Query Rewriting for *DL-Lite* with n-ary Concrete Domains

Franz Baader and **Stefan Borgwardt**
Faculty of Computer Science
Technische Universität Dresden, Germany
firstname.lastname@tu-dresden.de

Marcel Lippmann
TNG Technology Consulting GmbH
Unterföhring, Germany
marcel.lippmann@tngtech.com

Abstract

We investigate ontology-based query answering (OBQA) in a setting where both the ontology and the query can refer to concrete values such as numbers and strings. In contrast to previous work on this topic, the built-in predicates used to compare values are not restricted to being unary. We introduce restrictions on these predicates and on the ontology language that allow us to reduce OBQA to query answering in databases using the so-called combined rewriting approach. Though at first sight our restrictions are different from the ones used in previous work, we show that our results strictly subsume some of the existing first-order rewritability results for unary predicates.

1 Introduction

Ontology-based query answering (OBQA) (see, e.g., [Ortiz, 2013] for an overview) extends query answering in databases in two directions. On the one hand, in OBQA it is not assumed that the available data are complete, and thus facts that are not present are assumed to be unknown rather than false (*no* closed world assumption [CWA]). On the other hand, an ontology can be used to state background knowledge about the data and to translate between vocabularies (e.g., user-oriented versus system-oriented). Nevertheless, if the query and ontology languages are suitably restricted, then OBQA can be reduced to classical query answering in databases.

As the query language, one usually considers (unions of) *conjunctive queries ((U)CQs)* (i.e., select-project-join queries) in this setting. If the ontology language belongs to the so-called *DL-Lite family* of Description Logics (DLs) [Calvanese *et al.*, 2007; Artale *et al.*, 2009], then the ontology can often be compiled into the query, which can then be evaluated over the unchanged data using the CWA [Calvanese *et al.*, 2007; 2011]. If this approach is feasible, then one says that the query language is *first-order (FO) rewritable* w.r.t. the ontology language. FO rewritability implies that OBQA then has the same data complexity as query answering in databases, AC^0. For settings where the data complexity of OBQA is no longer in AC^0 (e.g., if the DL $\mathcal{EL}$ is used as ontology language), the *combined rewriting* approach, in which both the query and the data are changed, has turned out to be useful [Lutz *et al.*, 2009; Kontchakov *et al.*, 2011]. In case the data can be rewritten in polynomial time, this yields polynomial data complexity.

Real-world datasets frequently contain concrete data values (such as numbers and strings), and database queries use built-in predicates on these values to formulate restrictions on the tuples to be selected. When adopting concrete data values and built-in predicates for the OBQA setting, it makes sense to employ them not only in the query, but also in the ontology. In ontology languages based on DLs, one then talks about DLs with *concrete domains* [Baader and Hanschke, 1991; Lutz, 2003]. In addition to *concepts* and *roles* (i.e., unary and binary predicates on the abstract domain), such DLs employ *attributes* (i.e., binary relations between the abstract and the concrete domain) to assign concrete values to individuals, and *concrete predicates* (corresponding to built-in predicates in databases) to formulate constraints on these values.

Motivated by OBQA applications, several authors have introduced dialects of *DL-Lite* and CQs with concrete domains [Poggi *et al.*, 2008; Savković and Calvanese, 2012; Artale *et al.*, 2012]. However, like the standard Web Ontology Language OWL 2,[1] these extensions of *DL-Lite* with concrete domains consider only *unary* predicates on data values, which can be used to constrain a single value, but cannot require relationships between different values. With unary predicates one can, for example, express that the systolic blood pressure of a patient is >120 and the diastolic blood pressure is >80, but setting the systolic blood pressure into a relationship with the diastolic one requires a binary predicate. In this work, we lift this restriction, i.e., we define an extension of *DL-Lite* with concrete domains that may have predicates of arbitrary arity, and show that—for concrete domains satisfying certain properties—CQs with built-in predicates from the concrete domain allow for a combined rewriting w.r.t. ontologies formulated in this new language. For example, using an appropriate binary predicate we can then express that the pulse pressure, i.e., the difference between the systolic and the diastolic blood pressure, is 50.

We do not assume that attributes are functional, but our logic can express (local) functionality (e.g., a patient can have only one systolic blood pressure). We also show that concrete domains satisfying our restrictions are closed under disjoint

[1] see https://www.w3.org/TR/owl2-overview/

union and product. Using the product of two domains (one for pressure values and one for time), we can compare measurements at different time points; e.g., ask for patients whose systolic blood pressure increased by 20 in 30 seconds.

In addition to our combined rewriting approach, we also show that the FO rewritability results for the *DL-Lite* variant with unary concrete predicates in [Savković and Calvanese, 2012] follow from our results. Basically, we show that (i) concrete domains with unary predicates satisfying the restrictions in [Savković and Calvanese, 2012] can be turned into ones satisfying our restrictions, and (ii) in the unary case our combined rewriting boils down to an FO rewriting. The results in [Artale *et al.*, 2012] are orthogonal to ours since they are restricted to the unary case, but allow for more expressiveness on the DL side. In contrast to our work and [Savković and Calvanese, 2012; Artale *et al.*, 2012], in [Poggi *et al.*, 2008] queries do not contain built-in predicates. Finally, in [Hernich *et al.*, 2017] the authors also consider a setting with non-unary concrete domains, but where the data complexity is CO-NP-hard in general. They then investigate for which kinds of queries this complexity goes down to P. In contrast, our goal is to find restrictions that ensure combined rewritability, and thus polynomial data complexity, for all queries. Detailed proofs of our results can be found in [Baader *et al.*, 2017].

2 Concrete Domains

We first introduce the general notion of concrete domains, and then restrict it such that it fits our purpose. A *concrete domain* $\mathcal{D}$ consists of (i) a non-empty set $\Delta^{\mathcal{D}}$ of *values*, (ii) a collection of predicates Π_i with associated arities m_i containing the special unary predicate $\top_{\mathcal{D}}$, and (iii) interpretations $\Pi_i^{\mathcal{D}} \subseteq (\Delta^{\mathcal{D}})^{m_i}$ for all predicates, where $(\top_{\mathcal{D}})^{\mathcal{D}} = \Delta^{\mathcal{D}}$.

Let $\mathsf{N_V}$ be a set of *variables*. A *$\mathcal{D}$-formula* ϕ is a Boolean combination of *$\mathcal{D}$-atoms* $\Pi(v_1, \ldots, v_m)$, where Π is an m-ary predicate and $v_1, \ldots, v_m \in \Delta^{\mathcal{D}} \cup \mathsf{N_V}$. The set of variables in ϕ is denoted by $\mathsf{Var}(\phi)$. A *$\mathcal{D}$-conjunction* (*$\mathcal{D}$-disjunction*) is a conjunction (disjunction) of $\mathcal{D}$-atoms. The set $\mathsf{sol}_V(\phi)$ of *solutions* for a $\mathcal{D}$-formula ϕ, where $V \supseteq \mathsf{Var}(\phi)$, consists of all variable assignments $f\colon V \to \Delta^{\mathcal{D}}$ satisfying ϕ in $\mathcal{D}$ (using the standard notion of satisfaction in a relational structure). The $\mathcal{D}$-formula ϕ is *satisfiable* if $\mathsf{sol}_{\mathsf{Var}(\phi)}(\phi) \neq \emptyset$, and it *implies* the $\mathcal{D}$-formula ψ if $\mathsf{sol}_V(\phi) \subseteq \mathsf{sol}_V(\psi)$, where $V := \mathsf{Var}(\phi) \cup \mathsf{Var}(\psi)$.

