OWL2QL Meta-reasoning Using ASP-based Hybrid Knowledge Bases

Haya Majid Qureshi[0000−0002−6622−4950] and Wolfgang Faber[0000−0002−0330−5868]

University of Klagenfurt, Austria
{haya.qureshi,wolfgang.faber}@aau.at

Abstract. Metamodeling refers to scenarios in ontologies in which classes and roles can be members of classes or occur in roles. This is a desirable modelling feature in several applications, but allowing it without restrictions is problematic for several reasons, mainly because it causes undecidability. Therefore, practical languages either forbid metamodeling explicitly or treat occurrences of classes as instances to be semantically different from other occurrences, thereby not allowing metamodeling semantically. Several extensions have been proposed to provide metamodeling to some extent. Building on earlier work that reduces metamodeling query answering to Datalog query answering, recently reductions to query answering over hybrid knowledge bases were proposed with the aim of using the Datalog transformation only where necessary. Experiments with hybrid reasoners showed that these methods are viable in practice, but performance was not as good as one could have hoped for. In this work we show that using a suitable plugin for DLV2, rather than a dedicated hybrid knowledge base, can achieve much better performance.

Keywords: Ontology · Metamodeling · Rules · SPARQL.

1 Introduction

Metamodeling helps in specifying conceptual modelling requirements with the notion of meta-classes (for instance, classes that are instances of other classes) and meta-properties (relations between meta-concepts). These notions can be expressed in OWL Full. However, OWL Full is so expressive for metamodeling that it leads to undecidability [13]. OWL 2 DL and its sub-profiles guarantee decidability, but they provide a very restricted form of metamodeling [7] and give no semantic support due to the prevalent Direct Semantics (DS).

Consider an example adapted from [6], concerning the modeling of biological species, stating that all GoldenEagles are Eagles, all Eagles are Birds, and Harry is an instance of GoldenEagle, which further can be inferred as an instance of Eagle and Birds. However, in the species domain one can not just express properties of and relationships among species, but also express properties of the species themselves. For example "GoldenEagle is listed in the IUCN Red List of endangered species" states that GoldenEagle as a whole class is an

endangered species. Note that this is also not a subclass relation, as Harry is not an endangered species. To formally model this expression, we can declare GoldenEagle to be an instance of new class EndangeredSpecies.

$$\textit{Eagle} \sqsubseteq \textit{Birds}, \quad \textit{GoldenEagle} \sqsubseteq \textit{Eagle}, \quad \textit{GoldenEagle(Harry)}$$
$$\textit{EndangeredSpecies} \sqsubseteq \textit{Species}, \; \textit{EndangeredSpecies(GoldenEagle)}$$

Note that the two occurrences of the IRI for GoldenEagle (in a class position and in an individual position) are treated as different objects in the standard direct semantics *DS*[1], therefore not giving semantic support to punned[2] entities and treating them as independent of each other by reasoners. These restrictions significantly limit meta-querying as well, as the underlying semantics for SPARQL queries over OWL 2 QL is defined by the *Direct Semantic Entailment Regime* [5], which uses *DS*.

To remedy the limitation of metamodeling, Higher-Order Semantics (HOS) was introduced in [10] for OWL 2 QL ontologies and later referred to as Metamodeling Semantics (MS) in [11], which is the terminology that we will adopt in this paper. The interpretation structure of HOS follows the Hilog-style semantics of [1], which allows the elements in the domain to have polymorphic characteristics. Furthermore, to remedy the limitation of metaquerying, the Metamodeling Semantics Entailment Regime (MSER) was proposed in [2], which does allow meta-modeling and meta-querying using SPARQL by reduction from query-answering over OWL 2 QL to Datalog queries.

In [15] several methods were proposed that reduce query-answering over OWL 2 QL to queries over hybrid knowledge bases instead. The idea there was to split the input ontology into two parts, one involving metamodeling and one that does not. The former is transformed to Datalog using the method of [2], while the latter is kept as an ontology and linked to the Datalog program. The precise bridge rules to be created were either all possible or just those relevant to the query (using an established module notion). Experiments using *HEXLite-owl-api-plugin* as a hybrid reasoner showed this to be a viable approach, even if the observed performance was not as quick as hoped for. This appeared to be due to internals of the hybrid reasoner and the lack of any query-oriented optimisations such as the magic set technique. Indeed, results in [14] indicate that absence of a query-oriented method is detrimental for performance.

In this work, we use an extension of *DLV2* referred to as *Python external atoms*[3] as a hybrid reasoner. The system does support the magic set technique and our experiments show much better performance using this system.

In the following we recall the methods introduced in [15] and then turn to the new evaluation using DLV2.

[1] http://www.w3.org/TR/2004/REC-owl-semantics-20040210/
[2] http://www.w3.org/2007/OWL/wiki/Punning
[3] https://t.ly/dlv2-python-external-atoms

2 Preliminaries

This section gives a brief overview of the language and the formalism used in this work.

2.1 OWL 2 QL

This section recalls the syntax of the ontology language OWL 2 QL and the *Metamodeling Semantics* (MS) for OWL 2 QL, as given in [12].

