Virtual OBDA over Expressive Ontologies: Rewritings and Approximations

E. Botoeva\textsuperscript{1}, D. Calvanese\textsuperscript{1}, V. Santarelli\textsuperscript{2}, D. F. Savo\textsuperscript{2}, A. Solimando\textsuperscript{3}, and G. Xiao\textsuperscript{1}

\textsuperscript{1} Free University of Bozen-Bolzano, Italy, lastname@inf.unibz.it
\textsuperscript{2} Sapienza Università di Roma, Italy, lastname@dis.uniroma1.it
\textsuperscript{3} DIBRIS, University of Genova, Italy, alessandro.solimando@unige.it

Abstract. In this paper we extend virtual OBDA to more expressive ontology languages than that of DL-Lite\textsubscript{R}, which is the logic underpinning the W3C ontology language OWL 2. We achieve this by relying on two well-known mechanisms, namely conservative rewriting and approximation, and compiling some of the domain semantics into OBDA mappings.

1 Introduction

Ontology-Based Data Access (OBDA) is a popular paradigm that enables end users to access data sources through an ontology, abstracting away low-level details of the data sources themselves. The ontology provides a high-level description of the domain of interest, and is semantically linked to the data sources by means of a set of mapping assertions [7, 16]. Typically, the data sources are represented as relational data, the ontology is a set of logical axioms over concepts and roles, and each mapping assertion relates an SQL query over the database to a concept or role of the ontology.

Making OBDA work efficiently over large amounts of data, requires that query answering over the ontology is first-order (FO)-rewritable\textsuperscript{1} [8, 1], which in turn limits the expressiveness of the ontology language, and the degree of detail with which the domain of interest can be captured. The current language of choice for OBDA is DL-Lite\textsubscript{R}, the logic underlying OWL 2 QL [24], which has been specifically designed to ensure FO-rewritability of query answering. Hence, it does not allow one to express disjunctive information, or any form of recursion on the data (e.g., as resulting from qualified existentials on the left-hand side of concept inclusions), since using such constructs in general causes the loss of FO-rewritability [9]. For this reason, in many situations the expressive power of DL-Lite\textsubscript{R} is too restricted to capture real-world scenarios.

The aim of this work is to overcome these limitations of DL-Lite\textsubscript{R} by allowing the use of additional constructs in the ontology. To be able to exploit the added value coming from OBDA in real-world settings, an important requirement is the efficiency of query answering, achieved through a rewriting-based approach. This is only possible for ontology languages that are FO-rewritable. Two general mechanisms that have

\textsuperscript{*} This paper is an abridged version of [5].
\textsuperscript{1} Recall that FO queries constitute the core of SQL.
been proposed to cope with computational complexity coming from high expressiveness of ontology languages, and that allow one to regain FO-rewritability, are conservative rewriting [20] and approximation [26, 10]. In the former, an ontology in a powerful language is rewritten, when possible, into an equivalent one in a restricted language, while in the latter it is approximated, thus losing part of its semantics.

In this work, we significantly extend the practical impact of both approaches by bringing into the picture the mapping, an essential component of OBDA that has been ignored so far. Indeed, it is a fairly expressive component of an OBDA system, since it allows one to make use of arbitrary SQL (hence FO) queries to relate the content of the data source to the elements of the ontology. Hence, a natural question is how one can use the mapping component to capture as much as possible additional domain semantics, resulting in better approximations or more cases where conservative rewritings are possible, while maintaining a DL-Lite\textsubscript{R} ontology.

We illustrate how this can be done on an example. Consider a bank domain, where we can specify that a checking account in the name of a person is a simple account by means of the axiom CAcc \(\sqcap\exists\text{NameOf}\text{Person}\sqsubseteq\text{SAcc}\). Further, assume that the information about the accounts and their owners is stored in a database \(D\), and that the predicates CAcc, inNameOf, and Person are connected to \(D\) respectively via the mapping assertions \(sql_1(x) \leadsto CAcc(x)\), \(sql_2(x, y) \leadsto \text{NameOf}(x, y)\) and \(sql_3(x) \leadsto \text{Person}(x)\), where each \(sql_i\) is an SQL query over \(D\). Then this non-DL-Lite\textsubscript{R} axiom can be encoded by adding the assertion \(sql_1(x) \bowtie sql_2(x, y) \bowtie sql_3(y) \leadsto \text{SAcc}(x)\) to the mapping. This assertion connects \(D\) directly to the ontology term \(\text{SAcc}\) by making use of a join of the SQL queries in the original mapping. We observe that the resulting mapping, together with the ontology in which the non-DL-Lite\textsubscript{R} axiom has been removed, constitutes a conservative rewriting of the original OBDA specification.

In this paper, we elaborate on this idea, by introducing a novel framework for rewriting and approximation of OBDA specifications. Specifically, we provide a notion of rewriting based on query inseparability of OBDA specifications [3]. To deal with those cases where it is not possible to rewrite the OBDA specification into a query inseparable one whose ontology is in DL-Lite\textsubscript{R}, we give a notion of approximation that is sound for query answering. We develop techniques for rewriting and approximation of OBDA specifications based on compiling the extra expressiveness into the mappings. We target rather expressive ontology languages, and for Horn-ALCHIQ, a Horn fragment of OWL 2, we study decidability of existence of OBDA rewritings, and techniques to compute them when they exist, and to approximate them, otherwise.

