24th European Conference on Artificial Intelligence - ECAI 2020
Santiago de Compostela, Spain

Ontology Focusing: Knowledge-Enriched Databases on Demand

Tomasz Gogacz[1] **Víctor Gutiérrez-Basulto**[2] **Yazmín Ibáñez-García**[3]
Filip Murlak[4] **Magdalena Ortiz**[5] and **Mantas Šimkus**[6]

Abstract. We propose a novel use of ontologies to aid the on-demand design of data-centric systems. By means of a process that we call *focusing*, a schema for a (possibly knowledge-enriched) database can be obtained semi-automatically from an existing ontology and a specification of the scope of the desired system. We formalize the inputs and outputs of focusing, and identify relevant computational problems: finding a schema via focusing, testing its consistency, and answering queries in the knowledge-enriched databases it produces. These definitions are independent from the ontology language. We then study focusing for selected description logics as ontology languages, and popular classes of queries for specifying the scope of the system. For several representative combinations, we study the decidability and complexity of the identified computational problems. As a by-product, we isolate (and solve) *mixed* variants of the classical satisfiability and entailment problems, where selected predicates are required to have finite extension, as well as the *nullability problem*, which is closely related to query emptiness.

1 Introduction

In the design of a data-centric system, coming up with an adequate organization of the data to be managed by the system is crucial. If well-chosen, the database schemas, the integrity constraints, and the conceptual models can make the implementation of the remaining functionality more evident, as they bind the developers to a shared and unambiguous view. But unfortunately, coming up with the right data organization remains challenging and time-consuming, despite the many software engineering tools and techniques that are available. Indeed, modern systems need to manage increasingly complex structure of information, and face challenges like data incompleteness, or the need for interoperability with multiple other systems [1], together making the design tasks highly non-trivial.

Ontologies, understood here as logical theories expressing domain knowledge, can be valuable tools towards facing these challenges as they are able to provide a shared understanding of the domain to multiple users and systems. Successful applications of ontologies include data integration, where they provide a unified view of heterogeneous data sources [6, 30], as well as querying incomplete data sources, where they are used *on-line* during the system operation to infer additional facts from incomplete data [42].

[1] University of Warsaw, Poland, email: t.gogacz@mimuw.edu.pl
[2] Cardiff University, UK, email: gutierrezbasultov@cardiff.ac.uk
[3] Cardiff University, UK, email: ibanezgarciay@cardiff.ac.uk
[4] University of Warsaw, Poland, email: f.murlak@mimuw.edu.pl
[5] TU Wien, Austria, email: ortiz@kr.tuwien.ac.at
[6] TU Wien, Austria, email: simkus@dbai.tuwien.ac.at

Can we also exploit the domain knowledge captured in ontologies *during the design* of data-centric systems, to come up quickly and with moderate effort with a suitable data organization for the target application? Considerable resources have been invested in the last decade into constructing high-quality ontologies for many domains [4, 19], partly motivated by their expected reusability. These ontologies already contain a good share of the domain knowledge that would typically guide the database modeling phase, for example, which are the main entities to be managed, how they relate, and which constraints they must satisfy. This knowledge is usually given in a way that is easy to reuse and independent of any implementation specificities. It then seems natural to leverage this knowledge to reduce the development cost of situation-specific applications. For example, a large disaster management ontology, describing different types of disasters and responses, may be leveraged to compile different applications for different disaster response situations. Such an ontology can be maintained by the competent authorities in some region, along with knowledge and data about the region (districts, population, facilities such as hospitals and possible shelters, etc.); in fact, similar ontologies already exist (e.g., [28]). Another motivating example can be in healthcare, where the huge existing medical ontologies (such as SNOMED CT or GALEN) could be leveraged to develop personalized health applications.

We aim at *ontology-enriched data-centric applications*, which store structured data and query it leveraging ontological knowledge. A well-known obstacle when going from an ontology to an ontology-enriched data-centric system is the *open-world* semantics of ontologies [15, 25]: data is treated as incomplete and all that is not explicitly forbidden by the ontology is considered possible. As a result, an ontology represents a (usually very large) set of possible worlds or *models*. Queries apply the so-called *certain answers* semantics, where an answer is only retrieved if all models agree on it. This semantics is often too weak, as potentially useful answers may be discarded because of some possible yet irrelevant model that, for example, adds non-existing districts to a city. Concrete data-centric applications call for some completeness assumptions that allow to trim away irrelevant models. The most popular way to achieve this is by declaring some predicates *closed*, and considering only the models of the ontology that keep the extensions of these predicates precisely as in the data.

The setting we propose supports closed predicates and, more generally, *closed queries* (CL) where the completeness assumptions are applied to the answers of a query. It also supports a closely related, complementary type of completeness assumptions that we call *fixed* queries (FIX). When we declare a query fixed, we also assume its extension is complete, but unlike for closed queries, it will not be a *dynamic* extension that changes as the system evolves. Instead, the

extension of fixed queries is static and entailed by the ontology alone, so it does not need to be maintained in the database. In other words, closed queries express completeness assumptions about the data and fixed queries express completeness assumptions about the ontology. In our disaster management example, closed queries could include the ordered evacuations, open shelters, and available drivers; for these predicates, the application is committed to keep the stored content of the predicate identical to its real-world interpretation. Fixed queries will typically correspond to 'master data': immutable aspects of reality that can be accurately captured in the ontology, like all the districts of a city. But we can also fix queries that are *not* populated by the ontology (which is quite common as most ontologies focus on *terminological* rather than *assertional* knowledge), thus forcing their extension to remain empty. In this way, fixing can be used to avoid reasoning about irrelevant predicates and make a potentially broad ontology more specific on the fly, without actually modifying it.

We propose to describe schemas of partially complete ontology-enriched databases by means of *focusing solutions*. A focusing solution $\mathcal{F}$ pairs a set of predicate symbols Σ that describe a database schema (that is, a set of predicates), with a set of assumptions on the partial completeness on the data and the ontology (closed and fixed queries). Each database instance (i.e., concrete set of facts over Σ) is then interpreted as a set of intended 'relevant' models: the subset of its classical models that do not adopt a fully open-world view, but a partial open-closed view as specified in $\mathcal{F}$.

We want to identify focusing solutions that describe data organization appropriate for a concrete situation-specific application. While some parts of this data organization may be obvious to a designer, others may not, and we want to use automated reasoning to assist the designer's choices. The designer provides a partial description of the desired system by listing some closed queries (CL) and some fixed queries (FIX), as well as some queries that will be posed at runtime and whose answers should be determined regardless of the model of the ontology (DET). For a given ontology and a possibly partial description of a system, we want to find a focusing solution that guarantees that certain queries are determined (DET), assuming that certain queries are closed (CL) and certain other queries are fixed (FIX); we may need to fix or close more queries and we may be able to guarantee that more queries are determined. This focusing solution is in general not unique; we do not tackle the problem of finding good solutions yet, but concentrate on verifying if a given candidate is indeed a focusing solution.

Let us stress that focusing is about choosing which parts of the data and ontology to declare complete. While a few recent works have considered querying ontology-enriched databases with closed predicates, it is usually assumed that the closed predicates are given. In contrast, here we are interested in deciding whether certain predicates should be closed or can be fixed to guarantee certain behavior of the relevant queries.

We can summarize the contributions of this paper as follows:

- We give a precise definition of focusing solutions. It is independent of the ontology language and its generality gives many options for specifying the scope of the system.
- We identify key computational problems relevant for obtaining and using focusing solutions. The central reasoning problem is to decide if a candidate $\mathcal{F}$ is indeed a focusing solution for a given ontology. This comprises of checking whether closed queries (CL) are compatible with fixed queries (FIX), and whether the model set induced by $\mathcal{F}$ indeed determines appropriate queries (DET) for each input database instance. Another task is the (non)emptiness problem, which corresponds to checking whether there exists at least one database that is consistent with a focusing solution. The next natural problem is entailment of various queries in the ontology-enriched databases described by a focusing solution.
- We instantiate our general notions by considering a few choices of ontology languages and scope specifications. For these combinations, we study the decidability and complexity of the introduced computational problems. We consider Description Logics (DLs), which range from the members of the *DL-Lite* family (which is at the core of industry-grade ontology-based data access systems) up to expressive DLs like $\mathcal{ALCHOIF}$ (which is closely related to OWL 2, a standard for writing ontologies). In addition, we study different query languages for specifying the queries in focusing solutions. Overall our complexity results range from tractability to undecidability.
- As a by-product of our study of focusing-related reasoning tasks, we isolate (and solve) variants of classical reasoning tasks in DLs that are interesting in their own right. For instance, we study the *mixed satisfiability* problem for description logics, which requires finding a model of an input ontology where all concept and role names from a given set have finite extensions. The usual satisfiability problem and the *finite satisfiability* problem (which is often considered in the context of applications of DLs in the database setting) are special cases of mixed satisfiability. This reasoning task can be naturally used for the static analysis of description logic knowledge bases with *closed predicates* (also known as *DBoxes*). By non-trivial adaptations of techniques from finite model reasoning in DLs, we show worst-case optimal complexity results for this problem for a variety of DLs. Other relevant problems are: *mixed entailment*, defined analogously, and *nullability*, related to query emptiness.

