Optimizing Horn-$\mathcal{SHIQ}$ Reasoning for OBDA[⋆]

Labinot Bajraktari[1], Magdalena Ortiz[1], and Guohui Xiao[2]

[1] Faculty of Informatics, Vienna University of Technology, Austria
{bajraktari,ortiz}@kr.tuwien.ac.at
[2] Faculty of Computer Science, Free University of Bozen-Bolzano, Italy
xiao@inf.unibz.it

Abstract. The ontology-based data access (OBDA) paradigm can ease access to heterogeneous and incomplete data sources in many application domains. However, state-of-the-art tools are still based on the DL-Lite family of description logics (DLs) that underlies OWL 2 QL, which despite its usefulness is not sufficiently expressive for many domains. Accommodating more expressive ontology languages remains an open challenge, and the consensus is that Horn DLs like Horn-$\mathcal{SHIQ}$ are particularly promising. Query answering in Horn-$\mathcal{SHIQ}$, a prerequisite for OBDA, is supported in existing reasoners, but many ontologies cannot be handled. This is largely because algorithms build on an ABox-independent approach to ontological reasoning that easily incurs in an exponential behaviour. As an alternative to full ABox-independence, in this paper we advocate taking into account general information about the structure of the ABoxes of interest. This is especially natural in the setting of OBDA, where ABoxes are generated via mappings, and thus have a predictable structure. We present a simple yet effective approach that guides ontological reasoning using the possible combinations of concepts that may occur in the ABox, which can be obtained from the mappings of an OBDA specification. We implemented and tested our optimization in the Clipper reasoner with encouraging results.

1 Introduction

The *ontology-based data access (OBDA)* paradigm [22] eases access to possibly heterogeneous and incomplete data sources using an *ontology*, a formal representation of the conceptualization of the domain that is written in a shareable, machine-readable language. The user queries can be expressed over the familiar ontology vocabulary, and the knowledge in the ontology can be leveraged to infer implicit facts and obtain more query answers. Different sources can be linked to the same ontology, making OBDA a very effective alternative to the costly integration of data sources [23]. Let us consider a scenario where, to facilitate compliance, a regulatory body shares with financial institutions an ontology that

[⋆] This work was supported by the Austrian Science Fund (FWF) projects P30360, P30873, W1255 and by the Free University of Bozen-Bolzano projects STyLoLa, OBATS and QUADRO.

includes, among others, the axiom $\mathsf{Account} \sqcap \exists \mathsf{hasOwner.PEP} \sqsubseteq \mathsf{MonitoredAcc}$ which states that accounts of politically exposed persons (PEP) must be monitored. Queries of interest for the regulatory body could be similar to this one, which returns the owners of accounts that interact with a monitored account:

$$\mathsf{q}(y) \leftarrow \mathsf{Account}(x), \mathsf{hasOwner}(x,y), \mathsf{interactsWith}(x,z), \mathsf{MonitoredAcc}(z)$$

When evaluating the query we can infer which accounts are monitored, without this flag being explicitly stored in the data. Moreover, the actual storage of the accounts, owners and interactions may be rather complex, spanning different tables and databases, and is likely to differ between different (sub)organizations and financial institutions. In OBDA this is overcome by using *mappings* such as

$$\begin{aligned} \mathsf{sql}_1(x,y) &\rightsquigarrow \mathsf{interactedWith}(x,y), \mathsf{Account}(x), \mathsf{Account}(y) \\ \mathsf{sql}_2(x,y) &\rightsquigarrow \mathsf{hasOwner}(x,y) \\ \mathsf{sql}_3(x) &\rightsquigarrow \mathsf{PEP}(x) \end{aligned}$$

where $\mathsf{sql}_1 - \mathsf{sql}_3$ are (possibly complex) SQL queries that specify how the data in one specific organization's database is mapped to the vocabulary of the ontology.

The ontology of an OBDA specification is usually in the so-called *DL-Lite* family of description logics (DLs), which underlies OWL 2 QL [7]. DL-Lite is tailored so that queries over the ontology can be transformed, using reasoning, into standard SQL queries that already incorporate the relevant ontological knowledge and can be evaluated with existing database query engines. This central property is key to OBDA being efficiently implementable on top of current database management systems. However, many domains call for expressive features not supported in DL-Lite. For example, the axiom above uses conjunction $\sqcap$ and a *qualified existential restriction* $\exists r.B$ on the left-hand-side. Both constructs are not expressible in the DL-Lite family, but they are found in many ontologies [4], and they are in fact the basis of the OWL 2 EL profile popular for life science ontologies,[3] e.g. SNOMED CT, NCI, and GENE ontologies.

Considerable research efforts have been devoted to more expressive DLs. The so-called *Horn* DLs are particularly appealing: they can support the features above while remaining computationally manageable. In contrast to their non-Horn counterparts, the data complexity of reasoning in Horn DLs is in PTime [3]. Some advanced reasoning problems, like query emptiness and query inseparability, are more manageable for Horn DLs [6, 5, 1], and they have proved much more amenable to implementation [11, 13, 9]. Horn-$\mathcal{SHIQ}$, which can be seen as the Horn fragment of OWL Lite, is a very popular Horn DL that, additionally to the features above, supports *transitive roles* and some *number restrictions*. In our example, we could make $\mathsf{interactedWith}$ transitive to detect interactions through a chain of accounts, or we could use an axiom such as $\mathsf{Account} \sqsubseteq \leqslant 1\, \mathsf{hasOwner.Person}$ to say that an account can have only one owner.

Horn-$\mathcal{SHIQ}$ is relatively well understood, and there are existing reasoners for traditional reasoning problems like satisfiability and classification [13] as well as for *ontology mediated querying (OMQ)* which, in a nutshell, is the simplification of OBDA (and a prerequisite thereof) that omits the mapping layer: one assumes that the data is already an *ABox*, that is, a set of facts over the ontology

[3] `https://www.w3.org/TR/owl2-profiles/#OWL_2_EL`

vocabulary. Unlike DL-Lite, it is in general not possible to reduce query answering in the presence of a Horn-$\mathcal{SHIQ}$ ontology to plain SQL query evaluation. Some alternative approaches have been proposed in order to make OBDA with Horn-$\mathcal{SHIQ}$ feasible on top of existing database technologies. For example, to rewrite (exactly or approximately) an ontology into a weaker DL [17, 19], or to compile some of the extra expressivity into the mappings [4]. Another possibility is to compile the query and the ontology into a more expressive query language than SQL, like Datalog, as done in the Clipper system [11].

Clipper is a query rewriting engine that takes as input an ontology and possibly a set of queries. After a so-called 'saturation' step that uses a *consequence-driven* calculus to add axioms implied by the ontology, it generates a Datalog rewriting of the given queries. Clipper can handle realistic ontologies and queries, despite being a relatively simple prototype. It is among the richest query answering engines for Horn DLs, and has inspired recent adaptations [16, 8]. However, Clipper has stark limitations and there are many ontologies that it cannot process in reasonable time [9]. This is largely due to the ABox independence of the saturation step: some axioms that could be omitted for simpler tasks like *classification* [13], must be inferred by Clipper since they may be made relevant by the assertions in some input ABox.

To overcome this obstacle, we propose to mildly compromise ABox independence, by allowing the saturation step to depend on the structure of the possible ABoxes, but not on concrete assertions. Specifically, we propose a version (described in Section 3) of the Clipper saturation that is parametrized by a set of sets of concept names, which are used to guide the inference of new axioms. Intuitively, these concept names are the concept combinations that may occur in the relevant ABoxes. A nice feature of our approach is that if new ABoxes become relevant, new sets of concepts can be added and the saturation re-executed on top of the previous output, and all previous derivations remain valid. Our approach is particularly meaningful for OBDA, where the virtual ABoxes arising from the mappings have a restricted and predictable structure. In Section 4 we show that, meaningful sets of concept sets for guiding the algorithm can be easily extracted from the mappings of an OBDA specification. Our approach is simple yet effective: as we show with an implemented proof of concept, it significantly improves the efficiency of Clipper on existing ontologies; these results are reported in Section 5.

2 Preliminaries

We recall the definition of Horn-$\mathcal{SHIQ}$ and the basics of OBDA.