We have implemented our techniques in a prototype system called ONTOPROX, which exploits functionalities provided by the ONTOP [27] and CLIPPER systems [14] to rewrite or approximate an OBDA specification expressed in Horn-SHIQ to one that can be directly processed by any OBDA system. We have evaluated ONTOPROX over synthetic and real OBDA instances against (i) the default ONTOP behavior, (ii) local semantic approximation (LSA), (iii) global semantic approximation (GSA), and (iv) CLIPPER over materialized ABoxes. Using ONTOPROX, for a few queries we have been able to obtain more answers (in fact, complete answers, as confirmed by CLIPPER). However, for many queries ONTOPROX showed no difference with respect to the default ONTOP behavior. One reason for this is that in the considered real-world scenario, the mapping
designers put significant effort to manually create complex mappings that overcome the limitations of DL-LiteR. Essentially they followed the principle of the technique presented here, producing an OBDA specification that was already “complete” by design.

The observations above immediately suggest a significant practical value of our approach, which can be used to facilitate the design of new OBDA specifications for existing expressive ontologies: instead of a manual compilation, which is cumbersome, error-prone, and difficult to maintain, mapping designers can write straightforward mappings, and the resulting OBDA specification can then be automatically transformed into a DL-LiteR OBDA specification with rich mappings.

Omitted proofs can be found in the extended version of this paper [4].

2 Preliminaries

2.1 Ontologies

We assume to have the following pairwise disjoint countably infinite alphabets: \( \mathbb{N}_C \) of concept names, \( \mathbb{N}_R \) of role names, and \( \mathbb{N}_I \) of constants (also called individuals). We consider ontologies expressed in Description Logics (DLs). Here we present the logics \( \text{Horn-\ALCHIQ} \), the Horn fragment of \( \text{SHIQ} \) without role transitivity, and \( \text{DL-LiteR} \), for which we develop some of the technical results in the paper. However, the general approximation framework is applicable to any OWL 2 fragment.

A Horn-\( \text{ALCHIQ} \) TBox in normal form [18, 14] is a finite set of axioms of the following forms:

- concept inclusions (CIs) \( \bigwedge_i A_i \sqsubseteq C \),
- role inclusions (RIs) \( R_1 \sqsubseteq R_2 \),
- and role disjointness axioms \( R_1 \sqcap R_2 \sqsubseteq \bot \), where \( A, A_i \) denote concept names, \( R, R_1, R_2 \) denote role names \( P \) or their inverses \( P^{-} \), and \( C \) denotes a concept of the form \( \bot, A, \exists R.A, \forall R.A, \text{or } \leq 1 R.A \).

For an inverse role \( R = P^{-} \), we use \( R^{-} \) to denote \( P^{-} \). \( \bot \) denotes the empty concept/role. A DL-LiteR TBox is a finite set of axioms of the form \( B_1 \sqsubseteq B_2, B_1 \sqcap B_2 \sqsubseteq \bot, R_1 \sqsubseteq R_2, \) and \( R_1 \sqcap R_2 \sqsubseteq \bot \), where \( B_i \) denotes a concept of the form \( A \text{ or } \exists R.T \). In what follows, for simplicity we write \( \exists R \) instead of \( \exists R.T \), we use \( N \) to denote either a concept or a role name, and assume that all TBoxes are in normal form.

An ABox is a finite set of membership assertions of the form \( A(c) \) or \( P(c, c') \), where \( c, c' \in \mathbb{N}_I \). For a DL \( \mathcal{L} \), an \( \mathcal{L} \)-ontology is a pair \( \mathcal{O} = (\mathcal{T}, \mathcal{A}) \), where \( \mathcal{T} \) is an \( \mathcal{L} \)-TBox and \( \mathcal{A} \) is an ABox. A signature \( \Sigma \) is a finite set of concept and role names. An ontology \( \mathcal{O} \) is said to be defined over (or simply, over) \( \Sigma \) if all the concept and role names occurring in it belong to \( \Sigma \) (and likewise for TBoxes, ABoxes, concept inclusions, etc.). When \( \mathcal{T} \) is over \( \Sigma \), we denote by \( \text{sig}(\mathcal{T}) \) the subset of \( \Sigma \) actually occurring in \( \mathcal{T} \). Moreover we denote with \( \text{Ind}(\mathcal{A}) \), the set of individuals appearing in \( \mathcal{A} \).

The semantics, models, and the notions of satisfaction and consistency of ontologies are defined in the standard way. We only point out that we adopt the Unique Name Assumption (UNA), and for simplicity we also assume to have standard names, i.e., for every interpretation \( I \) and every constant \( c \in \mathbb{N}_I \) interpreted by \( I \), we have that \( c^I = c \).

2.2 OBDA and Mappings

Let \( S \) be a relational schema over a countably infinite set \( \mathbb{N}_S \) of database predicates. For simplicity, we assume to deal with plain relational schemas without constraints,
and with database instances that directly store abstract objects (as opposed to values). So, a database instance $D$ of $S$ is a set of ground atoms over the predicates in $N_S$ and the constants in $N_0$.\footnote{All our results easily extend to the case where objects are constructed from retrieved database values [7].} Queries over $S$ are expressed in SQL. We use $\varphi(x)$ to denote that query $\varphi$ has $x = x_1, \ldots, x_n$ as free (i.e., answer) variables, where $n$ is the arity of $\varphi$. Given a database instance $D$ of $S$ and a query $\varphi$ over $S$, $\text{ans}(\varphi, D)$ denotes the set of tuples of constants in $N_0$ computed by evaluating $\varphi$ over $D$.

In OBDA, one provides access to an (external) database through an ontology TBox, which is connected to the database by means of a mapping. Given a source schema $S$ and a TBox $T$, a (GAV) mapping assertion between $S$ and $T$ has the form $\varphi(x) \rightarrow A(x)$ or $\varphi'(x, x') \rightarrow P(x, x')$, where $A$ and $P$ are respectively concept and role names, and $\varphi(x), \varphi'(x, x')$ are arbitrary (SQL) queries expressed over $S$, called source queries. Intuitively, given a database instance $D$ of $S$ and a mapping assertion $m = \varphi(x) \rightarrow A(x)$, the instances of the concept $A$ generated by $m$ from $D$ is the set $\text{ans}(\varphi, D)$; similarly for a mapping assertion $\varphi(x, x') \rightarrow P(x, x')$.