An extended version of this paper with proofs is available [16].

2 Preliminaries

As ontology languages we use the DL $\mathcal{ALCHOIF}$ and its fragments. Let $\mathsf{N_I}$, $\mathsf{N_C}$, and $\mathsf{N_R}$ be countably infinite sets of *individual names*, *concept names*, and *role names*. If $r \in \mathsf{N_R}$, then both r and the *inverse* r^- of r are *roles*. $\mathcal{ALCHOIF}$ *concepts* are defined as

$$C, D ::= \top \mid \bot \mid A \mid \{c\} \mid \neg C \mid C \sqcap D \mid C \sqcup D \mid \exists p.C \mid \forall p.C$$

where $A \in \mathsf{N_C}$, $c \in \mathsf{N_I}$, and p is a role; concepts of the form $\{c\}$ are called *nominals*. A *concept inclusion* is an expression of the form $C \sqsubseteq D$, where C, D are concepts. A *role inclusion* is an expression of the form $r \sqsubseteq p$, where r, p are roles. A *functionality assertion* is an expression of the form $\mathsf{func}(p)$, where p is a role. A *fact* is an expression of the form $A(c)$ or $r(c, d)$ for $A \in \mathsf{N_C}$, $r \in \mathsf{N_R}$, and $c, d \in \mathsf{N_I}$. An ($\mathcal{ALCHOIF}$) *ontology* $\mathcal{O}$ is a finite set of concept inclusions, role inclusions, functionality assertions, and facts. The DL that is obtained by disallowing functionality assertions, role inclusions, or nominals is indicated by, respectively, dropping '$\mathcal{F}$', '$\mathcal{H}$', or '$\mathcal{O}$' from its name. We define the *semantics* of ontology $\mathcal{O}$ in terms of instances. A *(database) instance* $\mathcal{I}$ is a set of facts. A *signature* Σ is any set of concept and role names. An instance $\mathcal{I}$ is *over* a signature Σ, if facts in $\mathcal{I}$ use only concepts and role names from Σ. The *active domain* of $\mathcal{I}$, denoted by $\mathsf{adom}(\mathcal{I})$, is the set of all individual names in the facts of $\mathcal{I}$. For an instance $\mathcal{I}$, we define a function $\cdot^{\mathcal{I}}$ that maps every concept C to a set $C^{\mathcal{I}} \subseteq \mathsf{N_I}$, and every role p to a relation $p^{\mathcal{I}} \subseteq \mathsf{N_I} \times \mathsf{N_I}$. For each concept name A, and role name r,

we let

$$A^{\mathcal{I}} = \{c \mid A(c) \in \mathcal{I}\}\,, \quad r^{\mathcal{I}} = \{(c,d) \mid r(c,d) \in \mathcal{I}\}\,,$$

and for the remaining concepts and roles we set

$$\top^{\mathcal{I}} = \mathsf{adom}(\mathcal{I})\,, \qquad \bot^{\mathcal{I}} = \emptyset\,, \qquad \{c\}^{\mathcal{I}} = \{c\}\,,$$
$$(C \sqcup D)^{\mathcal{I}} = C^{\mathcal{I}} \cup D^{\mathcal{I}}, \qquad (C \sqcap D)^{\mathcal{I}} = C^{\mathcal{I}} \cap D^{\mathcal{I}},$$
$$(\neg C)^{\mathcal{I}} = \mathsf{adom}(\mathcal{I}) \setminus C^{\mathcal{I}}, \quad (r^{-})^{\mathcal{I}} = \{(c,d) \mid (d,c) \in r^{\mathcal{I}}\}\,,$$
$$(\exists p.C)^{\mathcal{I}} = \{c \mid \exists d : (c,d) \in p^{\mathcal{I}} \text{ and } d \in C^{\mathcal{I}}\}\,,$$
$$(\forall p.C)^{\mathcal{I}} = \{c \mid \forall d : (c,d) \in p^{\mathcal{I}} \text{ implies } d \in C^{\mathcal{I}}\}\,.$$

We say $\mathcal{I}$ is a *model* of $\mathcal{O}$, written $\mathcal{I} \models \mathcal{O}$, if $\alpha^{\mathcal{I}} \subseteq \beta^{\mathcal{I}}$ for each concept or role inclusion $\alpha \sqsubseteq \beta \in \mathcal{O}$, $p^{\mathcal{I}}$ is a partial function whenever $\mathsf{func}(p) \in \mathcal{O}$, and $F \in \mathcal{I}$ for each fact $F \in \mathcal{O}$. Note that by defining the semantics using sets of facts, we are effectively interpreting ontologies under the *Standard Name Assumption (SNA)*. That is, the domain of interpretation is always the set $\mathsf{N_I}$, and the interpretation of individual names is given by the identity function. However, the active domain of the instance may well be a proper subset of $\mathsf{N_I}$. Note that we adopt the SNA because it is the standard assumption in databases, not because it is needed to make our setting work.

As query languages we consider variations of *conjunctive queries (CQs)*. Let $\mathcal{CQ}$ be the class of conjunctive queries (primitive positive first-order formulas) over the signature $\mathsf{N_C} \cup \mathsf{N_R}$, with the usual semantics. We also consider the class $\mathcal{AQ} \subseteq \mathcal{CQ}$ of *atomic queries*, i.e. queries that retrieve the extension of a given relation symbol in a given database. If $Q \subseteq \mathcal{AQ}$, we may view Q as a set of predicates. We also use the class $\mathcal{IQ} \subseteq \mathcal{AQ}$ of *instance queries*, that is, atomic queries over unary relation symbols. We will also discuss $\mathcal{UCQ}$, the class of *unions of conjunctive queries (UCQs)*, corresponding to positive existential first-order formulas. For an instance $\mathcal{I}$ and a query q we write $[\![\mathcal{I}]\!]_q$ for the set of tuples selected by the query q over the instance $\mathcal{I}$; if q is Boolean, $[\![\mathcal{I}]\!]_q$ is either $\{\varepsilon\}$ or $\emptyset$ with ε being the empty tuple, depending on whether q does or does not hold in $\mathcal{I}$. We need the notion of *certain answers*, which is defined as follows. Given a query q, an ontology $\mathcal{O}$ and an instance $\mathcal{I}$, we let $[\![\mathcal{O},\mathcal{I}]\!]_q$ be the intersection of $[\![\mathcal{J}]\!]_q$ with $\mathcal{J}$ ranging over all models of $\mathcal{O}$ satisfying $\mathcal{I} \subseteq \mathcal{J}$.

3 Focusing and Associated Problems

In this section, we formally define the notion of focusing solutions, and present some basic reasoning problems that are relevant for obtaining and using them. We first elaborate on the role that (CL), (FIX) and (DET) play in our framework.

(CL) **Closing databases using queries.** Assume an ontology $\mathcal{O}$ and a finite database instance $\mathcal{I}$. Under the open-word assumption, the set of (classical) models of $\mathcal{O}$ and $\mathcal{I}$ captures all that is possible, i.e., all possible worlds. In our approach, we use queries to shrink the set of models by making assertions about the completeness of parts of the data. This is is inspired by [14] and generalizes the well-known *closed predicates* or DBoxes in DLs (see, e.g., [15, 25]).

Definition 1 (Query-based Closing)**.** *For an ontology $\mathcal{O}$, a finite instance $\mathcal{I}$, and a set Q of queries, we let $\mathsf{CL}(\mathcal{O},\mathcal{I},Q)$ be the set of all (possibly infinite) instances $\mathcal{J}$ such that*

$$\mathcal{I} \subseteq \mathcal{J}\,, \quad \mathcal{J} \models \mathcal{O}\,, \quad \textit{and} \quad [\![\mathcal{I}]\!]_q = [\![\mathcal{J}]\!]_q \quad \textit{for all } q \in Q\,.$$

Intuitively, $\mathsf{CL}(\mathcal{O},\mathcal{I},Q)$ contains exactly the models of $\mathcal{O}$ and $\mathcal{I}$ that provide no new information about the queries in Q compared to the information given by $\mathcal{I}$ alone. When we close a set Q_{CL} of queries, their contents in our application will depend on the current database alone, and be independent of the used ontology. For example, we may want to keep a complete table of operating shelters, or of the responders that are licensed to drive certain type of vehicle. We do not intend to use the open-world assumption to reason about scenarios where other shelters and drivers could exist, since they do not.