Syntax
We start by recalling some basic elements used for representing knowledge in ontologies: *Concepts*, a set of individuals with common properties, *Individuals*, objects of a domain of discourse, and *Roles*, a set of relations that link individuals. An OWL 2 ontology is a set of axioms that describes the domain of interest. The elements are classified into *literals* and *entities*, where *literals* are values belonging to datatypes and *entities* are the basic ontology elements denoted by *Internationalized Resource Identifiers* (IRI). The notion of the vocabulary V of an OWL 2 QL, constituted by the tuple $V = (V_e, V_c, V_p, V_d, D, V_i, L_{QL})$. In V, V_e is the union of V_c, V_p, V_d, V_i and its elements are called atomic expressions; V_c, V_p, V_d, and V_i are sets of IRIs, denoting, respectively, classes, object properties, data properties, and individuals, L_{QL} denotes the set of literals - characterized as OWL 2 QL datatype maps denoted as DM_{QL} and D is the set of datatypes in OWL 2 QL (including rdfs:Literal). Given a vocabulary V of an ontology $\mathcal{O}$, we denote by Exp the set of well formed expressions over V. For the sake of simplicity we use Description Logic (DL) syntax for denoting expressions in OWL 2 QL. Complex expressions are built over V, for instance, if $e_1, e_2 \in V$ then $\exists e_1.e_2$ is a complex expression. An OWL 2 QL Knowledge Base $\mathcal{O}$ is a pair $\langle \mathcal{T}, \mathcal{A} \rangle$, where $\mathcal{T}$ is the TBox (inclusion axioms) and $\mathcal{A}$ is the ABox (assertional axioms). Sometimes we also let $\mathcal{O}$ denote $\mathcal{T} \cup \mathcal{A}$ for simplicity. OWL 2 QL is a finite set of logical axioms. The axioms allowed in an OWL 2 QL ontology have one of the form: inclusion axioms $e_1 \sqsubseteq e_2$, disjointness axioms $e_1 \sqsubseteq \neg\, e_2$, axioms asserting property i.e., reflexive property $ref(e)$ and irreflexive property $irref(e)$ and assertional axioms i.e., $c(a)$ class assertion, , $p(a, b)$ object property assertion, and $d(a, b)$ data property assertion. We employ the following naming schemes (possibly adding subscripts if necessary): c,p,d,t denote a class, object property, data property and datatype. The above axiom list is divided into TBox axioms (further divided into positive TBox axioms and negative TBox axioms) and ABox axioms. The positive TBox axioms consist of all the inclusion and reflexivity axioms, the negative TBox axioms consist of all the disjointness and irreflexivity axioms and ABox consist of all the assertional axioms. For simplicity, we omit OWL 2 QL axioms that can be expressed by appropriate combinations of the axioms specified in the above axiom list. Also, for simplicity we assume to deal with ontologies containing no data properties.

Meta-modeling Semantics
The Meta-modeling Semantics (MS) is based on the idea that every entity in V may simultaneously have more than one type, so it can be a class, or an individual, or data property, or an object property or a data type. To formalise this idea, the Meta-modeling Semantics has been defined for OWL 2 QL. In what follows, $\mathbf{P}(S)$ denotes the power set of S. The meta-modeling semantics for $\mathcal{O}$ over V is based on the notion of interpretation, constituted by a tuple $\mathcal{I} = \langle \Delta, \cdot^I, \cdot^C, \cdot^P, \cdot^D, \cdot^T, \cdot^{\mathcal{I}} \rangle$, where

- Δ is the union of the two non-empty disjoint sets: $\Delta = \Delta^o \cup \Delta^v$, where Δ^o is the object domain, and Δ^v is the value domain defined by DM_{QL};
- $\cdot^I : \Delta^o \rightarrow \{True, False\}$ is a total function for each object $o \in \Delta^o$, which indicates whether o is an individual; if $\cdot^C, \cdot^P, \cdot^D, \cdot^T$ are undefined for an o, then we require $o^I = True$, also in other cases, e.g., if o is in the range of $\cdot^C$;
- $\cdot^C : \Delta^o \rightarrow \mathbf{P}(\Delta^o)$ is partial and can assign the extension of a class;
- $\cdot^P : \Delta^o \rightarrow \mathbf{P}(\Delta^o \times \Delta^o)$ is partial and can assign the extension of an object property;
- $\cdot^D : \Delta^o \rightarrow \mathbf{P}(\Delta^o \times \Delta^v)$ is partial and can assign the extension of a data property;
- $\cdot^T : \Delta^o \rightarrow \mathbf{P}(\Delta^v)$ is partial and can assign the extension of a datatype;
- $\cdot^{\mathcal{I}}$ is a function that maps every expression in Exp to Δ^o and every literal to Δ_v.

This allows for a single object o to be simultaneously interpreted as an individual via $\cdot^I$, a class via $\cdot^C$, an object property via $\cdot^P$, a data property via $\cdot^D$, and a data type via $\cdot^T$. For instance, for Example 1, $\cdot^C$, $\cdot^I$ would be defined for *GoldenEagle*, while $\cdot^P, \cdot^D$ and $\cdot^T$ would be undefined for it.

The semantics of logical axiom α is defined in accordance with the notion of axiom satisfaction for an MS interpretation $\mathcal{I}$. The complete set of notions is specified in Table 3.B in [12]. Moreover, $\mathcal{I}$ is said to be a model of an ontology $\mathcal{O}$ if it satisfies all axioms of $\mathcal{O}$. Finally, an axiom α is said to be logically implied by $\mathcal{O}$, denoted as $\mathcal{O} \models \alpha$, if it is satisfied by every model of $\mathcal{O}$.

2.2 Hybrid Knowledge Bases

Hybrid Knowledge Bases (HKBs) have been proposed for coupling logic programming (LP) and Description Logic (DL) reasoning on a clear semantic basis. Our approach uses HKBs of the form $\mathcal{K} = \langle \mathcal{O}, \mathcal{P} \rangle$, where $\mathcal{O}$ is an OWL 2 QL knowledge base and $\mathcal{P}$ is a hex program, as defined next.

Hex programs [3] extend answer set programs with external computation sources. We use hex programs with unidirectional external atoms, which import elements from the ontology of an HKB. For a detailed discussion and the semantics of external atoms, we refer to [4]. What we describe here is a simplification of the much more general hex formalism.

Regular atoms are of the form $p(X_1, \ldots, X_n)$ where p is a predicate symbol of arity n and $X_1, \ldots, X_n$ are terms, that is, constants or variables. An external

atom is of the form $\&g[X_1, \ldots, X_n](Y_1, \ldots, Y_m)$ where g is an external predicate name g (which in our case interfaces with the ontology), $X_1, \ldots, X_n$ are input terms and $Y_1, \ldots, Y_m$ are output terms.

Next, we define the notion of positive rules that may contain external atoms.

