Controlled Query Evaluation in Ontology-based Data Access*

Gianluca Cima[1][0000−0003−1783−5605], Domenico Lembo[1][0000−0002−0628−242X], Lorenzo Marconi[1], Riccardo Rosati[1][0000−0002−7697−4958], and Domenico Fabio Savo[2][0000−0002−8391−8049]

[1] Sapienza Università di Roma
{cima,lembo,marconi,rosati}@diag.uniroma1.it
[2] Università degli Studi di Bergamo
domenicofabio.savo@unibg.it

Abstract. In this paper we study the problem of information disclosure in ontology-based data access (OBDA). Following previous work on Controlled Query Evaluation, we introduce the framework of Policy-Protected OBDA (PPOBDA), which extends OBDA with data protection policies specified over the ontology and enforced through a *censor*, i.e., a function that alters answers to users' queries to avoid the disclosure of protected data. We consider PPOBDA systems in which the ontology is expressed in OWL 2 QL and the policies are denial constraints, and show that query answering under censors in such a setting can be reduced to standard query answering in OBDA (without data protection policies). The basic idea of our approach is to compile the policies of a PPOBDA system into the mapping of a standard OBDA system. To this aim, we analyze some notions of censor proposed in the literature, show that they are not suited for the above-mentioned compilation, and provide a new definition of censor that enables the effective realization of our idea. We have implemented our technique and evaluated it over the NPD benchmark for OBDA. Our results are very promising and show that controlled query evaluation in OBDA can be realized in the practice by using off-the-shelf OBDA engines.

Keywords: Ontology-based Data Access · Information Disclosure · Data Protection · First-Order Rewritability

1 Introduction

Controlled Query Evaluation (CQE) is an approach to privacy-preserving query answering that recently has gained attention in the context of ontologies [6,11,12,17]. In this paper, we consider the more general Ontology-based

* This work was supported by the EU within the H2020 Programme under the grant agreement 834228 (ERC WhiteMec) and the grant agreement 825333 (MOSAICrOWN), by Regione Lombardia within the Call Hub Ricerca e Innovazione under the grant agreement 1175328 (WATCHMAN), and by MUR (Ministero dell'Università e della Ricerca), through PRIN project HOPE (prot. 2017MMJJRE).

Data Access (OBDA) framework, where an ontology is coupled to external data sources through a mapping [20,23], and extend OBDA with CQE features. In this new framework, which we call *Policy-Protected Ontology-based Data Access (PPOBDA)*, a data protection policy is specified over the ontology of an OBDA system in terms of logical statements declaring confidential information that must not be revealed to the users. For instance, the following formula:

$$\forall x, y. OilComp(x) \wedge IssuesLic(x, y) \wedge Comp(y) \rightarrow \bot$$

says that the existence of an oil company issuing a license to another company (to operate over its properties) is a private information.

More formally, we define a PPOBDA specification $\mathcal{E}$ as a quadruple $\langle \mathcal{T}, \mathcal{S}, \mathcal{M}, \mathcal{P} \rangle$, where $\mathcal{T}$ is a Description Logic (DL) TBox [1], formalizing intensional domain knowledge, $\mathcal{S}$ is the relational schema at the sources, $\mathcal{M}$ is the mapping between the two, i.e., a set of logical assertions defining the semantic correspondence between $\mathcal{T}$ and $\mathcal{S}$, and $\mathcal{P}$ is the data protection policy expressed over $\mathcal{T}$. The components $\mathcal{T}$, $\mathcal{S}$, and $\mathcal{M}$ are exactly as in OBDA specifications, and, as in standard OBDA, a user can only ask queries over the TBox $\mathcal{T}$. Then, query answering is filtered through a *censor*, i.e., a function that alters the answers to queries, in such a way that no data are returned that may lead a malicious user to infer knowledge declared confidential by the policy, even in case he/she accumulates the answers he/she gets over time. Among possible censors, *optimal* ones are preferred, i.e., those altering query answers in a minimal way.

Within this framework, we initially consider two different notions of censor, called censor in either **CQ** or **GA**, previously defined for CQE over DL ontologies [12,17], and which can be naturally extended to PPOBDA. More precisely, given a PPOBDA specification $\mathcal{E} = \langle \mathcal{T}, \mathcal{S}, \mathcal{M}, \mathcal{P} \rangle$, an optimal censor in **CQ** (resp., **GA**) for $\mathcal{E}$ is a function that, taken as input a database instance D for the source schema $\mathcal{S}$, returns a maximal subset $\mathcal{C}$ of the set of Boolean conjunctive queries (resp., ground atoms) inferred by $\langle \mathcal{T}, \mathcal{S}, \mathcal{M} \rangle$ and D, such that $\mathcal{C} \cup \mathcal{T}$ does not entail information protected by the policy. Since in general, for such notions of censor, several of these maximal sets (incomparable to each other) exist, for both cases we define *query answering under optimal censors* in PPOBDA as a form of skeptical reasoning over all such sets, in the line of [17].