In the DL literature, concrete domains are usually required to satisfy additional properties that are tailored to the reasoning problems under consideration. For example, in order to obtain decidability of standard DL reasoning problems such as subsumption, Baader and Hanschke [1991] require the concrete domain to be *decidable*, which in our setting means that satisfiability of $\mathcal{D}$-conjunctions and implications between $\mathcal{D}$-conjunctions must be decidable. In the context of concrete domain extensions of $\mathcal{EL}$, this requirement is tightened by Baader *et al.* [2005] to decidability in *polynomial* time. However, to obtain polynomiality of subsumption, one additionally needs to require that the concrete domain is *convex*, i.e., whenever a $\mathcal{D}$-conjunction implies a (non-empty) $\mathcal{D}$-disjunction, then it should also imply one of its disjuncts. The papers [Savković and Calvanese, 2012; Artale *et al.*, 2012] among other things require $\mathcal{D}$ to be *unary*, which means that all its predicates must be unary.

Our combined rewritability results depend on the concrete domain $\mathcal{D}$ to be *cr-admissible*, i.e., polynomial, convex, and satisfying the following additional properties:

- $\mathcal{D}$ *has equality*: it contains all unary predicates $=_d$ with $d \in \Delta^{\mathcal{D}}$, which are interpreted as $\{d\}$, as well as a binary predicate $=$, interpreted as $\{(d, d) \mid d \in \Delta^{\mathcal{D}}\}$.
- $\mathcal{D}$ is *functional*: for any m-ary predicate Π, $d \in \Delta^{\mathcal{D}}$, and i, $1 \leq i \leq m$, the formula $\Pi(v_1, \ldots, v_m) \wedge =_d(v_i)$ has at most one solution.
- $\mathcal{D}$ is *constructive*: for all $\mathcal{D}$-conjunctions ϕ and $\mathcal{D}$-disjunctions ψ with $\mathsf{sol}_V(\phi) \setminus \mathsf{sol}_V(\psi) \neq \emptyset$, an element of this set can be computed in polynomial time.

The following concrete domains are known to be polynomial and convex [Baader *et al.*, 2005]:

- $\mathcal{D}_{\mathbb{Q}}$: The set $\mathbb{Q}$ of rational numbers with the unary predicates $\top_{\mathcal{D}_{\mathbb{Q}}}$, $=_q$, and $>_q$ (interpreted as $\{x \mid x > q\}$), and binary predicates $=$ and $+_q$ (with the interpretation $\{(x, y) \mid x = q + y\}$), for any $q \in \mathbb{Q}$.
- $\mathcal{D}_{\Sigma^*}$: The set Σ^* of words over an alphabet Σ with the predicates $\top_{\mathcal{D}_{\Sigma^*}}$, $=_w$, $=$, and conc_w (interpreted as $\{(x, y) \mid x = w \cdot y\}$), for any $w \in \Sigma^*$.

Both can also be shown to be functional and constructive, and hence cr-admissible. Moreover, we can show that the class of cr-admissible concrete domains is closed under disjoint union and product of concrete domains, which allows us to construct more complex domains without losing the above properties. For example, the product $\mathcal{D}_{\mathbb{Q}} \times \mathcal{D}_{\mathbb{Q}}$ can be used to model measurements that are associated with time stamps.

Unary Concrete Domains. The paper [Savković and Calvanese, 2012] about query answering in *DL-Lite* with unary concrete domains $\mathcal{D}$ imposes the following restriction.[2]

(infinitediff) For any $\mathcal{D}$-conjunction ϕ and $\mathcal{D}$-disjunction ψ, whenever $|\mathsf{sol}_V(\phi)| > 1$ and $\mathsf{sol}_V(\phi) \not\subseteq \mathsf{sol}_V(\psi')$ for every $\mathcal{D}$-atom ψ' in ψ (where $V := \mathsf{Var}(\phi) \cup \mathsf{Var}(\psi)$), then the cardinality of $\mathsf{sol}_V(\phi) \setminus \mathsf{sol}_V(\psi)$ is infinite.

The original definition actually does not include the condition $|\mathsf{sol}_V(\phi)| > 1$. However, it is easily checked that the constructions and results of [Savković and Calvanese, 2012] remain valid under our weaker version of *(infinitediff)*. In our setting, this modification is useful to accommodate the predicates $=_d$, whose presence would otherwise contradict *(infinitediff)*. To show that our results apply to the setting from [Savković and Calvanese, 2012], first note that one can add equality predicates to $\mathcal{D}$ without destroying *(infinitediff)*.

Lemma 2.1. *For any* unary *concrete domain $\mathcal{D}$ satisfying (infinitediff), the concrete domain $\mathcal{D}'$ obtained from $\mathcal{D}$ by adding the predicates $=$ and $=_d$ ($d \in \Delta^{\mathcal{D}}$) still satisfies (infinitediff).*

Surprisingly, in our setting *(infinitediff)* and convexity are equivalent, though they have been introduced for different

[2] The other restrictions in that paper, *(infinite)* and *(opendomain)*, are simply special cases with $\psi = \mathsf{false}$ and $\phi = \mathsf{true}$, respectively.

purposes in [Savković and Calvanese, 2012] and [Baader *et al.*, 2005], respectively. In general, convexity is a weaker restriction since it does not force non-singleton predicates to be infinite. But in the presence of the predicates $=_d$ we can show equivalence. In fact, if $\mathsf{sol}_V(\phi) \setminus \mathsf{sol}_V(\psi)$ is finite, then one can use the predicates $=_d$ to construct a counterexample to convexity. In contrast to the previous lemma, this result is not restricted to unary concrete domains.

Lemma 2.2. *A concrete domain $\mathcal{D}$ containing the predicates $=_d$ ($d \in \Delta^{\mathcal{D}}$) is convex iff it satisfies (infinitediff).*

As a further step towards showing that our results imply the ones in [Savković and Calvanese, 2012], observe that every unary concrete domain $\mathcal{D}$ is trivially functional. We will argue in Section 6 that, for unary concrete domains $\mathcal{D}$, we (i) need decidability only for unary predicates (not for $=$), and (ii) do not need polynomiality or constructivity of $\mathcal{D}$. In contrast, in the presence of predicates of higher arity, the predicates $=_d$, functionality, and constructivity are essential for our combined rewriting approach (see Section 6).