Definition 1. *A hex rule r is of the form*

$$a \leftarrow b_1, \ldots, b_k. \quad k \geq 0$$

where a is regular atom and $b_1, \ldots, b_k$ are regular or external atoms. We refer to a as the head of r, denoted as $H(r)$, while the conjunction $b_1, ..., b_k$ is called the body of r.

We call r ordinary if it does not contain external atoms. A program $\mathcal{P}$ containing only ordinary rules is called a positive program, otherwise a hex program. A hex program is a finite set of rules.

The semantics of hex programs generalizes the answer set semantics. The Herbrand base of $\mathcal{P}$, denoted $HB_{\mathcal{P}}$, is the set of all possible ground versions of atoms and external atoms occurring in $\mathcal{P}$ (obtained by replacing variables with constants). Note that constants are not just those in the standard Herbrand universe (those occuring in $\mathcal{P}$) but also those created by external atoms, which in our case will be IRIs from $\mathcal{O}$. Let the grounding of a rule r be $grd(r)$ and the grounding of program $\mathcal{P}$ be $grd(\mathcal{P}) = \bigcup_{r \in \mathcal{P}} grd(r)$. An interpretation relative to $\mathcal{P}$ is any subset $I \subseteq HB_{\mathcal{P}}$ containing only regular atoms. We write $I \models a$ iff $a \in I$. With every external predicate name $\&g \in G$ we associate an $(n + m + 1)$-ary Boolean function $f_{\&g}$ (called oracle function) assigning each tuple $(I, x_1, \ldots, x_n, y_1 \ldots, y_m)$ either 0 or 1, where I is an interpretation and x_i, y_j are constants. We say that $I \models \&g[x_1, \ldots, x_n](y_1, \ldots, y_m)$ iff $f_{\&g}(I, x_1 \ldots, x_n, y_1, \ldots, y_m) = 1$. For a ground rule r, $I \models B(r)$ iff $I \models a$ for all $a \in B(r)$ and $I \models r$ iff $I \models H(r)$ whenever $I \models B(r)$. We say that I is a model of $\mathcal{P}$, denoted $I \models \mathcal{P}$, iff $I \models r$ for all $r \in grd(\mathcal{P})$. The *FLP-reduct* of $\mathcal{P}$ w.r.t I, denoted as $f\mathcal{P}^I$, is the set of all $r \in grd(\mathcal{P})$ such that $I \models B(r)$. An interpretation I is an answer set of $\mathcal{P}$ iff I is a minimal model of $f\mathcal{P}^I$. By $AS(\mathcal{P})$ we denote the set of all answer sets of $\mathcal{P}$. If $\mathcal{K} = \langle \mathcal{O}, \mathcal{P} \rangle$, then we write $AS(\mathcal{K}) = AS(\mathcal{P})$ — note that $\mathcal{O}$ is implicitly involved via the external atoms in $\mathcal{P}$. In this paper, $AS(\mathcal{K})$ will always contain exactly one answer set, so we will abuse notation and write $AS(\mathcal{K})$ to denote this unique answer set.

We will also need the notion of query answers of HKBs that contain rules defining a dedicated query predicate q. Given a hybrid knowledge base $\mathcal{K}$ and a query predicate q, let $ANS(q, \mathcal{K})$ denote the set $\{\langle x_1, \ldots, x_n \rangle \mid q(x_1, \ldots, x_n) \in AS(\mathcal{K})\}$.

3 Query Answering Using MSER

We consider SPARQL queries, a W3C standard for querying ontologies. While SPARQL query results can in general either be result sets or RDF graphs, we

have restricted ourselves to simple **SELECT** queries, so it is sufficient for our purposes to denote results by set of tuples. For example, consider the following SPARQL query:

SELECT $?x$ $?y$ $?z$ WHERE {
$?x$ $rdf:type$ $?y$.
$?y$ $rdfs:SubClassOf$ $?z$
}

This query will retrieve all triples $\langle x, y, z \rangle$, where x is a member of class y that is a subclass of z. In general, there will be several variables and there can be multiple matches, so the answers will be sets of tuples of IRIs.

Now, we recall query answering under the Meta-modeling Semantics Entailment Regime (MSER) from [2]. This technique reduces SPARQL query answering over OWL 2 QL ontologies to Datalog query answering. The main idea of this approach is to define (i) a translation function τ mapping OWL 2 QL axioms to datalog facts and (ii) a fixed datalog rule base $\mathcal{R}^{ql}$ that captures inferences in OWL 2 QL reasoning.

The reduction employs a number of predicates, which are used to encode the basic axioms available in OWL 2 QL. This includes both axioms that are explicitly represented in the ontology (added to the Datalog program as facts via τ) and axioms that logically follow. In a sense, this representation is closer to a meta-programming representation than other Datalog embeddings that translate each axiom to a rule.

The function τ transforms an OWL 2 QL assertion α to a fact. For a given ontology $\mathcal{O}$, we will denote the set of facts obtained by applying τ to all of its axioms as $\tau(\mathcal{O})$; it will be composed of two portions $\tau(\mathcal{T})$ and $\tau(\mathcal{A})$, as indicated in Table 1.