Our basic idea to solve query answering under censors is to transform a PPOBDA specification $\mathcal{E}$ into a classical OBDA specification $\mathcal{J}$ (i.e., without policies), in such a way that, whatever database D instantiates the source schema $\mathcal{S}$, query answering under censors in $\mathcal{E}$ over D is equivalent to standard query answering in $\mathcal{J}$ over D. In this transformation, we require that $\mathcal{J}$ has the same TBox of $\mathcal{E}$, so that this reduction is transparent to the user, who can continue asking to $\mathcal{J}$ exactly the same queries he/she could ask to $\mathcal{E}$. We also impose that $\mathcal{J}$ maintains the same source schema as $\mathcal{E}$, since, as typical in OBDA, the data sources to be accessed are autonomous, and cannot be modified for OBDA purposes. Moreover, we aim at a transformation that is independent from the underlying data and from the user queries, so that it can be computed only once, at design-time. This enables us to use off-the-shelf OBDA engines, like

Mastro[3] or Ontop[4] to realize CQE in OBDA. The problem we study can be thus summarized as follows: Given a PPOBDA specification $\mathcal{E} = \langle \mathcal{T}, \mathcal{M}, \mathcal{S}, \mathcal{P} \rangle$, construct an OBDA specification $\mathcal{J} = \langle \mathcal{T}, \mathcal{S}, \mathcal{M}' \rangle$ such that, for any database D for $\mathcal{S}$, conjunctive query answering under censors in $\mathcal{E}$ over D is equivalent to standard conjunctive query answering in $\mathcal{J}$ over D.

We investigate the above problem for the relevant case in which the TBox is expressed in *DL-Lite*$_{\mathcal{R}}$, the DL underpinning OWL 2 QL [18], the standard profile of OWL 2 designed for ontology-based data management and prominently used in OBDA, and the policy is a set of denial assertions, i.e., conjunctive queries for which an empty answer is imposed due to confidential reasons (as in our initial example). Our contributions are as follows.

- We introduce the framework for PPOBDA (Section 4).
- We show that the problem above defined has in general no solution when censors in either **CQ** or **GA** are considered. We in fact prove this result for an empty TBox, and thus it holds for TBoxes in any DL, and not only for OWL 2 QL ones (Section 5).
- To solve this issue, we propose a further, semantically well-founded approximated notion of censor, which we call IGA censor. Intuitively, an IGA censor for a PPOBDA specification $\mathcal{E}$ is a function that, for any database D instantiating the source schema $\mathcal{S}$ of $\mathcal{E}$, returns the intersection of the sets of ground atoms computed by the optimal censors in **GA** for $\mathcal{E}$ applied to D. We then provide an algorithm that solves our problem for OWL 2 QL PPOBDA specifications under IGA censors. (Section 6)
- We provide an experimental evaluation of our approach. We have implemented our algorithm in Java, and tested it over the OBDA NPD benchmark [15], whose TBox has been suitably approximated from OWL 2 to OWL 2 QL. We have compared query answering in the case in which no data protection policy is specified (i.e., in standard OBDA) with query answering under IGA censors for an increasing number of policy assertions. We have used Mastro as OBDA engine. Our results show that the cost of the off-line transformation performed by our tool is negligible, and answering queries in the presence of a data protection policy in our approach does not cause a significant overhead with respect to the case without policy (Section 7).

2 Related Work

Existing OBDA solutions do not provide any explicit support to the protection of confidential data, and the research has so far produced only initial theoretical contributions in this direction. In [3], the authors study the problem of determining whether information that is declared confidential at the sources through a protection policy, as in CQE, can be inferred by a user on the basis of the answers to the queries posed over the OBDA system, assuming that he/she is

[3] http://obdasystems.com/mastro
[4] https://ontop-vkg.org/

knowledgeable about the OBDA specification. Both [3] and the present paper focus on the role of the mapping in filtering data coming from the sources with respect to a declarative data protection policy. However, we consider the policy expressed over the TBox of the OBDA specification and look at the mapping as a means to enforce data protection, whereas in [3] the policy is declared at the source level and the mapping is seen as a potential cause for secret disclosure. Possible disclosure of confidential source-level information has also been studied in [2,19,8], in the context of data integration or exchange, possibly in the presence of integrity constraints at the sources. In these works, the integrated target schema is a flat relational one, thus not an expressive TBox, as in OBDA, and secrets are specified in terms of queries over the sources, thus not policies over the target schema, as in our framework. Also, the focus is on disclosure analysis and not confidentiality enforcement.

Initially, CQE has been studied in the context of propositional theories under closed world assumption (see, e.g., [21,4]), thus in a framework substantially different from ours. The more recent works on CQE over DL ontologies are instead closer to our research. In [6], the authors propose a method for computing secure knowledge views over DL ontologies in the presence of user background knowledge and investigate the computational complexity of the approach for ontologies and policies specified in various expressive DLs. In [11], the authors generalize the CQE paradigm for incomplete databases proposed in [5], and study CQE for OWL 2 RL ontologies and policies represented by a set of ground atoms. The same authors continued their investigation in [12], for ontologies and policies specified in Datalog or in one of the OWL 2 profiles [18], mainly focusing on the problem of the existence of a censor under two incomparable different notions of censors. In [17], the authors revisited CQE as the problem of computing the answers to a query that are returned by all optimal censors, which is also the approach we adopt in this paper. However, like all the above mentioned papers on CQE over ontologies, [17] does not consider OBDA mappings.

We finally point out that forms of privacy-preserving query answering over DL ontologies have been studied also, e.g., in [10,22], but not according to the CQE approach, or in an OBDA context.

3 Preliminaries

We use standard notions of function-free first-order (FO) logic and relational databases. We assume to have the pairwise disjoint countably infinite sets Σ_R, Σ_T, Σ_C, and Σ_V for relational database predicates, ontology predicates, constants (a.k.a. individuals), and variables, respectively.