Example 1. *Consider the following ontology $\mathcal{O}$, instance $\mathcal{I}$ and singleton query set Q:*

$$\mathcal{O} = \left\{ \begin{array}{r} \mathsf{Driver} \equiv \exists \mathsf{hasLicense}.\top\,, \\ \exists \mathsf{hasLicense}^{-}.\top \sqsubseteq \mathsf{License}\,, \\ \mathsf{License} \sqsubseteq \mathsf{Prof} \sqcup \mathsf{Generic} \end{array} \right\},$$

$$\mathcal{I} = \left\{ \begin{array}{rr} \mathsf{Driver}(\mathsf{Tim})\,, & \mathsf{hasLicense}(\mathsf{Tom},\mathsf{id}_1)\,, \\ \mathsf{Prof}(\mathsf{id}_2)\,, & \mathsf{hasLicense}(\mathsf{Ann},\mathsf{id}_2) \end{array} \right\},$$

$$Q = \left\{ \; \mathsf{q}(X,Y) \leftarrow \mathsf{hasLicense}(X,Y), \mathsf{Prof}(\mathsf{Y}) \; \right\}.$$

The ontology $\mathcal{O}$ says that drivers are exactly those objects that have a driving license, and that a driving license can be professional or generic. Consider the consequences of $\mathcal{O}$ and $\mathcal{I}$. We can infer that Tim *and* Tom *are drivers that have driving licenses, but due to the open-world nature of $\mathcal{O}$ we do not know if they are professional drivers. Suppose we know (have complete knowledge of) all objects possessing a professional license. This can be stated via the conjunctive query in Q, which retrieves all pairs of objects X, Y such that X has a professional license Y. The set $\mathsf{CL}(\mathcal{O},\mathcal{I},Q)$ contains only the models of $\mathcal{O}$ and $\mathcal{I}$ that are compatible with our completeness assertion, and consequently allows us to infer additional information. For example, in all structures in $\mathsf{CL}(\mathcal{O},\mathcal{I},Q)$ both* Tim *and* Tom *have generic licenses, a fact that does not follow from $\mathcal{O}$ and $\mathcal{I}$ alone.*

(FIX) **Freezing query answers.** We can also use queries to express completeness assertions about the ontology, similarly as we did for the data using (CL), again restricting the set of possible worlds.

Definition 2 (Query-based Fixing)**.** *For an ontology $\mathcal{O}$, a finite instance $\mathcal{I}$, and a set Q of queries, we let $\mathsf{FIX}(\mathcal{O},\mathcal{I},Q)$ be the set of all (possibly infinite) instances $\mathcal{J}$ such that*

$$\mathcal{I} \subseteq \mathcal{J}\,, \quad \mathcal{J} \models \mathcal{O}\,, \quad \textit{and} \quad [\![\mathcal{J}]\!]_q = [\![\mathcal{O},\emptyset]\!]_q \quad \textit{for all } q \in Q\,.$$

Intuitively, $\mathsf{FIX}(\mathcal{O},\mathcal{I},Q)$ is the set of models of $\mathcal{O}$ and $\mathcal{I}$ that provide no new information about the queries in Q compared to the information produced by $\mathcal{O}$ alone. A set Q_{FIX} of fixed queries enables us to specify "master data", or information to be frozen during the design of a system, i.e. information that is independent of database contents at runtime. In our running example it could be the districts of a city, or the details of key locations like train stations and airports.

Example 2. *Consider the following DL ontology $\mathcal{O}$:*

$$\mathsf{Hospital} \equiv \mathsf{PermHospital} \sqcup \mathsf{FieldHospital}\,,$$
$$\mathsf{PermHospital} \sqsubseteq \exists \mathsf{hasAddress}.\top\,,$$
$$\mathsf{PermHospital}(\mathsf{h}_1)\,, \quad \mathsf{Hospital}(\mathsf{h}_2)\,, \quad \mathsf{hasAddress}(\mathsf{h}_2,\mathsf{a})\,.$$

The ontology $\mathcal{O}$ says that hospitals can either be permanent hospitals or field hospitals, and that a permanent hospital must always

have an address. The three facts (which are a part of the ontology $\mathcal{O}$) say that (i) h_1 is a permanent hospital, (ii) h_2 is a hospital, and (iii) h_2 has the address a. Let q be the query $\mathsf{q}(X) \leftarrow \mathsf{Hospital}(X), \mathsf{hasAddress}(X, Y)$, which we use as a completeness statement for the ontology $\mathcal{O}$, saying that all hospitals that have an address must be known in advance, i.e., must follow from $\mathcal{O}$ alone. We have $[\![\mathcal{O}, \emptyset]\!]_q = \{\mathsf{h}_1, \mathsf{h}_2\}$. In consequence, h_1 and h_2 are frozen to be the only hospitals with an address, i.e. for any actual database instance $\mathcal{I}$, an instance in $\mathsf{FIX}(\mathcal{O}, \mathcal{I}, \{q\})$ can describe further hospitals, but they must all be field hospitals.

We can also use the fixed queries to spare our situation-specific application from reasoning about irrelevant aspects of our potentially broad ontology. If a query does not have any instances (which is common in ontologies that define many concepts but only assert facts about a few) then the effect of fixing a query is that its extension is empty in all models considered by the system. By fixing aspects of reality that are captured in the ontology but irrelevant to our application, we restrict the ontological reasoning to the specific situation.

Example 3. *Assume an ontology $\mathcal{O}'$ containing the axiom*

$$\mathsf{NaturalDisaster} \equiv \mathsf{Flood} \sqcup \mathsf{Earthquake} \sqcup \mathsf{Wildfire} \sqcup \mathsf{Hurricane}$$

If the query $\mathsf{q}(X) \leftarrow \mathsf{Flood}(X) \vee \mathsf{Wildfire}(X) \vee \mathsf{Hurricane}(X)$ gets fixed, then $\mathsf{NaturalDisaster} \equiv \mathsf{Earthquake}$ holds in each model in $\mathsf{FIX}(\mathcal{O}', \mathcal{I}, \{q\})$ for every instance $\mathcal{I}$, and our system effectively ignores the possible existence of other natural disasters.

(DET) **Determined Queries.** For an ontology $\mathcal{O}$ and an instance $\mathcal{I}$, the above points (CL) and (FIX) describe a natural way to shrink the set of models of $\mathcal{O}$ and $\mathcal{I}$. We use the notion of *determined* queries to guide this shrinking process. There may be queries Q_{DET} that are neither closed nor frozen, as they depend on both the data and the ontology, but we would still like their contents to be independent of the possible worlds (specific models). For example, we may want to use the ontology to derive the existence of certain facilities in all shelters and be sure that none other are present in any model, even though the full facilities are neither explicitly stored in the database nor hard-coded in the ontology.

Definition 3. *Assume a query q and a set M of database instances. We say q is* determined *in M, if $[\![\mathcal{J}_1]\!]_q = [\![\mathcal{J}_2]\!]_q$ holds for all $\mathcal{J}_1, \mathcal{J}_2 \in M$. A set Q of queries is* determined *in M, if every $q \in Q$ is determined in M.*

Intuitively, the above notion helps us address the ambiguity arising when a query returns different answers in different possible worlds. From the computational view-point, if a query q is determined in a set M, then computing the certain answer to q over M (i.e., the tuples that are in the answer to q in all instances in M) can be done by simply evaluating q in a arbitrarily chosen instance from M.

Example 4. *Recall Example 2. Observe that the query $\mathsf{q}(X) \leftarrow \mathsf{PermHospital}(X)$ is not determined in the models of $\mathcal{O}$ (under the usual DL semantics), but is determined in the restricted set $\mathsf{FIX}(\mathcal{O}, \mathcal{I}, Q)$, for any database instance $\mathcal{I}$.*

Focusing Configurations and Solutions. We now formally define our notion of focusing, which simply combines the ideas described in points (CL), (FIX), and (DET) above.