Table 1: τ Function

$\tau(\mathcal{O})$	α	$\tau(\alpha)$	α	$\tau(\alpha)$
$\tau(\mathcal{T})$	c1 $\sqsubseteq$ c2	isacCC(c1, c2)	r1 $\sqsubseteq \neg$ r2	disjrRR(r1,r2)
	c1 $\sqsubseteq \exists$r2$^-$.c2	isacCI(c1,r2,c2)	c1 $\sqsubseteq \neg$ c2	disjcCC(c1,c2)
	$\exists$r1 $\sqsubseteq \exists$r2.c2	isacRR(r1,r2,c2)	c1 $\sqsubseteq \neg\exists$r2$^-$	disjcCI(c1,r2)
	$\exists$r1$^-$ $\sqsubseteq$ c2	isacIC(r1,c2)	$\exists$r1$\sqsubseteq \neg$ c2	disjcRC(r1,c2)
	$\exists$r1$^-$ $\sqsubseteq \exists$r2.c2	isacIR(r1,r2,c2)	$\exists$r$_1$ $\sqsubseteq \neg\exists$r2	disjcRR(r1,r2)
	$\exists$r1$^-$ $\sqsubseteq \exists$r2$^-$.c2	isacII(r1,r2,c2)	$\exists$r1 $\sqsubseteq \neg\exists$r2$^-$	disjcRI(r1,r2)
	r1 $\sqsubseteq$ r2	isarRR(r1,r2)	$\exists$r1$^-$ $\sqsubseteq \neg$ c2	disjcIC(r1,c2)
	r1 $\sqsubseteq$ r2$^-$	isarRI(r1,r2)	$\exists$r1$^-$ $\sqsubseteq \neg\exists$r2	disjcIR(r1,r2)
	c1 $\sqsubseteq \exists$r2.c2	isacCR(c1,r2,c2)	$\exists$r1$^-$ $\sqsubseteq \neg\exists$r2$^-$	disjcII(r1,r2)
	$\exists$r1$\sqsubseteq$ c2	isacRC(r1,c2)	r1 $\sqsubseteq \neg$ r2$^-$	disjrRI(r1,r2)
	$\exists$r1 $\sqsubseteq \exists$r2$^-$.c2	isacRI(r1,r2,c2)	irref(r)	irrefl(r)
	refl(r)	refl(r)		
$\tau(\mathcal{A})$	c(x)	instc(c,x)	x $\neq$ y	diff(x,y)
	r(x, y)	instr(r,x,y)		

The fixed program $\mathcal{R}^{ql}$ can be viewed as an encoding of axiom saturation in OWL 2 QL. The full set of rules provided by authors of [2] are reported in the online repository of [14]. We will consider one rule to illustrate the underlying ideas:

$$\text{isacCR(C1,R2,C2)} \leftarrow \text{isacCC(C1,C3)}, \text{isacCR(C3,R2,C2)}.$$

The above rule encodes the following inference rule:

$$\mathcal{O} \models \text{C1} \sqsubseteq \text{C3}, \mathcal{O} \models \text{C3} \sqsubseteq \exists \text{R2.C2} \Rightarrow \mathcal{O} \models \text{C1} \sqsubseteq \exists \text{R2.C2}$$

Finally, the translation can be extended in order to transform conjunctive SPARQL queries under MS over OWL 2 QL ontologies into a Datalog query. SPARQL queries will be translated to Datalog rules using a transformation τ^q. τ^q uses τ to translate the triples inside the body of the SPARQL query $\mathcal{Q}$ and adds a fresh datalog predicate q in the head to account for projections. In the following we assume q to be the query predicate created in this way.

For example, the translation of the SPARQL query given earlier will be

$$\text{q(X,Y,Z)} \leftarrow \text{instc(X,Y)}, \text{isacCC(Y,Z)}.$$

Given an OWL 2 QL ontology $\mathcal{O}$ and a SPARQL query $\mathcal{Q}$, let $ANS(\mathcal{Q}, \mathcal{O})$ denote the answers to $\mathcal{Q}$ over $\mathcal{O}$ under MSER, that is, a set of tuples of IRIs. In the example above, the answers will be a set of triples.

4 MSER Query Answering via Hybrid Knowledge Bases

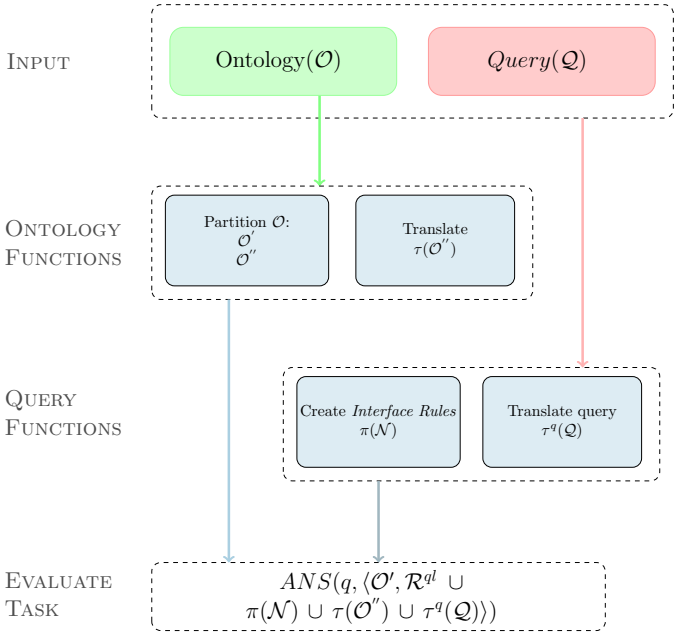
We propose four variants for answering MSER queries by means of Hybrid Knowledge Bases. We first describe the general approach and then define each of the four variants.

4.1 General Architecture

The general architecture is outlined in Figure 1. In all cases, the inputs are an OWL 2 QL ontology $\mathcal{O}$ and a SPARQL query $\mathcal{Q}$. We then differentiate between **OntologyFunctions** and **QueryFunctions**. The **OntologyFunctions** achieves two basic tasks: first, the ontology is split into two partitions $\mathcal{O}^{'}$ and $\mathcal{O}^{''}$, then $\tau(\mathcal{O}^{''})$ is produced.

The **QueryFunctions** work mainly on the query. First, a set $\mathcal{N}$ of IRIs is determined for creating *Interface Rules* (IR, simple hex rules), denoted as $\pi(\mathcal{N})$ for importing the extensions of relevant classes and properties from $\mathcal{O}^{'}$. In the simplest case, $\mathcal{N}$, consist of all IRIs in $\mathcal{O}^{'}$, but we also consider isolating those IRIs that are relevant to the query by means of Logic-based Module Extraction (LME) as defined in [8]. Then, τ^q translates $\mathcal{Q}$ into a datalog query $\tau^q(\mathcal{Q})$. Finally, the created hex program components are united (plus the fixed inference rules), yielding the rule part $\mathcal{P} = \mathcal{R}^{ql} \cup \pi(\mathcal{N}) \cup \tau(\mathcal{O}^{''}) \cup \tau^q(\mathcal{Q})$, which together with $\mathcal{O}^{'}$ forms the HKB $\mathcal{K} = \langle \mathcal{O}^{'}, \mathcal{P} \rangle$, for which we then determine $ANS(q, \mathcal{K})$, where q is the query predicate introduced by $\tau^q(\mathcal{Q})$.

