly more complex and of little merit for the main insights reported in this paper. We thus rather provide a summary of the relevant results and methods in Section 6. Readers who are interested in the full technical presentation of this result are referred to an independent report [21].

### 2. Description Logic and Datalog

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

<table>
<thead>
<tr>
<th>Syntax</th>
<th>Semantics</th>
</tr>
</thead>
<tbody>
<tr>
<td>atomic concept</td>
<td>(A \sqsubseteq A')</td>
</tr>
<tr>
<td>top</td>
<td>(\top)</td>
</tr>
<tr>
<td>bottom</td>
<td>(\bot)</td>
</tr>
<tr>
<td>conjunction</td>
<td>(C \sqcap D)</td>
</tr>
<tr>
<td>disjunction</td>
<td>(C \sqcup D)</td>
</tr>
<tr>
<td>negation</td>
<td>(\neg C)</td>
</tr>
<tr>
<td>role restrictions</td>
<td>existential</td>
</tr>
<tr>
<td></td>
<td>universal</td>
</tr>
</tbody>
</table>

### Description Logics

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

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

An ALC knowledge base is a set of (terminological and assertional) axioms of ALC.

The semantics of DLs are based on a Tarski-style model theory. An interpretation \(I\) over a domain \(A\) assigns a set \(A' \subseteq A\) to each atomic concept \(A \in I\), a binary relation \(R' \subseteq A' \times A'\) to each role \(R \in I\), and an element \(a' \in A'\) to each individual \(a \in I\). The interpretation of concept expressions is defined recursively as in Table 1. A GCI \(C \sqsubseteq D\) is satisfied by \(I\), written \(I \models C \sqsubseteq D\), if \(C' \subseteq D'\). An assertion \(C(a)\) is satisfied by \(I\), written \(I \models C(a)\), if \(a' \in C'\). A knowledge base KB is satisfied by \(I\), written \(I \models KB\), if \(I \models a\) for all axioms \(a \in KB\). When an interpretation satisfies an axiom/a knowledge base, we also say that it is a model of that axiom/knowledge base.

Entailment is defined as usual. A knowledge base KB_{1} entails a knowledge base KB_{2}, written KB_{1} \models KB_{2}, if every model of KB_{1} is a model of KB_{2}. A knowledge base is unsatisfiable (or inconsistent) if it has no models, and satisfiable (or consistent) otherwise.
wise. We use the same terminology for axioms, treated as singleton knowledge bases.

It is well known in the folklore of DL and easy to see that there exists a translation $\pi$ of $\mathcal{ALC}$ (and also of $\mathcal{SROIQ}$) into $\mathcal{FOL}_e$ that preserves logical consequences and non-consequences, i.e., $KB_1 \models KB_2$ if and only if $\pi(KB_1) \models \pi(KB_2)$. A definition of $\pi$ may e.g., be found in [21, Figure 3.4]. Also note that every DL interpretation as defined above can be considered as an interpretation of $\mathcal{FOL}_e$ by considering atomic concepts as unary predicates, roles as binary predicates, and individuals as constant symbols.

**Datalog** We use the term “Datalog” to refer to the function-free Horn logic fragment of $\mathcal{FOL}_e$. A Datalog program is a first-order theory which contains only formulae of the form $\forall x_1 \ldots \forall x_n \rightarrow B$ where $A_i, B$ are atoms without function symbols of arity greater than 0, and universal quantifies over all variables occurring in the implications. We generally omit the quantifier, we simply write $B$ if $n = 0$, and we use $\bot$ to denote the empty head.

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

### 3. Semantic correspondences between logical theories

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

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

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

$$\exists \text{supervisor}.\text{Prof}(\text{tom}).$$

(9)

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

$$\text{supervisor}(\text{tom}, c_{\text{tomsprof}}).$$

(10)

$$\text{Prof}(c_{\text{tomsprof}}).$$

(11)

Then (10) and (11) together are just another way of writing the Skolemisation of (9) with $c_{\text{tomsprof}}$ used as the nullary Skolem function symbol. Thus both forms are equisatisfiable, but they are not semantically equivalent.