The Description Logic Horn-$\mathcal{SHIQ}$. We consider countably infinite pairwise disjoint sets $\mathsf{N_C} \supset \{\top, \bot\}$ of *atomic concepts*, $\mathsf{N_R}$ of *role names*, and $\mathsf{N_I}$ of *individual names*. The set of *roles* is $\mathsf{N}_\mathsf{R}^\pm = \mathsf{N_R} \cup \{r^- \mid r \in \mathsf{N_R}\}$. If $r \in \mathsf{N_R}$, then $\mathsf{inv}(r) = r^-$ and $\mathsf{inv}(r^-) = r$. *Concepts* are inductively defined as follows: (a) each $A \in \mathsf{N_C}$ is a concept, and (b) if C, D are concepts and r is a role, then $C \sqcap D$, $C \sqcup D$, $\neg C$, $\forall r.C$, $\exists r.C$, $\geqslant n\, r.C$ and $\leqslant n\, r.C$, for $n \geq 1$, are concepts.

An expression $C \sqsubseteq D$, where C, D are concepts, is a *general concept inclusion axiom (GCI)*. An expression $r \sqsubseteq s$, where r, s are roles, is a *role inclusion (RI)*. A *transitivity axiom* is an expression *trans*(r), where r is a role. A TBox $\mathcal{T}$ is a finite set of GCIs, RIs and transitivity axioms. We let $\sqsubseteq^*_{\mathcal{T}}$ denote the reflexive transitive closure of $\{(r,s) \mid r \sqsubseteq s \in \mathcal{T} \text{ or } \mathsf{inv}(r) \sqsubseteq \mathsf{inv}(s) \in \mathcal{T}\}$. We assume w.l.o.g. that there are no $r \neq s$ in $\mathsf{N}^{\pm}_{\mathsf{R}}$ such that $r \sqsubseteq^*_{\mathcal{T}} s$ and $s \sqsubseteq^*_{\mathcal{T}} r$. A role s is *transitive* in $\mathcal{T}$ if *trans*$(s) \in \mathcal{T}$ or *trans*$(s^-) \in \mathcal{T}$. A role s is *simple* in $\mathcal{T}$ if there is no transitive r in $\mathcal{T}$ s.t. $r \sqsubseteq^*_{\mathcal{T}} s$.

A TBox $\mathcal{T}$ is a Horn-$\mathcal{SHIQ}$ TBox (in normalized form), if each GCI in $\mathcal{T}$ takes one the following forms:

$$\begin{array}{llll} \text{(F1)} & A_1 \sqcap \ldots \sqcap A_n \sqsubseteq B & \text{(F3)} & A_1 \sqsubseteq \forall r.B \\ \text{(F2)} & A_1 \sqsubseteq \exists r.B & \text{(F4)} & A_1 \sqsubseteq \leqslant 1\, r.B \end{array}$$

where $A_1, \ldots, A_n, B$ are concept names and r is a role, and all roles in concepts of the form $\leqslant 1\, r.B$ are simple. Axioms (F2) are called *existential*. W.l.o.g. we treat here only Horn-$\mathcal{SHIQ}$ TBoxes in normalized form; our results generalize to full Horn-$\mathcal{SHIQ}$ by means of TBox *normalization*; see e.g. [13, 14] for a definition and normalization procedures. A Horn-$\mathcal{ALCHIQ}$ TBox is a Horn-$\mathcal{SHIQ}$ TBox with no transitivity axioms, and Horn-$\mathcal{ALCHIQ}^{\sqcap}$ TBoxes are obtained by additionally allowing *role conjunction* $r_1 \sqcap r_2$, where r_1, r_2 are roles. We let $\mathsf{inv}(r_1 \sqcap r_2) = \mathsf{inv}(r_1) \sqcap \mathsf{inv}(r_2)$ and assume w.l.o.g. that for each role inclusion $r \sqsubseteq s$ of an Horn-$\mathcal{ALCHIQ}^{\sqcap}$ TBox $\mathcal{T}$, $s \in \mathsf{N}^{\pm}_{\mathsf{R}}$ and $\mathsf{inv}(r) \sqsubseteq \mathsf{inv}(s) \in \mathcal{T}$.

An assertion is $A(a)$ or $r(a,b)$, where $A \in \mathsf{N_C}$, $r \in \mathsf{N_R}$, and a, $b \in \mathsf{N_I}$. An ABox $\mathcal{A}$ is a finite set of assertions. Abusing notation, we may write $r(a,b) \in \mathcal{A}$ for $r \in \mathsf{N}^{\pm}_{\mathsf{R}}$, meaning $r(a,b) \in \mathcal{A}$ if $r \in \mathsf{N_R}$, and $\mathsf{inv}(r)(b,a) \in \mathcal{A}$ otherwise.

The semantics for TBoxes and ABoxes is given by *interpretations* $\mathcal{I} = \langle \Delta^{\mathcal{I}}, \cdot^{\mathcal{I}} \rangle$ which map each $a \in \mathsf{N_I}$ to some $a^{\mathcal{I}} \in \mathcal{I}$, each $A \in \mathsf{N_C}$ to some $A^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$, and each $r \in \mathsf{N_R}$ to some $r^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$, such that $\top^{\mathcal{I}} = \Delta^{\mathcal{I}}$, $\bot^{\mathcal{I}} = \emptyset$. The map $\cdot^{\mathcal{I}}$ is extended to all concepts and remaining roles as usual, and modelhood is defined in the usual way, see [2] for details. As common in OBDA, we make the unique name assumption (UNA), i.e., we require $a^{\mathcal{I}} \neq b^{\mathcal{I}}$ for all $a, b \in \mathsf{N_I}$ and every interpretation. We remark that our theoretical results also hold in the absence of the UNA, however, the proposed optimization trivializes in that case.

Databases and Ontology Based Data Access. A database schema $\mathcal{S}$ consists of a set of relations R and a set of functional dependencies (FD) $\mathcal{F}$. The columns of a relation R are identified by their positions $1, \ldots, n$. For a set $\boldsymbol{i}$ of columns of R, and a tuple t of R, $t[\boldsymbol{i}]$ denotes the projection of t over $\boldsymbol{i}$. An FD F over R has the form $R : \boldsymbol{i} \to \boldsymbol{j}$, where $\boldsymbol{i}$ and $\boldsymbol{j}$ are tuples of columns in R; we call each $j \in \boldsymbol{j}$ a functional attribute in F. This FD holds in an instance $\mathcal{D}$ if the values of $\boldsymbol{i}$ determine the values of $\boldsymbol{j}$, i.e. $\boldsymbol{t_1}[\boldsymbol{i}] = \boldsymbol{t_2}[\boldsymbol{i}]$ implies $\boldsymbol{t_1}[\boldsymbol{j}] = \boldsymbol{t_2}[\boldsymbol{j}]$ for every pair of tuples $\boldsymbol{t_1}$ and $\boldsymbol{t_2}$ such that $\{R(\boldsymbol{t_1}), R(\boldsymbol{t_2})\} \subseteq \mathcal{D}$.

An *OBDA specification* is a triple $\mathcal{P} = (\mathcal{T}, \mathcal{M}, \mathcal{S})$, where $\mathcal{T}$ is a TBox (in e.g, DL-Lite or Horn-$\mathcal{SHIQ}$), $\mathcal{S}$ is a database schema, $\mathcal{M}$ is a mapping consisting of mapping assertions that link predicates in $\mathcal{T}$ to queries over $\mathcal{S}$. The standard W3C language for mappings is R2RML [10], however here we use a more concise syntax that is common in the OBDA literature. Formally, a *mapping* $\mathcal{M}$ is a set of *mapping assertions* m that take the form

$$\text{conj}(\boldsymbol{y}) \rightsquigarrow X(\boldsymbol{f}, \boldsymbol{x})$$

consisting of a *source* part $\text{conj}(\boldsymbol{y})$, which is a conjunction of database atoms whose variables are $\boldsymbol{y}$, and a *target* part $X(\boldsymbol{f}, \boldsymbol{x})$, which is an atom whose predicate is X over terms built using function symbols $\boldsymbol{f}$ and variables $\boldsymbol{x} \subseteq \boldsymbol{y}$. In this paper $X(\boldsymbol{f}, \boldsymbol{x})$ takes either the form $C(f(\boldsymbol{x_1}))$ for a concept name C, or the form $r(f(\boldsymbol{x_1}), g(\boldsymbol{x_2}))$ for a role name r. We say that such mapping assertion m *defines the predicate* X. We use *body*(m) to refer to the source part $\text{conj}(\boldsymbol{y})$ of a mapping m as above, and *head*(m) to refer to its head $X(\boldsymbol{f}, \boldsymbol{x})$.