Fig. 1: The Overall Architecture of Hybrid-Framework



4.2 Basic Notions

Before defining the specific variations of our approach, we first define some auxiliary notions. The first definition identifies meta-elements.

Definition 2. *Given an Ontology $\mathcal{O}$, IRIs in $(V_c \cup V_p) \cap V_i$ are meta-elements, i.e., IRIs that occur both as individuals and classes or object properties.*

In our example, *GoldenEagle* is a meta-element. Meta-elements form the basis of our main notion, clashing axioms.

Definition 3. *Clashing Axioms in $\mathcal{O}$ are axioms that contain meta-elements, denoted as* $\mathrm{CA}(\mathcal{O})$*. To denote clashing and non-clashing parts in TBox ($\mathcal{T}$) and ABox ($\mathcal{A}$), we write $\mathcal{A}^N = \mathcal{A} \setminus \mathrm{CA}(\mathcal{O})$ as non-clashing ABox, $\mathcal{A}^C = \mathrm{CA}(\mathcal{O}) \cap \mathcal{A}$ as clashing ABox; and likewise $\mathcal{T}^N = \mathcal{T} \setminus \mathrm{CA}(\mathcal{O})$ as non-clashing TBox and $\mathcal{T}^C = \mathrm{CA}(\mathcal{O}) \cap \mathcal{T}$ as clashing TBox.*

The clashing axiom notion allows for splitting $\mathcal{O}$ into two parts and generate $\mathcal{O}'$ without clashing axioms.

We would also like to distinguish between standard queries and meta-queries. A meta-query is an expression consisting of meta-predicates p and meta-variables v, where p can have other predicates as their arguments and v can appear in predicate positions. The simplest form of meta-query is an expression where variables appear in class or property positions also known as *second-order queries*.

More interesting forms of meta-queries allow one to extract complex patterns from the ontology, by allowing variables to appear simultaneously in individual object and class or property positions. We will refer to non-meta-queries as standard queries. Moving towards *Interface Rules*, we first define signatures of queries, ontologies, and axioms.

Definition 4. *A* signature $\mathbf{S}(\mathcal{Q})$ *of a SPARQL query* $\mathcal{Q}$ *is the set of* IRIs *occurring in* $\mathcal{Q}$*. If no IRIs occur in* $\mathcal{Q}$*, we define* $\mathbf{S}(\mathcal{Q})$ *to be the signature of* $\mathcal{O}$*. Let* $\mathbf{S}(\mathcal{O})$ *(or* $\mathbf{S}(\alpha)$*) be the set of atomic classes, atomic roles and individuals that occur in* $\mathcal{O}$ *(or in axiom* α*).*

As hinted earlier, we can use $\mathbf{S}(\mathcal{O}^{'})$ for creating interface rules ($\mathcal{O}^{'}$ being the ontology part in the HKB), or use $\mathbf{S}(\mathcal{Q})$ for module extraction via *LME* as defined in [8] for singling out the identifiers relevant to the query, to be imported from the ontology via interface rules. We will denote this signature as $\mathbf{S}(LME(\mathbf{S}(\mathcal{Q}), \mathcal{O}^{'}))$.

We next define the *Interface Rules* for a set of IRIs $\mathcal{N}$.

Definition 5. *For a set a of IRIs* $\mathcal{N}$*, let* $\pi(\mathcal{N})$ *denote the hex program containing a rule*

$$instc(C, X) \leftarrow \&g[C](X).$$

for each class identifier $C \in \mathcal{N}$*, and a rule*

$$instr(R, X, Y) \leftarrow \&g[R](X, Y).$$

for each property identifier $R \in \mathcal{N}$*. Here* $\&g$ *is a shorthand for the external atom that imports the extension of classes or properties from the ontology* $\mathcal{O}^{'}$ *of our framework.*

The rules in $\pi(\mathcal{N})$ are called *Interface Rules* (IR) and serve as the bridge from $\mathcal{O}^{'}$ to $\mathcal{P}$.

4.3 Variants

Now we define the four variants for the ontology functions, and two for the query functions. Since for one ontology function $\mathcal{O}^{'}$ is empty, the two query functions have the same effect, and we therefore arrive at seven different variants for creating the hybrid knowledge bases (HKB).

The difference in the ontology functions is which axioms of $\mathcal{O} = \langle \mathcal{A}, \mathcal{T} \rangle$ stay in $\mathcal{O}^{'}$ and which are in $\mathcal{O}^{''}$, the latter of which is translated to Datalog. We use a simple naming scheme, indicating these two components:

$\mathcal{A}-\mathcal{T}$**:** $\mathcal{O}^{'} = \mathcal{A}$, $\mathcal{O}^{''} = \mathcal{T}$.
$N\mathcal{A}\mathcal{T}-C\mathcal{A}\mathcal{T}$**:** $\mathcal{O}^{'} = \langle \mathcal{A}^N, \mathcal{T} \rangle$, $\mathcal{O}^{''} = \langle \mathcal{A}^C, \mathcal{T} \rangle$.
$N\mathcal{A}\mathcal{T}-C\mathcal{A}C\mathcal{T}$**:** $\mathcal{O}^{'} = \langle \mathcal{A}^N, \mathcal{T} \rangle$, $\mathcal{O}^{''} = \langle \mathcal{A}^C, \mathcal{T}^C \rangle$.
$E-\mathcal{A}\mathcal{T}$**:** $\mathcal{O}^{'} = \emptyset$, $\mathcal{O}^{''} = \mathcal{O} = \langle \mathcal{A}, \mathcal{T} \rangle$.