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

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

**Definition 3.2** Consider $\mathcal{FOL}_e$ theories $T$ and $T'$ with signatures $\mathcal{J} \subseteq \mathcal{J}'$. Then $T'$ semantically emulates $T$ if the following conditions hold:

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

It is usually not necessary to mention the signatures of $T$ and $T'$ explicitly, since it is always possible to find minimal signatures for $T$ and $T'$ that satisfy condition (1) of Definition 3.2. The concept of semantic emulation is related to the notion of conservative extensions [27] which, however, additionally assumes $T \subseteq T'$ and hence requires syntactic compatibility of the involved logics. This justifies to introduce the new notion of semantic emulation for our setting. Also note that, in contrast to equivalence and equisatisfiability, semantic emulation is not a symmetric relation, since one of the theories introduces additional “internal” symbols to its signature. It is not hard to see that the Datalog program consisting of (10) and (11) above semantically emu-
lates the DL fact (9) since we can always find a suitable interpretation for the fresh constant \(c_{\text{consprof}}\).

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

**Definition 3.3** Consider \(\text{FOL}_\text{ax}\) theories \(T\) and \(T'\) with signatures \(\mathcal{S} \subseteq \mathcal{S}'\). Then \(T'\) syntactically emulates \(T\) if for every first-order formula \(\varphi\) over \(\mathcal{S}\): \(T \models \varphi\) iff \(T' \models \varphi\).

It is easy to see that semantic emulation implies syntactic emulation. This illustrates the strength and significance of semantic emulation 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 \(T\) when restricting to those symbols.

Note that syntactic emulation of \(T\) by \(T'\) can equivalently be characterized by the requirement that for every formula \(\varphi\) over \(\mathcal{S}\) the sets \(T \cup \{\varphi\}\) and \(T' \cup \{\varphi\}\) be equisatisfiable.

We conclude this section with a useful lemma, which generalises the well-known least model property of Datalog. Therein, we assume the intersection of interpretations being defined in the obvious and usual way based on the intersection of predicate extensions. The proof of this lemma is straightforward by unraveling of the definitions.

**Lemma 3.4** Let \(I_1, I_2\) be interpretations over the same domain which agree on the interpretation of constant and function symbols, and let \(T\) be a first-order theory that is satisfied by \(I_1\) and \(I_2\).

1. If \(T\) is a Datalog program then also the intersection \(I_1 \cap I_2\) satisfies \(T\).
2. If \(T\) can be semantically emulated by a Datalog program then also the intersection \(I_1 \cap I_2\) satisfies \(T\).

4. The DLP fragment of a description logic

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

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

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

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

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

A knowledge base \(KB'\) is a variant of a knowledge base \(KB\) if it is obtained from \(KB\) by replacing each axiom with a variant.

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

Combining these ideas, we can formally define DLP fragments:

---

3The converse is not true in general, but this fact is not essential for our work so we do not discuss the details.
Definition 4.2 Given description logics $\mathcal{L}$ and $\mathcal{D}$, we call $\mathcal{D}$ a DLP fragment of $\mathcal{L}$ if

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

As discussed above, item (1) of this definition fixes the syntactic framework for DLP fragments. Item (2) states the property that motivates the study of DLP languages: every axiom of a DLP fragment can be expressed in Datalog. DLP languages as discussed in the literature may require the use of auxiliary symbols for the translation to Datalog [37], and the Datalog program can no longer be semantically equivalent to the original knowledge base in this case, even if all consequences with respect to the original predicates are still the same. This motivates the use of semantic emulation as introduced in Definition 3.2.

Item (3) of Definition 4.2 reflects our desire to obtain fragments that correspond to well-behaved logical languages as opposed to being arbitrary collections of axioms. An obvious way to implement this would be to require DLP fragments to be described by a context-free grammar. A typical feature of grammars for logical languages is that they are parametrised by a logical signature that can be modified without changing the essential structural features of the language. This effect is mirrored by the requirement of item (3) without introducing detailed requirements on a suitable logical grammar. We will find grammatical descriptions in the cases we consider, though item (3) as such does not imply that this is possible.