We make the following assumptions: *(i)* $\boldsymbol{a} \neq \boldsymbol{b}$ implies $f(\boldsymbol{a}) \neq f(\boldsymbol{b})$, for any f, and *(ii)* $f_1 \neq f_2$ implies $f_1(\boldsymbol{a}) \neq f_2(\boldsymbol{b})$, for any $\boldsymbol{a}, \boldsymbol{b}$. Both assumptions are in-line with the use of function symbols in OBDA systems [18], where they act as *templates* for producing a unique identifier for each input value. Assumption *(i)* is ensured in the R2RML standard using "safe separators", and although *(ii)* is not built into R2RML, it is assumed in existing OBDA tools like Ontop version 1 (implicitly in [20]).

For a database instance $\mathcal{D}$ and mapping $\mathcal{M}$, the ABox $\mathcal{M}(\mathcal{D})$ is the set of atoms $X(\boldsymbol{f}, \boldsymbol{a})$, for all $\text{conj}(\boldsymbol{y}) \rightsquigarrow X(\boldsymbol{f}, \boldsymbol{x}) \in \mathcal{M}$ and all tuples $\mathbf{a}$ of constants in $\mathcal{D}$ such that $\text{conj}(\mathbf{a})$ holds in $\mathcal{D}$. An *OBDA instance* is a pair $(\mathcal{P}, \mathcal{D})$, where $\mathcal{P}$ is an OBDA specification and $\mathcal{D}$ is a database instance that satisfies the dependencies in $\mathcal{S}$ from $\mathcal{P}$. The semantics of $(\mathcal{P}, \mathcal{D})$ is given by the models of $\mathcal{T}$ and $\mathcal{M}(\mathcal{D})$).

3 Restricting Horn-$\mathcal{SHIQ}$ Saturation

The query rewriting algorithm for Horn-$\mathcal{SHIQ}$ described in [11] first saturates a given TBox $\mathcal{T}$ by adding axioms, and then uses the saturated TBox to rewrite a given input query. The saturation step is critical and can be costly. Before describing how we can improve it using information about the ABox structure, we discuss in more detail how ABox-independence impacts scalability.

3.1 Bottleneck of ABox-independent saturation

The calculus of [11] is as shown in Table 1, but without the side conditions after ":" and rules Λ^*, Λ^+, Λ^-, which represent the core of our optimization discussed below. This algorithm is sound and complete for every possible ABox, and it can be implemented in a relatively simple way. However, it is computationally expensive. It is well-known that the algorithm is (unavoidably) worst-case exponential, but the problem is that this is not just a hypothetical worst-case: an unmanageable combinatorial explosion of inferred axioms may occur for realistic ontologies as well. Roughly, this is because there may be many axioms that are relevant for building the universal model of some ABox, but which are not relevant for the ABoxes we are interested in. We illustrate this through the example below:

Table 1. Optimized inference calculus. Possibly primed M, N are conjunctions of concept names, and S of roles; Λ is a set of activators and α, α' are activators in Λ. The calculus in [11] omits the side conditions and the rules Λ^*, Λ^+, Λ^-.

$(\mathbf{R}^c_{\sqsubseteq})$	$\dfrac{M \sqsubseteq \exists S.(N \sqcap N') \quad N \sqsubseteq A}{M \sqsubseteq \exists S.(N \sqcap N' \sqcap A)}$	$: \mathcal{M} \subseteq \alpha, \alpha \in \Lambda$
$(\mathbf{R}^r_{\sqsubseteq})$	$\dfrac{M \sqsubseteq \exists (S \sqcap S').N \quad S \sqsubseteq r}{M \sqsubseteq \exists (S \sqcap S' \sqcap r).N}$	$: M \subseteq \alpha, \alpha \in \Lambda$
$(\mathbf{R}_{\bot})$	$\dfrac{M \sqsubseteq \exists S.(N \sqcap \bot)}{M \sqsubseteq \bot}$	$: M \subseteq \alpha, \alpha \in \Lambda$
$(\mathbf{R}_{\forall})$	$\dfrac{M \sqsubseteq \exists (S \sqcap r).N \quad A \sqsubseteq \forall r.B}{M \sqcap A \sqsubseteq \exists (S \sqcap r).(N \sqcap B)}$	$: M \cup A \subseteq \alpha, \alpha \in \Lambda$
$(\mathbf{R}^-_{\forall})$	$\dfrac{M \sqsubseteq \exists (S \sqcap \mathsf{inv}(r)).(N \sqcap A) \quad A \sqsubseteq \forall r.B}{M \sqsubseteq B}$	$: M \subseteq \alpha, \alpha \in \Lambda$
$(\mathbf{R}_{\leq})$	$\dfrac{\begin{array}{l} M \sqsubseteq \exists (S \sqcap r).(N \sqcap B) \quad A \sqsubseteq\, \leqslant 1\, r.B \\ M' \sqsubseteq \exists (S' \sqcap r).(N' \sqcap B) \end{array}}{M \sqcap M' \sqcap A \sqsubseteq \exists (S \sqcap S' \sqcap r).(N \sqcap N' \sqcap B)}$	$: M \cup M' \cup A \subseteq \alpha, \alpha \in \Lambda$
$(\mathbf{R}^-_{\leq})$	$\dfrac{\begin{array}{l} M \sqsubseteq \exists (S \sqcap \mathsf{inv}(r)).(N_1 \sqcap N_2 \sqcap A) \quad A \sqsubseteq\, \leqslant 1\, r.B \\ N_1 \sqsubseteq \exists (S' \sqcap r).(N' \sqcap B \sqcap C) \end{array}}{M \sqcap B \sqsubseteq C \quad M \sqcap B \sqsubseteq \exists (S \sqcap \mathsf{inv}(S' \sqcap r)).(N_1 \sqcap N_2 \sqcap A)}$	$: M \cup B \subseteq \alpha, \alpha \in \Lambda$
(Λ^*) $\dfrac{M \sqsubseteq B \quad M \subseteq \alpha}{\Lambda = \Lambda \cup \{\alpha \cup \{B\}\}}$	(Λ^+) $\dfrac{M \sqsubseteq \exists R.N}{\Lambda = \Lambda \cup \{N\}}$	(Λ^-) $\dfrac{\alpha' \subseteq \alpha}{\Lambda = \Lambda \setminus \alpha'}$

Example 1. The following axioms stipulate different flags for monitored accounts:

$$\mathsf{MonitoredAccount} \sqsubseteq \exists \mathsf{hasFlag}.\top \qquad \mathsf{IndividualAcc} \sqsubseteq \forall \mathsf{hasFlag}.\mathsf{YellowFlag}$$

$$\mathsf{SmallBusinessAcc} \sqsubseteq \forall \mathsf{hasFlag}.\mathsf{RedFlag} \quad \mathsf{BigBusinessAcc} \sqsubseteq \forall \mathsf{hasFlag}.\mathsf{YellowFlag}$$

When running the calculus on this ontology we obtain several axioms of the form $\mathsf{MonitoredAccount} \sqcap C \sqsubseteq \exists \mathsf{hasFlag}.D$, where C is some conjunction of account types such as $\mathsf{IndividualAcc} \sqcap \mathsf{SmallBusinessAcc}$, $\mathsf{IndividualAcc} \sqcap \mathsf{BigBusinessAcc}$, etc., and D is some conjunction of flag colors. However, if we know that an account will only have one account type, we do not need all these axioms. If the ontology includes axioms stating the disjointness of account and flag types, the algorithm would discard some of the axioms after inferring them. Our approach, in contrast, is effective even when such axioms are not given, and allows us to save the computation of this axioms in advance if we know that an account is never declared to have two types in the data. Note also that the same role $\mathsf{hasFlag}$ could be used to flag something other than accounts, such as transactions, and then the calculus would also derive axioms for all combinations of types of transactions and types of accounts. Our observations suggest that such patterns are not uncommon, and obvious "common sense" disjointness assertions (such as saying that transactions are not accounts, or that accounts are not persons) are often omitted. This may not affect other reasoning techniques (such as tableau for consistency testing), but can have a major impact on this kind of ABox independent saturation.

3.2 Constraining the Derivation

We propose to use knowledge about the structure of relevant ABoxes to guide the calculus and avoid inferring irrelevant axioms as illustrated above.