Definition 4 (Focusing solutions)**.** *A* focusing configuration *is a tuple*

$$\mathcal{F} = (\Sigma, Q_{\mathsf{CL}}, Q_{\mathsf{FIX}}, Q_{\mathsf{DET}}),$$

where Σ is a signature, and $Q_{\mathsf{CL}}, Q_{\mathsf{FIX}}, Q_{\mathsf{DET}}$ are sets of queries. An instance is legal *for $\mathcal{F}$ if it is a* finite *instance over Σ. For such instance $\mathcal{I}$ and an ontology $\mathcal{O}$, let*

$$\mathsf{MOD}(\mathcal{O}, \mathcal{F}, \mathcal{I}) = \mathsf{CL}(\mathcal{O}, \mathcal{I}, Q_{\mathsf{CL}}) \cap \mathsf{FIX}(\mathcal{O}, \mathcal{I}, Q_{\mathsf{FIX}}).$$

We call $\mathcal{F}$ a focusing solution *for $\mathcal{O}$, if the following conditions are satisfied* for all *instances $\mathcal{I}$ legal for $\mathcal{F}$:*

1. *if $\mathsf{CL}(\mathcal{O}, \mathcal{I}, Q_{\mathsf{CL}}) \neq \emptyset$, then $\mathsf{MOD}(\mathcal{O}, \mathcal{F}, \mathcal{I}) \neq \emptyset$;*
2. *Q_{DET} is determined in $\mathsf{MOD}(\mathcal{O}, \mathcal{F}, \mathcal{I})$.*

To understand Definition 4, assume an ontology $\mathcal{O}$ and a focusing configuration $\mathcal{F} = (\Sigma, Q_{\mathsf{CL}}, Q_{\mathsf{FIX}}, Q_{\mathsf{DET}})$. Intuitively, $\mathcal{F}$ can be seen as a description of the data that will be managed by a concrete topic-specific application derived from the ontology $\mathcal{O}$ (possibly covering a broader range of terms). In particular, such an application will consider only instances over Σ as legal instances. For any such legal instance $\mathcal{I}$, the open-world view of the target application is restricted to the set $\mathsf{MOD}(\mathcal{O}, \mathcal{F}, \mathcal{I})$, i.e. the application employs completeness assumptions specified using queries, as explained in (CL) and (FIX). For $\mathcal{F}$ to be a focusing solution, it has to behave well for all possible legal database instances. In particular, following the intuition expressed in (DET), we require that the queries in Q_{DET} are determined in $\mathsf{MOD}(\mathcal{O}, \mathcal{F}, \mathcal{I})$ for each legal $\mathcal{I}$. Finally, the point 1 in Definition 4 requires that the assertions in Q_{FIX} are compatible with the assertions in Q_{CL}, for each legal database instance. For this, each instance that is consistent with the ontology $\mathcal{O}$ and the completeness assertions in Q_{CL}, must be also consistent with $\mathcal{O}$ and all completeness assertions applied simultaneously.

Example 5. *Suppose our goal is to create an application that collects evacuation requests from people affected by an emergency. Naturally, the kind of information that needs to be collected depends heavily on the concrete circumstances, e.g., the type of an emergency, its location, the available resources, etc. Consider the following evacuation management ontology $\mathcal{O}$:*

$$\begin{aligned}
\mathsf{ER} &\sqsubseteq \exists \mathsf{requestedBy}.\mathsf{Person}\,,\\
\mathsf{ER} &\sqsubseteq \exists \mathsf{hasLoc}.\mathsf{Location}\,,\\
\mathsf{ER} &\sqsubseteq \exists \mathsf{hasEvent}.\mathsf{Evac}\,,\\
\mathsf{Evac} &\equiv \mathsf{PreEvac} \sqcup \mathsf{PostEvac}\,,\\
\mathsf{PreEvac} &\sqsubseteq \neg \mathsf{PostEvac}\,,\\
\mathsf{ER} \sqcap \exists \mathsf{hasEvent}.\mathsf{PreEvac} &\sqsubseteq \exists \mathsf{hasSeniors}.(\{\mathsf{yes}\} \sqcup \{\mathsf{no}\})\,,\\
\mathsf{ER} \sqcap \exists \mathsf{hasEvent}.\mathsf{PostEvac} &\sqsubseteq \exists \mathsf{hasInjured}.(\{\mathsf{yes}\} \sqcup \{\mathsf{no}\})\,,\\
\mathsf{hasInjured} &\sqsubseteq \mathsf{hasVulnerable}\,,\\
\mathsf{hasSeniors} &\sqsubseteq \mathsf{hasVulnerable}\,,\\
\mathsf{PriorityER} &\equiv \mathsf{ER} \sqcap \exists \mathsf{hasVulnerable}.\{\mathsf{yes}\}\,.
\end{aligned}$$

The ontology captures the following knowledge. An evacuation request (ER) must be associated to a person who requested an evacuation and a location (of such person). Moreover, each request is issued in a context of some response to an emergency. Here, a response is simply an evacuation, which can either be a preemptive evacuation (PreEvac), or an evacuation after a disaster strikes (PostEvac). An evacuation request in the context of a preemptive (resp., post-event) evacuation must have a Boolean flag to determine the presence of

senior people (resp., injured persons). A request that includes such vulnerable people is considered a high priority request. This general ontology can be focused on (at least) two different scenarios (e.g., for two different applications): handling preemptive evacuations only, or handling post-event evacuations only. This can be done using a pair of focusing configurations as follows:

$$\mathcal{F}_1 = (\Sigma, \{\mathsf{ER}, \mathsf{hasSeniors}\}, \{\mathsf{PostEvac}\}, \{\mathsf{PriorityER}\}),$$
$$\mathcal{F}_2 = (\Sigma, \{\mathsf{ER}, \mathsf{hasInjured}\}, \{\mathsf{PreEvac}\}, \{\mathsf{PriorityER}\}).$$

Here Σ can be any signature (but it is natural to omit, respectively, PreEvac *and* PostEvac *from it). It is easy to check that both $\mathcal{F}_1$ and $\mathcal{F}_2$ are focusing solutions for the above ontology. The first configuration focuses on preemptive evacuations. Since the ontology does not entail any objects in* PostEvac*, 'freezing' this query causes us to focus on preemptive evacuations. Concretely, the axiom* $\mathsf{Evac} \equiv \mathsf{PreEvac}$ *is entailed: it holds in all instances in* $\mathsf{MOD}(\mathcal{O}, \mathcal{F}_1, \mathcal{I})$ *for all finite instances $\mathcal{I}$ over Σ. In addition, $\mathcal{F}_1$ tells us to store complete information about the relations* ER *and* hasSeniors*. If we do so, the answer to the query* PriorityER *becomes determined, i.e., there is no ambiguity about evacuations requests that have a high priority. The features of $\mathcal{F}_2$ are symmetric to those of $\mathcal{F}_1$; in particular, 'freezing'* PreEvac *leads to the entailment of* $\mathsf{Evac} \equiv \mathsf{PostEvac}$*.*

Reasoning Problems. We now identify key reasoning problems crucial in obtaining and using focusing solutions. For each problem the input includes a focusing configuration $\mathcal{F}$; some problems additionally take ontologies, database instances, and queries. To be able to speak about concrete formalisms, we parameterize the problems by query and ontology languages. We write $\mathsf{O}: \mathcal{L}_{\mathsf{ONT}}$ to indicate that the language used for expressing ontologies is $\mathcal{L}_{\mathsf{ONT}}$ (in our case, a fragment of $\mathcal{ALCHOIF}$). Similarly, we use $\mathsf{C}: \mathcal{L}_{\mathsf{CL}}$, $\mathsf{F}: \mathcal{L}_{\mathsf{FIX}}$, $\mathsf{D}: \mathcal{L}_{\mathsf{DET}}$, and $\mathsf{Q}: \mathcal{L}_{\mathsf{Q}}$ for sets $\mathcal{L}_{\mathsf{CL}}, \mathcal{L}_{\mathsf{FIX}}, \mathcal{L}_{\mathsf{DET}}, \mathcal{L}_{\mathsf{Q}}$ of queries.

The main problem is recognizing focusing solutions among focusing configurations.

$\mathsf{FOCUS}(\mathsf{O}: \mathcal{L}_{\mathsf{ONT}}, \mathsf{C}: \mathcal{L}_{\mathsf{CL}}, \mathsf{F}: \mathcal{L}_{\mathsf{FIX}}, \mathsf{D}: \mathcal{L}_{\mathsf{DET}})$

Input: A pair $(\mathcal{O}, \mathcal{F})$ where $\mathcal{O} \in \mathcal{L}_{\mathsf{ONT}}$ and $\mathcal{F} = (\Sigma, Q_{\mathsf{CL}}, Q_{\mathsf{FIX}}, Q_{\mathsf{DET}})$ with $\Sigma \subseteq \mathsf{N_C} \cup \mathsf{N_R}$, $Q_{\mathsf{CL}} \subseteq \mathcal{L}_{\mathsf{CL}}$, $Q_{\mathsf{FIX}} \subseteq \mathcal{L}_{\mathsf{FIX}}$, $Q_{\mathsf{DET}} \subseteq \mathcal{L}_{\mathsf{DET}}$.