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

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

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

Proof. We start with an auxiliary construction: if the concept expression $C$ is satisfiable and does not contain the symbols $R, A_1, A_2$, and $c$, then no Datalog program semantically emulates the expression $\alpha_C := (C \land \forall R. (A_1 \lor A_2)) \exists R(c)$. For a contradiction, suppose that $\alpha_C$ is semantically emulated by a Datalog theory $\text{datalog}(\alpha_C)$. By construction, $\alpha_C$ is satisfiable, and so is $\{\alpha_C, A_i \subseteq \bot\}$ for each $i = 1, 2$. By Definition 3.3, we find that $\text{datalog}(\alpha_C) \cup \{A_i \subseteq \bot\}$ is satisfiable, too. Thus, there are models $I_i$ of $\text{datalog}(\alpha_C)$ such that $A_{i,1} = 0$. By the least model property of Datalog, there is also a model $I$ of $\text{datalog}(\alpha_C)$ such that $A_{1} = A_{2} = 0$. But then $\text{datalog}(\alpha_C) \cup \{A_1 \cup A_2 \subseteq \bot\}$ is satisfiable although $\{\alpha, A_1 \cup A_2 \subseteq \bot\}$ is not, contradicting the supposed semantic emulation.

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

Therefore, there is an unsatisfiable expression $\alpha$ with $\alpha \notin \mathcal{D}$. Now let $\mathcal{D}'$ be defined as $\mathcal{D} \cup \{\alpha\}$. The transformation is given by $\text{datalog}'(\alpha) = \text{datalog}(\alpha)$ if $\alpha \in \mathcal{D}$, and $\text{datalog}'(\alpha) = \{T \rightarrow A(x), A(x) \rightarrow \bot\}$ otherwise, where $A$ is a new predicate symbol. It is immediate that $\mathcal{D}'$ is a P-DLP fragment of $\mathcal{ALC}$ strictly greater than $\mathcal{D}$. \qed

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

We can also obtain an interesting general result on the complexity of reasoning in DLP fragments:

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

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

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

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

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

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

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

Using Definition 4.2, it is now possible to investigate DLP fragments of relevant description logics. In this paper, we detail this approach for $\mathcal{ALC}$; some remarks on the more complex case of $\mathcal{SROIQ}$ are given in Section 6 below. It turns out that the largest DLP fragment of $\mathcal{ALC}$ exists, and can be defined as follows, where we use the negation normal form NNF for simplifying our presentation.

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

- $\text{GCIs} C \subseteq D$ such that $\text{NNF}(\neg C \sqcup D)$ is an $L^a_H$ concept as defined in Fig. 1, or
- $\text{ABox} axioms C(a)$ where $\text{NNF}(C) = a L^a_H$ concept as defined in Fig. 1.

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

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

\[ \neg A \sqcap \forall R.(\neg B \sqcup \neg C) \in L^a_H, \quad (12) \]

\[ A \sqcup (B \sqcap \forall R.C) \in L^a_H, \quad (13) \]

\[ \neg A \sqcup \exists R.B \in L^a_H. \quad (14) \]
Though name separation prevents most forms of semantic interactions within concepts, we still require grammars for $L^{A}_\forall$ and $L^{A}_\exists$ to characterise concepts all variants of which are equivalent to $\top$ and $\bot$, respectively. This includes concept expressions such as $\forall A \bot$, $\exists A \bot$, and $\forall A \exists A \bot$.

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

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

1. If $C \in L^{A}_\forall$ then $C$ has a conjunctive normal form $\bigwedge_1 \bigwedge_j C_{i,j}$ with $C_{i,j}$ a negated atom for all $i, j$.
2. If $C \in L^{A}_\exists$ or $C \in L^{A}_\bot$ then $C$ has a conjunctive normal form $\bigwedge_1 \bigwedge_j C_{i,j}$ with $C_{i,j}$ negated or non-negated atoms and for every $i$ there is at most one $j$ such that $C_{i,j}$ is an non-negated atom.

(Since the assumptions require that $C$ does not contain quantifiers there is no difference here between $C \subseteq L^{A}_\forall$ and $C \subseteq L^{A}_\bot$.)

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

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

1. $C \in L^{A}_\bot$. Then the claim follows from part (1) of the lemma.
2. $C$ is an atom. Then the claim is obviously true.
3. $C = C_1 \sqcap C_2$ with $C_i \in L^{A}_\forall$. If $C_i$ is a conjunctive normal form of $C_i$ satisfying the claim then $C_i \sqcap C_2$ is a conjunctive normal form of $C$ satisfying the claim.
4. $C = C_1 \sqcup C_2$ with $C_i \in L^{A}_\forall$ and $C_1 \in L^{A}_\bot$. Let $\bigwedge_1 \bigwedge_j C_{i,j}$ and $\bigwedge_m \bigwedge_n C_{m,n}$ be the conjunctive normal forms that exist by induction hypothesis satisfying the respective claims. A conjunctive normal form of $C = C_1 \sqcup C_2$ is obtained as the conjunction of all $\bigwedge_1 \bigwedge_j C_{i,j}$ for all combinations of $i, m$. Since $\bigwedge_j C_{i,j}$ contains at most one positive atom and $\bigwedge_m C_{m,n}$ contains only negative atoms we are finished.