Definition 1 (Propagating concepts and activators). *Concept names in* $\mathsf{N_C}$ *and expressions of the form* $\exists r$ *with* r *a role are called* basic concepts*.*

Let a *be an individual and* $\mathcal{A}$ *an ABox. The* ABox type of a in $\mathcal{A}$ *is the following set of basic concepts:*

$$atyp_{\mathcal{A}}(a) = \{A \mid A(a) \in \mathcal{A}\} \cup \{\exists r \mid r(a,b) \in \mathcal{A}\}$$

For a TBox $\mathcal{T}$ *and a set* τ *of basic concepts, the* $\mathcal{T}$-propagating concepts of τ *are:*

$$prop(\tau, \mathcal{T}) = \{B | A \sqsubseteq \forall s.B \in \mathcal{T}, r \sqsubseteq^*_{\mathcal{T}} s, \exists \mathsf{inv}(r) \in \tau\}$$

An activator α *is just a set of concept names. We say that a set* Λ *of activators* covers *an ABox* $\mathcal{A}$ *w.r.t. a TBox* $\mathcal{T}$ *if for each individual* a *occurring in* $\mathcal{A}$*, there is some activator* $\alpha \in \Lambda$ *such that*

$$\tau_a|_{\mathsf{N_C}} \cup prop(\tau_a, \mathcal{T}) \subseteq \alpha$$

where $\tau_a = atyp_{\mathcal{A}}(a)$ *and* $\tau_a|_{\mathsf{N_C}}$ *denotes its restriction to concept names only.*

Note that in covering sets of activators, we require that for each individual there is an activator that contains not only its type, but also its propagating concepts. In a way, this over-approximates its actual type in the universal model. Given a TBox $\mathcal{T}$, the singleton set that contains exactly the set of all concept names occurring in $\mathcal{T}$ is a set of activators that covers any ABox over the signature of $\mathcal{T}$. However, such a large activator would not prevent the derivation of irrelevant axioms in our calculus. For a concrete ABox we can be more accurate, and take as set of activators precisely the set of all $\alpha_a = \tau_a|_{\mathsf{N_C}} \cup prop(\tau_a, \mathcal{T})$ where a is an individual in $\mathcal{A}$ with $\tau_a = atyp_{\mathcal{A}}(a)$. In fact, we use such activators sets in our experiments in Section 5.

In Table 1 we present the optimized version of the calculus, which takes as input and maintains a set of activators. Each rule has a side condition that checks if the left hand side of the axiom we want to derive is contained in an existing activator. There are three additional rules Λ^*, Λ^+ and Λ^- not present in [11]. The rule Λ^* is used to close the maintained activators under axioms of the form $(F1)$, while Λ^+ is used to create fresh activators for inferred axioms of the form $(F2)$, and Λ^- drops redundant activators.

Before saturation, we drop transitivity axioms from the input TBox, and instead add axioms of the form (F3) that ensure the effect of transitivity is accounted for during saturation. The saturation starts from the resulting Horn-$\mathcal{ALCHIQ}$ TBox, but since it may introduce role conjunctions, it keeps a set of Horn-$\mathcal{ALCHIQ}^{\sqcap}$ axioms.

Definition 2. *Given an Horn-$\mathcal{SHIQ}$ ontology, let $\mathcal{T}^*$ be the result of dropping all transitivity axioms trans(r) from $\mathcal{T}$, and adding, for every $A \sqsubseteq \forall s.B \in \mathcal{T}$ and every transitive role r with $r \sqsubseteq^*_{\mathcal{T}} s$, the axioms $A \sqsubseteq \forall r.B^r$, $B^r \sqsubseteq \forall r.B^r$ and $B^r \sqsubseteq B$, where B^r is a fresh concept name.*

We denote by $\nabla(\mathcal{T}, \Lambda)$ the result of saturating $\mathcal{T}^*$ with the rules in Table 1 and set of initial activators Λ.

Formally $\nabla(\mathcal{T}, \Lambda)$ is a pair of an Horn-$\mathcal{ALCHIQ}^\sqcap$ TBox and a set of activators, but we may abuse notation and use $\nabla(\mathcal{T}, \Lambda)$ to denote the TBox alone.

Similarly as in [11], the saturated set of axioms contains all inferences from the ontology that are relevant for reasoning about any covered ABox; not only for checking consistency, but also for other problems like query rewriting.

Definition 3. *For an ABox $\mathcal{A}$, we denote by $\mathcal{A}^{\nabla(\mathcal{T},\Lambda)}$ the result of closing $\mathcal{A}$ under the following rules:*

- $A_1 \sqcap \ldots \sqcap A_n \sqsubseteq B \in \nabla(\mathcal{T}, \Lambda)$ *and* $\{A_1(a), \ldots A_n(a)\} \in \mathcal{A}$, *then* $B(a) \in \mathcal{A}$;
- $A \sqsubseteq \forall r.B \in \nabla(\mathcal{T}, \Lambda)$ *and* $r(a, b) \in \mathcal{A}, A(a) \in \mathcal{A}$, *then* $B(b) \in \mathcal{A}$;
- $r \sqsubseteq s \in \mathcal{T}$ *and* $r(a, b) \in \mathcal{A}$, *then* $s(a, b) \in \mathcal{A}$;
- $A \sqsubseteq\ \leqslant 1\, rB \in \mathcal{T}$ *and* $A(a), r(a, b), r(a, c), B(b), B(c)$, *then* $\bot(a) \in \mathcal{A}$
- $A_1 \sqcap \ldots \sqcap A_n \sqsubseteq \exists(r_1 \sqcap \ldots \sqcap r_m).(B_1 \sqcap \ldots \sqcap B_k)$, $A \sqsubseteq\ \leqslant 1\, r.B \in \nabla(\mathcal{T}, \Lambda)$ *such that for some* i, j *we have* $r{=}r_i, B{=}B_j$ *and* $A(a), r(a, b) \in \mathcal{A}$, *then* $\{B_1(b), \ldots, B_k(b), r_1(a, b), \ldots, r_k(a, b)\} \subseteq \mathcal{A}$.

We call $\mathcal{A}^{\nabla(\mathcal{T},\Lambda)}$ contradiction-free *if there are no assertions of the form* $\bot(a)$.

To test if a given ABox covered by Λ is consistent with $\mathcal{T}$, it is enough to check $\mathcal{A}^{\nabla(\mathcal{T},\Lambda)}$ for contradiction-freeness.

Proposition 1. *Let $\mathcal{A}$ be an ABox covered by Λ. Then $(\mathcal{T}, \mathcal{A})$ is consistent iff $\mathcal{A}^{\nabla(\mathcal{T},\Lambda)}$ is contradiction-free.*

However, we do not want to only test consistency. Our motivation is OBDA, and we want support for instance and conjunctive queries for different ABoxes. We thus provide the standard guarantee one would expect in this setting: from the computed axioms and a consistent ABox, we can build a *universal model*. As usual, the Horn-$\mathcal{ALCHIQ}^\sqcap$ TBox that results from saturation is used to build so-called *pre-models* with standard chase techniques, which become models once the extensions of non-simple roles are updated to satisfy transitivity axioms.

Definition 4 ($\mathcal{T}$-chase, universal model). *Let $\mathcal{T}$ be a Horn-$\mathcal{ALCHIQ}^\sqcap$ TBox and $\mathcal{I}$ an interpretation. We say that an axiom of the form $M \sqsubseteq \exists S.N$ is* applicable at $e \in \Delta^\mathcal{I}$ *if*

(a) $e \in M^\mathcal{I}$,

(b) there is no $e' \in \Delta^\mathcal{I}$ *with* $(e, e') \in S^\mathcal{I}$ *and* $e' \in N^\mathcal{I}$,

(c) there is no axiom $M' \sqsubseteq \exists S'.N' \in \mathcal{T}$ *such that* $e \in (M')^\mathcal{I}$, $S \subseteq S'$, $N \subseteq N'$, *and* $S \subset S'$ *or* $N \subset N'$.