An OBDA specification is a triple $P = \langle T, M, S \rangle$, where $T$ is a DL TBox, $S$ is a relational schema, and $M$ is a finite set of mapping assertions. Without loss of generality, we assume that all concept and role names appearing in $M$ are contained in $\text{sig}(T)$. An OBDA instance is a pair $\langle P, D \rangle$, where $P$ is an OBDA specification, and $D$ is a database instance of $S$. The semantics of the OBDA instance $\langle P, D \rangle$ is specified in terms of interpretations of the concepts and roles in $T$. We define it by relying on the (virtual\footnote{We call such an ABox ‘virtual’, because we are not interested in materializing its facts.}) ABox $A_{M,D} = \{ N(o) \mid o \in \text{ans}(\varphi, D) \text{ and } \varphi(x) \rightarrow N(x) \in M \}$ generated by $M$ from $D$, where $N$ is a concept or role name in $T$. Then, a model of $\langle P, D \rangle$ is simply a model of the ontology $\langle T, A_{M,D} \rangle$.

To abstract away the low-level SQL details in source queries, following [13], we split each mapping assertion $m = \varphi(x) \rightarrow N(x)$ in $M$ into two parts. By introducing an intermediate view name $V_m$ for the SQL query $\varphi(x)$, we obtain a low-level mapping assertion of the form $\varphi(x) \rightarrow V_m(x)$, and a high-level mapping assertion of the form $V_m(x) \rightarrow N(x)$. Hence, in our technical development we deal only with the high-level mappings, and directly consider the intermediate views as our data sources.

2.3 Query Answering

We consider conjunctive queries, which are the basic and most important querying mechanism in relational database systems and ontologies. A conjunctive query (CQ) $q(x)$ over a signature $\Sigma$ is a formula $\exists y. \varphi(x, y)$, where $\varphi$ is a conjunction of atoms $N(z)$, such that $N$ is a concept or role name in $\Sigma$, and $z$ are variables from $x$ and $y$. The set of certain answers to a CQ $q(x)$ over an ontology $\langle T, A \rangle$, denoted $\text{cert}(q, (T,A))$, is the set of tuples $e$ of elements from $\text{Ind}(A)$ of the same length as $x$, such that $q(e)$ (considered as a FO sentence) holds in every model of $\langle T, A \rangle$. We mention two more query classes. An atomic query (AQ) is a CQ consisting of exactly one atom whose variables are all free. A CQ with inequalities (CQ$\geq$) is a CQ that may contain inequality atoms between the variables of the predicate atoms.
Given a CQ $q$, an OBDA specification $P = \langle T, M, S \rangle$ and a database instance $D$ of $S$, the answer to $q$ over the OBDA instance $(P, D)$, denoted $\text{cert}(q, P, D)$, is defined as $\text{cert}(q, (T, A_M, D))$. Observe that, when $D$ is inconsistent with $P$ (i.e., $(P, D)$ does not have a model), then $\text{cert}(q, P, D)$ is the set of all possible tuples of constants in $A_M, D$ (of the same arity as $q$).

3 An OBDA Rewriting Framework

We extend the notion of query inseparability of ontologies [6] to OBDA specifications. We adopt the proposal by [3], but we do not enforce preservation of inconsistency.

**Definition 1.** Let $\Sigma$ be a signature. Two OBDA specifications $P_1 = \langle T_1, M_1, S \rangle$ and $P_2 = \langle T_2, M_2, S \rangle$ are $\Sigma$-CQ inseparable if $\text{cert}(q, P_1, D) = \text{cert}(q, P_2, D)$, for every CQ $q$ over $\Sigma$ and every database instance $D$ of $S$.

In OBDA, one must deal with the trade-off between the computational complexity of query answering and the expressiveness of the ontology language. Suppose that in an OBDA specification $P = \langle T, M, S \rangle$, $T$ is expressed in an ontology language $\mathcal{L}$ that does not allow efficient query answering. A possible solution is to exploit the expressive power of the mapping to compute a new OBDA specification $P' = \langle T', M', S \rangle$ in which $T'$ is expressed in a language $\mathcal{L}_t$ more suitable for query answering than $\mathcal{L}$. The aim is to encode in $\mathcal{M}'$ not only $\mathcal{M}$ but also part of the semantics of $T$, so that $P'$ is query-inseparable from $P$. This leads to the notion of rewriting of OBDA specifications.

**Definition 2.** Let $\mathcal{L}_t$ be an ontology language. The OBDA specification $P' = \langle T', M', S \rangle$ is a CQ-rewriting in $\mathcal{L}_t$ of the OBDA specification $P = \langle T, M, S \rangle$ if (i) $\text{sig}(T) \subseteq \text{sig}(T')$, (ii) $T'$ is an $\mathcal{L}_t$-TBox, and (iii) $P$ and $P'$ are $\Sigma$-CQ inseparable, for $\Sigma = \text{sig}(T)$. If such $P'$ exists, we say that $P$ is CQ-rewritable into $\mathcal{L}_t$.

We observe that the new OBDA specification can be defined over a signature that is an extension of that of the original TBox. This is specified by condition (i). In condition (ii), we impose that the new ontology is specified in the target language $\mathcal{L}_t$. Finally, condition (iii) imposes that the OBDA specifications cannot be distinguished by CQs over the original TBox. Note that the definition allows for changing the ontology and the mappings, but not the source schema, accounting for the fact that the data sources might not be under the control of the designer of the OBDA specification.