$E-\mathcal{A}\mathcal{T}$ serves as a baseline, as it boils down to the Datalog encoding of [2]. $E-\mathcal{A}\mathcal{T}$ is also the only variant for which the query functions will not create any Interface Rules, i.e. $\pi(\mathcal{N}) = \emptyset$. We next describe and define each of the ontology functions.

A−T In this approach, $\mathcal{O}'$ consists only of the ABox $\mathcal{A}$ of $\mathcal{O}$ and the TBox $\mathcal{T}$ of $\mathcal{O}$ is translated into Datalog. Here the main difference to the Datalog encoding of [2] is that the ABox is not translated to facts but stays in the ontology part and can be accessed via Interface Rules.

Definition 6. *Given $\mathcal{O} = \langle \mathcal{A}, \mathcal{T} \rangle$, let the $\mathcal{A}-\mathcal{T}$ HKB be $\mathcal{K}^{\mathcal{A}-\mathcal{T}}(\mathcal{O}) = \langle \mathcal{A}, \mathcal{R}^{ql} \cup \tau(\mathcal{T}) \rangle$.*

NAT−CAT In this approach, the notion of clashing axioms (cf. Definition 3) is used to separate the ABox into two parts, each of which is combined with the same TBox. One of the resulting ontologies is clash-free and can therefore be reliably treated by a standard ontology reasoner. The other ontology has the meta-assertions and will be treated by the datalog transformation. The link between the two will later be provided by the query functions in the form of Interface Rules.

Definition 7. *Given $\mathcal{O} = \langle \mathcal{A}, \mathcal{T} \rangle$, let the NAT–CAT HKB be $\mathcal{K}^{NAT-CAT}(\mathcal{O}) = \langle \langle \mathcal{A}^N, \mathcal{T} \rangle, \mathcal{R}^{ql} \cup \tau(\langle \mathcal{A}^C, \mathcal{T} \rangle) \rangle$.*

NAT−CACT This approach is similar to the previous one except that it does not pair the same $\mathcal{T}$ with both $\mathcal{A}^N$ and $\mathcal{A}^C$. Instead, it associates $\mathcal{A}^N$ with the original $\mathcal{T}$ as before, but associates only $\mathcal{T}^C$, the clashing part of $\mathcal{T}$ to $\mathcal{A}^C$, yielding a potentially smaller ontology to be translated to Datalog. Also here, the linking Interface Rules will be added later by the query functions.

Definition 8. *Given $\mathcal{O} = \langle \mathcal{A}, \mathcal{T} \rangle$, let the NAT–CACT HKB be $\mathcal{K}^{NAT-CACT}(\mathcal{O}) = \langle \langle \mathcal{A}^N, \mathcal{T} \rangle, \mathcal{R}^{ql} \cup \tau(\langle \mathcal{A}^C, \mathcal{T}^C \rangle) \rangle$.*

E−AT This approach is the baseline, has an empty ontology part in the HKB and translates the entire given ontology to Datalog, just like in [2]. Note that here no Interface Rules are necessary.

Definition 9. *Given $\mathcal{O}$, let the $E-\mathcal{AT}$ HKB be $\mathcal{K}^{E-\mathcal{AT}}(\mathcal{O}) = \langle \emptyset, \mathcal{R}^{ql} \cup \tau(\mathcal{O}) \rangle$.*

Next we turn to the query functions. As hinted at earlier, we will consider two versions, which differ in the Interface Rules they create. Both create query rules $\tau^q(\mathcal{Q})$ for the given query, but one (All) will create interface rules for all classes and properties in the ontology part of the HKB, while the other (Mod) will extract the portion of the ontology relevant to query using LME and create Interface Rules only for classes and properties in this module.

For notation, we will overload the $\cup$ operator for HKBs, so we let $\langle \mathcal{O}, \mathcal{P} \rangle \cup \langle \mathcal{O}', \mathcal{P}' \rangle = \langle \mathcal{O} \cup \mathcal{O}', \mathcal{P} \cup \mathcal{P}' \rangle$ and we also let $\langle \mathcal{O}, \mathcal{P} \rangle \cup \mathcal{P}' = \langle \mathcal{O}, \mathcal{P} \cup \mathcal{P}' \rangle$ for ontologies $\mathcal{O}$ and $\mathcal{O}'$ and hex programs $\mathcal{P}$ and $\mathcal{P}'$.

Definition 10. *Given an HKB $\langle \mathcal{O}, \mathcal{P} \rangle$ and query $\mathcal{Q}$, let the All HKB be defined as $\mathcal{K}_{All}(\langle \mathcal{O}, \mathcal{P} \rangle, \mathcal{Q}) = \langle \mathcal{O}, \mathcal{P} \cup \tau^q(\mathcal{Q}) \cup \pi(\mathbf{S}(\mathcal{O})) \rangle$.*

Definition 11. *Given an HKB $\langle \mathcal{O}, \mathcal{P} \rangle$ and query $\mathcal{Q}$, let the Mod HKB be defined as $\mathcal{K}_{All}(\langle \mathcal{O}, \mathcal{P} \rangle, \mathcal{Q}) = \langle \mathcal{O}, \mathcal{P} \cup \tau^q(\mathcal{Q}) \cup \pi(\mathbf{S}(LME(\mathbf{S}(\mathcal{Q}), \mathcal{O})))\rangle$.*

We will combine ontology functions and query functions, and instead of $\mathcal{K}_\beta(\mathcal{K}^\alpha(\mathcal{O}), \mathcal{Q})$ we will write $\mathcal{K}^\alpha_\beta(\mathcal{O}, \mathcal{Q})$. We thus get eight combinations, but we will not use $\mathcal{K}^{E-\mathcal{AT}}_{Mod}$, as it unnecessarily introduces Interface Rules. Also note that $\mathcal{K}^{E-\mathcal{AT}}_{All}(\mathcal{O}, \mathcal{Q})$ does not contain any Interface Rules, because the ontology part of $\mathcal{K}^{E-\mathcal{AT}}(\mathcal{O})$ is empty.