If $M \sqsubseteq \exists S.N$ is applicable at $e \in \Delta^\mathcal{I}$, then we obtain an interpretation $\mathcal{I}'$ by applying $M \sqsubseteq \exists S.N$ in $\mathcal{I}$ as follows:

- $\Delta^{\mathcal{I}'} = \Delta^{\mathcal{I}} \cup \{e'\}$ *with* e' *a new element not present in* $\Delta^{\mathcal{I}}$,
- *for each* $A \in \mathsf{N_C}$, *we have* $A^{\mathcal{I}'} = A^{\mathcal{I}} \cup \{e'\}$ *if* $A \in N$, *and* $A^{\mathcal{I}'} = A^{\mathcal{I}}$ *otherwise,*
- *for each* $r \in \mathsf{N_R}$, *we have* $r^{\mathcal{I}'} = r^{\mathcal{I}} \cup \{(e, e')\}$ *if* $r \in S$, $r^{\mathcal{I}'} = r^{\mathcal{I}} \cup \{(e', e)\}$ *if* $r^- \in S$, $r^{\mathcal{I}'} = r^{\mathcal{I}}$ *otherwise.*

For a contradiction-free ABox $\mathcal{A}$, *we let* $\mathcal{I}^{\mathcal{A}}$ *denote the interpretation whose domain are the individuals in* $\mathcal{A}$, *and that has* $A^{\mathcal{I}^{\mathcal{A}}} = \{a \mid A(a) \in \mathcal{A}\}$ *for all* $A \in \mathsf{N_C}$, *and* $r^{\mathcal{I}^{\mathcal{A}}} = \{(a, b) \mid r(a, b) \in \mathcal{A}\}$ *for all* $r \in \mathsf{N_R}$.

The $\mathcal{T}$*-chase of a contradiction-free ABox* $\mathcal{A}$ *is the possibly infinite interpretation obtained from* $\mathcal{I}^{\mathcal{A}}$ *by fairly applying the existential axioms in* $\mathcal{T}$ *(that is, every applicable axiom is eventually applied).*

Let $\mathcal{J}$ *denote the* $\mathcal{T}$*-chase of* $\mathcal{A}^{\nabla(\mathcal{T},\Lambda)}$. *Then* $\mathcal{I}^{(\mathcal{T},\Lambda,\mathcal{A})}$ *is the interpretation with* $\Delta^{(\mathcal{T},\Lambda,\mathcal{A})} = \Delta^{\mathcal{J}}$, $A^{\mathcal{I}^{(\mathcal{T},\Lambda,\mathcal{A})}} = A^{\mathcal{J}}$ *for every* $A \in \mathsf{N_C}$, *and*

$$r^{\mathcal{I}^{(\mathcal{T},\Lambda,\mathcal{A})}} = \bigcup_{s \sqsubseteq^*_{\mathcal{T}} r} s^{\mathcal{J}}_{+} \qquad \text{for every } r \in \mathsf{N}^{\pm}_{\mathsf{R}}$$

where $s^{\mathcal{J}}_{+}$ *is the transitive closure of* $s^{\mathcal{J}}$ *if* $trans(s) \in \mathcal{T}$, *and* $s^{\mathcal{J}}_{+} = s^{\mathcal{J}}$ *otherwise.*

It is standard to show that $\mathcal{I}^{(\mathcal{T},\Lambda,\mathcal{A})}$ is indeed a universal model:

Proposition 2. *Let* $(\mathcal{T}, \mathcal{A})$ *be a consistent Horn-*$\mathcal{SHIQ}$ *KB, and* Λ *a set of activators, such that* Λ *covers* $\mathcal{A}$ *w.r.t.* $\mathcal{T}$. *The following hold:*

(a) if $(\mathcal{T}, \mathcal{A})$ *is consistent then* $\mathcal{I}^{(\mathcal{T},\Lambda,\mathcal{A})} \models (\mathcal{T}, \mathcal{A})$, *and*
(b) $\mathcal{I}^{(\mathcal{T},\Lambda,\mathcal{A})}$ *can be homomorphically embedded into any model of* $\mathcal{K}$.

This proposition is analogous to Proposition 2 in [11]. By guaranteeing that we can build a universal model of any ABox that is consistent with $\mathcal{T}$, we can use $\nabla(\mathcal{T}, \Lambda)$ as a representation of models that is sufficient for query answering. As in the work of Eiter et al., the finite $\nabla(\mathcal{T}, \Lambda)$ allows us to rewrite the query in such a way that it can be evaluated over a small and easy to compute part of the possibly infinite represented universal model, we refer to [11] for details.

Note that the output of the original algorithm in [11] coincides with $\nabla(\mathcal{T}, \Lambda)$ if Λ contains only the set of all the concept names appearing in $\mathcal{T}$. In terms of computational complexity, the same upper bounds apply for the size of the saturated sets obtained with either version of the calculus: it may be single exponential in $\mathcal{T}$, and this exponential blow-up is in general unavoidable. But as we discuss in Section 5.1, in practice the version with activators is faster, builds smaller sets, and can handle more ontologies.

Incremental Computation. Assume that we have computed $\nabla(\mathcal{T}, \Lambda)$, and let $\mathcal{T}_1$ and Λ_1 be the obtained TBox and set of activators. Now, suppose that we want to query an ABox $\mathcal{A}$ that is not covered by Λ (this could be, for example, because the underlying OBDA specification changed). Then we can simply take a set Λ' of activators that covers the new ABox profiles. To compute $\mathcal{T}_2 = \nabla(\mathcal{T}, \Lambda \cup \Lambda')$, we can reuse the previous output and simply compute $\mathcal{T}_2 = \nabla(\mathcal{T}_1, \Lambda_1 \cup \Lambda')$. All axioms in $\mathcal{T}_1$ are preserved, and new ones may be derived.

4 Activators from Mappings

In this section we show how one can obtain initial activators for our optimized calculus directly from an OBDA specification $\mathcal{P} = (\mathcal{T}, \mathcal{M}, \mathcal{S})$, so that the extracted activators cover the ABox $\mathcal{M}(\mathcal{D})$ for any database $\mathcal{D}$. In this section, we will rely on our assumptions about the function symbols in OBDA specifications, namely, that $\boldsymbol{a} \neq \boldsymbol{b}$ implies $f(\boldsymbol{a}) \neq f(\boldsymbol{b})$ for every f, and that $f_1 \neq f_2$ implies $f_1(\boldsymbol{a}) \neq f_2(\boldsymbol{b})$, for any $\boldsymbol{a}, \boldsymbol{b}$.

One simple way to obtain the activators would be take as an ABox type all the head predicates of mappings that share the same functional symbol, roughly treating each function symbol as the same constant in the ABox. After all, if all the mappings that share some $f(x)$ in the head would fire for some x, they can all yield assertions for the same individual. Such an approach would be complete, but could potentially generate quite large ABox types. As we report in Section 5.2, we observed that real-world specifications do contain mappings that share the same function symbol in the head, but frequently they cannot fire for the same value of x due to the functional dependencies. To leverage this and obtain a more fine-grained set of activators, we first define conflicting mappings.

Definition 5 (Conflicting mapping assertions). *Let $F = R : \boldsymbol{i} \to \boldsymbol{j}$ a functional dependency and let $j \in \boldsymbol{j}$ be one of its functional attributes. We call a pair m, m' of mapping assertions (F, j)-*conflicting *if the following hold:*

– *there are terms $f(\boldsymbol{x})$ in head(m) and $f(\boldsymbol{y})$ in head(m'), for some function symbol f, and*
– *there are atoms $R(\boldsymbol{t}) \in$ body(m) and $R(\boldsymbol{t'}) \in$ body(m') such that, for each $i \in \boldsymbol{i}$ we have $\boldsymbol{t}[i] = x_i$, $\boldsymbol{t'}[i] = y_i$, where $x_i \in \boldsymbol{x}$, $y_i \in \boldsymbol{y}$, and there exists a $j \in \boldsymbol{j}$, such that $\boldsymbol{t}[j]$, $\boldsymbol{t'}[j]$ are two different constants.*

Example 2. Consider the following $\mathcal{M}$ for the ontology from Example 1.

$$\begin{aligned}
m_1 : \mathsf{transfers}(x, y) &\rightsquigarrow \mathsf{interactedWith}(f_1(x), f_1(y)),\\
&\ \mathsf{Account}(f_1(x)), \mathsf{Account}(f_1(y))\\
m_2 : \mathsf{account\_owners}(x, y, `politician') &\rightsquigarrow \mathsf{PEP}(f_2(x))\\
m_3 : \mathsf{account\_owners}(x, y, z) &\rightsquigarrow \mathsf{hasOwner}(f_1(x), f_2(y))\\
m_4 : \mathsf{account\_details}(x, `business', `big') &\rightsquigarrow \mathsf{BigBusinessAcc}(f_1(x))\\
m_5 : \mathsf{account\_details}(x, `business', `small') &\rightsquigarrow \mathsf{SmallBusinessAcc}(f_1(x))\\
m_6 : \mathsf{account\_details}(x, `private', y) &\rightsquigarrow \mathsf{IndividualAcc}(f_1(x))
\end{aligned}$$

Now consider a functional dependency $F_1 : id \to type, size$ over the relation $\mathsf{account\_details}(id, type, size)$. Then, according to Definition 5, pairs m_4,m_6 and m_5,m_6 are $(F_1, type)$-conflicting, while the pair m_4,m_5 is $(F_1, size)$-conflicting.