Question: Is $\mathcal{F}$ a focusing solution to $\mathcal{O}$?

Thus, in FOCUS a candidate focusing configuration is given, and the task is to decide if it is a focusing solution. Overall, we envision the following focusing process. When a need for some topic-specific application arises, a designer picks an existing ontology $\mathcal{O}$, and comes up with three (possibly empty) query sets Q^0_{CL}, Q^0_{FIX}, and Q^0_{DET}, which express a basic specification of data to be handled by the target application. The goal of a focusing reasoner is then to come up with a focusing solution $\mathcal{F} = (\Sigma, Q_{\mathsf{CL}}, Q_{\mathsf{FIX}}, Q_{\mathsf{DET}})$ that is compatible with the given specification, i.e., satisfies $Q^0_{\mathsf{CL}} \subseteq Q_{\mathsf{CL}}$, $Q^0_{\mathsf{FIX}} \subseteq Q_{\mathsf{FIX}}$, and $Q^0_{\mathsf{DET}} \subseteq Q_{\mathsf{DET}}$. The sets Q^0_{CL} and Q^0_{FIX} are extended when stronger assumptions are needed, and Q^0_{DET} is extended when stronger guarantees can be given. We do not prescribe how to produce Σ, but one obvious option is to take for Σ the set of predicates appearing in Q_{CL}. An ordinary database may be obtained by including in Σ and in Q_{CL} all terms that are known to be relevant, i.e., occur in Q^0_{DET} and Q^0_{CL}; Q_{DET} may then contain all queries over Σ and Q_{FIX} is irrelevant.

If we restrict ourselves to query languages that give us a finite number of candidates (e.g., atomic queries), FOCUS can be used directly in the search for a solution by applying exhaustive search. This search can be guided by some preferences of the designer. Note that there are many trade-offs involved that we leave for future investigation. For example, the more queries are closed, the easier it is for a query to be determined, but we must pay the maintenance costs for the queries we close. On the other hand, if suitable queries are already determined, it may be desirable to fix as much as is possible without making any instances inconsistent. A basic strategy could then be to minimize the set Q_{CL} of queries, and then maximize the set Q_{FIX}. More sophisticated strategies could involve a specified order in which the designer prefers queries to be added to Q_{CL}, reflecting, for instance, the cost of maintaining them (size statistics, availability, acquisition costs, etc). One could also consider semi-automated approaches, like a dialog approach where successive solutions are proposed to the designer, who adjusts the specification, or even the ontology, and accepts some suggested choices while rejecting others, thus converging to a satisfactory focusing solution. We leave investigating such strategies to future research.

An additional criterion that is natural in validating candidate focusing solutions is the existence of at least one database instance that is consistent with a given focusing solution. This is embodied in the following decision problem.

$\mathsf{EMPTINESS}(\mathcal{P})$

Input: A pair $(\mathcal{O}, \mathcal{F})$ where $(\mathcal{O}, \mathcal{F}) \in \mathsf{FOCUS}(\mathcal{P})$.

Question: Is $\mathsf{MOD}(\mathcal{O}, \mathcal{F}, \mathcal{I}) = \emptyset$ for each $\mathcal{I}$ legal for $\mathcal{F}$?

The symbol $\mathcal{P}$ above stands for a collection of parameters. Note that we identify the decision problem $\mathsf{FOCUS}(\mathcal{P})$ with the set of its positive instances; that is, $(\mathcal{O}, \mathcal{F}) \in \mathsf{FOCUS}(\mathcal{P})$ means that $\mathcal{O}$ and $\mathcal{F}$ are expressed in the languages specified in $\mathcal{P}$ and $\mathcal{F}$ is a focusing solution for $\mathcal{O}$. Clearly, a single consistent database instance does not guarantee that a given focusing solution is useful, but the criterion can help eliminate some utterly useless solutions. We remark that for focusing configurations of the form $\mathcal{F} = (\Sigma, Q_{\mathsf{CL}}, \emptyset, \emptyset)$ with $Q_{\mathsf{CL}} \subseteq \mathcal{AQ}$, non-EMPTINESS amounts to finding a model of $\mathcal{O}$ where each predicate from a given set (concretely, Q_{CL}) has a finite extension. The latter problem, dubbed *mixed satisfiability*, is a common generalization of the well-known satisfiability and finite satisfiability problems; it reappears in Section 4.

The previous decision problems are geared towards the system design phase (they are static analysis problems). The following two are natural on-line reasoning tasks, which assume that a satisfactory focusing solution is given. They are reminiscent of the standard satisfiability and entailment problems in logic, and thus also of the basic problems in ontology-based data access.

$\mathsf{CONSISTENCY}(\mathcal{P})$

Input: A triple $(\mathcal{O}, \mathcal{F}, \mathcal{I})$, where $\mathcal{I}$ is legal for $\mathcal{F}$ and $(\mathcal{O}, \mathcal{F}) \in \mathsf{FOCUS}(\mathcal{P})$.

Question: $\mathsf{MOD}(\mathcal{O}, \mathcal{F}, \mathcal{I}) \neq \emptyset$?

$\mathsf{ENTAILMENT}(\mathsf{Q}: \mathcal{L}_{\mathsf{Q}}, \mathcal{P})$

Input: A tuple $(\mathcal{O}, \mathcal{F}, \mathcal{I}, q)$, where $\mathcal{I}$ is legal for $\mathcal{F}$, $(\mathcal{O}, \mathcal{F}) \in \mathsf{FOCUS}(\mathcal{P})$, and $q \in \mathcal{L}_{\mathsf{Q}}$ is Boolean.

Question: Is q true in all $\mathcal{J} \in \mathsf{MOD}(\mathcal{O}, \mathcal{F}, \mathcal{I})$?

Note that CONSISTENCY is a special case of non-ENTAILMENT.

4 Concrete problems

In this section we instantiate the abstract reasoning problems with selected query and ontology languages. We discuss how these specific problems can be solved, and provide complexity results for them.

In the absence of functionality assertions, FOCUS can be solved within typical complexity bounds if completeness assertions about data are expressed using instance queries or if there are no completeness assertions about the ontology. If neither is assumed, the problem is undecidable already for a restricted DL $\mathcal{ELI}_\bot$, which allows only concept inclusions, and these must be built using only $\sqcap$ and $\exists$.

Theorem 1. *The problems*

- $\mathsf{FOCUS}(\mathsf{O}\colon \mathcal{ALCHOI}, \mathsf{C}\colon \mathcal{IQ}, \mathsf{F}\colon \mathcal{AQ}, \mathsf{D}\colon \mathcal{CQ})$,
- $\mathsf{FOCUS}(\mathsf{O}\colon \mathcal{ALCHOI}, \mathsf{C}\colon \mathcal{AQ}, \mathsf{F}\colon \emptyset, \mathsf{D}\colon \mathcal{CQ})$,

are 2ExpTime*-complete, but*

- $\mathsf{FOCUS}(\mathsf{O}\colon \mathcal{ELI}_\bot, \mathsf{C}\colon \mathcal{AQ}, \mathsf{F}\colon \mathcal{IQ}, \mathsf{D}\colon \emptyset)$

is undecidable.

Logics allowing nominals and disjunction, like $\mathcal{ALCHOI}$, are able to express fixing concepts and roles. Consequently, the conditions imposed on models by a set $Q_{\mathsf{FIX}} \subseteq \mathcal{AQ}$ of fixed queries can be compiled into the ontology. This allows to reinterpret the two points in Definition 4 as natural variants of two classical problems in DLs. Point 1 can be cast as the following NULLABILITY problem.

$\mathsf{NULLABILITY}(\mathsf{O}\colon \mathcal{L}_{\mathsf{ONT}}, \mathsf{C}\colon \mathcal{L}_{\mathsf{CL}}, \mathsf{Q}\colon \mathcal{L}_{\mathsf{Q}})$	
Input:	A tuple $(\mathcal{O}, \Sigma, Q_{\mathsf{CL}}, q)$ with $\mathcal{O} \in \mathcal{L}_{\mathsf{ONT}}$, $\Sigma \subseteq \mathsf{N_C} \cup \mathsf{N_R}$, $Q_{\mathsf{CL}} \subseteq \mathcal{L}_{\mathsf{CL}}$, and $q \in \mathcal{L}_{\mathsf{Q}}$.
Question:	Do all $\mathcal{I}$ over Σ with $\mathsf{CL}(\mathcal{O}, \mathcal{I}, Q_{\mathsf{CL}}) \neq \emptyset$ admit $\mathcal{J} \in \mathsf{CL}(\mathcal{O}, \mathcal{I}, Q_{\mathsf{CL}})$ with $[\![\mathcal{J}]\!]_q = \emptyset$?