It is obvious that $DLP_{\mathcal{ALC}}$ satisfies items (1) and (3) of Definition 4.2, so what remains to show is that $DLP_{\mathcal{ALC}}$ knowledge bases can indeed be expressed in Datalog. Following the grammatical structure of $DLP_{\mathcal{ALC}}$, we specify three auxiliary functions for constructing Datalog programs to semantically emulate a $DLP_{\mathcal{ALC}}$ knowledge base. The following two lemmata can be proven by simple inductions as detailed in [21].

**Lemma 5.4** Given a concept name $A$, and a concept $C \subseteq L^{A}_\bot$, Fig. 2 recursively defines a Datalog program $\text{dig}_{\bot}^A(A \subseteq C)$ that semantically emulates $A \subseteq C$.

**Example 5.5** Let $E$ be the $L^{A}_\bot$ concept $\neg A \sqcup (B \sqcap \forall R.C)$ as in (13). Then $\text{dig}_{\bot}^A(D \subseteq E)$ consists of the following rules:

\[
\begin{align*}
D(x) \land X_1(x) & \rightarrow X_2(x) \\
A(x) & \rightarrow X_3(x) \\
X_2(x) & \rightarrow B(x) \\
X_3(x) \land R(x, y) & \rightarrow C(x) \\
X_2(x) & \rightarrow C(x)
\end{align*}
\]

Clearly, this rule set could be further simplified to obtain the three rules $D(x) \land A(x) \rightarrow X_3(x)$, $X_2(x) \rightarrow B(x)$, $X_3(x) \land R(x, y) \rightarrow C(x)$ which are easily seen to semantically emulate $D \subseteq E$.

**Lemma 5.6** Given a constant $a$ and a concept $C \subseteq L^{A}_\bot$, Fig. 3 recursively defines a Datalog program $\text{dig}_{\bot}^A(C(a), \bot)$ that semantically emulates $C(a)$.

Again, this transformation is designed for a concise definition, not for optimised output.
<table>
<thead>
<tr>
<th>C</th>
<th>$\text{dig}_B(A \sqsubseteq C)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$D \in L^A_B$</td>
<td>$\text{dig}_B(\neg X \sqsubseteq D) \cup {A(x) \land X(x) \rightarrow \bot}$</td>
</tr>
<tr>
<td>$B$</td>
<td>$[A(x) \rightarrow B(x)]$</td>
</tr>
<tr>
<td>$\forall R.D$</td>
<td>$\text{dig}_B([X \sqsubseteq D) \cup {A(x) \land R(x,y) \rightarrow X(y)]$</td>
</tr>
<tr>
<td>$D_1 \cap D_2$</td>
<td>$\text{dig}_B(A \sqsubseteq D_1) \cup \text{dig}_B^A(A \sqsubseteq D_2)$</td>
</tr>
<tr>
<td>$D_1 \cup D_2 \in (L^A_{\mu} \sqcap L^A_{\nu})$</td>
<td>$\text{dig}_B^A([X_2 \sqsubseteq D_1] \cup \text{dig}_B^A(\neg X_1 \sqsubseteq D_2) \cup {A(x) \land X_1(x) \rightarrow X_2(x)}$</td>
</tr>
</tbody>
</table>

Fig. 2. Transforming axioms $A \sqsubseteq L^A_B$ and $\neg A \sqsubseteq L^A_B$ to Datalog

<table>
<thead>
<tr>
<th>C</th>
<th>$\text{dig}_B^A(C(a), E)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$D \in L^A_B$</td>
<td>$\text{dig}_B^A([X \sqsubseteq D \sqcup E] \cup {X(a)}$</td>
</tr>
<tr>
<td>$D_1 \cap D_2$</td>
<td>$\text{dig}_B^A(D_1(a), E) \cup \text{dig}_B^A(D_2(a), E)$</td>
</tr>
<tr>
<td>$D_1 \cup D_2 \in (L^A_{\mu} \sqcap L^A_{\nu})$</td>
<td>$\text{dig}_B^A(\neg X \sqsubseteq D_1) \cup \text{dig}_B^A(\neg X \sqsubseteq D_2) \cup \text{dig}_B^A(D_1(a), E) \sqcup \neg X$</td>
</tr>
<tr>
<td>$\exists R.D$</td>
<td>$\text{dig}_B^A([X \sqsubseteq E] \cup {X(a) \rightarrow R(a,b), X(a) \rightarrow Y(b)}$</td>
</tr>
</tbody>
</table>