As expected, it is not always possible to obtain a CQ-rewriting of $P$ in an ontology language $\mathcal{L}_t$ that allows for efficient query answering. Indeed, the combined expressiveness of $\mathcal{L}_t$ with the new mappings might not be sufficient to simulate query answering over $P$ without loss. In these cases, we can resort to approximating query answers over $P$ in a sound way, which means that the answers to queries posed over the new specification are contained in those produced by querying $P$. Hence, we say that the OBDA specification $P' = \langle T', M', S \rangle$ is a sound CQ-approximation in $\mathcal{L}_t$ of the OBDA specification $P = \langle T, M, S \rangle$ if $P'$ satisfies (i), (ii), and $\text{cert}(q, P', D) \subseteq \text{cert}(q, P, D)$, for each CQ $q$ over $\text{sig}(T)$ and for each instance $D$ of $S$.

Next, we study CQ-rewritability of OBDA specifications into $DL$-Lite$\mathcal{R}$, developing suitable techniques.
4   Rewriting OBDA Specifications

In this section, we develop our OBDA rewriting technique, which relies on Datalog rewritings of the TBox (and mappings). Recall that a Datalog program (with inequalities) is a finite set of definite Horn clauses without functions symbols, i.e., rules of the form \( \text{head} \leftarrow \varphi \), where \( \varphi \) is a finite non-empty list of predicate atoms and guarded inequalities called the body of the rule, and \( \text{head} \) is an atom, called the head of the rule, all of whose variables occur in the body. The predicates that occur in rule heads are called intensional (IDB), the other predicates are called extensional (EDB).

4.1 ET-mappings

Now, we extend the notion of T-mappings introduced by [27], and define the notion of an ET-mapping that results from compiling into the mapping the expressiveness of ontology languages that are Datalog rewritable, as introduced below.

We first introduce notation we need. Let \( \Pi \) be a Datalog program and \( N \) an IDB predicate. For a database \( D \) over the EDB predicates of \( \Pi \), let \( N^i_\Pi (D) \) denote the set of facts about \( N \) that can be deduced from \( D \) by at most \( i \geq 1 \) applications of the rules in \( \Pi \), and let \( N^\infty_\Pi (D) = \bigcup_{i \geq 1} N^i_\Pi (D) \). It is known that the predicate \( N^\infty_\Pi (\cdot) \) defined by \( N \) in \( \Pi \) can be characterized by a possibly infinite union of CQ\( ^\infty \)'s [11], i.e., there exist CQ\( ^\infty \)'s \( \varphi^N_0, \varphi^N_1, \ldots \) such that \( N^\infty_\Pi (D) = \bigcup_{i \geq 0} \{ N(\alpha) \mid \alpha \in \text{ans}(\varphi^N_i, D) \} \), for every \( D \).

The \( \varphi^N_i \)'s are called the expansions of \( N \) and can be described in terms of expansion trees; cf. [4, Appendix A]. We denote by \( \Phi_\Pi(N) \) the set of expansion trees for \( N \) in \( \Pi \), and abusing notation also the (possibly infinite) union of CQ\( ^\infty \)'s corresponding to it. Note that \( \Phi_\Pi(N) \) might be infinite due to the presence of IDB predicates that are recursive, i.e., either directly or indirectly refer to themselves.

We call a TBox \( T \) Datalog rewritable if it admits a translation \( \Pi_T \) to Datalog that preserves consistency and answers to AQs (see, e.g., the translations by [17], [14], and [28] for Horn-SHIQ, and by [12] for SHIQ). We assume that \( \Pi_T \) makes use of a special nullary predicate \( \bot \) that encodes inconsistency, i.e., for an ABox \( A \), \( \langle T, A \rangle \) is consistent iff \( \bot^\infty_\Pi, (A) \) is empty.\(^4\) We also assume that \( \Pi_T \) includes the following auxiliary rules, which ensure that \( \Pi_T \) derives all possible facts constructed over \( \text{sig}(T) \) and \( \text{Ind}(A) \) whenever \( \langle T, A \rangle \) is inconsistent:

\[
\begin{align*}
T_{\Delta}(x) & \leftarrow A(x); \quad T_{\Delta}(x) \leftarrow P(x,y); \quad T_{\Delta}(y) \leftarrow P(x,y); \\
A(x) & \leftarrow \bot, \quad T_{\Delta}(x); \quad P(x,y) \leftarrow \bot, \quad T_{\Delta}(x), T_{\Delta}(y);
\end{align*}
\]

where \( A \) and \( P \) respectively range over concept and role names in \( \text{sig}(T) \), and \( T_{\Delta} \) is a fresh unary predicate denoting the set of all the individuals appearing in \( A \).

In the following, we denote with \( \Pi_M \) the (high-level) mapping \( M \) viewed as a Datalog program, and with \( \Pi_{T,M} \) the Datalog program \( \Pi_T \cup \Pi_M \) associated to a Datalog rewritable TBox \( T \) and a mapping \( M \). From the properties of the translation \( \Pi_T \) (and the simple structure of \( \Pi_M \)), we obtain that \( \Pi_{T,M} \) satisfies the following:

\(^4\) Here we simply consider \( A \) as a database.
Input: Horn-$\text{ALCHIQ}$ TBox $\mathcal{T}$ and mapping $\mathcal{M}$.
Output: $\text{DL-Lite}_\mathcal{R}$ TBox $\mathcal{T}_c$ and ET-mapping $\mathcal{M}_c$.

Step 1: $\mathcal{T}_1$ is obtained from $\mathcal{T}$ by adding all CIs of the form $\exists R_i(\bigwedge_{j=1}^m A_j)$ entailed by $\mathcal{T}$, for concept names $A_i, A_j \in \text{sig}(\mathcal{T})$.

Step 2: $\mathcal{T}_2 = \text{norm}(\mathcal{T}_1)$.