Ontologies. With **FO** we indicate the language of all FO sentences over Σ_T, Σ_C, and Σ_V. An FO ontology $\mathcal{O}$ is a finite set of FO sentences, i.e., $\mathcal{O} \subseteq \mathbf{FO}$. With $Mod(\mathcal{O})$ we denote the set of the models of $\mathcal{O}$, i.e., the FO interpretations $\mathcal{I}$ such that $\phi^{\mathcal{I}}$ (i.e., the interpretation of ϕ in $\mathcal{I}$) evaluates to true, for each sentence $\phi \in \mathcal{O}$. We say that $\mathcal{O}$ is consistent if $Mod(\mathcal{O}) \neq \emptyset$, inconsistent otherwise, and that $\mathcal{O}$ entails an FO sentence ϕ, denoted $\mathcal{O} \models \phi$, if $\phi^{\mathcal{I}}$ is true in every

$\mathcal{I} \in \mathit{Mod}(\mathcal{O})$. The set of logical consequences of an ontology $\mathcal{O}$ in a language $\mathcal{L} \subseteq \mathbf{FO}$, denoted $\mathsf{cl}_{\mathcal{L}}(\mathcal{O})$, is the set of sentences in $\mathcal{L}$ entailed by $\mathcal{O}$.

Queries. A query q is a (possibly open) FO formula $\phi(\vec{x})$, where $\vec{x}$ are the free variables of q. The number of variables in $\vec{x}$ is the *arity* of q. We consider queries over either relational databases or ontologies. Given a query q of arity n over a database D, we use $\mathit{Eval}(q, D)$ to denote the evaluation of q over D, i.e., the set of tuples $\vec{t} \in \Sigma_C^n$ such that $D \models \phi(\vec{t})$, where $\phi(\vec{t})$ is the sentence obtained by substituting $\vec{x}$ with $\vec{t}$ in q.

A conjunctive query (CQ) q is an FO formula of the form $\exists \vec{y}.\alpha_1(\vec{x}, \vec{y}) \wedge \ldots \wedge \alpha_n(\vec{x}, \vec{y})$, where $n \geq 1$, $\vec{x}$ is the sequence of free variables, $\vec{y}$ is the sequence of existential variables, and each $\alpha_i(\vec{x}, \vec{y})$ is an atom (possibly containing constants) with predicate α_i and variables in $\vec{x} \cup \vec{y}$. Each variable in $\vec{x} \cup \vec{y}$ occurs in at least one atom of q. Boolean CQs (BCQs) are queries whose arity is zero (i.e., BCQs are sentences). The length of a CQ q is the number of its atoms. The set of *certain answers* to a CQ q of arity n over an ontology $\mathcal{O}$ is the set $\mathit{cert}(q, \mathcal{O})$ of tuples $\vec{c} \in \Sigma_C^n$ such that $\mathcal{O}$ entails the sentence $\exists \vec{y}.\alpha_1(\vec{c}, \vec{y}) \wedge \ldots \wedge \alpha_n(\vec{c}, \vec{y})$. As usual, when a BCQ q is entailed by $\mathcal{O}$, i.e., $\mathcal{O} \models q$, we may also say $\mathit{cert}(q, \mathcal{O}) = \{\langle\rangle\}$, i.e., the set of certain answers contains only the empty tuple, $\mathit{cert}(q, \mathcal{O}) = \emptyset$, otherwise.

For ease of exposition, in our technical development we will focus on the entailment of BCQs from DL ontologies. However, our results can be straightforwardly extended to non-Boolean CQs through a standard encoding of open formulas into closed ones. In the following, we denote by **CQ** the languages of BCQs, and by **GA** the language of ground atoms, i.e., BCQs with only one atom and no variables, both specified over Σ_T, Σ_C, and Σ_V.

OWL 2 QL and *DL-Lite*$_{\mathcal{R}}$. We consider ontologies expressed in *DL-Lite*$_{\mathcal{R}}$ [7], i.e., the DL that provides the logical underpinning of OWL 2 QL [18]. DLs are decidable FO languages using only unary and binary predicates, called concepts and roles, respectively [1]. Concepts denote sets of objects, whereas roles denote binary relationships between objects. A DL ontology $\mathcal{O}$ is a set $\mathcal{T} \cup \mathcal{A}$, where $\mathcal{T}$ is the *TBox* and $\mathcal{A}$ is the *ABox*, specifying intensional and extensional knowledge, respectively. A TBox $\mathcal{T}$ in *DL-Lite*$_{\mathcal{R}}$ is a finite set of axioms of the form: $B_1 \sqsubseteq B_2$, $B_1 \sqsubseteq \neg B_2$, $R_1 \sqsubseteq R_2$, and $R_1 \sqsubseteq \neg R_2$, where each R_i, with $i \in \{1, 2\}$ is an atomic role $Q \in \Sigma_T$, or its inverse Q^-; each B_i, with $i \in \{1, 2\}$ is an atomic concept $A \in \Sigma_T$, or a concept of the form $\exists Q$ or $\exists Q^-$, i.e., unqualified existential restrictions, which denote the set of objects occurring as first or second argument of Q, respectively. Assertions of the form $B_1 \sqsubseteq B_2$ and $R_1 \sqsubseteq R_2$ indicate subsumption between predicates, those of the form $B_1 \sqsubseteq \neg B_2$ and $R_1 \sqsubseteq \neg R_2$ indicate disjointness between predicates. An ABox $\mathcal{A}$ is a finite set of ground atoms, i.e., assertions of the form $A(a)$, $Q(a, b)$, where $A, Q \in \Sigma_T$, and $a, b \in \Sigma_C$. The semantics of a *DL-Lite*$_{\mathcal{R}}$ ontology $\mathcal{O}$ is given in terms of FO models over the signature of $\mathcal{O}$ in the standard way [7].