Fig. 3. Transforming axioms $C(a)$ with $C \in L^A_B$ to Datalog

Example 5.7 Let $E$ be the $L^A_B$ concept $\neg A \sqcap \exists R.B$ as in (14). Then $\text{dig}_B^A(E(a), \bot)$ consists of the following rules ($X$, and $Y$ indicating fresh concept names as in the definition of the transformation):

$A(x) \rightarrow X_1(x) \quad X_3(a) \rightarrow R(a,b)$

$X_5(a) \rightarrow Y(b) \quad X_3(x) \land X_4(x) \rightarrow X_2(x)$

$X_5(x) \rightarrow X_4(x) \quad X_5(x) \rightarrow X_6(x)$

$X_5(x) \rightarrow B(x) \quad X_5(x) \land X_6(x) \rightarrow X_7(x)$

$X_7(x) \rightarrow B(x) \quad Y(x) \rightarrow X_6(x)$

As before, this rule set can be simplified significantly by eliminating most of the introduced auxiliary concept symbols. Doing this, we obtain the three rules $A(x) \rightarrow X_2(x)$, $X_3(a) \rightarrow R(a,b)$, and $X_5(x) \rightarrow B(b)$, which again are easily seen to semantically emulate $E(a)$ as claimed. Here, the fresh constant symbol $b$ acts as a Skolem constant that represents the individual that the existential concept expression may require to exist.

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

Theorem 5.8 For every $DLP_{\mathcal{ALC}}$ axiom $\alpha$ as in Definition 5.1, one can construct a Datalog program $\text{dig}(\alpha)$ that emulates $\alpha$.

Proof. If $\alpha = C \sqsubseteq D$ is a TBox axiom, define $\text{datalog}(\alpha) := \text{dig}_B^A(A \sqsubseteq \text{NNF}(\neg C \sqcup D)) \cup \{A(x)\}$. If $\alpha = C(a)$ is an ABox axiom, define $\text{datalog}(\alpha) := \text{dig}_B^A(C(a), \bot)$. The result then follows by Lemma 5.4 and 5.6. □

We still need to show that $DLP_{\mathcal{ALC}}$ is indeed the largest DLP fragment of $\mathcal{ALC}$. We first introduce two
transformation – etb and qe –, and make some basic observations that allow us to use these transformations for showing maximality of DLP_{ALC}.

**Definition 5.9** Let C be an arbitrary ALC concept expression. The expression etb(C) (eliminate top and bottom) is obtained from C by elimination of top and bottom symbols, achieved by applying exhaustively the following rewrite rules:

\[
\begin{align*}
    & \top \land D \rightarrow D \\
    & \top \lor D \rightarrow \top \\
    & D \land \top \rightarrow D \\
    & D \lor \top \rightarrow \top \\
    & \exists R. \bot \rightarrow \bot \\
    & \forall R. \top \rightarrow \top \\
\end{align*}
\]

Note, that etb(C) may still contain subexpressions of the form \(\forall R. \bot\) and \(\exists R. \top\).

The next lemma summarises some easy observations on etb.

**Lemma 5.10** For any ALC concept expression C:

1. etb(C) is logically equivalent to C, i.e., for any interpretation \((\Delta^I, I)\) and any \(a \in \Delta^I\), we have \(a \in C^I\) iff \(a \in \text{etb}(C)^I\);
2. for every \(L \in \{L^A_0, L^A_1, L^A_2, L^A_3, L^A_{10}\}\), we have \(C \in L\) iff \(\text{etb}(C) \in L\);  
3. if C does not contain subexpressions of the form \(\forall R. \bot\) or \(\exists R. \top\) then \(\text{etb}(C) = \bot\), or \(\text{etb}(C) = \top\), or \(\text{etb}(C)\) does neither contain \(\bot\) nor \(\top\).