Step 3: $\mathcal{T}_3 = \text{norm}(\mathcal{T}_2)$.

Step 4: $\mathcal{M}_c = \text{etm}_{\mathcal{T}_3}(\mathcal{M})$, and $\mathcal{T}_c$ is the $\text{DL-Lite}_\mathcal{R}$ TBox consisting of all $\text{DL-Lite}_\mathcal{R}$ axioms over $\text{sig}(\mathcal{T}_3)$ entailed by $\mathcal{T}_3$ (including the trivial ones $\forall R \subseteq N$).

Fig. 1. OBDA specification rewriting procedure RewObda.

Lemma 1. Let $\langle \mathcal{T}, \mathcal{M}, S \rangle$ be an OBDA specification where $\mathcal{T}$ is Datalog rewritable. Then, for every database instance $\mathcal{D}$ of $S$, concept or role name $N$ of $\mathcal{T}$, and $a$ in $\text{Ind}(\mathcal{A}_{\mathcal{M}, \mathcal{D}})$, we have that $\langle \mathcal{T}, \mathcal{A}_{\mathcal{M}, \mathcal{D}} \rangle \models N(a)$ if and only if $N(a) \in \text{norm}(\mathcal{T}, \mathcal{M})(\mathcal{D})$.

For a predicate $N$, we say that an expansion $\varphi^N \in \Phi_{\mathcal{T}, \mathcal{M}}(N)$ is $\text{DB}$-defined if $\varphi^N$ is defined over database predicates. Now we are ready to define ET-mappings.

Definition 3. Let $\langle \mathcal{T}, \mathcal{M}, S \rangle$ be an OBDA specification where $\mathcal{T}$ is Datalog rewritable. The ET-mapping for $\mathcal{M}$ and $\mathcal{T}$, denoted $\text{etm}_{\mathcal{T}}(\mathcal{M})$, is defined as the set of assertions of the form $\varphi^N(x) \models N(x)$ such that $N$ is a concept or role name in $\mathcal{T}$, and $\varphi^N \in \Phi_{\mathcal{T}, \mathcal{M}}(N)$ is $\text{DB}$-defined.

It is easy to show that, for $\mathcal{M}' = \text{etm}_{\mathcal{T}}(\mathcal{M})$ and each database instance $\mathcal{D}$, the virtual ABox $\mathcal{A}_{\mathcal{M}', \mathcal{D}}$ (which can be defined for ET-mappings as for ordinary mappings) contains all facts entailed by $\langle \mathcal{T}, \mathcal{A}_{\mathcal{M}, \mathcal{D}} \rangle$. In this sense, the ET-mapping $\text{etm}_{\mathcal{T}}(\mathcal{M})$ plays for a Datalog rewritable TBox $\mathcal{T}$ the same role as T-mappings play for (the simpler) $\text{DL-Lite}_\mathcal{R}$ TBoxes. Note that, in general, an ET-mapping is not a mapping, as it may contain infinitely many assertions. However, $\mathcal{A}_{\mathcal{M}', \mathcal{D}}$ is still finite, given that it is constructed over the finite number of constants appearing in $\mathcal{D}$.

4.2 Rewriting Horn-$\text{ALCHIQ}$ OBDA Specifications to $\text{DL-Lite}_\mathcal{R}$

Let $\langle \mathcal{T}, \mathcal{M}, S \rangle$ be an OBDA specification, where $\mathcal{T}$ is a Horn-$\text{ALCHIQ}$ TBox over a signature $\Sigma$. Procedure RewObda$(\mathcal{T}, \mathcal{M})$, in Figure 1, constructs a $\text{DL-Lite}_\mathcal{R}$ TBox $\mathcal{T}_c$ and an ET-mapping $\mathcal{M}_c$ such that $\langle \mathcal{T}_c, \mathcal{M}_c, S \rangle$ is $\Sigma$-CQ inseparable from $\langle \mathcal{T}, \mathcal{M}, S \rangle$.

In Step 2, the procedure applies to $\mathcal{T}_1$ the normalization step $\text{norm}_3$, which gets rid of concepts of the form $\exists R_i(\bigwedge_{j=1}^m A_j)$ in the right-hand side of CIs. This is achieved by the following well-known substitution [1]: every CI $\bigwedge_{i=1}^m A_i \subseteq \exists R_i(\bigwedge_{j=1}^n A'_j)$ in $\mathcal{T}_1$ is replaced with $\bigwedge_{i=1}^m A_i \subseteq \exists P_{new}, P_{new} \subseteq R, \mathcal{T} \not\subseteq \forall P_{new} A'$, for $1 \leq j \leq n$, where $P_{new}$ is a fresh role name. Notice that the latter two forms of inclusions introduced by $\text{norm}_3$ are actually in $\text{DL-Lite}_\mathcal{R}$, as $\mathcal{T} \subseteq \forall P_{new} A'$ is equivalent to $\exists P_{new} \subseteq A'$. In Step 3, the procedure applies to $\mathcal{T}_2$ a further normalization step, $\text{norm}_7$, which introduces a fresh concept name $A_{1\cap \cdots \cap A_n}$ for each concept conjunction $A_1 \cap \cdots \cap A_n$ appearing in $\mathcal{T}_2$, and adds $A_1 \cap \cdots \cap A_n \equiv A_{1\cap \cdots \cap A_n}$ to the TBox. Note that $\text{norm}_7(\mathcal{T}_1)$

5 We use $\equiv$ to abbreviate inclusion in both directions.
and \( \text{norm}_\Sigma(T_2) \) are model-conservative extensions of \( T_1 \) and \( T_2 \), respectively [21], as one can easily show. We denote by \( \text{rew}(T) \) the resulting TBox \( T_r \), which in general is exponential in the size of \( T \), and by \( \text{comp}(T, \mathcal{M}) \) the resulting ET-mapping \( \mathcal{M}_c \), which in general is infinite.