Note that we define conflicts in a way that they are easy to identify, and that we can guarantee that conflicting mapping assertions cannot fire to create assertions about the same constant. There may be other reasons why two mappings do not fire together that we disregard, but this does not compromise the correctness of our approach, it may simply result in larger activators, which may lead to more irrelevant inferences.

For a functional symbol f, we denote by $\mathcal{M}(f)$ the set of all mapping assertions in $\mathcal{M}$ such that f occurs in the head. A subset $\mathcal{M}'$ of $\mathcal{M}(f)$ is conflict-free if there are no mapping assertions m and m' in $\mathcal{M}'$ that are (F, j)-conflicting for some F and j. With $\mathbb{M}_f$ we denote the set of maximal conflict-free subsets of $\mathcal{M}(f)$. Then we can guarantee the coverage of $\mathcal{M}(\mathcal{D})$ by creating an ABox type and an activator for each function symbol f and each $M_i \in \mathbb{M}_f$.

The problem of computing maximal conflict-free subsets can be solved by using the notions of maximal cliques from graph theory and the hitting set problem. Recall that a clique in an undirected graph is a subset of the vertices such that every two distinct vertices are adjacent; a maximal clique is a clique that cannot be extended by adding one more vertex. For a set of sets Ω, H is a hitting set of Ω if for all $S \in \Omega, H \cap S \neq \emptyset$; a hitting set H is minimal if there exists no other hitting set H', such that $H' \subseteq H$. To compute $\mathbb{M}_f$, we first create a graph G_f where the node set is $\mathcal{M}(f)$ and the edge set is $\{(m, m') \mid m, m'$ are (F, j)-conflicting for some $(F, j)\}$. Next let Ω_f be the set of maximal cliques of G_f. Note that each set in Ω_f also includes the set of conflict free mapping assertions. Then every minimal hitting set of Ω_f is a maximal conflict-free subset of $\mathcal{M}(f)$. One can use any hitting set algorithm, e.g. [21], for this task. We note that despite the lack of tractability, there are efficient algorithms available for the maximal clique and minimal hitting set problems, and in the sizes of the generated instances in this case are relatively small. Moreover, the minimality of the hitting sets is not so critical, an efficient approximation algorithm can also be employed.

Definition 6 (Activators from an OBDA specification). *Given an OBDA specification $\mathcal{P} = (\mathcal{T}, \mathcal{M}, \mathcal{S})$, let f be a function symbol occurring in $\mathcal{M}$, and let $\mathbb{M}_f$ be the set of maximal conflict-free subsets of $\mathcal{M}(f)$. Then we define, for each $M_i \in \mathbb{M}_f$:*

$$\begin{aligned} atyp_{\mathcal{M}}(f, M_i) = & \{A \mid \textit{some } m \in M_i \textit{ has } head(m) \textit{ of the form } A(f(x))\} \cup \\ & \{\exists r \mid \textit{some } m \in M_i \textit{ has } head(m) \textit{ of the form } r(f(x), t)\} \cup \\ & \{\exists r^- \mid \textit{some } m \in M_i \textit{ has } head(m) \textit{ of the form } r(t, f(x))\} \end{aligned}$$

Finally, we denote by $\Lambda(\mathcal{P})$ the set of activators whose elements are, for each function symbols f occurring in $\mathcal{M}$ and each $M_i \in \mathbb{M}_f$:

$$atyp_{\mathcal{M}}(f, M_i)|_{\mathsf{N_C}} \cup prop(atyp_{\mathcal{M}}(f, M_i), \mathcal{T})$$

where $atyp_{\mathcal{M}}(f, M_i)|_{\mathsf{N_C}}$ denotes the restriction of $atyp_{\mathcal{M}}(f, M_i)$ to $\mathsf{N_C}$ only.

Using the fact that, for all $\mathcal{D}$, each assertion in $\mathcal{M}(\mathcal{D})$ comes from some mapping in $\mathcal{M}$, and that by our assumptions only non-conflicting mappings that share a function symbol can produce assertions about a common individual, the following is not hard to show.

Proposition 3. *For every OBDA instance $(\mathcal{P}, \mathcal{D})$, $\Lambda(\mathcal{P})$ covers the ABox $\mathcal{M}(\mathcal{D})$.*

Example 3. Let's consider our running example. For the functional symbol f_1, the graph $G_{f_1} = (E_{f_1}, V_{f_1})$, where $E_{f_1} = \{m_1, m_3, m_4, m_5, m_6\}$, and $V_{f_1} = \{(m_4, m_5), (m_4, m_6), (m_5, m_6)\}$. The maximal cliques are $\Omega_{f_1} = \{\{m_1\}, \{m_3\}, \{m_4, m_5, m_6\}\}$. Thus, the maximal conflict-free sets $\mathbb{M}_{f_1}$ are the minimal hitting sets of Ω_{f_1}, i.e. $\mathbb{M}_{f_1} = \{M_1, M_2, M_3\}$, where $M_1 = \{m_1, m_3, m_4\}$, $M_2 = \{m_1, m_3, m_5\}$, $M_3 = \{m_1, m_3, m_6\}$. Similarly, the maximal conflict-free sets for f_2 is $\mathbb{M}_{f_2} = \{M_4\}$ where $M_4 = \{m_2, m_3\}$. Then by Definition 6 we get:

$\mathsf{atyp}_{\mathsf{f}_1}(M_1) = \{\exists\mathsf{interactedWith}, \exists\mathsf{interactedWith}^-, \mathsf{Account}, \mathsf{hasOwner}, \mathsf{BigBusinessAcc}\}$
$\mathsf{atyp}_{\mathsf{f}_1}(M_2) = \{\exists\mathsf{interactedWith}, \exists\mathsf{interactedWith}^-, \mathsf{Account}, \mathsf{hasOwner}, \mathsf{BigBusinessAcc}\}$
$\mathsf{atyp}_{\mathsf{f}_1}(M_3) = \{\exists\mathsf{interactedWith}, \exists\mathsf{interactedWith}^-, \mathsf{Account}, \mathsf{hasOwner}, \mathsf{IndividualAcc}\}$
$\mathsf{atyp}_{\mathsf{f}_2}(M_4) = \{\exists\mathsf{hasOwner}^-, \mathsf{Politician}\}$

and, the following set of activators:

$$\begin{aligned}\Lambda = \{&\{\mathsf{Account}, \mathsf{BigBusinessAcc}\}, \{\mathsf{Account}, \mathsf{SmallBusinessAcc}\},\\ &\{\mathsf{Account}, \mathsf{IndividualAcc}\}, \{\mathsf{Politician}\}\}\end{aligned}$$

Note that with this Λ, the $\mathbf{R}_\forall$ rule of the optimized calculus will not derive the irrelevant axioms discussed in Example 1.

5 Evaluation

We implemented the optimized calculus in Table 1 in the Clipper reasoner. From here on we refer to Clipper with C-orig and to our implementation with C-opt. We tested the feasibility of our approach in two directions:

Optimized TBox saturation We tested C-opt on a large test set of ontologies from the Oxford Ontology Repository,[4] and compared its performance to C-orig. The activators that were given as input to C-opt were obtained from the respective ABoxes that these ontologies include.

Activator extraction from mappings We extracted activators from three large OBDA specifications, and analysed the resulting activators.

We remark that, on the one hand, the OBDA specifications that we used in the second part come with *DL-Lite* ontologies, for which the combinatorial behaviour of Horn-$\mathcal{SHIQ}$ does not arise and C-opt brings no improvement. The expressive ontologies of the Oxford repository, on the other hand, have no accompanying mapping specifications. The lack of test cases with both an expressive ontology and realistic mappings, while unfortunate, is not surprising, as existing OBDA engines only support *DL-Lite*.