OBDA. An *OBDA specification* is a triple $\mathcal{J} = \langle \mathcal{T}, \mathcal{S}, \mathcal{M} \rangle$, where $\mathcal{T}$ is a DL TBox over the alphabet Σ_T, $\mathcal{S}$, called *source schema*, is a relational schema over the alphabet Σ_R, and $\mathcal{M}$ is a *mapping* between $\mathcal{S}$ and $\mathcal{T}$.

The mapping $\mathcal{M}$ is a finite set of *mapping assertions* from $\mathcal{S}$ to $\mathcal{T}$. Each of these assertions m has the form $\phi(\vec{x}) \rightsquigarrow \psi(\vec{x})$, where $\phi(\vec{x})$, called the *body of* m, and $\psi(\vec{x})$, called the *head of* m, are queries over $\mathcal{S}$ and $\mathcal{T}$, respectively, both with free variables $\vec{x}$. We consider the case in which $\phi(\vec{x})$ is an FO query, and $\psi(\vec{x})$ is a single-atom query without constants and existential variables (i.e., each m is a GAV mapping assertion [14]). This is the form of mapping commonly adopted in OBDA, and a special case of the W3C standard R2RML [13].

In the above definition, we have assumed that the source database directly stores the identifiers (e.g., the URIs) of the instances of the ontology predicates. However, all our results hold also when such identifiers are constructed in the mapping using the database values, as usual in OBDA [20] and in R2RML.

The semantics of $\mathcal{J}$ is given with respect to a database instance for $\mathcal{S}$, called source database for $\mathcal{J}$. Given one such database D, the *retrieved ABox* for $\mathcal{J}$ w.r.t. D, denoted $ret(\mathcal{J}, D)$, is the ABox that contains all and only the facts $\psi(\vec{t})$ such that $\psi(\vec{x})$ occurs in the head of some mapping assertion $m \in \mathcal{M}$, and $\vec{t}$ is a tuple of constants such that $\vec{t} \in Eval(\phi(\vec{x}), D)$, where $\phi(\vec{x})$ is the body of m. Then, a *model* for $\mathcal{J}$ w.r.t. D is a model of the ontology $\mathcal{T} \cup ret(\mathcal{J}, D)$. The set of models of $\mathcal{J}$ w.r.t. D is denoted by $Mod(\mathcal{J}, D)$. Also, we call $(\mathcal{J}, D)$ an *OBDA setting* and say that $(\mathcal{J}, D)$ *is inconsistent* if $Mod(\mathcal{J}, D) = \emptyset$, and denote by $(\mathcal{J}, D) \models \alpha$ the entailment of a sentence α by $(\mathcal{J}, D)$, i.e., the fact that $\alpha^{\mathcal{I}}$ is true in every $\mathcal{I} \in Mod(\mathcal{J}, D)$.

4 Framework

We start by introducing the formal notion of policy-protected OBDA specification. Our framework is a generalization to the OBDA context of the CQE framework for DL ontologies provided in [9,17].

We first define a *denial assertion* (or simply a denial) as an FO sentence of the form $\forall \vec{x}.\phi(\vec{x}) \rightarrow \bot$, such that $\exists \vec{x}.\phi(\vec{x})$ is a BCQ. Given one such denial δ and a DL ontology $\mathcal{O}$, then $\mathcal{O} \cup \{\delta\}$ is a consistent FO theory if $\mathcal{O} \not\models \exists \vec{x}.\phi(\vec{x})$, and is inconsistent otherwise. We then give the following definition.

Definition 1 (PPOBDA specification). *A* policy-protected ontology-based data access (PPOBDA) *specification is a quadruple* $\mathcal{E} = \langle \mathcal{T}, \mathcal{S}, \mathcal{M}, \mathcal{P} \rangle$ *such that* $\langle \mathcal{T}, \mathcal{S}, \mathcal{M} \rangle$ *is an OBDA specification, and* $\mathcal{P}$ *is a* policy*, i.e., a set of denial assertions over the signature of* $\mathcal{T}$*, such that* $\mathcal{T} \cup \mathcal{P}$ *is a consistent FO theory.*

Example 1. Consider the following PPOBDA specification $\mathcal{E} = \langle \mathcal{T}, \mathcal{S}, \mathcal{M}, \mathcal{P} \rangle$, where

$\mathcal{T}$ = { $OilComp \sqsubseteq Comp$, $\exists IssuesLic^- \sqsubseteq Comp$, $\exists PipeOp \sqsubseteq Pipeline$, $\exists PipeOp^- \sqsubseteq Comp$ }

$\mathcal{S}$ = {company, license, operator}

$\mathcal{M}$ = {m_1: $\exists y.\mathsf{company}(x, y) \rightsquigarrow Comp(x)$, m_2: $\mathsf{company}(x, \text{'oil'}) \rightsquigarrow OilComp(x)$, m_3: $\mathsf{license}(x, y) \rightsquigarrow IssuesLic(x, y)$, m_4: $\mathsf{operator}(x, y) \rightsquigarrow PipeOp(x, y)$ }