5 Evaluation

In [15] we conducted experiments using HEXLite with the OWL-API plugin. While it did show drastic improvements when using one of the hybrid approaches with respect to the baseline $E-\mathcal{AT}$and with using Mod rather then All, the absolute performance was disappointing. In particular, with the larger ontologies considered, no answer could be obtained even after hours. This contrasts sharply with the findings in [14], in which the best systems took only seconds to answer queries even on the larger ontologies. The main reasons appeared to be inefficiencies in the OWL-API plugin, paired with a lack of query-oriented computation.

In the meantime we became aware of *DLV2* with *Python external atoms*[4].

The version of DLV2 that we obtained from the developers directly supports the Turtle format of ontologies, and one can use ontology IRIs directly as predicate names. The rules of Definition 5 can then directly use class and role identifiers:

Definition 12. *For a set a of IRIs $\mathcal{N}$, let $\pi(\mathcal{N})$ denote the DLV2 program containing a rule*

$$instc(C, X) \leftarrow \quad C(X).$$

for each class identifier $C \in \mathcal{N}$, and a rule

$$instr(R, X, Y) \leftarrow \quad R(X, Y).$$

for each property identifier $R \in \mathcal{N}$.

For transforming our ontologies to Turtle format, we have used a utility called *ont-converter*[5] that automatically transforms the source ontology in different formats (RDF/XML, OWL/XML, N3, etc).

The experimental setting is the same as in [15]: we conducted two sets of experiments on the widely used Lehigh University Benchmark (LUBM) dataset and on the Making Open Data Effectively USable (MODEUS) Ontologies[6]. We

[4] https://dlv.demacs.unical.it/home
[5] https://github.com/sszuev/ont-converter
[6] http://www.modeus.uniroma1.it/modeus/node/6

only use the query function Mod, as it was evident in [15] that All has no advantage over Mod.

The **LUBM** datasets describe a university domain with information like departments, courses, students, and faculty. This dataset comes with 14 queries with different characteristics (low selectivity vs high selectivity, implicit relationships vs explicit relationships, small input vs large input, etc.). We have also considered the meta-queries *mq1*, *mq4*, *mq5*, and *mq10* from [9] as they contain variables in-property positions and are long conjunctive queries. We have also considered two special-case queries *sq1* and *sq2* from [2] to exercise the MSER features and identify the new challenges introduced by the additional expressivity over the ABox queries. Basically, in special-case queries, we check the impact of `DISJOINTWITH` and meta-classes in a query. For this, like in [2], we have introduced a new class named *TypeOfProfessor* and make *FullProfessor*, *AssociateProfessor* and *AssistantProfessor* instances of this new class and also define *FullProfessor*, *AssociateProfessor* and *AssistantProfessor* to be disjoint from each other. Then, in *sq1* we are asking for all those y and z, where y is a professor, z is a type of professor and y is an instance of z. In *sq2*, we have asked for different pairs of professors.

The **MODEUS** ontologies describe the *Italian Public Debt* domain with information like financial liability or financial assets to any given contracts [11]. It comes with 8 queries. These queries are pure meta-queries as they span over several levels of the knowledge base. MODEUS ontologies are meta-modeling ontologies with meta-classes and meta-properties.

We have done the experiments on a Linux batch server, running Ubuntu 20.04.3 LTS (GNU/Linux 5.4.0-88-generic x86_64) on one AMD EPYC 7601 (32-Core CPU), 2.2GHz, Turbo max. 3.2GHz. The machine is equipped with 512GB RAM and a 4TB hard disk. Java applications used OpenJDK 11.0.11 with a maximum heap size of 25GB. During the course of the evaluation of the proposed variants we have used the time resource limitation as the benchmark setting on our data sets to examine the behavior of different variants. If not otherwise indicated, in both experiments, each benchmark had 3600 minutes (excluding the $\mathcal{K}$ generation time). For simplicity, we have not included queries that contain data properties in our experiments. We also have included the generation time of the hybrid knowledge base $\mathcal{K}$ including the loading of ontology and query, τ translation, module extraction, generating IR and translating queries. All material of experiments and results are available at https://gitlab.com/hmqq/hkb-dlv2.git. In the figures, $E-\mathcal{AT}$ is labelled E-T.

In Figure 2 and 3, it can be seen that *DLV2* shows regular performance across all datasets and all variants of HKB with a slight increase in time depending on the size of the dataset. There is one outlier, meta-query MQ5 on LUBM(1) with $N\mathcal{AT}-C\mathcal{AT}$, which we were not expecting and might be a measurement error. In any case, this a massive improvement over the performance with HEXLite, where some of these queries required thousands of seconds to evaluate.

In Figures 4 to 7 the performance on MODEUS queries is reported. All the variants show consistent performance; however, the behaviour of the $N\mathcal{AT}-$