Convexity is necessary for our rewritability results both in the general and in the unary case. In fact, these results imply polynomial data complexity. If the concrete domain is not convex, then answering conjunctive queries that can refer to concrete domain predicates is CO-NP-hard in the data complexity (and hence neither FO nor combined rewritable, unless P=NP), even in the unary case [Savković, 2011; Savković and Calvanese, 2012; Artale *et al.*, 2012].

3 The Ontology Language

For any cr-admissible concrete domain $\mathcal{D}$, we introduce the logic *DL-Lite*$_{core}^{(\mathcal{HF})}(\mathcal{D})$, a common extension of *DL-Lite*$_{core}^{(\mathcal{HF})}$ and *DL-Lite*$_{\mathcal{A}}$ [Artale *et al.*, 2009; Poggi *et al.*, 2008].

Syntax. Let $\mathsf{N_C}$, $\mathsf{N_R}$, $\mathsf{N_A}$, and $\mathsf{N_I}$ denote disjoint sets of *concept*, *role*, *attribute*, and *individual names*. *Roles* R and *concepts* B are defined as

$$R ::= P \mid P^- \qquad B ::= \top \mid A \mid \exists R \mid \exists U_1, \ldots, U_m.\Pi,$$

where $P \in \mathsf{N_R}$, $A \in \mathsf{N_C}$, $U_1, \ldots, U_m \in \mathsf{N_A}$, and Π is an m-ary predicate of $\mathcal{D}$. A *TBox* (or *ontology*) is a finite set of *inclusions* $X_1 \sqsubseteq X_2$, *disjointness constraints* $\mathsf{disj}(X_1, X_2)$, *functionality constraints* $\mathsf{funct}(R)$, and *attribute range constraints* $B \sqsubseteq \forall U_1, \ldots, U_m.\Pi$ where X_1 and X_2 are both either concepts, roles, or attribute names. As usual, role names occurring in functionality constraints are not allowed to occur on the right-hand side of inclusions [Artale *et al.*, 2009]. In contrast to *DL-Lite*$_{core}^{(\mathcal{HF})}$, we do not explicitly have role (a)symmetry or (ir)reflexivity axioms here; they can, however, be simulated as described in [Artale *et al.*, 2009].

An *ABox* is a finite set of *concept assertions* $A(a)$, *role assertions* $P(a, b)$, and *attribute assertions* $U(a, d)$, where $a, b \in \mathsf{N_I}$ and $d \in \Delta^{\mathcal{D}}$. A *knowledge base (KB)* $\mathcal{K} := \langle \mathcal{A}, \mathcal{T} \rangle$ consists of a TBox $\mathcal{T}$ and an ABox $\mathcal{A}$ that uses only the concept, role, and attribute names occurring in $\mathcal{T}$.

Semantics. The semantics is the standard one [Poggi *et al.*, 2008], based on *interpretations* $(\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ that assign distinct elements $a^{\mathcal{I}} \in \Delta^{\mathcal{I}}$ to all individual names, sets $C^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$ to concepts, binary relations on $\Delta^{\mathcal{I}}$ to roles, and relations $U^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{D}}$ to attribute names. For example, the interpretation of an attribute restriction $\exists U_1, U_2.\Pi$ is a set that contains all $e \in \Delta^{\mathcal{I}}$ for which there are $d_1, d_2 \in \Delta^{\mathcal{D}}$ with $(d_1, d_2) \in \Pi^{\mathcal{D}}$, $(e, d_1) \in U_1^{\mathcal{I}}$, and $(e, d_2) \in U_2^{\mathcal{I}}$. The semantics of axioms is also standard; e.g., an interpretation *satisfies* a disjointness constraint $\mathsf{disj}(X_1, X_2)$ if $X_1^{\mathcal{I}} \cap X_2^{\mathcal{I}} = \emptyset$. The *models* of a KB $\mathcal{K}$ are those interpretations satisfying all its axioms, and $\mathcal{K}$ is *consistent* if it has a model.

Other *DL-Lite* Logics. Our logic extends those from [Poggi *et al.*, 2008; Savković and Calvanese, 2012]. In fact, the missing functionality restrictions on attributes can be expressed using attribute range constraints for binary equality. On top of that, we even allow functional attributes to occur on the right-hand sides of inclusions. In contrast to [Artale *et al.*, 2012], we do not support number restriction on roles or attributes. But we can at least simulate *conjunctions* in inclusions via the concrete domain. For example, $B_1 \sqcap B_2 \sqsubseteq B_3$ can be expressed by $B_1 \sqsubseteq \exists U_1.=_d$, $\top \sqsubseteq \exists U_2.\top_{\mathcal{D}}$, $B_2 \sqsubseteq \forall U_1, U_2.=$, and $\exists U_2.=_d \sqsubseteq B_3$, where U_1, U_2 are fresh attribute names, and d is a fresh constant.

Example 3.1. The *DL-Lite*$_{core}^{(\mathcal{HF})}(\mathcal{D}_{\mathbb{Q}})$ TBox

$$\{\exists \mathsf{age}.=_{60} \sqsubseteq \exists \mathsf{maxHR}.=_{160},\ \exists \mathsf{maxHR}, \mathsf{hr}.+_5 \sqsubseteq \mathsf{Alert}\}$$

says that the maximum heart rate for persons aged 60 is 160, and that for any person an alert should be raised when the measured heart rate rises to only 5 below the maximum heart rate. A corresponding ABox contains actual data such as

$$\{\mathsf{Patient}(p_1),\ \mathsf{age}(p_1, 60),\ \mathsf{hr}(p_1, 155),\\ \mathsf{Patient}(p_2),\ \mathsf{hr}(p_2, 155),\ \mathsf{maxHR}(p_2, 180)\},$$

which implies the assertion $\mathsf{Alert}(p_1)$, but not $\mathsf{Alert}(p_2)$.

This example illustrates a prominent advantage of attribute restrictions using predicates of arity greater than 1. Here, they allow us to express an *alert* by comparing the current measurement with a maximum value. Using unary predicates, one could express hard-coded limits like $\exists \mathsf{hr}.>_{180} \sqsubseteq \mathsf{Alert}$, but not comparisons with an (age-dependent) maximum rate, unless one writes a huge (finite) case distinction.

As in OWL 2 QL, but in contrast to many *DL-Lite* dialects, we allow qualified attribute restrictions on the *left-hand side* of inclusions. This is possible without causing undecidability (as in [Baader and Hanschke, 1991; Lutz, 2002]) since they only refer to values for a single abstract domain element.