$\mathcal{P}$ = { d_1: $\forall x, y.OilComp(x) \wedge IssuesLic(x, y) \wedge Comp(y) \rightarrow \bot$, d_2: $\forall x, y.PipeOp(x, y) \wedge OilComp(y) \rightarrow \bot$}

In words, the TBox $\mathcal{T}$ specifies that oil companies (concept *OilComp*) are a special kind of companies (concept *Comp*) and that companies can issue licenses (role $IssuesLic$) to other companies (over their properties) and be operators (role *PipeOp*) of pipelines (concept *Pipeline*). The schema $\mathcal{S}$ has three tables, each with two columns: company, which contains data about companies and their type, license, which contains data about license issuance, and operator, which contains operators of pipelines. The policy $\mathcal{P}$ specifies as confidential the fact that an oil company issues a license to a company, and the fact that an oil company is the operator of a pipeline. □

The semantics of a PPOBDA specification $\mathcal{E} = \langle \mathcal{T}, \mathcal{S}, \mathcal{M}, \mathcal{P} \rangle$ coincides with that of the OBDA specification $\langle \mathcal{T}, \mathcal{S}, \mathcal{M} \rangle$, and thus we naturally extend to PPOBDA the notion of source database D, retrieved ABox (denoted $ret(\mathcal{E}, D)$), set of models (denoted $Mod(\mathcal{E}, D)$), and setting (denoted $(\mathcal{E}, D)$).

We now give a notion of censor in PPOBDA that is parametric with respect to the language $\mathcal{L}$ used for enforcing the policy (similarly to [17]). In the following, given a TBox $\mathcal{T}$, with $\mathcal{L}(\mathcal{T})$ we denote the subset of $\mathcal{L}$ containing all and only the sentences specified only over the predicates occurring in $\mathcal{T}$ and the constants in Σ_C. For instance, with $\mathbf{FO}(\mathcal{T})$ we denote the set of FO sentences having the above mentioned characteristics. Moreover, given a database D, with $\mathcal{L}_D$ we denote the formulas in $\mathcal{L}$ mentioning only constants in D.

Definition 2 (censor in $\mathcal{L}$). *Given a PPOBDA specification $\mathcal{E} = \langle \mathcal{T}, \mathcal{S}, \mathcal{M}, \mathcal{P} \rangle$ and a language $\mathcal{L} \subseteq \mathbf{FO}(\mathcal{T})$, a* censor *for $\mathcal{E}$ in $\mathcal{L}$ is a function* $\mathsf{cens}(\cdot)$ *such that, for each source database D for $\mathcal{E}$, returns a set $\mathsf{cens}(D) \subseteq \mathcal{L}_D$ such that:*

(i) $(\langle \mathcal{T}, \mathcal{S}, \mathcal{M} \rangle, D) \models \phi$*, for each $\phi \in \mathsf{cens}(D)$, and*
(ii) $\mathcal{T} \cup \mathcal{P} \cup \mathsf{cens}(D)$ *is a consistent FO theory.*

We call $\mathcal{L}$ the censor language*.*

Given two censors $\mathsf{cens}(\cdot)$ and $\mathsf{cens}'(\cdot)$ for $\mathcal{E}$ in $\mathcal{L}$, we say that $\mathsf{cens}'(\cdot)$ is *more informative* than $\mathsf{cens}(\cdot)$ if:

(i) for every database instance D for $\mathcal{E}$, $\mathsf{cens}(D) \subseteq \mathsf{cens}'(D)$, and
(ii) there exists a database instance D' for $\mathcal{E}$ such that $\mathsf{cens}(D') \subset \mathsf{cens}'(D')$.

Then, a censor $\mathsf{cens}(\cdot)$ for $\mathcal{E}$ in $\mathcal{L}$ is *optimal* if there does not exist a censor $\mathsf{cens}'(\cdot)$ for $\mathcal{E}$ in $\mathcal{L}$ such that $\mathsf{cens}'(\cdot)$ is more informative than $\mathsf{cens}(\cdot)$. The set of all optimal censors in $\mathcal{L}$ for a PPOBDA specification $\mathcal{E}$ is denoted by $\mathcal{L}\text{-}\mathsf{OptCens}_{\mathcal{E}}$.

In this paper, we consider censors in the languages $\mathbf{CQ}(\mathcal{T})$ and $\mathbf{GA}(\mathcal{T})$, i.e., we instantiate $\mathcal{L}$ in Definition 2 to either the language of BCQs or the language of ground atoms, respectively, both over the predicates occurring in $\mathcal{T}$. These are the censor languages studied in [17] over DL ontologies. In the following, when $\mathcal{T}$ is clear from the context, we simply denote them as $\mathbf{CQ}$ and $\mathbf{GA}$, respectively.

Example 2. Consider the PPOBDA specification $\mathcal{E}$ of Example 1, and let cens_1 be the function such that, given a source database D for $\mathcal{E}$, $\mathsf{cens}_1(D)$ is the set of ground atoms $\mathsf{cl}_{\mathbf{GA}}(\mathcal{T} \cup \mathcal{A}_1)$, where $\mathcal{A}_1$ is the ABox obtained from $ret(\mathcal{E}, D)$ by adding the assertion $Comp(c)$ and removing the assertion $OilComp(c)$, for

each individual c such that $\mathcal{T} \cup ret(\mathcal{E}, D) \models (OilComp(c) \wedge \exists x.IssuesLic(c, x) \wedge Comp(x)) \vee (\exists x.PipeOp(x, c) \wedge OilComp(c))$. It is easy to verify that cens_1 is an optimal censor for $\mathcal{E}$ in $\mathbf{GA}$, i.e. $\mathsf{cens}_1 \in \mathbf{GA}\text{-}\mathsf{OptCens}_{\mathcal{E}}$. □

For censors in **CQ** and **GA** we define the following entailment problems.