**Example 1** Assume that the domain knowledge is represented by the TBox \( T = \{ \text{CAcc} \sqsubseteq \exists \text{NameOf}. \text{Person} \sqsubseteq \text{SAcc} \} \) about bank accounts. Its normalization is \( T^b = \{ \text{Person} \sqsubseteq \exists \text{NameOf}. -A_1, \text{CAcc} \sqcap A_1 \sqsubseteq \text{SAcc} \}. Assume that the database schema \( S^b \) consists of the two relations \( \text{ENT}(\text{ID}, \text{TYPE}, \text{EMPID}) \), \( \text{PROD}(\text{NUM}, \text{TYPE}, \text{CUSTID}) \), whose data are mapped to the ontology terms by means of the following mapping \( \mathcal{M} \):

\[
\begin{align*}
\text{mp: } & \text{SELECT ID AS X FROM ENT WHERE ENT.TYPE='P'} \rightarrow \text{Person}(X) \\
\text{mq: } & \text{SELECT NUM AS X, CUSTID AS Y FROM PROD} \rightarrow \text{NameOf}(X, Y) \\
\text{mc: } & \text{SELECT NUM AS X FROM PROD P WHERE P.TYPE='B'} \rightarrow \text{CAcc}(X)
\end{align*}
\]

The corresponding high-level mapping \( \mathcal{M}^b \) consists of the assertions:

\[
\begin{align*}
h_p & : \{ x \mid \text{Person}(x) \} \rightarrow \text{Person}(x) \\
h_q & : \{ x, y \mid \text{NameOf}(x, y) \} \rightarrow \text{NameOf}(x, y) \\
h_c & : \{ x \mid \text{CAcc}(x) \} \rightarrow \text{CAcc}(x)
\end{align*}
\]

Now, consider the OBDA specification \( \mathcal{P}^b = \langle T^b, \mathcal{M}^b, S^b \rangle \). The \text{RewObda} procedure invoked on \( (T^b, \mathcal{M}^b) \) produces:

- The intermediate TBoxes \( T_1^b \) and \( T_2^b \) coinciding with \( T^b \), and \( T_3^b \) extending \( T^b \) with \( A_{\text{CAcc}\sqcap A_1} \equiv \text{CAcc} \sqcap A_1 \).
- The ET-mapping \( \mathcal{M}^c = \text{etm}_{T_3^b}(\mathcal{M}^b) \), which extends \( \mathcal{M}^b \) with the following three assertions:
  \[
  \begin{align*}
  \{ x \mid \text{NameOf}(x, y), \text{Person}(y) \} & \rightarrow A_1(x); \\
  \{ x \mid \text{CAcc}(x), \text{NameOf}(x, y), \text{Person}(y) \} & \rightarrow \text{SAcc}(x); \\
  \{ x \mid \text{CAcc}(x), \text{NameOf}(x, y), \text{Person}(y) \} & \rightarrow A_{\text{CAcc}\sqcap A_1}(x).
  \end{align*}
  \]

The procedure returns the DL-Lite\(_R\) TBox \( T^b_r = \{ A_{\text{CAcc}\sqcap A_1} \sqsubseteq \text{CAcc} \sqcap A_1, A_{\text{CAcc}\sqcap A_1} \sqsubseteq \text{SAcc} \} \) and the mapping \( \mathcal{M}^c_r \). It is possible to show that \( \mathcal{P}^b_{\text{DL-Lite}_R} = \langle T^b_r, \mathcal{M}^c_r, S^b \rangle \) is a CQ-rewriting of \( \mathcal{P}^b \) into DL-Lite\(_R\).

The TBox \( T_3 \) obtained as an intermediate result in Step 3 of \text{RewObda}(T, \mathcal{M}) \), is a model-conservative extension of \( T \), tailored towards capturing in DL-Lite\(_R\) the answers to tree-shaped CQs. This is obtained by introducing in Step 2 sufficiently many new role names, and in Step 3 new concept names, so as to capture entailed axioms that generate the tree-shaped parts of models. Note that at-most number restrictions do not participate in the generation of the tree-shaped parts of models, hence are not considered explicitly in Steps 1 to 3. On the other hand, the ET-mapping \( \mathcal{M}_c = \text{comp}(T, \mathcal{M}) \) generates from a database instance a virtual ABox \( \mathcal{A}^v \) that is complete with respect to all ABox facts that might be involved in the generation of the tree-shaped parts of models of \( T_r \) and \( \mathcal{A}^v \). This allows us to prove the main result of this section.

**Theorem 2.** Let \( \langle T, \mathcal{M}, S \rangle \) be an OBDA specification such that \( T \) is a Horn-\( \text{ACC}\)\( \text{ILQ} \) TBox, and let \( \langle T_r, \mathcal{M}_c \rangle = \text{RewObda}(T, \mathcal{M}) \). Then \( \langle T, \mathcal{M}, S \rangle \) and \( \langle T_r, \mathcal{M}_c, S \rangle \) are \( \Sigma \)-CQ inseparable, for \( \Sigma = \text{sig}(T) \).

Clearly, \( \langle T_r, \mathcal{M}_c, S \rangle \) is a candidate for being a CQ-rewriting of \( \langle T, \mathcal{M}, S \rangle \) into DL-Lite\(_R\). However, since \( \mathcal{M}_c \) might be an infinite set, \( \langle T_r, \mathcal{M}_c, S \rangle \) might not be an OBDA
specification and hence might not be effectively usable for query answering. Next we address this issue, and show that in some cases we obtain proper CQ-rewritings, while in others we have to resort to approximations.

5 Approximating OBDA Specifications

To obtain from an ET-mapping a proper mapping, we exploit the notion of predicate boundedness in Datalog, and use a bound on the depth of Datalog expansion trees.