4 Conjunctive Queries with Built-ins

Let $\mathsf{N_V}$ be a set of variables, partitioned into *object variables* $\mathsf{N_{OV}}$ and *concrete domain variables* $\mathsf{N_{CV}}$. A *conjunctive query (CQ)* ϕ is of the form $(\vec{x}, \vec{v}) \leftarrow \psi(\vec{y}, \vec{w})$, where $\vec{x}, \vec{y}$ are vectors over $\mathsf{N_{OV}}$, $\vec{v}, \vec{w}$ are vectors over $\mathsf{N_{CV}}$, the variables $\vec{x}, \vec{v}$ are included in $\vec{y}, \vec{w}$, and $\psi(\vec{y}, \vec{w})$ is a conjunction of *atoms* of the forms $A(x)$ (concept atom), $R(x, y)$ (role atom), $U(x, v)$ (attribute atom), $x = y$ (object equality atom), or $\Pi(v_1, \ldots, v_m)$ (value comparison atom). In addition to variables, atoms may also contain constants from $\mathsf{N_I}$ and $\Delta^{\mathcal{D}}$ at appropriate places. The variables in $\vec{x}, \vec{v}$ are the *distinguished* variables of ϕ; all others are *nondistinguished*. A CQ is called *Boolean* if $\vec{x}$ and $\vec{v}$ are empty. We write $\alpha \in \phi$ to denote that

α is an atom occurring in the CQ ϕ. The set $\mathsf{terms}(\phi)$ consists of the elements of $\mathsf{N_I}$, $\Delta^{\mathcal{D}}$, and $\mathsf{N_V}$ occurring in ϕ.

An interpretation $\mathcal{I}$ *satisfies* a Boolean CQ ϕ ($\mathcal{I} \models \phi$) if there is a *homomorphism* of ϕ into $\mathcal{I}$, i.e., a function $\pi\colon \mathsf{terms}(\phi) \to \Delta^{\mathcal{I}} \cup \Delta^{\mathcal{D}}$ that maps object variables into $\Delta^{\mathcal{I}}$ and concrete domain variables into $\Delta^{\mathcal{D}}$, preserves the interpretations of individual names and concrete values, and satisfies all atoms of ϕ w.r.t. $\cdot^{\mathcal{I}}$ and $\cdot^{\mathcal{D}}$. A KB $\mathcal{K}$ *entails* ϕ ($\mathcal{K} \models \phi$) if every model of $\mathcal{K}$ also satisfies ϕ. A *potential answer* $\mathfrak{a}$ to a CQ $\phi\colon (\vec{x}, \vec{v}) \leftarrow \psi(\vec{y}, \vec{w})$ w.r.t. $\mathcal{K}$ maps $\vec{x}$ to individual names from $\mathcal{K}$ and $\vec{v}$ to $\Delta^{\mathcal{D}}$. A *certain answer* to ϕ w.r.t. $\mathcal{K}$ is a tuple of the form $\mathfrak{a}(\vec{x}, \vec{v})$, where $\mathfrak{a}$ is a potential answer for which $\mathcal{K}$ entails $\mathfrak{a}(\phi)\colon () \leftarrow \psi(\mathfrak{a}(\vec{y}, \vec{w}))$. The set of certain answers to ϕ w.r.t. $\mathcal{K}$ is denoted by $\mathsf{cert}(\phi, \mathcal{K})$. Similarly, for an interpretation $\mathcal{I}$, the set $\mathsf{ans}(\phi, \mathcal{I})$ contains all tuples $\mathfrak{a}(\vec{x}, \vec{v})$ where $\mathfrak{a}$ is a potential answer to ϕ w.r.t. $\mathcal{K}$ such that $\mathcal{I} \models \mathfrak{a}(\phi)$.

Rewritability. A CQ ϕ is *FO rewritable* (or, equivalently, UCQ rewritable; see [Bienvenu *et al.*, 2013]) w.r.t. a TBox $\mathcal{T}$ if there is a finite set $\Phi_{\mathcal{T}}$ of CQs such that for every consistent KB $\mathcal{K} = \langle \mathcal{A}, \mathcal{T} \rangle$ we have

$$\mathsf{cert}(\phi, \mathcal{K}) = \bigcup_{\phi' \in \Phi_{\mathcal{T}}} \mathsf{ans}(\phi', \mathcal{I}(\mathcal{A})),$$

where $\mathcal{I}(\mathcal{A})$ is the finite interpretation that satisfies exactly the assertions in $\mathcal{A}$. One can view $\mathcal{I}(\mathcal{A})$ as a (closed-world) database over which the union of the CQs in $\Phi_{\mathcal{T}}$ (called a *rewriting* of ϕ w.r.t. $\mathcal{T}$) is evaluated. The CQ ϕ is *combined rewritable* w.r.t. $\mathcal{T}$ if the above property holds with $\mathcal{I}(\mathcal{K})$ instead of $\mathcal{I}(\mathcal{A})$, where $\mathcal{I}(\mathcal{K})$ is a finite interpretation that is constructed from $\mathcal{A}$ and $\mathcal{T}$ in polynomial time. Since database queries can be evaluated in AC^0 [Abiteboul *et al.*, 1995], FO rewritability yields a data complexity in AC^0, and combined rewritability raises this to P.

Safety. We follow the approach used in databases and assume that concrete domain predicates are *built-in* predicates of the database system, i.e., their full (possibly infinite) extensions are known [Klug, 1988; Brisaboa *et al.*, 1998; Afrati *et al.*, 2006; Savković and Calvanese, 2012]. Although this means that the interpretation $\mathcal{I}(\mathcal{K})$ is not finite anymore, i.e., not a database, for so-called *domain-independent* queries it suffices to check satisfiability of $\mathcal{D}$-conjunctions, which is usually implemented by using a dedicated solver, e.g., for integer arithmetic. Domain-independence requires that the answers should not depend on the chosen domain $\Delta^{\mathcal{D}}$ of available values, but only on the values used in $\mathcal{K}$ [Abiteboul *et al.*, 1995]. To ensure this condition in our setting, we restrict ourselves to *safe* queries, as in [Savković and Calvanese, 2012; Afrati *et al.*, 2006]. A concrete domain variable v in a CQ ϕ is *safe* if it occurs in ϕ in an atom of the form

a) $U(x, v)$ for some $U \in \mathsf{N_A}$ and object variable x, or

b) $=_d(v)$ for some $d \in \Delta^{\mathcal{D}}$.

The CQ ϕ is *safe* if all its concrete domain variables are safe. A variable v that occurs in an atom $U(x, v)$ in ϕ is *bound* to x (in ϕ). Condition b) is not essential for our results, since such variables can be replaced by constants, but it is more convenient for formulating the rewriting in Section 6. Apart from ensuring domain-independence, we can show that safety is a necessary condition for combined rewritability (unless P=NP). In fact, in *DL-Lite*$_{core}^{(\mathcal{HF})}(\mathcal{D}_{\{a\}^*})$, non-safe CQs can express that an attribute value starts with letter a. Together with $=_\varepsilon$, one can thus simulate truth values by using "empty word" versus "non-empty word," which can then be employed to express the satisfaction of propositional formulas.

Lemma 4.1. *In DL-Lite*$_{core}^{(\mathcal{HF})}(\mathcal{D}_{\{a\}^*})$*, entailment of (non-safe) Boolean CQs is* CO-NP*-hard in data complexity.*

5 Canonical Models

Usually, rewritability results are proved using the notion of *canonical models* of knowledge bases. Given a KB $\mathcal{K}$, a canonical model $\mathcal{I}_{\mathcal{K}}$ is a model of $\mathcal{K}$ with the property that $\mathsf{ans}(\phi, \mathcal{I}_{\mathcal{K}}) = \mathsf{cert}(\phi, \mathcal{K})$ holds for all CQs ϕ. Unfortunately, such canonical models need not exist, even for unary $\mathcal{D}$.