Definition 3. *Given a PPOBDA specification $\mathcal{E} = \langle \mathcal{T}, \mathcal{S}, \mathcal{M}, \mathcal{P} \rangle$, a database instance D for $\mathcal{S}$, and a BCQ q, we consider the following decision problems:*

***(CQ-Cens-Entailment)**: decide whether $\mathcal{T} \cup \mathsf{cens}(D) \models q$ for every $\mathsf{cens} \in \mathbf{CQ}\text{-}\mathsf{OptCens}_{\mathcal{E}}$. If this is the case, we write $(\mathcal{E}, D) \models^{cqe}_{\mathbf{CQ}} q$.*

***(GA-Cens-Entailment)**: decide whether $\mathcal{T} \cup \mathsf{cens}(D) \models q$ for every $\mathsf{cens} \in \mathbf{GA}\text{-}\mathsf{OptCens}_{\mathcal{E}}$. If this is the case, we write $(\mathcal{E}, D) \models^{cqe}_{\mathbf{GA}} q$.*

Our ultimate goal is to solve the above problems by reducing them to classical entailment of BCQs in OBDA. To this aim, we define below the notion of query equivalence under censor between PPOBDA and OBDA specifications.

Definition 4 (query equivalence). *Given a PPOBDA specification $\mathcal{E} = \langle \mathcal{T}, \mathcal{S}, \mathcal{M}, \mathcal{P} \rangle$ and an OBDA specification $\mathcal{J} = \langle \mathcal{T}, \mathcal{S}, \mathcal{M}' \rangle$, we say that $\mathcal{E}$ and $\mathcal{J}$ are* query-equivalent under censors in **CQ** (resp. **GA**) *if for every database instance D for $\mathcal{S}$ and every BCQ q, $(\mathcal{E}, D) \models^{cqe}_{\mathbf{CQ}} q$ (resp. $(\mathcal{E}, D) \models^{cqe}_{\mathbf{GA}} q$) iff $(\mathcal{J}, D) \models q$.*

Based on the above definition, we can decide **CQ**-cens-entailment of a BCQ q from a PPOBDA $\mathcal{E}$ coupled with a source database D for $\mathcal{S}$ by constructing an OBDA specification $\mathcal{J}$ such that $\mathcal{E}$ and $\mathcal{J}$ are query-equivalent under censors in **CQ** and checking whether $(\mathcal{J}, D) \models q$ (analogously for **GA**-cens-entailment). We remark that, besides the policy, the mapping is the only component in which $\mathcal{E}$ and $\mathcal{J}$ differ (see also Section 1). Intuitively, $\mathcal{M}'$ in $\mathcal{J}$ implements a censor (in either **CQ** or **GA**) for $\mathcal{E}$.

5 Inexpressibility results

In this section, we start investigating how to reduce query entailment in PPOBDA to query entailment in OBDA, based on the query equivalence definition given in the previous section.

Before proceeding further, we notice that, given a PPOBDA specification $\mathcal{E} = \langle \mathcal{T}, \mathcal{S}, \mathcal{M}, \mathcal{P} \rangle$, a natural question is whether the OBDA specification $\mathcal{J} = \langle \mathcal{T}, \mathcal{S}, \mathcal{M} \rangle$, i.e., obtained by simply eliminating the policy $\mathcal{P}$ from $\mathcal{E}$, is query-equivalent to $\mathcal{E}$ under censors in either **CQ** or **GA**. In other terms, one might wonder whether the mapping $\mathcal{M}$ is already realizing a filter on the data such that denials in $\mathcal{P}$ are never violated by the underlying data retrieved through $\mathcal{M}$, whatever source database for $\mathcal{J}$ is considered[5]. If this would be the case, the entailment problems we are studying would become trivial. However, since

[5] Note that this is not the problem studied in [3] (see also the discussion in Section 2).

the bodies of mapping assertions are FO queries, to answer the above question we should decide entailment in FO, which is an undecidable problem.

The following result says that, under censors in **CQ**, constructing an OBDA specification query-equivalent to $\mathcal{E}$ is in general not possible, already for the case of a TBox that does not contain axioms. As a consequence, entailment of BCQs under censors in **CQ** cannot be solved through transformation in a query-equivalent OBDA specification, whatever logic is used for the TBox.

Theorem 1. *There exists a PPOBDA specification $\mathcal{E} = \langle \mathcal{T}, \mathcal{S}, \mathcal{M}, \mathcal{P} \rangle$ with $\mathcal{T} = \emptyset$ for which there does not exist an OBDA specification $\mathcal{J}$ such that $\mathcal{E}$ and $\mathcal{J}$ are query-equivalent under censors in* **CQ**.