NULLABILITY is closely related to the query emptiness problem, which is known to be NExpTime-complete for atomic queries, $\mathcal{ALCI}$ ontologies, and no closed predicates [3]. We show that $\mathsf{NULLABILITY}(\mathsf{O}\colon \mathcal{ALCHOI}, \mathsf{C}\colon \mathcal{IQ}, \mathsf{Q}\colon \mathcal{UCQ})$ is in 2ExpTime; this is done by reduction to query entailment for $\mathcal{ALCHOI}$ with an exponential instance [29]. Allowing closed roles ($\mathsf{C}\colon \mathcal{AQ}$) quickly leads to undecidability, which carries over to FOCUS. 2ExpTime-hardness carries over from query entailment for $\mathcal{ALCO}$ [27]. Point 2 in Definition 4 boils down to a variant of the classical query entailment problem, dubbed MIXED-ENTAILMENT, that only considers models where *selected* predicates have finite extensions.

$\mathsf{MIXED\text{-}ENTAILMENT}(\mathsf{O}\colon \mathcal{L}_{\mathsf{ONT}}, \mathsf{Q}\colon \mathcal{L}_{\mathsf{Q}})$	
Input:	A tuple $(\mathcal{O}, \Sigma, \mathcal{I}, q)$ with $\mathcal{O} \in \mathcal{L}_{\mathsf{ONT}}$, $\Sigma \subseteq \mathsf{N_C} \cup \mathsf{N_R}$, $\mathcal{I}$ an instance, and $q \in \mathcal{L}_{\mathsf{Q}}$ Boolean.
Question:	Does q hold in each model $\mathcal{J}$ of $\mathcal{O}$ such that $\mathcal{I} \subseteq \mathcal{J}$ and $R^{\mathcal{J}}$ is finite for all $R \in \Sigma$?

MIXED-ENTAILMENT generalizes both finite and unrestricted entailment, but for logics enjoying finite controllability, like $\mathcal{ALCHOI}$, the three variants coincide, which makes MIXED-ENTAILMENT 2ExpTime-complete [10] and establishes the lower bounds for FOCUS.

For the problem of entailment with focusing solutions, we handle fixed atomic queries and closed conjunctive queries.

Theorem 2. *The problem*

- $\mathsf{ENTAILMENT}(\mathsf{O}\colon \mathcal{ALCHOI}, \mathsf{C}\colon \mathcal{CQ}, \mathsf{F}\colon \mathcal{AQ}, \mathsf{Q}\colon \mathcal{CQ})$

is 2ExpTime*-complete in combined complexity, and* coNP*-complete in data complexity.*

For combined complexity, the lower bound carries over from CQ entailment in $\mathcal{ALCI}$ [23] and the upper bound is obtained by a careful reduction to UCQ entailment for $\mathcal{ALCHOI}$ [10]. For data complexity, the lower bound carries over from IQ entailment in a sublogic of $\mathcal{ALCHOI}$ [37]. For the upper bound, the approach is similar to that used in [13, 25], where a guess-and-check algorithm is used to find concise representations of tree-like counter-models.

The emptiness problem can be tackled for full $\mathcal{ALCHOIF}$, if all completeness assertions are expressed using atomic queries.

Theorem 3. *The problem*

- $\mathsf{EMPTINESS}(\mathsf{O}\colon \mathcal{ALCHOIF}, \mathsf{C}\colon \mathcal{AQ}, \mathsf{F}\colon \mathcal{AQ}, \mathsf{D}\colon \mathcal{CQ})$

is in $\mathrm{PTIME}^{\mathrm{NEXPTIME}}$,

- $\mathsf{EMPTINESS}(\mathsf{O}\colon \mathcal{ALCHOIF}, \mathsf{C}\colon \mathcal{AQ}, \mathsf{F}\colon \emptyset, \mathsf{D}\colon \mathcal{CQ})$

is coNExpTime*-complete, and for ontologies without facts*

- $\mathsf{EMPTINESS}(\mathsf{O}\colon \mathcal{ALCHIF}, \mathsf{C}\colon \mathcal{AQ}, \mathsf{F}\colon \emptyset, \mathsf{D}\colon \mathcal{CQ})$

is ExpTime*-complete.*

The two lower bounds above are inherited from general satisfiability in $\mathcal{ALCHOIF}$ [41] and $\mathcal{ALCHIF}$ [38]. For the upper bounds, we cast EMPTINESS as the dual of a variant of the classical satisfiability problem, dubbed MIXED-SAT, where one looks for a model in which selected predicates have finite extension.

$\mathsf{MIXED\text{-}SAT}(\mathcal{L}_{\mathsf{ONT}})$	
Input:	A pair $(\mathcal{O}, \Sigma)$ with $\mathcal{O} \in \mathcal{L}_{\mathsf{ONT}}$, $\Sigma \subseteq \mathsf{N_C} \cup \mathsf{N_R}$.
Question:	Is there a model $\mathcal{J}$ of $\mathcal{O}$ such that $R^{\mathcal{J}}$ is finite for all $R \in \Sigma$?

We solve this problem by adapting a method previously used for finite and unrestricted satisfiability [24, 31, 32]. This involves a reduction to a generalization of the integer programming problem, which, as we argue, can be solved algorithmically and yields worst-case optimal upper bounds for $\mathcal{ALCHOIF}$ and $\mathcal{ALCHIF}$. In the end, we can show that $\mathsf{MIXED\text{-}SAT}(\mathsf{O}\colon \mathcal{ALCHOIF})$ is NExpTime-complete, while $\mathsf{MIXED\text{-}SAT}(\mathsf{O}\colon \mathcal{ALCHIF})$ is ExpTime-complete. This shows that, in terms of computational complexity, mixed satisfiability in the above two DLs is computationally not more expensive than ordinary or finite satisfiability. We remark that this problem can be naturally seen as a static analysis problem for DL knowledge bases with closed predicates. Following the terminology of [25], let us consider a pair $(\mathcal{T}, \Sigma_C)$ of a DL TBox $\mathcal{T}$ and a set Σ_C of closed predicates. A natural static analysis problem is to test if there is at least one *finite* ABox that is consistent with $(\mathcal{T}, \Sigma_C)$, i.e., an ABox $\mathcal{A}$ such that $\mathcal{T}$ and $\mathcal{A}$ have a model that respects the closed predicates in Σ_C. This problem is non-trivial for DLs that do not have the finite model property (like $\mathcal{ALCHOIF}$ or $\mathcal{ALCHIF}$), and is equivalent to checking whether $(\mathcal{T}, \Sigma_C)$ is a positive instance of MIXED-SAT.

The case of *DL-Lite*. We now turn our attention to the *DL-Lite* family of DLs. We look at three members of this family, the first and the third of which do not enjoy the finite model property (i.e. satisfiability does not imply the existence of a *finite* model). A $\mathit{DL\text{-}Lite}_{\mathrm{Bool}}^{\mathcal{HOF}}$ ontology contains only statements of the forms

$$A_1 \sqcap \ldots \sqcap A_k \sqsubseteq B_1 \sqcup \ldots \sqcup B_m\,,$$
$$A_1 \sqsubseteq \exists r.\top\,, \quad \top \sqsubseteq \forall r.B_1\,, \quad r \sqsubseteq s\,, \quad \mathsf{func}(r)\,,$$

where $k, m \geq 1$, A_i and B_j are $\top$, $\bot$, nominals or concept names, r, s are roles, and functional roles have no subroles: $\mathsf{func}(s) \in \mathcal{O}$ and $r \sqsubseteq s \in \mathcal{O}$ imply $r = s$. The DL $DL\text{-}Lite_{\text{Bool}}^{\mathcal{O}}$ is obtained by additionally prohibiting role inclusions and functionality assertions. The DL $DL\text{-}Lite^{\mathcal{HF}}$ is obtained from $DL\text{-}Lite_{\text{Bool}}^{\mathcal{HOF}}$ by prohibiting nominals, as well as $\sqcup$ and $\sqcap$; that is, all concept inclusions are of the form $A \sqsubseteq B$, $A \sqsubseteq \exists r.\top$, or $\top \sqsubseteq \forall r.B$. We have

$$DL\text{-}Lite^{\mathcal{HF}},\ DL\text{-}Lite_{\text{Bool}}^{\mathcal{O}} \subseteq DL\text{-}Lite_{\text{Bool}}^{\mathcal{HOF}} \subseteq \mathcal{ALCHOIF}\,.$$

For $DL\text{-}Lite_{\text{Bool}}^{\mathcal{O}}$ we regain decidability of FOCUS with completeness assertions about both the data and the ontology given by atomic queries (cf. Theorem 1).