An IDB predicate \( N \) is said to be bounded in a Datalog program \( \Pi \), if there exists a constant \( k \) depending only on \( \Pi \) such that, for every database \( D \), we have \( \mathcal{N}_k^\Pi (D) = \mathcal{N}^{\infty}_k (D) \) [11]. If \( N \) is bounded in \( \Pi \), then there exists an equivalent Datalog program \( \Pi' \) such that \( \Phi_\Pi (N) \) is finite, and thus represents a finite union of CQ\( \neq \)'s. It is well known that predicate boundedness for Datalog is undecidable in general [15]. We say that \( \Omega \) is a boundedness oracle if for a Datalog program \( \Pi \) and a predicate \( N \) it returns one of the three answers: \( N \) is bounded in \( \Pi \), \( N \) is not bounded in \( \Pi \), or unknown. When \( N \) is bounded, \( \Omega \) returns also a finite union of CQ\( \neq \)'s, denoted \( \Omega_\Pi (N) \), defining \( N \). Given a constant \( k \), \( \Phi_\Pi^k (N) \) denotes the set of trees (and the corresponding union of CQ\( \neq \)'s) in \( \Phi_\Pi (N) \) of depth at most \( k \), hence \( \Phi_\Pi^k (N) \) is always finite.

We introduce a cutting operator \( \text{cut}^k_\Omega \), which is parametric with respect to the cutting depth \( k > 0 \) and the boundedness oracle \( \Omega \), which, when applied to a predicate \( N \) and a Datalog program \( \Pi \), returns a finite union of CQ\( \neq \)'s as follows:

\[
\text{cut}^k_\Omega (N, \Pi) = \begin{cases} 
\Omega_\Pi (N), & \text{if } N \text{ is bounded in } \Pi \text{ w.r.t. } \Omega \\
\Phi_\Pi^k (N), & \text{otherwise}.
\end{cases}
\]

We apply cutting also to ET-mappings: given an ET-mapping \( \text{etm}_T (M) \), the mapping \( \text{cut}^k_\Omega (\text{etm}_T (M)) \) is the (finite) set of mapping assertions \( \varphi^N(x) \rightarrow N(x) \) s.t. \( N \) is a concept or role name in \( T \), and \( \varphi^N \in \text{cut}^k_\Omega (N, \Pi_{T,M}) \) is DB-defined.

The following theorem provides a sufficient condition for CQ-rewritability into DL-Lite\( \square \) in terms of the well-known notion of first-order (FO)-rewritability, which we recall here: a query \( q \) is FO-rewritable with respect to a TBox \( T \), if there exists a FO query \( q' \) such that \( \text{cert}(q, \langle T, A \rangle) = \text{ans}(q', A) \), for every ABox \( A \) over \( \text{sig}(T) \) (viewed as a database). It uses the fact that if an AQ is FO-rewritable with respect to a Horn-ALCHIQ TBox \( T \), then it is actually rewritable into a union of CQ\( \neq \)'s, and the fact that if \( T \) is FO-rewritable for AQs (i.e., every AQ is FO-rewritable with respect to \( T \)), then each concept and role name is bounded in \( \Pi_T \) [22, 2].

**Theorem 3.** Let \( \langle T, M, S \rangle \) be an OBDA specification such that \( T \) is a Horn-ALCHIQ TBox. Further, let \( T_r = \text{rew}(T) \) and \( M' = \text{cut}^k_\Omega (\text{comp}(T, M)) \), for a boundedness oracle \( \Omega \) and some \( k > 0 \). If \( T \) is FO-rewritable for AQs, then \( \langle T, M, S \rangle \) is CQ-rewritable into DL-Lite\( \square \), and \( \langle T_r, M', S \rangle \) is its CQ-rewriting. Otherwise, \( \langle T_r, M', S \rangle \) is a sound CQ-approximation of \( \langle T, M, S \rangle \) in DL-Lite\( \square \).

This gives us decidable conditions for rewritability of OBDA specifications in several significant cases. It is shown by [2] and [22] that FO-rewritability of AQs relative to Horn-SHIL TBoxes, Horn-ALCF TBoxes, and Horn-ALCF TBoxes of depth two is decidable. In fact, these FO-rewritability algorithms provide us with a boundedness
oracle $\Omega$: for each concept and role name $N$ in $\mathcal{T}$, they return a FO-rewriting of the AQ $N(x)$ that combined with the mapping $\mathcal{M}$ results in $\Omega_{\Pi_{\mathcal{T},\mathcal{M}}}(N)$.

Unfortunately, a complete characterization of CQ-rewritability into $\text{DL-Lite}_R$ is not possible if arbitrary FO-queries are allowed in the (low-level) mapping.

**Theorem 4.** The problem of checking whether an OBDA specification with an $\mathcal{EL}$ ontology and FO source queries in the mapping is CQ-rewritable into $\text{DL-Lite}_R$ is undecidable.

However, if we admit only unions of CQs in the (low-level) mapping, it follows from decidability of boundedness of monadic Datalog programs [11] that we can fully characterize CQ-rewritability.

**Theorem 5.** The problem of checking whether an OBDA specification with a Horn-$\mathcal{ALCHI}$ ontology of depth one and unions of CQs as source queries in the mapping is CQ-rewritable into $\text{DL-Lite}_R$ is decidable.

### 6 Implementation

To demonstrate the feasibility of our OBDA specification rewriting technique, we implemented the $\text{ONTOPROX}^6$ prototype system and evaluated it over synthetic and real OBDA instances. It relies on the OBDA reasoner $\text{ONTOP}^7$ and the complete Horn-$\mathcal{SHIQ}$ CQ-answering system $\text{CLIPPER}^8$, used as Java libraries, on a standard Prolog engine ($\text{SWI-Prolog}^9$), and on an OWL 2 reasoner ($\text{HERMIT}^{10}$).