Proof. Consider the PPOBDA specification $\mathcal{E} = \langle \mathcal{T}, \mathcal{S}, \mathcal{M}, \mathcal{P} \rangle$ such that $\mathcal{T} = \emptyset$, $\mathcal{S}$ contains the binary relation T, where $T \in \Sigma_R$, $\mathcal{M} = \{T(x, y) \rightsquigarrow Q(x, y)\}$, where $Q \in \Sigma_T$, and $\mathcal{P} = \{\forall x \,.\, Q(a, x) \rightarrow \bot, \quad \forall x \,.\, Q(x, a) \rightarrow \bot\}$, where a belongs to Σ_C. Assume that $\mathcal{J}$ is an OBDA specification such that $\mathcal{E}$ and $\mathcal{J}$ are query-equivalent under censors in **CQ**, and let $\mathcal{M}'$ be the mapping of $\mathcal{J}$. Consider now the case when the source database D consists of the fact $T(a, a)$. First, it is immediate to see that, given the policy $\mathcal{P}$, no BCQ mentioning the individual a can belong to any censor $\mathsf{cens}(\cdot)$ in **CQ**-$\mathsf{OptCens}_{\mathcal{E}}$. Then, since a is the only individual appearing in D, it follows that no BCQ mentioning any individual can belong to any censor $\mathsf{cens}(\cdot)$ in **CQ**-$\mathsf{OptCens}_{\mathcal{E}}$. This implies that the mapping $\mathcal{M}'$ of $\mathcal{J}$ cannot retrieve any instance from D, i.e., $ret(\mathcal{J}, D)$ is empty, and therefore no BCQ is entailed by $(\mathcal{J}, D)$. On the other hand, the OBDA setting $(\langle \mathcal{T}, \mathcal{S}, \mathcal{M} \rangle, D)$ infers purely existential BCQs. For instance, all the BCQs expressing existential cycles of any length over the role Q, that is all the queries of the form $\exists x_0, \ldots, x_n \,.\, Q(x_0, x_1) \wedge Q(x_1, x_2) \wedge \ldots \wedge Q(x_n, x_0)$, where $n \in \mathbb{N}$. All such queries can be positively answered by the PPOBDA setting $(\mathcal{E}, D)$ without revealing a secret: so, all such queries belong to every censor $\mathsf{cens}(\cdot)$ in **CQ**-$\mathsf{OptCens}_{\mathcal{E}}$. Since they are not entailed by $(\mathcal{J}, D)$, this contradicts the hypothesis that $\mathcal{E}$ and $\mathcal{J}$ are query-equivalent under censors in **CQ**, thus proving the theorem. □

Hereinafter, we focus on *DL-Lite*$_{\mathcal{R}}$ PPOBDA specifications, i.e., whose TBox is expressed in the logic *DL-Lite*$_{\mathcal{R}}$. The following theorem states that the same issue of Theorem 1 arises also under censors in **GA**.

Theorem 2. *There exists a DL-Lite*$_{\mathcal{R}}$ *PPOBDA specification $\mathcal{E}$ for which there does not exist an OBDA specification $\mathcal{J}$ such that $\mathcal{E}$ and $\mathcal{J}$ are query-equivalent under censors in* **GA**.

Proof. From Theorem 6 in [17], it follows that, for *DL-Lite*$_{\mathcal{R}}$ PPOBDA specifications, **GA**-Cens-Entailment is coNP-hard in data complexity. Instead, standard conjunctive query entailment for OBDA specifications with a *DL-Lite*$_{\mathcal{R}}$ TBox is in AC^0 in data complexity [20]. This clearly shows the thesis. □

6 Embedding a policy into the mapping

Towards the identification of a notion of censor that allows us to always transform a PPOBDA specification $\mathcal{E}$ into a query-equivalent OBDA one, we define below a new notion of censor that suitably approximates censors for $\mathcal{E}$ in **GA**.

Definition 5 (Intersection GA censor). *Given a PPOBDA specification* $\mathcal{E} = \langle \mathcal{T}, \mathcal{S}, \mathcal{M}, \mathcal{P} \rangle$*, the* intersection GA (IGA) censor *for* $\mathcal{E}$ *is the function* $\mathsf{cens}_{IGA}(\cdot)$ *such that, for every database instance* D *for* $\mathcal{S}$*,* $\mathsf{cens}_{IGA}(D) = \bigcap_{\mathsf{cens} \in \mathbf{GA}\text{-}\mathsf{OptCens}_{\mathcal{E}}} \mathsf{cens}(D)$.

Example 3. Let $\mathcal{E}$ be the PPOBDA specification of Example 1, and let $D = \{\mathsf{company}(c_1, \text{'oil'}), \mathsf{company}(c_2, \text{'oil'}), \mathsf{company}(c_3, \text{'oil'}), \mathsf{license}(c_1, c_4), \mathsf{operator}(p_1, c_2)\}$ be a source database for $\mathcal{E}$. One can verify that $\mathsf{cens}_{IGA}(D) = \{Comp(c_1), Comp(c_2), Comp(c_3), OilComp(c_3), Comp(c_4), Pipeline(p_1)\}$. □

Notice that, differently from the previous notions of censors, the IGA censor is unique. Then, given a source database instance D for $\mathcal{E}$ and a BCQ q, *IGA-Cens-Entailment* is the problem of deciding whether $\mathcal{T} \cup \mathsf{cens}_{IGA}(D) \models q$. If this is the case, we write $(\mathcal{E}, D) \models^{cqe}_{IGA} q$.

The following proposition, whose proof is straightforward, says that IGA-Cens-Entailment is a sound approximation of GA-Cens-Entailment.