Example 5.1. Consider the simple *DL-Lite*$_{core}^{(\mathcal{HF})}(\mathcal{D}_{\mathbb{Q}})$ KB $\mathcal{K} = \langle \{A(a)\}, \{A \sqsubseteq \exists U.{>_0}\} \rangle$. A canonical model $\mathcal{I}_{\mathcal{K}}$ of $\mathcal{K}$ must satisfy $(a^{\mathcal{I}_{\mathcal{K}}}, q) \in U^{\mathcal{I}_{\mathcal{K}}}$ for some $q > 0$. But then the safe Boolean CQ $\phi_q\colon () \leftarrow \exists v.U(a, v) \wedge {>_{q/2}}(v)$ is satisfied in $\mathcal{I}_{\mathcal{K}}$, but not entailed by $\mathcal{K}$.

Savković and Calvanese [2012] try to solve this problem by selecting the "most general" q that does not satisfy any $\mathcal{D}$-atoms except those implied by ${>_0}(q)$. They choose $q > 0$ such that "for any m predicates $\Pi_1, \ldots, \Pi_m$ in $\mathcal{D}_{\mathbb{Q}}$ such that $\left(\bigcup_{i=1}^m \Pi_i^{\mathcal{D}_{\mathbb{Q}}}\right) \subsetneq (>_0)^{\mathcal{D}_{\mathbb{Q}}}$ it holds that $q \notin \left(\bigcup_{i=1}^m \Pi_i^{\mathcal{D}_{\mathbb{Q}}}\right)$" [Savković and Calvanese, 2012, page 725]. For a given choice of $\Pi_1, \ldots, \Pi_m$, such a value q must exist due to *(infinitediff)*. However, regardless of the value of q, the CQ ϕ_q remains a counterexample. Hence, this construction is incorrect already for the unary predicates ${>_q}$, $q \in \mathbb{Q}$, contrary to the claim in [Savković and Calvanese, 2012, Example 2].

To overcome this problem, we weaken the requirements on the canonical model by considering only those CQs that use concrete domain predicates from a fixed, finite set of predicates. This solves the issue in Example 5.1 since there are infinitely many predicates ${>_{q/2}}$, $q \in \mathbb{Q}$, and thus not all CQs ϕ_q satisfy this restriction. For ease of presentation, we assume in the following that all CQs use only the concrete domain predicates from $\mathcal{T}$. We call such CQs *$\mathcal{T}$-restricted*. Similarly, we can assume as usual that all other symbols occurring in ϕ also occur in $\mathcal{T}$. This does not affect the complexity of query answering, but in practice restricts the kind of queries a user can ask over a given KB, which is usually fixed in advance.

Abstract Interpretations. Here we cannot describe our construction in detail, but only explain the general ideas. First, we build an *abstract* canonical model $\mathcal{I}_{\mathcal{K}}$ of the KB $\mathcal{K}$, where attributes may use variables as place-holders for actual values, and sets of $\mathcal{D}$-atoms are used to constrain the possible solutions for these variables. This is constructed using *chase rules* extending the ones in [Calvanese *et al.*, 2007], and is very similar to the *universal pre-models* in [Hernich *et al.*, 2017]. In this process, the attribute restrictions from the TBox are translated into $\mathcal{D}$-atoms over the variables occurring in $\mathcal{I}_{\mathcal{K}}$. For example, if in $\mathcal{I}_{\mathcal{K}}$ it already holds that $(e, v_1) \in U^{\mathcal{I}_{\mathcal{K}}}$ and $(e, v_2) \in V^{\mathcal{I}_{\mathcal{K}}}$, and $\top \sqsubseteq \forall U, V.\Pi$ occurs in $\mathcal{K}$, then we add the constraint $\Pi(v_1, v_2)$ to $\mathcal{I}_{\mathcal{K}}$. We denote by $\mathcal{I}_{\mathcal{A}}^*$ the initial part of this model, which is constructed by applying the chase rules only to the individual names from $\mathcal{A}$.

In the next step, we construct a *canonical solution* $f_{\mathcal{K}}$ for all variables in $\mathcal{I}_{\mathcal{K}}$, i.e., one that satisfies all constraints, but does not unnecessarily satisfy any of the "relevant" $\mathcal{D}$-atoms (defined similarly to $\mathfrak{R}_{\mathcal{T}} \cup \mathfrak{R}_{\mathcal{T},2}$ in the next section). As in [Savković and Calvanese, 2012; Artale *et al.*, 2012], the convexity of $\mathcal{D}$ is crucial for this construction, but we also need functionality here (see [Baader *et al.*, 2017] for details). In our combined rewriting approach, the finite interpretation $f_{\mathcal{K}}(\mathcal{I}^*_{\mathcal{A}})$, which is obtained from the abstract interpretation $\mathcal{I}^*_{\mathcal{A}}$ by replacing all variables by their value under $f_{\mathcal{K}}$, plays the role of $\mathcal{I}(\mathcal{K})$ in the definition of combined rewritability.

6 Rewriting CQs with Built-in Predicates

To obtain our rewriting, we extend the approach of [Calvanese *et al.*, 2007; Poggi, 2006; Savković and Calvanese, 2012]. The idea is to construct the rewriting $\Phi_{\mathcal{T}}$ of the initial CQ ϕ w.r.t. the TBox $\mathcal{T}$ by iterative application of several operators (called reduce, split, $\mathsf{infer}_{\mathcal{T}}$, and $\mathsf{infer}_{\mathcal{D}}$). Variants of the two basic operators reduce and $\mathsf{infer}_{\mathcal{T}}$ have first been used in [Calvanese *et al.*, 2007; Poggi, 2006]. The former tries to unify atoms in CQs, while the latter applies the TBox inclusions as rewrite rules. Intuitively, $A \sqsubseteq B \in \mathcal{T}$ means that any certain answer to $A(x)$ is also a certain answer to $B(x)$, and hence $A(x)$ is included in the rewriting of $B(x)$. We extend $\mathsf{infer}_{\mathcal{T}}$ to deal with attribute range restrictions, which behave similarly. A special case of this extension can be found in [Savković and Calvanese, 2012].

Two new operators deal with concrete domain predicates of higher arity. The operator split can "split" two occurrences of a concrete domain variable into separate variables, as long as they are restricted to the same value by a predicate $=_d$; this is needed for technical reasons (see [Baader *et al.*, 2017]). The operator $\mathsf{infer}_{\mathcal{D}}$ behaves like $\mathsf{infer}_{\mathcal{T}}$, but takes care of implications in the concrete domain instead of the abstract domain.