The test ontologies, the compiled C-opt, and files with the mapping analysis can be found in `https://github.com/ghxiao/clipper-materials/tree/master/iswc-2019`.

5.1 Optimized TBox Saturation

The instances for these tests came from Oxford Ontology Repository, which contains 797 ontologies. From those, only 370 had ABoxes, and out of them

[4] http://www.cs.ox.ac.uk/isg/ontologies/UID/

Table 2. Distribution of ontologies by their respective ABox and TBox sizes.

		TBox Sizes				
		S	**M**	**L**	**VL**	**Total**
ABox Sizes	**S**	5.12%	6.51%	4.65%	0.47%	16.75%
	M	0%	9.3%	3.72%	0.93%	13.95%
	L	0%	5.12%	12.09%	0.47%	17.68%
	VL	0.47%	11.63%	14.88%	24.64%	51.62%
	Total	5.59%	32.56%	35.34%	26.51%	

18 yielded exceptions while loading on C-orig. From the remaining ontologies, 131 were uninteresting since their normalized TBoxes did not contain existential axioms (F2) in which case the saturation step trivializes. From the resulting 221 ontologies we dropped all axioms not expressible in Horn-$\mathcal{SHIQ}$. Table 2 shows the distribution of our 221 ontologies with respect to TBox and ABox size. We categorized them into (S)mall, (M)edium, (L)arge and (V)ery (L)arge, with boundaries of up to 100, up to 1000, up to 10000, and above 10000 axioms/assertions. There is a fair mix of sizes, and around half of the ontologies have both ABox and TBox that are large or very large.

All experiments were run on a PC with an Intel i7 2.4 GHz CPU with 4 cores running 64 bit LinuxMint 17, and a Java heap of 4 GB. A time-out limit of 2 minutes was set. This was the total time allowed for loading and normalizing the TBox, saturating it, and in the case of C-opt, also the time used to obtain activators from the ABox.

Additionally to successfully saturating all ontologies that C-orig succeeded on, C-opt showed a 37.96% increase in the success rate: C-opt succeeded in 149 out of 221 (67.71%), while C-orig in 108 out of 221 (49.33%). Our of the 221 ontologies, 52 are in the *DL-Lite* or $\mathcal{EL}$ profiles. For them, C-opt succeeded in 49 vs 48 for C-orig. If we take into account only ontologies in more expressive fragments, beyond *DL-Lite* and $\mathcal{EL}$, the performance improvement is even more pronounced: our C-opt succeeded in 100 cases our of 169 ontologies (59.17%), while C-orig succeeds only in 60 cases (35.5%), resulting in an increase of 66.67% in the success rate.

In what follows, we zoom in into three aspects of these tests. First we make a more fine grained comparison of C-orig and C-opt, by comparing their behavior on the 108 ontologies that both succeeded on **(P1)**. Then we look at the performance of C-opt on the 41 ontologies that it succeeded on while C-orig did not **(P2)**. Finally, to shed some light on the limits of our optimization, we look more closely at the ontologies that C-opt could not saturate **(P3)**.

(P1) We compare the sizes of the saturated sets and the run times of both versions, on the 108 ontologies that both succeeded on. To understand how much smaller the saturated set is in the optimized version, independently of the size of the original TBox, we show the *TBox rate of growth* given by the number of axioms of (F1)-(F2) in the saturated TBox, divided by their total in the initial (normalized) TBox. Figure 1 depicts the TBox rate of growth for all ontologies;

Fig. 1. TBox rate of growth for both versions (C-ORIG =red, C-OPT =blue).

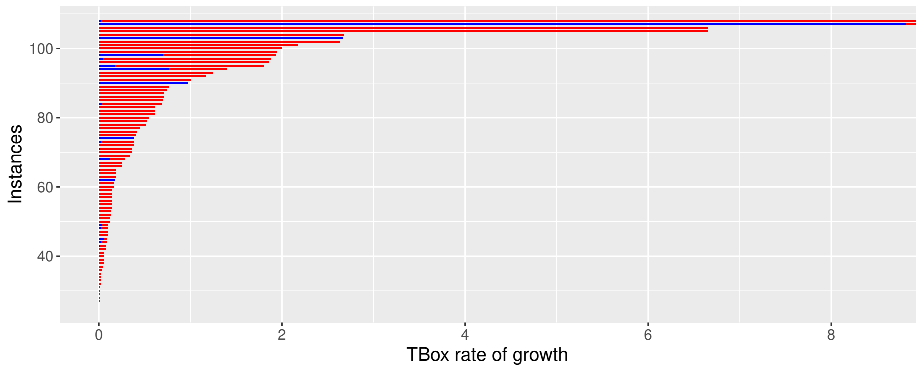
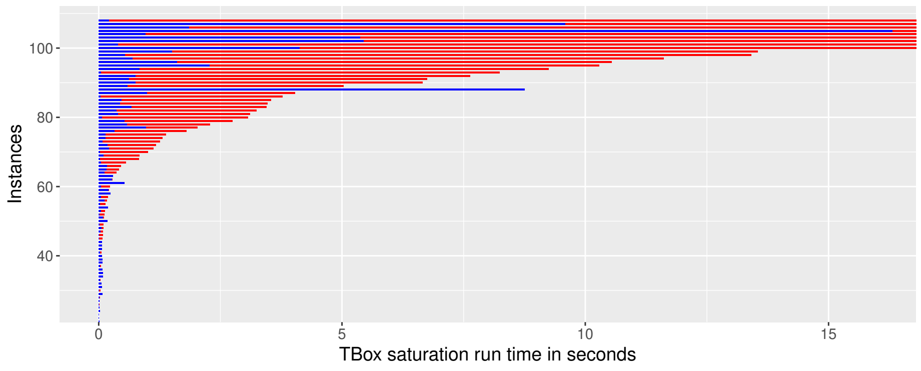


Fig. 2. TBox saturation run time for both versions (C-ORIG =red, C-OPT =blue).



blue bars for C-OPT are plotted over red bars for C-ORIG. Note that the y-axis is simply the 108 ontologies, ordered by the rate of growth of C-ORIG for better visualization. The x-axis is cut-off; the rate of growth of C-ORIG in fact reached a 20-fold growth. The rate of growth of the optimized version was nearly always smaller, often just a small fraction of the original, in the worst case they are equal. For 88 ontologies instances the rate of growth for C-OPT was 0, i.e. no new axioms were derived (see the bars without blue color). This means that all the axioms derived by C-ORIG were irrelevant for the ABox in the ontology, and for any ABox covered by the same profiles.

Figure 2 shows the run time of the TBox saturation (x-axis) over all ontologies (y-axis, ordered by runtime of C-ORIG). Similarly as in Figure 1, the run time of C-OPT (blue) is plotted on top of C-ORIG (red). C-OPT outperformed C-ORIG in most of the cases. With one exception, these were ontologies where C-OPT was so fast (typically under 100 milliseconds), that the overhead of handling the activators did not pay off.

(P2) The growth rate and saturation run times of the 41 ontologies gained with C-OPT are shown in the Figure 3. In both graphs, the ontologies are ordered by the TBox rate of growth. The left graph shows that the rate of growth for these

Fig. 3. Gained instances with C-OPT.

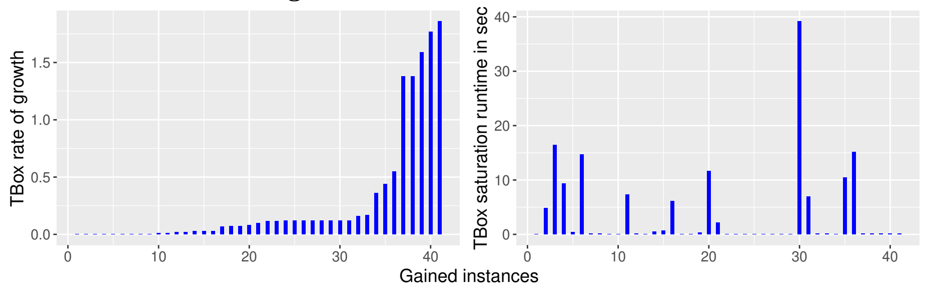
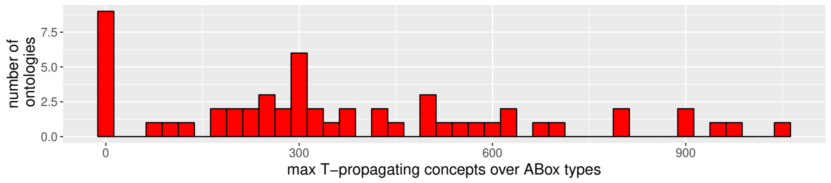


Fig. 4. Failed instances with both versions.



ontologies is in line with the growth reported in **(P1)**, remaining below double the original size even in the worst-case. On the right we see that the run time for most ontologies was under 10 seconds, also in line with the run times in **(P1)**.