**Definition 5.11** Let C be an arbitrary ALC concept expression. The expression qe(C) is obtained from C by quantifier elimination:

\[
\begin{align*}
    & \text{qe}(A) = A \quad \text{(concept name)} \\
    & \text{qe}(\neg C_1) = \neg \text{qe}(C_1) \\
    & \text{qe}(C_1 \cap C_2) = \text{qe}(C_1) \cap \text{qe}(C_2) \\
    & \text{qe}(C_1 \cup C_2) = \text{qe}(C_1) \cup \text{qe}(C_2) \\
    & \text{qe}(\forall R. C_1) = \forall R. \text{qe}(C_1) \\
    & \text{qe}(\exists R. C_1) = \exists R. \text{qe}(C_1) \\
\end{align*}
\]

**Lemma 5.12** Let \((I, A, R)\) be a signature and fix a domain \(\Delta\). There is an interpretation \(I_0\) on \(\Delta\) of the role symbols in \(R\) such that for any interpretation \(I_0\) on \(\Delta\) of the signature \((I, A, \emptyset)\), and for any concept C of \((I, A, R)\), we find \(C^I = \text{qe}(C)^{I_0}\) with \(I = I_0 \cup I_1\).

**Proof.** Setting \(I_1(R) = \{(a, a) \mid a \in \Delta\}\) for all \(R \in R\), we obtain:

\[
\begin{align*}
    & (\forall R. D)^I = \{a \in \Delta \mid b \in D^{I_0} \text{ for all } (a, b) \in D^I\} \\
    & = \{a \in \Delta \mid a \in D^{I_0}\} = D^{I_0}, \\
    & (\exists R. D)^I \subseteq \{a \in \Delta \mid \text{there is } (a, b) \in D^I \text{ with } b \in D^{I_0}\} \\
    & = \{a \in \Delta \mid a \in D^{I_0}\} = D^{I_0}. \\
\end{align*}
\]

Note, that Lemma 5.12 is true for arbitrary ALC concept expressions, they need neither belong to DLP_{ALC} nor be name-separated.

**Lemma 5.13** Let C be an arbitrary ALC concept expression. Then

\[
\begin{align*}
    & C \in L^A_0 \quad \text{iff} \quad \text{qe}(C) \in L^A_0, \\
    & C \in L^A_1 \quad \text{iff} \quad \text{qe}(C) \in L^A_1, \\
    & C \in L^A_2 \quad \text{iff} \quad \text{qe}(C) \in L^A_2. \\
\end{align*}
\]

**Proof.** Here is a sample from the inductive proof for the first equivalence. The goal in this case is to show that \((\forall R. D) \in L^A_0\) iff \(D \in L^A_0\).

The “if” direction is directly covered by a grammar rule. For the “only if” direction, we observe that there are only two grammar rules that can produce a formula of the form \((\forall R. D)\). The first is \(\forall R. L^A_{10}\), for which we directly find that \((\forall R. D) \in L^A_0\) implies \(D \in L^A_0\). The second rule is \(\forall R. L^A_1\). Thus \((\forall R. D) \in L^A_1\) implies \(D \in L^A_1\), which suffices since \(L^A_2 \subseteq L^A_{10}\). \(\square\)

**Theorem 5.14** DLP_{ALC} is the largest DLP fragment of ALC.

**Proof.** For a contradiction, suppose that there is a DLP fragment F of ALC that is strictly larger than DLP_{ALC}. Then there is some GCI \(C' \subseteq D' \in F\) but not in DLP_{ALC}. The other possibility that there is an ABox axiom \(C'(a) \in F\) with \(C'(a) \in L^A_{10}\) is completely analogous. By Definition 4.2, any name-separated variant \(C \subseteq D\) of \(C' \subseteq D'\) is still in F. Since DLP_{ALC} is closed under variants, \(C \subseteq D\) in not in DLP_{ALC}. By Definition 5.1 this means that the negation normal form \(E = \neg C \cup D\) is not in L_{10}. By Lemmas 5.10 and 5.13 also etb(qe(E)) is not in L_{10}. Let \(E'\) be a conjunctive normal form of etb(qe(E)). Thus \(E' = C_{01} \cap \ldots \cap C_{0k}\) with \(C_{0i} = L_{01} \cup \ldots \cup L_{0k}\), where each \(L_{ij}\) is a concept name or the negation of a concept name. Again, it can be verified that \(E \in L_{10}\) iff \(E' \in L_{10}\). Furthermore, for one \(i, 1 \leq i \leq k\) there are
two non-negated concept names among \{L_{i1}, \ldots, L_{in}\}. Otherwise, we could show \(E^{cnf} \in L^A_{\Pi,1} \). For this we need the extended grammar of \(L^A_{\Pi,1} \). Without loss of generality let \(i = 1 \) and \(L_{1,1} = A_1, L_{1,2} = A_2 \) positive. The name separation of \(E \) may have been lost by building the transformation to conjunctive normal form \(E^{cnf} \), but we still have the following:

1. For any atom \(A \), if \(A \) occurs in \(E^{cnf} \) then \(\neg A \) does not occur in \(E^{cnf} \), and vice versa.
2. For any two different conjuncts \(Con_i \) and \(Con_j \) of \(E^{cnf} \), there is a literal \(l \) occurring in \(Con_i \) and not in \(Con_j \) (and by symmetry also a literal \(l' \) occurring in \(Con_j \) and not in \(Con_i \)).

Claim 1 can be easily seen since the transformation from \(E \) to \(E^{cnf} \) is effected by repeated application of the rewriting rule \((C_1 \sqcap C_2) \sqcup C_3 \rightarrow (C_1 \sqcup C_3) \sqcap (C_2 \sqcup C_3) \)

Claim 2 can be proven by induction on the structural complexity of \(E \). In the simplest case \(E \) already is a conjunctive normal form. Then name separation of \(E \) even implies that different conjuncts \(Con_i \) are disjoint.

Next assume that \(E = E_1 \sqcup \ldots \sqcup E_n \) and by induction hypothesis each \(E_i \) has a conjunctive normal form \(E_i = Con_{i1} \sqcap \ldots \sqcap Con_{in} \), such that for \(j \neq k \) the conjunct \(Con_{ij} \) contains a literal, that does not occur in \(Con_{jk} \). Furthermore, name separation of \(E \) tells us that different \(E_{i1}, E_{i2} \) do not share a literal. By elementary computation we have

\[
E^{cnf} = \bigcap_{1 \leq i \leq m_1} \ldots \bigcap_{1 \leq i \leq m_n} (Con_{i1} \sqcup \ldots \sqcup Con_{in}).
\]

Let us look at two different conjuncts in \(E^{cnf} \). Typically we may consider \(Con_{i1} \sqcup C_i \) and \(Con_{i2} \sqcup C_i \) with \(C_i = Con_{i2} \sqcap \ldots \sqcap Con_{m_2}\). By induction hypothesis there is a literal \(l \) in \(Con_{i1} \) that does not occur in \(Con_{i2} \). Under the present assumptions \(l \) occurs in \(Con_{i1} \sqcup C_i \) and not in \(Con_{i2} \sqcup C_i \). This completes our proof of claim 2. Returning to our main line of reasoning we define interpretations \(I_1 \) and \(I_2 \) on a universe \(A \) by

\[
A_1^I = \Delta \quad L_{i1}^{I_1} = \emptyset \text{ for all } 2 \leq j \leq n_1, \quad A_2^I = \Delta \quad L_{i2}^{I_2} = \emptyset \text{ for all } 1 \leq j \leq n_1, j \neq 2.
\]

Thus

\[
Con_{i1}^I = Con_{i2}^I = \Delta \quad \text{and} \quad Con_{i1}^{I \cap I_2} = \emptyset.
\]

By property 2 it is possible to extend the interpretations \(I_i \) such that \(Con_{ij}^I = \Delta \) for \(i \in [i, 1] \) and \(2 \leq j \leq k \). In total we have

\[
(E^{cnf})^{I_1} = (E^{cnf})^{I_2} = \Delta \quad \text{and} \quad (E^{cnf})^{I \cap I_2} = \emptyset.
\]

Since the normal form and the etb transformation preserve logical equivalence, we also have \(\pi(E^{I_i}) = \Delta \) for \(i = 1, 2 \) and \(\pi(E^{I \cap I_2}) = \emptyset \). By Lemma 5.12 there are expansions \(I_i^* \) of \(I_i \) such that \(E_i^* = \pi(E^{I_i}) = \Delta \) for \(i \in [1, 2] \) and \(E_i^* \cap I_2 = E_i^{I \cap I_2} = \pi(E^{I \cap I_2}) = \emptyset \). By Lemma 3.4, this contradicts the possibility that \(\pi(E)\) can be emulated by a Datalog formula.