The Basic Operators. $\Phi_{\mathcal{T}}$ is the result of iteratively applying

$$\mathsf{step}(\Phi) \coloneqq \Phi \cup \mathsf{reduce}(\Phi) \cup \mathsf{split}(\Phi) \cup \mathsf{infer}_{\mathcal{T}}(\Phi) \cup \mathsf{infer}_{\mathcal{D}}(\Phi)$$

to the initial set $\{\phi\}$, until we reach a fixed-point. We define

$$\mathsf{reduce}(\Phi) \coloneqq \{\sigma(\phi') \mid \phi' \in \Phi,\ \sigma \in \mathsf{subst}(\phi')\},$$

where $\mathsf{subst}(\phi')$ contains all substitutions of variables by terms from ϕ'. The set $\mathsf{split}(\Phi)$ contains all CQs obtained from any $\phi' \in \Phi$ with $=_d(v) \in \phi'$ by replacing one other occurrence of v with a fresh variable v' and adding the atom $=_d(v')$. Before we introduce the infer operators, we want to illustrate them on an example.

Example 6.1. Consider the KB $\mathcal{K} = \langle \mathcal{A}, \mathcal{T} \rangle$ from Example 3.1 and the CQ $\phi\colon (x) \leftarrow \mathsf{Alert}(x)$ that asks for all patients with alerts. The only certain answer to ϕ w.r.t. $\mathcal{K}$ is p_1. To obtain this answer without referring to the TBox $\mathcal{T}$, we have to apply several rewriting steps.

First, $\mathsf{infer}_{\mathcal{T}}$ applies the inclusion $\exists \mathsf{maxHR}, \mathsf{hr}.{+_5} \sqsubseteq \mathsf{Alert}$ by replacing $\mathsf{Alert}(x)$ by the left-hand side of the axiom:

$$(x) \leftarrow \mathsf{maxHR}(x, v) \wedge \mathsf{hr}(x, w) \wedge {+_5}(v, w).$$

Note that the existential quantifiers in the inclusion are made explicit by introducing fresh nondistinguished variables v, w.

In $\mathcal{D}_{\mathbb{Q}}$, it holds that $=_{160}(v) \wedge =_{155}(w)$ implies ${+_5}(v, w)$. Hence, $\mathsf{infer}_{\mathcal{D}}$ replaces ${+_5}(v, w)$ by the former two atoms:

$$(x) \leftarrow \mathsf{maxHR}(x, v) \wedge =_{160}(v) \wedge \mathsf{hr}(x, w) \wedge =_{155}(w).$$

This step introduces the predicate $=_{155}$, which is not present in ϕ or $\mathcal{T}$. To avoid an infinite rewriting, we obviously have to restrict the implications that can be applied in this way.

Since the maximum heart rate is 160 for all patients that are 60 years old, we can again apply $\mathsf{infer}_{\mathcal{T}}$ to obtain

$$(x) \leftarrow \mathsf{age}(x, u) \wedge =_{60}(u) \wedge \mathsf{hr}(x, w) \wedge =_{155}(w).$$

Evaluating this query over $\mathcal{A}$ yields the expected answer p_1.

Based on this intuition, we define the operator

$$\mathsf{infer}_{\mathcal{T}}(\Phi) \coloneqq \{\sigma(\phi'') \mid \phi' \in \Phi,\ \phi' \to_{\mathcal{T}} \phi'',\ \sigma \in \mathsf{subst}(\phi'')\},$$

where the relation $\phi' \to_{\mathcal{T}} \phi''$ holds for two safe CQs ϕ', ϕ'' if one of the following cases applies:

- There exist an atom $X_2(\vec{x})$ in ϕ' and $X_1 \sqsubseteq X_2$ in $\mathcal{T}$ such that ϕ'' is obtained by replacing $X_2(\vec{x})$ with $X_1(\vec{x})$. Here, $(\exists R)(x)$ stands for $R(x, y)$, where y is a nondistinguished unique variable, and $(\exists U_1, \dots, U_m.\Pi)(x)$ abbreviates the *set* of atoms $\{U_1(x, v_1), \dots, U_m(x, v_m), \Pi(v_1, \dots, v_m)\}$, where $v_1, \dots, v_m$ are unique nondistinguished variables. We also allow that $X_2(\vec{x})$ comprises only a subset of these atoms, as long as it includes at least one attribute atom.
- There exist $\Pi(v_1, \dots, v_m)$ in ϕ' and $B \sqsubseteq \forall U_1, \dots, U_m.\Pi$ in $\mathcal{T}$ such that ϕ'' is obtained by replacing $\Pi(v_1, \dots, v_m)$ with the atoms $B(x), U_1(x, v_1), \dots, U_m(x, v_m)$, where x is an object variable of ϕ'.

As in previous rewriting algorithms, this operator does not introduce new object variables (except if they occur only once).

Concrete Domain Implications. The operator $\mathsf{infer}_{\mathcal{D}}$ is based on a similar relation $\to_{\mathcal{D}}$ on safe CQs. A first naive idea would be to define $\phi' \to_{\mathcal{D}} \phi''$ as follows:

- There is an atom $\Pi(v_1, \dots, v_m)$ in ϕ' that is implied by a $\mathcal{D}$-conjunction ψ such that ϕ'' is obtained by replacing $\Pi(v_1, \dots, v_m)$ by ψ and adding atoms $U(x, v)$ for the fresh variables v in ψ (where U occurs in $\mathcal{T}$ and x occurs in ϕ').

The new attribute atoms ensure safety of the resulting CQ. However, without further restrictions, this operation may yield an unbounded number of new atoms and variables, and hence an "infinite FO rewriting". To avoid this, we introduce a bound on the number of concrete domain variables occurring in CQs: we only allow $n_{\mathcal{T}}$ concrete domain variables bound to each object variable x, where $n_{\mathcal{T}}$ is the number of occurrences of attribute names on the right-hand side of inclusions in $\mathcal{T}$. This is becuase only such inclusions can cause new values to be created in the canonical model, and hence $n_{\mathcal{T}}$ is the maximal number of values relevant for reasoning about the concrete values of a fixed domain element. We further restrict the CQs to the set $\mathfrak{R}_{\mathcal{T}}$ of concrete domain predicates occurring in $\mathcal{T}$, and hence call a CQ *bounded* if all its value comparison atoms are of the form

(B1) $\Pi(v_1, \dots, v_m)$, where $\Pi \in \mathfrak{R}_{\mathcal{T}}$ and the variables among $v_1, \dots, v_m$ are bound to at most one object variable x. In the set of all such atoms $\Pi(v_1, \dots, v_m)$, there may occur only $n_{\mathcal{T}}$ concrete domain variables bound to the same x.

We amend the definitions of $\mathsf{infer}_{\mathcal{T}}$ and $\mathsf{infer}_{\mathcal{D}}$ by allowing only bounded CQs (e.g., new atoms $U(x, v)$ do not bind v to two different variables). Note that this restriction may be temporarily violated, as long as it can be restored by an immediate application of a substitution (see the definition of $\mathsf{infer}_{\mathcal{T}}$).