(P3) We analysed the ontologies that we could not saturate, and observed that the maximal size of $\mathcal{T}$-propagating concepts over ABox types plays a key role, for both C-ORIG and C-OPT. C-ORIG always failed when this number was above 20. C-OPT could handle some ontologies with up to almost a hundred, but failed in all ontologies with more than that. The maximal size of $\mathcal{T}$-propagating concepts over ABox types for all ontologies C-OPT failed on is shown in Figure 4. Note that there are a few ontologies with no propagating concepts for which both versions fail. These are hard to saturate for other reasons not related to our optimization.

From the evidence in **(P1)**–**(P3)**, we can conclude that our approach yields improvements in three dimensions: the number of ontologies we can saturate, the run time of the calculus, and the size of the resulting saturated TBox.

5.2 Analysis of Activators Obtained from OBDA Specifications

With the purpose of understanding the feasibility of our approach when ABoxes come from real OBDA specifications, we analysed three existing benchmarks that have large specifications: NPD [15] (1173 mapping assertions), Slegge [12] (62 mapping assertions), and UOBM [4] (96 mapping assertions). The latter is synthetic, while the other two are from real-world scenarios. As already discussed, these OBDA specifications are paired with *DL-Lite* ontologies for which testing C-OPT is not meaningful. Our main goal here was to understand how the activators obtained from OBDA specifications look, and to verify if they are in line with the activators from ABoxes that we used for testing C-OPT.

Using the approach in Section 4, that exploits functional dependencies, we obtained a set of activators where the largest set has size 9 for NPD, 3 for Slegge and 11 for UOBM. We expect these activator sizes to be quite manageable for C-opt, since its average runtime was under 2 seconds over all ontologies from the Oxford repository with similar ABoxes (namely, 5 or more assertions in the ABox type of some individual). We remark than in all these cases, since the ontologies are simple there are no propagating concepts, so the activators coincide with the actual ABox types that the mappings induce.

The size of the obtained activator sets was over 600 for for NPD, but only 4 for Slegge and 12 for UOBM. We note that the number of mappings sharing the same function symbol f in the head was up to 30 for NPD, 4 for Slegge, and 15 for UOBM, hence taking only one activator per function symbol would probably result in a rather poor performance of C-opt.

6 Conclusion

In this paper we proposed an optimization of Horn-$\mathcal{SHIQ}$ reasoning that can be useful for OBDA. We illustrated the problems of current ABox-independent approaches to TBox saturation, which often manifests exponential behaviour, and proposed a way to overcome this. In a nutshell, we avoid the derivation of axioms that are useless since they consider combinations of concepts that can not occur in the real data. We achieve this by constraining the axiom derivation with *activators* that reflect the possible structure of the data. We implemented our approach as an optimization of the Clipper reasoner [11], which scales well in general, and can handle considerably more ontologies than the original Clipper. Crucially, the ABox structure information that is needed in our approach can be obtained directly from the mapping assertions of an OBDA specification; we corroborated this on existing OBDA specifications. We hope this work brings closer the goal of realizing OBDA with ontologies beyond DL-Lite.

In future work, we plan to explore more fine-grained algorithms for extracting activators from mappings, and more generally, to further develop approaches to optimize reasoning techniques that are almost data-independent, but that allow to leverage some general features of the data to improve performance.

References

1. Baader, F., Bienvenu, M., Lutz, C., Wolter, F.: Query and predicate emptiness in ontology-based data access. J. Artif. Intell. Res. **56**, 1–59 (2016)
2. Baader, F., Calvanese, D., McGuinness, D., Nardi, D., Patel-Schneider, P. (eds.): The Description Logic Handbook: Theory, Implementation, and Applications. Cambridge University Press, second edn. (2007)
3. Bienvenu, M., Ortiz, M.: Ontology-mediated query answering with data-tractable description logics. In: Reasoning Web. Web Logic Rules - 11th International Summer School 2015, Berlin, Germany, July 31 - August 4, 2015, Tutorial Lectures. pp. 218–307 (2015)

4. Botoeva, E., Calvanese, D., Santarelli, V., Savo, D.F., Solimando, A., Xiao, G.: Beyond OWL 2 QL in OBDA: rewritings and approximations. In: Proc. of AAAI. pp. 921–928. AAAI Press (2016)
5. Botoeva, E., Kontchakov, R., Ryzhikov, V., Wolter, F., Zakharyaschev, M.: Games for query inseparability of description logic knowledge bases. Artif. Intell. **234**, 78–119 (2016)
6. Botoeva, E., Lutz, C., Ryzhikov, V., Wolter, F., Zakharyaschev, M.: Query inseparability for ALC ontologies. CoRR **abs/1902.00014** (2019)
7. Calvanese, D., De Giacomo, G., Lembo, D., Lenzerini, M., Rosati, R.: Tractable reasoning and efficient query answering in description logics: The *DL-Lite* family. J. Autom. Reasoning **39**(3), 385–429 (2007)
8. Carral, D., Dragoste, I., Krötzsch, M.: The combined approach to query answering in Horn-ALCHOIQ. In: Proc. of KR. pp. 339–348. AAAI Press (2018)
9. Carral, D., González, L., Koopmann, P.: From Horn-SRIQ to datalog: A data-independent transformation that preserves assertion entailment. In: Proc. of AAAI. AAAI Press (2019)
10. Das, S., Sundara, S., Cyganiak, R.: R2RML: RDB to RDF mapping language. W3C recommendation, W3C (2012)
11. Eiter, T., Ortiz, M., Simkus, M., Tran, T., Xiao, G.: Query rewriting for Horn-SHIQ plus rules. In: Proc. of AAAI. pp. 726–733. AAAI Press (2012)
12. Hovland, D., Kontchakov, R., Skjæveland, M., Waaler, A., Zakharyaschev, M.: Ontology-based data access to slegge. In: Proc. of ISWC. pp. 120–129. Springer (2017)
13. Kazakov, Y.: Consequence-driven reasoning for Horn SHIQ ontologies. In: Proc. of IJCAI. pp. 2040–2045 (2009)
14. Krötzsch, M., Rudolph, S., Hitzler, P.: Complexity boundaries for Horn description logics. In: Proc. of AAAI. pp. 452–457. AAAI Press (2007)
15. Lanti, D., Rezk, M., Xiao, G., Calvanese, D.: The NPD benchmark: Reality check for OBDA systems. In: Proc. of EDBT. ACM Press (2015)
16. Leone, N., Manna, M., Terracina, G., Veltri, P.: Fast query answering over existential rules. ACM Trans. Comput. Log. **20**(2), 12:1–12:48 (2019). https://doi.org/10.1145/3308448, `https://doi.org/10.1145/3308448`
17. Lutz, C., Walther, D., Wolter, F.: Conservative extensions in expressive description logics. In: Proc. of IJCAI. pp. 453–458 (2007)
18. Poggi, A., Lembo, D., Calvanese, D., De Giacomo, G., Lenzerini, M., Rosati, R.: Linking data to ontologies. J. Data Semantics **10**, 133–173 (2008)
19. Ren, Y., Pan, J.Z., Zhao, Y.: Soundness preserving approximation for tbox reasoning. In: Proc. of AAAI. pp. 351–356. AAAI Press (2010)
20. Rodriguez-Muro, M., Rezk, M.: Efficient SPARQL-to-SQL with R2RML mappings. J. Web Semant. **33**, 141–169 (2015)
21. Shi, L., Cai, X.: An exact fast algorithm for minimum hitting set. In: Third International Joint Conference on Computational Science and Optimization. vol. 1, pp. 64–67 (2010)
22. Xiao, G., Calvanese, D., Kontchakov, R., Lembo, D., Poggi, A., Rosati, R., Zakharyaschev, M.: Ontology-based data access: A survey. In: Proc. of IJCAI. pp. 5511–5519 (2018)
23. Xiao, G., Ding, L., Cogrel, B., Calvanese, D.: Virtual knowledge graphs: An overview of systems and use cases. Data Intelligence **1**, 201–223 (2019)