Essentially, $\text{ONTOPROX}$ implements the rewriting and compiling procedure described in Figure 1, but instead of computing the (possibly infinite) ET-mapping $\text{comp}(\mathcal{T}, \mathcal{M})$, it computes its finite part $\text{cut}_k(\text{comp}(\mathcal{T}, \mathcal{M}))$. So, it gets as input an OWL 2 OBDA specification $\langle \mathcal{T}_{\text{OWL2}}, \mathcal{M}, \mathcal{S} \rangle$ and a positive integer $k$, and produces a $\text{DL-Lite}_R$ OBDA specification that can be used with any OBDA system. Below we describe some of the implementation details:

1. $\mathcal{T}_{\text{OWL2}}$ is first approximated to the Horn-$\mathcal{SHIQ}$ TBox $\mathcal{T}$ by dropping the axioms outside this fragment.
2. $\mathcal{T}$ is translated into a (possibly recursive) Datalog program $\Pi$ and saturated with all CIs of the form $\forall A_i \subseteq \exists R_i (\forall A'_i)$, using functionalities provided by $\text{CLIPPER}$.
3. The expansions $\text{cut}_k(\Phi_{\Pi}(X))$ are computed by an auxiliary Prolog program using Prolog meta-programming.
4. To produce actual mappings that can be used by an OBDA reasoner, the views in the high-level mapping $\text{cut}_k(\text{comp}(\mathcal{T}, \mathcal{M}))$ are replaced with their original SQL definitions using functionalities of $\text{ONTOP}$.
5. The $\text{DL-Lite}_R$ closure is computed by relying on the OWL 2 reasoner for Horn-$\mathcal{SHIQ}$ TBox classification.

---

6 [https://github.com/ontop/ontoprox/](https://github.com/ontop/ontoprox/)

7 [http://ontop.inf.unibz.it/](http://ontop.inf.unibz.it/)

8 [http://www.kr.tuwien.ac.at/research/systems/clipper/](http://www.kr.tuwien.ac.at/research/systems/clipper/)


For the experiments, we have considered two scenarios:

**UOBM.** The university ontology benchmark (UOBM) \[23\] comes with a SHOIN ontology (with 69 concepts, 35 roles, 9 attributes, and 204 TBox axioms), and an ABox generator. We have designed a suitable database schema for the generated ABox, converted the ABox to a 10MB database instance for the schema, and manually created the mapping, consisting of 96 assertions\[11\].

**Telecom benchmark.** The telecommunications ontology models a portion of the network of a leading telecommunications company, namely the portion connecting subscribers to the operating centers of their service providers. The current specification consists of an OWL 2 ontology with 152 concepts, 53 roles, 73 attributes, 458 TBox axioms, and of a mapping with 264 mapping assertions. The database instance contains 32GB of real-world data.

For each OBDA instance \(\langle\mathcal{T}, M, S\rangle\), we have evaluated the number of query answers and the query answering time with respect to five different setups:

1. The default behavior of ONTOP v1.15, which simply ignores all non-\(DL-Lite_\mathcal{R}\) axioms in \(\mathcal{T}\), i.e., using \(\langle\mathcal{T}^1, M, S\rangle\) where \(\mathcal{T}^1\) are all the \(DL-Lite_\mathcal{R}\) axioms in \(\mathcal{T}\).
2. The local semantic approximation (LSA) of \(\mathcal{T}\) in \(DL-Lite_\mathcal{R}\), i.e., using \(\langle\mathcal{T}^2, M, S\rangle\) where \(\mathcal{T}^2\) is obtained as the union, for each axiom \(\alpha \in \mathcal{T}\), of the set of \(DL-Lite_\mathcal{R}\) axioms \(\mathcal{T}(\alpha)\) entailed by \(\alpha\) [10].
3. The global semantic approximation (GSA) of \(\mathcal{T}\) in \(DL-Lite_\mathcal{R}\), i.e., using \(\langle\mathcal{T}^3, M, S\rangle\) where \(\mathcal{T}^3\) is the \(DL-Lite_\mathcal{R}\) closure of \(\mathcal{T}\) [25].
4. Result of ONTOPROX, i.e., \(\langle\text{rew}(\mathcal{T}), \text{cut}_5(\text{comp}(\mathcal{T}, M)), S\rangle\).
5. CLIPPER over the materialization of the virtual ABox.

The details of the evaluation can be found in [4].

## 7 Conclusions

We proposed a novel framework for rewriting and approximating OBDA specifications in an expressive ontology language to specifications in a weaker language, in which the core idea is to exploit the mapping layer to encode part of the semantics of the original specification, and we developed techniques for \(DL-Lite_\mathcal{R}\) as the target language.

We plan to continue our work along the following directions: (i) extend our technique to Horn-SHIQ, and, more generally, to Datalog rewritable TBoxes [12]; (ii) deepen our understanding of the computational complexity of deciding CQ-rewritability of OBDA specifications into \(DL-Lite_\mathcal{R}\); (iii) extend our technique to SPARQL queries under different OWL entailment regimes [19]; (iv) carry out more extensive experiments, considering queries that contain existentially quantified variables. This will allow us to verify the effectiveness of RewObda, which was designed specifically to deal with existentially implied objects.

**Acknowledgement.** This work is partially supported by the EU under the large-scale integrating project (IP) Optique (Scalable End-user Access to Big Data), grant agreement n. FP7-318338. We thank Martin Rezk for insightful discussions, and Benjamin Cogrel and Elem Güzel for help with the experimentation.

\[11\] https://github.com/ontop/ontop-examples/tree/master/aaai-2016-ontoprox/uobm
References