Unfortunately, this is still not enough to obtain the desired rewriting. The reason is that the initial CQ ϕ itself need not satisfy (B1). In particular, it may contain value comparison atoms whose variables are bound to different object variables. However, due to the functionality of $\mathcal{D}$, such atoms can only be implied by the TBox if all of their variables already satisfy atoms of the form $=_d(v)$. It remains to find a *finite* set of values d that are relevant in these situations. It turns out that it suffices to consider such values d that are implied by some set of atoms of the form (B1). More precisely, we collect in $\mathfrak{R}_{\mathcal{T},1}$ all predicates $=_d$ for which $=_d(v)$ is implied by a conjunction ψ using only predicates from $\mathfrak{R}_{\mathcal{T}}$ and at most $n_{\mathcal{T}}$ variables. Since $\mathcal{D}$ is polynomial and constructive, we can, for all such (finitely many) conjunctions ψ, compute a solution for each of their variables v, and check whether this is the only possible solution for v. We obtain $\mathfrak{R}_{\mathcal{T},2}$ by a similar construction, but now allowing all predicates in $\mathfrak{R}_{\mathcal{T}} \cup \mathfrak{R}_{\mathcal{T},1}$ to occur in the conjunctions ψ (see [Baader *et al.*, 2017] for details). We now relax the definition of boundedness by allowing also the following kinds of value comparison atoms:

(B2) Atoms from ϕ, possibly after applying reduce or split.

(B3) Atoms of the form $=_d(v)$, where $=_d \in \mathfrak{R}_{\mathcal{T},2}$.

In $\rightarrow_{\mathcal{D}}$, we now allow atoms of the form (B2) to be rewritten into atoms satisfying either (B1) or (B3) (possibly after applying a substitution). Atoms of the form (B3) cannot be rewritten further. This concludes the description of $\Phi_{\mathcal{T}}$.

Correctness. We can show that, if the CQ ϕ is safe, then every CQ in $\Phi_{\mathcal{T}}$ is safe and bounded, from which we obtain that $\Phi_{\mathcal{T}}$ is finite. Moreover, this rewriting is correct in the sense that, for any consistent KB $\mathcal{K} = \langle \mathcal{A}, \mathcal{T} \rangle$, we have

$$\mathsf{cert}(\phi, \langle \mathcal{A}, \mathcal{T} \rangle) = \bigcup_{\phi' \in \Phi_{\mathcal{T}}} \mathsf{ans}(\phi', \mathcal{I}^*_{\mathcal{A}}),$$

where $\mathcal{I}^*_{\mathcal{A}}$ is the finite *abstract* interpretation that can be constructed from $\mathcal{A}$ and $\mathcal{T}$ in polynomial time (see Section 5).

In order to obtain an actual combined rewriting, the last step is to replace the abstract interpretation $\mathcal{I}^*_{\mathcal{A}}$ by an ordinary finite interpretation, i.e., a database. We use the canonical solution $f_{\mathcal{K}}$ described in Section 5 to obtain the desired finite interpretation by instantiating all variables in $\mathcal{I}^*_{\mathcal{A}}$. To construct this solution, we need to solve polynomial-sized $\mathcal{D}$-conjunctions over the predicates in $\mathfrak{R}_{\mathcal{T}} \cup \mathfrak{R}_{\mathcal{T},2}$ (plus some others) and the constants in $\mathcal{A}$, which is possible in polynomial time since $\mathcal{D}$ is polynomial and constructive. Using the resulting finite interpretation $f_{\mathcal{K}}(\mathcal{I}^*_{\mathcal{A}})$ as $\mathcal{I}(\mathcal{K})$, we can show that $\Phi_{\mathcal{T}}$ satisfies the definition of combined rewritability.

Theorem 6.2. *If $\mathcal{D}$ is cr-admissible, safe and $\mathcal{T}$-restricted CQs are combined rewritable w.r.t. DL-Lite$_{core}^{(\mathcal{HF})}(\mathcal{D})$ TBoxes $\mathcal{T}$, and the rewritings are computable.*

This shows that the entailment problem for safe Boolean CQs in *DL-Lite*$_{core}^{(\mathcal{HF})}(\mathcal{D})$ is in P in data complexity. From a practical point of view, this result allows us to combine an off-line (polynomial) computation of the database $f_{\mathcal{K}}(\mathcal{I}^*_{\mathcal{A}})$ with an on-line rewriting of incoming queries.

So far, we have ignored the side condition that $\mathcal{K}$ should be consistent, since query answering over an inconsistent KB is meaningless. As usual, KB consistency can be checked by an off-line test (see [Baader *et al.*, 2017]).

Unary Concrete Domains. We again consider the special case of a unary, decidable concrete domain $\mathcal{D}$ satisfying *(infinitediff)*, as in [Savković and Calvanese, 2012]. Recall from Section 2 that adding the equality predicates does not affect *(infinitediff)*. Since neither $\mathcal{T}$ nor ϕ contain $=$, for $\rightarrow_{\mathcal{D}}$ it suffices to decide implications that do not contain this predicate; moreover, the unary predicates $=_d$ do not affect decidability. Furthermore, *(infinitediff)* implies convexity, and $\mathcal{D}$ is trivially functional. The two remaining properties of cr-admissibility, polynomiality and constructivity, are only needed to construct $\mathfrak{R}_{\mathcal{T},2}$ and the canonical solution $f_{\mathcal{K}}$. In [Baader *et al.*, 2017], we define a weaker notion of boundedness without $\mathfrak{R}_{\mathcal{T},2}$ that suffices for unary concrete domains, and show that one can directly use $\mathcal{I}(\mathcal{A})$ instead of $f_{\mathcal{K}}(\mathcal{I}^*_{\mathcal{A}})$.

Theorem 6.3. *If $\mathcal{D}$ is unary, decidable, and satisfies (infinitediff), then safe and $\mathcal{T}$-restricted CQs are FO rewritable w.r.t. DL-Lite$_{core}^{(\mathcal{HF})}(\mathcal{D})$ TBoxes $\mathcal{T}$, and the rewritings are computable.*

This extends the results of [Savković and Calvanese, 2012] to a more expressive ontology language, and adds the missing condition of $\mathcal{T}$-restrictedness.

7 Conclusion

Our combined rewritability result for CQs with built-in predicates over *DL-Lite*$_{core}^{(\mathcal{HF})}(\mathcal{D})$ ontologies establishes for the first time a polynomial data complexity for query answering w.r.t. ontologies formulated in an ontology language with n-ary concrete domains. These results subsume the ones of [Savković and Calvanese, 2012] for the case of unary concrete domains, and they are orthogonal to the results in [Hernich *et al.*, 2017]. In the latter work, the data complexity is in general CO-NP, and the authors investigate for which queries this goes down to P. Until now, our focus was on showing rewritability and complexity results. To be useful in practice, the size of the rewriting needs to be reduced, e.g., by investigating whether more concise rewritings [Kontchakov *et al.*, 2010] or alternative target languages [Rosati and Almatelli, 2010] can be employed in our setting. Instead of considering all possible implications in the concrete domain, it may also be possible to realize the operator $\mathsf{infer}_{\mathcal{D}}$ by a dedicated solving engine for the concrete domain. In addition to considering minor extensions, like allowing for concrete domain variables and predicates in the ABox as in [Lutz, 2002], we will also try to extend the language by local identification constraints (keys) [Calvanese *et al.*, 2008] and functional roles on the right-hand side of inclusions, and investigate whether FO rewritability holds in the general case.