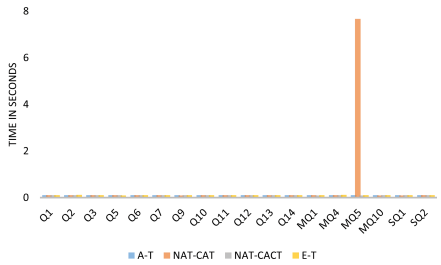


Fig. 2: LUBM(1) experiments with standard and meta queries

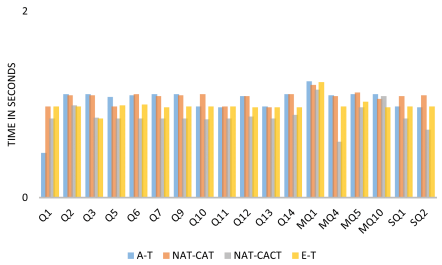


Fig. 3: LUBM(9) experiments with Standard and Meta Queries

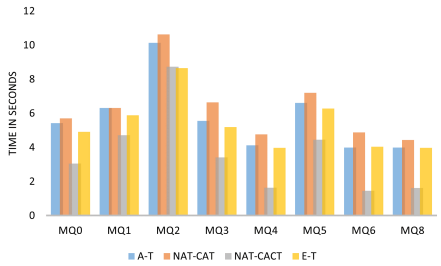


Fig. 4: MODEUS(00) with Meta Queries

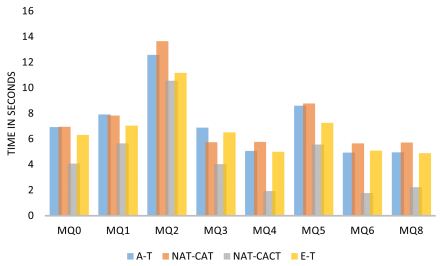


Fig. 5: MODEUS(01) with Meta Queries

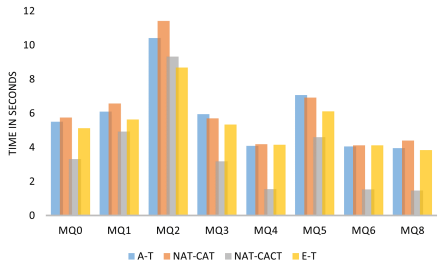


Fig. 6: MODEUS(02) with Meta Queries

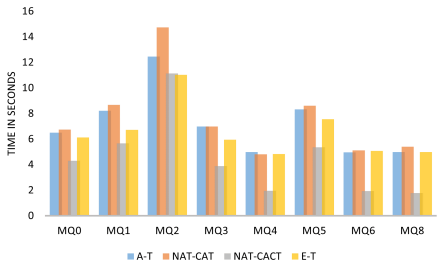


Fig. 7: MODEUS(03) with Meta Queries

$\mathcal{CACT}$ variant seems to be usually best. These results are very satisfactory with respect to the results observed with HEXLite, where none of these queries were answered even after a few hours of runtime.

It should also be noted that $\mathcal{NAT}$–$\mathcal{CACT}$ with DLV2 also outperforms non-hybrid query answering using DLV2 as reported in [14], making it the fastest known method on these ontologies and queries.

6 Discussion and Conclusion

This work shows that the methods introduced in [14] do not only have a positive relative impact when using a hybrid reasoner, but that they can also yield the best known performance when using a suitable tool for hybrid reasoning.

It seems clear from the result that there is a benefit of keeping some portions in the ontology rather than transforming the entire ontology to facts. This is, however, contingent of the availability of a query-aware method (in this case magic sets). Among the variants, $\mathcal{NAT}$–$\mathcal{CACT}$ showed best performance, which is also the one that hybridizes most.

In the future, we hope to identify more hybrid reasoning systems that are query aware.

References

1. Chen, W., Kifer, M., Warren, D.S.: Hilog: A foundation for higher-order logic programming. The Journal of Logic Programming **15**(3), 187–230 (1993)
2. Cima, G., De Giacomo, G., Lenzerini, M., Poggi, A.: On the SPARQL metamodeling semantics entailment regime for OWL 2 QL ontologies. In: Proceedings of the 7th International Conference on Web Intelligence, Mining and Semantics. pp. 1–6 (2017)
3. Eiter, T., Fink, M., Ianni, G., Krennwallner, T., Redl, C., Schüller, P.: A model building framework for answer set programming with external computations. Theory and Practice of Logic Programming **16**(4), 418–464 (2016)
4. Eiter, T., Ianni, G., Schindlauer, R., Tompits, H.: Effective integration of declarative rules with external evaluations for semantic-web reasoning. In: European Semantic Web Conference. pp. 273–287. Springer (2006)
5. Glimm, B.: Using SPARQL with RDFS and OWL entailment. In: Reasoning Web International Summer School. pp. 137–201. Springer (2011)
6. Guizzardi, G., Almeida, J.P.A., Guarino, N., de Carvalho, V.A.: Towards an ontological analysis of powertypes. In: JOWO@ IJCAI (2015)
7. Hitzler, P., Krotzsch, M., Rudolph, S.: Foundations of semantic web technologies. CRC press (2009)
8. Jiménez-Ruiz, E., Grau, B.C., Sattler, U., Schneider, T., Berlanga, R.: Safe and economic re-use of ontologies: A logic-based methodology and tool support. In: European Semantic Web Conference. pp. 185–199. Springer (2008)
9. Kontchakov, R., Rezk, M., Rodriguez-Muro, M., Xiao, G., Zakharyaschev, M.: Answering SPARQL queries over databases under OWL 2 QL entailment regime. In: International Semantic Web Conference. pp. 552–567. Springer (2014)

10. Lenzerini, M., Lepore, L., Poggi, A.: A higher-order semantics for OWL 2 QL ontologies. In: Description Logics (2015)
11. Lenzerini, M., Lepore, L., Poggi, A.: Metaquerying made practical for OWL 2 QL ontologies. Information Systems **88**, 101294 (2020)
12. Lenzerini, M., Lepore, L., Poggi, A.: Metamodeling and metaquerying in OWL 2 QL. Artificial Intelligence **292**, 103432 (2021)
13. Motik, B.: On the properties of metamodeling in owl. In: International Semantic Web Conference. pp. 548–562. Springer (2005)
14. Qureshi, H.M., Faber, W.: An evaluation of meta-reasoning over OWL 2 QL. In: Moschoyiannis, S., Peñaloza, R., Vanthienen, J., Soylu, A., Roman, D. (eds.) Rules and Reasoning - 5th International Joint Conference, RuleML+RR 2021, Leuven, Belgium, September 13-15, 2021, Proceedings. Lecture Notes in Computer Science, vol. 12851, pp. 218–233. Springer (2021). https://doi.org/10.1007/978-3-030-91167-6_15, https://doi.org/10.1007/978-3-030-91167-6_15
15. Qureshi, H.M., Faber, W.: Using hybrid knowledge bases for meta-reasoning over OWL 2 QL. In: Hanus, M., Inclezan, D. (eds.) Practical Aspects of Declarative Languages - 25th International Symposium, PADL 2023, Boston, MA, USA, January 16-17, 2023, Proceedings. Lecture Notes in Computer Science, vol. 13880, pp. 216–231. Springer (2023). https://doi.org/10.1007/978-3-031-24841-2_14