\[\square\]

6. The Datalog Fragment of SROIQ

The previous section showed that syntactic descriptions tend to become rather complex when maximising languages in a canonical way, but the situation is substantially more intricate when considering SROIQ instead of ALC as an underlying DL. Here, we summarise the conclusions that have been obtained in [21] for this case. There, a maximal DLP fragment of SROIQ\textsuperscript{free} – the DL obtained by discarding the restrictions of regularity and simplicity from SROIQ – has been developed under the additional requirement of closure under disjunctive normal forms (DNF):

**Theorem 6.1** The largest DLP fragment of SROIQ\textsuperscript{free} that is also closed under DNF exists, and it can be characterised by a parametrised set of grammar productions. We call this DL DLP*.

Disjunctive normal forms here are mainly required to curtail the syntactic complexity of the obtained fragment, and we conjecture that a maximal DLP fragment of SROIQ\textsuperscript{free} that does not have this property also exists. Rather than in the concrete description of this fragment, we are interested here in the general insights that are obtained from proofs of such results. The above result consists of three parts: (1) specifying an explicit syntactic characterisation, (2) showing that all DLP axioms can be FOλ\textsubscript{e}-emulated in Datalog, (3) showing that DLP* is the largest such DL. Here we give an overview of the main methods that are used in each step.

**Syntactic Characterisation** The main challenge here is to reduce the presentational complexity as far as possible. A DLP normal form is introduced that incorporates DNF and an improved form of NNF, and which ignores concepts that, like \(L^A_{\Pi,1}^A \) above, are always
equivalent to $\top/\bot$. The syntax of $DLP$ in normal form is still very complex due to the interplay of number restrictions and nominals that is possible even in name-separated axioms.

**Datalog Emulation** A recursive Datalog transformation as in the case of $DLP_{ALC}$ above is provided. The individual steps are substantially more involved, and even lead to exponentially large Datalog programs in various cases, although these programs are very regular and can be constructed in a single pass without complex computations. We conjecture that this blow-up is unavoidable but this issue has not been investigated further.

**Maximality** The least model property of Datalog was used for showing maximality of $DLP_{ALC}$, but no extension of this direct approach to $DLP$ has been found. Instead, additional model-theoretic properties of Datalog were used that incorporate submodels and product models [7]. Using various inductive arguments, it has then been shown that any extension of $DLP$ leads to axioms that cannot be FOL$_m$-emulated in Datalog.

**Example 6.2** Figure 4 provides some examples to illustrate the issues that occur in the general case. Each line shows one SROIQ axiom and a Datalog program that semantically emulates it. Cases like (1) are well-known and are already covered by $DLP_{ALC}$. In case (2), the nominal class $c$ denotes the set that contains the single element represented by $c$, which explains why the existential quantifier can be translated as in the example. This case is also rather common: the on-

```
\begin{align*}
\text{Example 6.2} & \quad \text{Figure 4 provides some examples to illustrate the issues that occur in the general case. Each line shows one SROIQ axiom and a Datalog program that semantically emulates it. Cases like (1) are well-known and are already covered by DLP_{ALC}. In case (2), the nominal class } c \text{ denotes the set that contains the single element represented by } c, \text{ which explains why the existential quantifier can be translated as in the example. This case is also rather common: the on-}
\end{align*}
```

In summary, one can say that the interplay of nominal classes, counting quantifiers, and finite amounts of disjunctions leads to many additional cases of axioms that can be expressed in Datalog. The restriction to name-separated axioms in Definition 4.2 cannot prevent this interaction, but it still suffices to ensure the existence of a maximal DLP fragment that can be described by a context-free grammar.

**7. Conclusions and Outlook**

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

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

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

This work also explicitly introduces a notion of \textit{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 this can be very different from semantic equivalence. Yet, emulation can be argued to define minimal requirements for preserving a theory’s semantics even in combination with additional information, so it appears to be a natural tool for enabling information exchange in distributed knowledge systems. We think that the articulation of this notion is useful for studying the semantic interplay of heterogeneous logical formalisms in general.

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

Acknowledgements The first author has been working at Karlsruhe Institute of Technology when conducting the research reported herein. This work was supported by DFG in project \textit{ExpresST} and by EPSRC in project \textit{ConDOR} (EP/G02085X/1).

References