Acknowledgments

This work was supported by DFG in the CRC 912 (HAEC) and the project BA 1122/19-1 (GoAsQ).

References

[Abiteboul *et al.*, 1995] Serge Abiteboul, Richard Hull, and Victor Vianu. *Foundations of Databases*. Addison-Wesley, 1995.

[Afrati *et al.*, 2006] Foto Afrati, Chen Li, and Prasenjit Mitra. Rewriting queries using views in the presence of arithmetic comparisons. *Theor. Comput. Sci.*, 368(1-2):88–123, 2006.

[Artale *et al.*, 2009] Alessandro Artale, Diego Calvanese, Roman Kontchakov, and Michael Zakharyaschev. The *DL-Lite* family and relations. *J. Artif. Intell. Res.*, 36:1–69, 2009.

[Artale *et al.*, 2012] Alessandro Artale, Vladislav Ryzhikov, and Roman Kontchakov. *DL-Lite* with attributes and datatypes. In *Proc. of the 20th Eur. Conf. on Artificial Intelligence (ECAI)*, pages 61–66, 2012.

[Baader and Hanschke, 1991] Franz Baader and Philipp Hanschke. A scheme for integrating concrete domains into concept languages. In *Proc. of the 12th Int. Joint Conf. on Artificial Intelligence (IJCAI)*, pages 452–457, 1991.

[Baader *et al.*, 2005] Franz Baader, Sebastian Brandt, and Carsten Lutz. Pushing the $\mathcal{EL}$ envelope. In *Proc. of the 19th Int. Joint Conf. on Artificial Intelligence (IJCAI)*, pages 364–369, 2005.

[Baader *et al.*, 2017] Franz Baader, Stefan Borgwardt, and Marcel Lippmann. Query rewriting for *DL-Lite* with n-ary concrete domains (extended version). LTCS-Report 17-04, TU Dresden, Germany, 2017. see https://lat.inf.tu-dresden.de/research/reports.html.

[Bienvenu *et al.*, 2013] Meghyn Bienvenu, Balder ten Cate, Carsten Lutz, and Frank Wolter. Ontology-based data access: A study through disjunctive datalog, CSP, and MMSNP. In *Proc. of the 32nd Symp. on Principles of Database Systems (PODS)*, pages 213–224, 2013.

[Brisaboa *et al.*, 1998] Nieves R. Brisaboa, Héctor J. Hernández, José R. Paramá, and Miguel R. Penabad. Containment of conjunctive queries with built-in predicates with variables and constants over any ordered domain. In *Proc. of the 2nd East Eur. Symp. on Advances in Databases and Information Systems (ADBIS)*, pages 46–57, 1998.

[Calvanese *et al.*, 2007] Diego Calvanese, Giuseppe De Giacomo, Domenico Lembo, Maurizio Lenzerini, and Riccardo Rosati. Tractable reasoning and efficient query answering in description logics: The *DL-Lite* family. *J. Autom. Reas.*, 39(3):385–429, 2007.

[Calvanese *et al.*, 2008] Diego Calvanese, Giuseppe De Giacomo, Domenico Lembo, Maurizio Lenzerini, and Riccardo Rosati. Path-based identification constraints in description logics. In *Proc. of the 11th Int. Conf. on Principles of Knowledge Representation and Reasoning (KR)*, pages 231–241, 2008.

[Calvanese *et al.*, 2011] Diego Calvanese, Giuseppe De Giacomo, Domenico Lembo, Maurizio Lenzerini, Antonella Poggi, Mariano Rodriguez-Muro, Riccardo Rosati, Marco Ruzzi, and Domenico Fabio Savo. The MASTRO system for ontology-based data access. *Sem. Web*, 2:43–53, 2011.

[Hernich *et al.*, 2017] André Hernich, Julio Lemos, and Frank Wolter. Query answering in DL-Lite with datatypes: A non-uniform approach. In *Proc. of the 31st AAAI Conf. on Artificial Intelligence (AAAI)*, pages 1142–1148, 2017.

[Klug, 1988] Anthony Klug. On conjunctive queries containing inequalities. *J. ACM*, 35(1):146–160, 1988.

[Kontchakov *et al.*, 2010] Roman Kontchakov, Carsten Lutz, David Toman, Frank Wolter, and Michael Zakharyaschev. The combined approach to query answering in *DL-Lite*. In *Proc. of the 12th Int. Conf. on Principles of Knowledge Representation and Reasoning (KR)*, pages 247–257, 2010.

[Kontchakov *et al.*, 2011] Roman Kontchakov, Carsten Lutz, David Toman, Frank Wolter, and Michael Zakharyaschev. The combined approach to ontology-based data access. In *Proc. of the 22nd Int. Joint Conf. on Artificial Intelligence (IJCAI)*, pages 2656–2661, 2011.

[Lutz *et al.*, 2009] Carsten Lutz, David Toman, and Frank Wolter. Conjunctive query answering in the description logic $\mathcal{EL}$ using a relational database system. In *Proc. of the 21st Int. Joint Conf. on Artificial Intelligence (IJCAI)*, pages 2070–2075, 2009.

[Lutz, 2002] Carsten Lutz. *The Complexity of Description Logics with Concrete Domains*. PhD thesis, RWTH Aachen, Germany, 2002.

[Lutz, 2003] Carsten Lutz. Description logics with concrete domains - a survey. In *Advances in Modal Logic 4 (AiML)*, pages 265–296. King's College Publications, 2003.

[Ortiz, 2013] Magdalena Ortiz. Ontology based query answering: The story so far. In *Proc. of the 7th Alberto Mendelzon Int. Workshop on Foundations of Data Management (AMW)*, 2013.

[Poggi *et al.*, 2008] Antonella Poggi, Domenico Lembo, Diego Calvanese, Giuseppe De Giacomo, Maurizio Lenzerini, and Riccardo Rosati. Linking data to ontologies. *J. Data Semant.*, X:133–173, 2008.

[Poggi, 2006] Antonella Poggi. *Structured and Semi-Structured Data Integration*. PhD thesis, Università degli Studi di Roma "La Sapienza" and Université de Paris Sud, Italy/France, 2006.

[Rosati and Almatelli, 2010] Riccardo Rosati and Alessandro Almatelli. Improving query answering over *DL-Lite* ontologies. In *Proc. of the 12th Int. Conf. on Principles of Knowledge Representation and Reasoning (KR)*, pages 290–300, 2010.

[Savković and Calvanese, 2012] Ognjen Savković and Diego Calvanese. Introducing datatypes in *DL-Lite*. In *Proc. of the 20th Eur. Conf. on Artificial Intelligence (ECAI)*, pages 720–725, 2012.

[Savković, 2011] Ognjen Savković. Managing data types in ontology-based data access. Master's thesis, Free University of Bozen-Bolzano, Italy, 2011.