Theorem 4. *The problem*
- $\mathsf{FOCUS}(\mathsf{O}\colon DL\text{-}Lite_{\text{Bool}}^{\mathcal{O}}, \mathsf{C}\colon \mathcal{AQ}, \mathsf{F}\colon \mathcal{AQ}, \mathsf{D}\colon \mathcal{CQ})$

is in 2ExpTime.

The proof essentially boils down to a reduction to $\mathsf{FOCUS}(\mathsf{O}\colon \mathcal{ALCOI}, \mathsf{C}\colon \mathcal{IQ}, \mathsf{F}\colon \mathcal{AQ}, \mathsf{D}\colon \mathcal{CQ})$, relying on the syntactic restrictions of $DL\text{-}Lite$.

The same restrictions open the way to different algorithms and lower complexity for EMPTINESS.

Theorem 5. *The problem*
- $\mathsf{EMPTINESS}(\mathsf{O}\colon DL\text{-}Lite_{\text{Bool}}^{\mathcal{HOF}}, \mathsf{C}\colon \mathcal{AQ}, \mathsf{F}\colon \mathcal{AQ}, \mathsf{D}\colon \mathcal{CQ})$

is in PTime^{NP},
- $\mathsf{EMPTINESS}(\mathsf{O}\colon DL\text{-}Lite_{\text{Bool}}^{\mathcal{HOF}}, \mathsf{C}\colon \mathcal{AQ}, \mathsf{F}\colon \emptyset, \mathsf{D}\colon \mathcal{CQ})$

is coNP*-complete, and*
- $\mathsf{EMPTINESS}(\mathsf{O}\colon DL\text{-}Lite^{\mathcal{HF}}, \mathsf{C}\colon \mathcal{AQ}, \mathsf{F}\colon \emptyset, \mathsf{D}\colon \mathcal{CQ})$

is PTime*-complete.*

The upper bounds for $\mathsf{O}\colon DL\text{-}Lite_{\text{Bool}}^{\mathcal{HOF}}$ above are obtained by refining our reduction to generalized integer programming and limiting the number of variables with non-zero values [12, 33]; the lower bound is inherited from satisfiability in propositional logic. As a side result we obtain that $\mathsf{MIXED\text{-}SAT}(\mathsf{O}\colon DL\text{-}Lite_{\text{Bool}}^{\mathcal{HOF}})$ is NP-complete. The upper bound for $DL\text{-}Lite^{\mathcal{HF}}$ can be obtained by applying the *cycle reversion* technique [35]. In particular, mixed satisfiability in $DL\text{-}Lite^{\mathcal{HF}}$ can be reduced in polynomial time to ordinary satisfiability, which is known to be tractable.

5 Related Work

The focusing framework relates to multiple notions and problems investigated within the KR&R and database theory fields: *module extraction*, *closed predicates*, *implicit definability*, *query emptiness*, and *finite model reasoning*.

In the context of *module extraction*, roughly speaking, a fragment $\mathcal{O}_1$ of an ontology $\mathcal{O}_2$ is called a *module* if the terms defined in $\mathcal{O}_1$ have precisely the meaning they have in $\mathcal{O}_2$. Further, one usually requires that $\mathcal{O}_1$ and $\mathcal{O}_2$ agree on the entailment of inclusions over a given signature Σ. In this way, an application whose scope is limited to the entities in Σ can safely use $\mathcal{O}_1$ instead of the full $\mathcal{O}_2$; see e.g. [9, 11, 18, 21, 22, 34, 39] and references therein for more details. Unlike module extraction, focusing will in general change the meaning of terms (see Example 5, where the equivalence $\mathsf{Evac} \equiv \mathsf{PreEvac}$ is implied by first focusing solution, but not by the initial ontology).

Closed predicates (aka *DBoxes*) are used in DLs to combine closed-world and open-world reasoning by declaring certain predicates as closed and therefore forcing them to close their extensions [15, 25]. Definition 1 thus generalizes the idea of closed predicates in DLs. Closing extensions of predicates is similar in spirit to *circumscription* [26], but instead of minimizing the inference of new tuples in selected predicates, such inferences are prohibited altogether. Circumscription has been studied both for expressive and lightweight DLs [7, 8]. Note that closed predicates, or other kinds of statements to assert information completeness have also been studied in databases (see, e.g., [2, 5, 14]). In particular, Fan and Geerts [14] study completeness assertions made using queries, and explored reasoning about databases and queries in their presence.

When $Q_{\mathsf{FIX}} = \emptyset$ and $Q_{\mathsf{CL}} \subseteq \mathcal{AQ}$, the determinacy condition in Definition 4 can be cast as mixed *implicit definability*: each query in Q_{DET} must be implicitly definable over Q_{CL} under $\mathcal{O}$ over models where predicates from Σ have finite extensions. The reduction of mixed implicit definability to mixed query entailment, underlying a part of the proof of Theorem 1, is the same as in the finite or unrestricted case [40].

Our nullability problem is related to the *query emptiness* and *schema-level positive query implication* ($\exists$PQI) problems [3, 5]. For Boolean queries, these three problems effectively collapse. For non-Boolean queries, there is a slight divergence. Consider the ontology $\mathcal{O} = \{\{c\} \sqsubseteq \exists r.A\}$, the instance query $A(x)$, and suppose the signature of legal databases is $\Sigma = \emptyset$. In the sense of [3, 5], this yields a positive instance, i.e., there is a database (the empty database) in which the certain answer to $A(x)$ is empty. In our setting, this is a negative instance of the nullability problem (the extension of A always has an element, still we cannot identify it via a constant). Note that the undecidability result for nullability with $\mathcal{ELI}_\bot$ ontologies already holds for Boolean CQs of the form $\exists x\, A(x)$. This result thus contributes to the research on the $\exists$PQI problem since $\mathcal{ELI}_\bot$ can be translated into guarded TGDs without constants but extended with constraints, providing a new class of constraints for which $\exists$PQI is undecidable.

Related to mixed satisfiability are the works on finite model reasoning in DLs and fragments of first-order logic [17, 20, 24, 31, 33, 35, 36]. In particular, we adapt and extend the inequations-based technique from [24, 31] to establish our upper bounds.

6 Discussion

In this paper, we have introduced *focusing*, which makes it possible to reuse the knowledge in an ontology as a basis for the on-demand design of data-centric applications. We have isolated relevant computational problems and provided complexity and undecidability results for selected combinations of description logics and query languages. These results are not meant to paint a complexity landscape, or to identify the best formalisms to be used for focusing. Rather, they constitute a preliminary study of the potential and the limits of the focusing framework. From here one could go in several directions.

One is to push the decidability results towards more and more expressive logics, for instance, involving transitive roles or counting restrictions. A more practical goal is to find languages for which all key problems are decidable with reasonable complexity bounds. Here, a natural strategy would be to try to pinpoint the complexity of FOCUS and ENTAILMENT for logics from the $DL\text{-}Lite$ family.

A more open-ended goal is finding the best focusing solutions. This calls for suitable ways to capture the designer's preferences and to derive additional queries that may be closed or fixed. A related challenge is to understand how completeness assertions can be used to optimize queries or to alleviate the complexity of reasoning by ensuring that the specific ontology is expressible in a simpler logic. E.g., by freezing a predicate in every disjunction one could make the ontology effectively Horn. Are there ways to decide whether this is the case, and if so, can this be leveraged by algorithms?

Acknowledgements

Gogacz and Murlak were funded by Poland's National Science Centre grant 2018/30/E/ST6/00042. This work was also supported by the Vienna Business Agency and the Austrian Science Fund (FWF) projects P30360 and P30873.

REFERENCES

[1] Serge Abiteboul, Marcelo Arenas, Pablo Barceló, Meghyn Bienvenu, Diego Calvanese, Claire David, Richard Hull, Eyke Hüllermeier, Benny Kimelfeld, Leonid Libkin, Wim Martens, Tova Milo, Filip Murlak, Frank Neven, Magdalena Ortiz, Thomas Schwentick, Julia Stoyanovich, Jianwen Su, Dan Suciu, Victor Vianu, and Ke Yi, 'Research Directions for Principles of Data Management', *Dagstuhl Manifestos*, **7**(1), (2018).

[2] Serge Abiteboul and Oliver M. Duschka, 'Complexity of answering queries using materialized views', in *Proc. PODS'98*, pp. 254–263. ACM, (1998).

[3] Franz Baader, Meghyn Bienvenu, Carsten Lutz, and Frank Wolter, 'Query and predicate emptiness in ontology-based data access', *J. Artif. Intell. Res. (JAIR)*, **56**, 1–59, (2016).

[4] Franz Baader, Ian Horrocks, Carsten Lutz, and Ulrike Sattler, *An Introduction to Description Logic*, Cambridge Uni. Press, 2017.

[5] Michael Benedikt, Pierre Bourhis, Balder ten Cate, and Gabriele Puppis, 'Querying visible and invisible information', in *Proc. LICS 2016*, pp. 297–306. ACM, (2016).

[6] Michael Benedikt, Bernardo Cuenca Grau, and Egor V. Kostylev, 'Logical foundations of information disclosure in ontology-based data integration', *Artif. Intell.*, **262**, 52–95, (2018).

[7] Piero A. Bonatti, Marco Faella, and Luigi Sauro, 'Defeasible inclusions in low-complexity DLs', *J. Artif. Intell. Res. (JAIR)*, **42**, 719–764, (2011).

[8] Piero A. Bonatti, Carsten Lutz, and Frank Wolter, 'The complexity of circumscription in DLs', *J. Artif. Intell. Res. (JAIR)*, **35**, 717–773, (2009).

[9] Elena Botoeva, Boris Konev, Carsten Lutz, Vladislav Ryzhikov, Frank Wolter, and Michael Zakharyaschev, 'Inseparability and conservative extensions of description logic ontologies: A survey', in *Reasoning Web*, (2016).

[10] Diego Calvanese, Thomas Eiter, and Magdalena Ortiz, 'Regular path queries in expressive description logics with nominals', in *Proc. IJCAI 2009*, pp. 714–720, (2009).

[11] Chiara Del Vescovo, Bijan Parsia, Uli Sattler, and Thomas Schneider, 'The modular structure of an ontology: Atomic decomposition', in *Proc. IJCAI 2011*, (2011).

[12] Friedrich Eisenbrand and Gennady Shmonin, 'Carathéodory bounds for integer cones', *Oper. Res. Lett.*, **34**(5), 564–568, (2006).

[13] Thomas Eiter, Magdalena Ortiz, and Mantas Šimkus, 'Conjunctive query answering in the description logic SH using knots', *J. Comput. Syst. Sci.*, **78**(1), 47–85, (2012).

[14] Wenfei Fan and Floris Geerts, 'Relative information completeness', *ACM Trans. Database Syst.*, **35**(4), 27:1–27:44, (October 2010).

[15] Enrico Franconi, Yazmin Ibáñez-García, and Inanç Seylan, 'Query answering with DBoxes is hard', *Electr. Notes Theor. Comput. Sci.*, **278**, 71–84, (2011).

[16] Tomasz Gogacz, Víctor Gutiérrez-Basulto, Yazmin Angélica Ibáñez-García, Filip Murlak, Magdalena Ortiz, and Mantas Simkus, 'Ontology focusing: Knowledge-enriched databases on demand', *CoRR*, **abs/1904.00195**, (2019).

[17] Tomasz Gogacz, Yazmin Ibáñez-García, and Filip Murlak, 'Finite query answering in expressive description logics with transitive roles', in *Proc. KR-18*, pp. 369–378, (2018).

[18] Bernardo Cuenca Grau, Ian Horrocks, Yevgeny Kazakov, and Ulrike Sattler, 'Modular reuse of ontologies: Theory and practice', *J. Artif. Int. Res.*, **31**(1), 273–318, (February 2008).

[19] Ian Horrocks, 'Ontologies and the semantic web', *Commun. ACM*, **51**(12), 58–67, (2008).

[20] Yazmín Ibáñez-García, Carsten Lutz, and Thomas Schneider, 'Finite model reasoning in Horn description logics', in *Proc. KR-14*, (2014).

[21] Boris Konev, Carsten Lutz, Dirk Walther, and Frank Wolter, 'Model-theoretic inseparability and modularity of description logic ontologies', *Artif. Intell.*, **203**, 66–103, (2013).

[22] Roman Kontchakov, Frank Wolter, and Michael Zakharyaschev, 'Logic-based ontology comparison and module extraction, with an application to DL-Lite', *Artif. Intell.*, **174**(15), 1093–1141, (2010).

[23] Carsten Lutz, 'The complexity of conjunctive query answering in expressive description logics', in *Proc. IJCAR-08*, (2008).

[24] Carsten Lutz, Ulrike Sattler, and Lidia Tendera, 'The complexity of finite model reasoning in description logics', *Inf. Comput.*, **199**(1-2), 132–171, (2005).

[25] Carsten Lutz, Inanc Seylan, and Frank Wolter, 'The data complexity of ontology-mediated queries with closed predicates', *Logical Methods in Computer Science*, **Volume 15, Issue 3**, (August 2019).

[26] John McCarthy, 'Circumscription - A form of non-monotonic reasoning', *Artif. Intell.*, **13**(1-2), 27–39, (1980).

[27] Nhung Ngo, Magdalena Ortiz, and Mantas Šimkus, 'Closed predicates in description logics: Results on combined complexity', in *Proc. KR 2016*, pp. 237–246. AAAI Press, (2016).

[28] OpenSensingCity Project. Smart City Artifacts, 2014. Accessed in February 2020 at http://opensensingcity.emse.fr/scans/.

[29] Magdalena Ortiz, Diego Calvanese, and Thomas Eiter, 'Data complexity of query answering in expressive description logics via tableaux', *J. Autom. Reasoning*, **41**(1), 61–98, (2008).

[30] Antonella Poggi, Domenico Lembo, Diego Calvanese, Giuseppe De Giacomo, Maurizio Lenzerini, and Riccardo Rosati, 'Linking data to ontologies', *J. Data Semantics*, **10**, 133–173, (2008).

[31] Ian Pratt-Hartmann, 'Complexity of the two-variable fragment with counting quantifiers', *J. Logic Lang. Inform.*, **14**(3), 369–395, (2005).

[32] Ian Pratt-Hartmann, 'Complexity of the guarded two-variable fragment with counting quantifiers', *J. Log. Comput.*, **17**(1), 133–155, (2007).

[33] Ian Pratt-Hartmann, 'On the computational complexity of the numerically definite syllogistic and related logics', *Bulletin of Symbolic Logic*, **14**(1), 1–28, (2008).

[34] Ana Armas Romero, Mark Kaminski, Bernardo Cuenca Grau, and Ian Horrocks, 'Module extraction in expressive ontology languages via datalog reasoning', *J. Artif. Intell. Res.*, **55**, 499–564, (2016).

[35] Riccardo Rosati, 'Finite model reasoning in DL-Lite', in *Proc. ESWC 2008*, volume 5021, pp. 215–229, (2008).

[36] Sebastian Rudolph, 'Undecidability results for database-inspired reasoning problems in very expressive description logics', in *Proc. KR-16*, pp. 247–257, (2016).

[37] Andrea Schaerf, 'On the complexity of the instance checking problem in concept languages with existential quantification', *J. Intell. Inf. Syst.*, **2**(3), 265–278, (1993).

[38] Klaus Schild, 'A correspondence theory for terminological logics: Preliminary report', in *Proc. IJCAI 1991*, pp. 466–471. Morgan Kaufmann, (1991).

[39] Heiner Stuckenschmidt, Christine Parent, and Stefano Spaccapietra, *Modular Ontologies: Concepts, Theories and Techniques for Knowledge Modularization*, Springer, 2009.

[40] Balder ten Cate, Enrico Franconi, and İnanç Seylan, 'Beth definability in expressive description logics', *J. Artif. Int. Res.*, **48**(1), 347–414, (October 2013).

[41] Stephan Tobies, 'The complexity of reasoning with cardinality restrictions and nominals in expressive description logics', *J. Artif. Intell. Res. (JAIR)*, **12**, 199–217, (2000).

[42] Guohui Xiao, Diego Calvanese, Roman Kontchakov, Domenico Lembo, Antonella Poggi, Riccardo Rosati, and Michael Zakharyaschev, 'Ontology-based data access: A survey', in *Proc. IJCAI-18*, (2018).