Proposition 1. *Given a PPOBDA specification* $\mathcal{E}$*, a source database* D *for* $\mathcal{E}$ *and a BCQ* q*, if* $(\mathcal{E}, D) \models^{cqe}_{IGA} q$ *then* $(\mathcal{E}, D) \models^{cqe}_{\mathbf{GA}} q$.

We naturally extend Definition 4 to IGA censors. Given a PPOBDA specification $\mathcal{E} = \langle \mathcal{T}, \mathcal{S}, \mathcal{M}, \mathcal{P} \rangle$ and an OBDA specification $\mathcal{J} = \langle \mathcal{T}, \mathcal{M}', \mathcal{S} \rangle$, we say that $\mathcal{E}$ and $\mathcal{J}$ are *query-equivalent under IGA censor* if for every source database D for $\mathcal{E}$ and every BCQ q, $(\mathcal{E}, D) \models^{cqe}_{IGA} q$ iff $(\mathcal{J}, D) \models q$.

We now prove that every *DL-Lite*$_{\mathcal{R}}$ PPOBDA specification $\mathcal{E}$ admits an OBDA specification $\mathcal{J}$ that is query-equivalent under IGA censor to $\mathcal{E}$, and provide an algorithm to build $\mathcal{J}$. The intuition behind our algorithm is as follows. For any source database D, we want that $ret(\mathcal{J}, D)$ does not contain all those facts of $ret(\mathcal{E}, D)$ that together with the TBox $\mathcal{T}$ lead to the violation of the policy $\mathcal{P}$. At the same time, we want this elimination of facts to be done in a minimal way, according to our definition of IGA censor. Thus only "really dangerous" facts have to be dropped from $ret(\mathcal{E}, D)$. These facts actually belong to at least one minimal (w.r.t. set containment) ABox $\mathcal{A}$ such that $\mathcal{T} \cup \mathcal{A} \cup \mathcal{P}$ is inconsistent. Note that in this case, for each fact $\alpha \in \mathcal{A}$ there is at least a censor $\mathsf{cens}(\cdot) \in \mathbf{GA}\text{-}\mathsf{OptCens}_{\mathcal{E}}$ such that $\mathsf{cens}(D)$ does not contain α. Therefore α does not belong to the set $\mathsf{cens}_{IGA}(D)$, where $\mathsf{cens}_{IGA}(\cdot)$ is the IGA censor for $\mathcal{E}$.

Identifying such facts is easier if we can reason on each denial in isolation. For this to be possible, the policy $\mathcal{P}$ must enjoy the following property: for every denial $\delta \in \mathcal{P}$, every minimal (w.r.t. set containment) ABox $\mathcal{A}$ such that $\{\delta\} \cup \mathcal{T} \cup \mathcal{A}$ is inconsistent is also a minimal ABox such that $\mathcal{P} \cup \mathcal{T} \cup \mathcal{A}$ is inconsistent. This is, however, not always the case. Consider, e.g., the policy $\mathcal{P} = \{\forall x.A(x) \wedge$

$B(x) \rightarrow \bot; \forall x.A(x) \rightarrow \bot\}$. The ABox $\{A(d), B(d)\}$ is a minimal ABox violating the first denial, but is not a minimal ABox violating $\mathcal{P}$, since $\{A(d)\}$ violates the second denial (in this example $\mathcal{T} = \emptyset$). We thus first transform $\mathcal{P}$ into a policy $\mathcal{P}'$ enjoying the above property.

To this aim we introduce the notion of *extended denial assertion* (or simply extended denial), which is a formula of the form $\forall \vec{x}.\phi(\vec{x}) \wedge \neg\pi(\vec{x}) \rightarrow \bot$ such that $\exists \vec{x}.\phi(\vec{x})$ is a BCQ and $\pi(\vec{x})$ is a (possibly empty) disjunction of conjunctions of equality atoms of the form $t_1 = t_2$, where t_1 and t_2 are either variables in $\vec{x}$ or constants in Σ_C. An extended policy is a finite set of extended denials.

Definition 6. *Given a policy $\mathcal{P}$ and an extended policy $\mathcal{P}'$. We say that $\mathcal{P}'$ is a* non-redundant representation of $\mathcal{P}$ *if the following conditions hold: (i) for every ABox $\mathcal{A}$, $\mathcal{P} \cup \mathcal{A}$ is inconsistent iff $\mathcal{P}' \cup \mathcal{A}$ is inconsistent; (ii) for every extended denial δ' occurring in $\mathcal{P}'$, every minimal ABox $\mathcal{A}$ such that $\{\delta'\} \cup \mathcal{A}$ is inconsistent is also a minimal ABox such that $\mathcal{P} \cup \mathcal{A}$ is inconsistent.*

One might think that computing a non-redundant representation of $\mathcal{P}$ means simply eliminating from $\mathcal{P}$ each denial δ such that $\mathcal{P} \setminus \{\delta\} \cup \mathcal{T} \models \delta$. In fact, only eliminating denials that are (fully) logically inferred by other denials (and the TBox) is not sufficient, since some redundancies can occur for specific instantiations of the denials. For example, $\delta_1 = \forall x, y.Q(x, y) \wedge C(y) \rightarrow \bot$ is not inferred by $\delta_2 = \forall x.Q(x, x) \rightarrow \bot$, but it becomes inferred when $x = y$. This implies that a minimal violation of δ_1 where the two arguments of Q are the same (e.g., $\{Q(a, a), C(a)\}$) is not a minimal violation of $\{\delta_1, \delta_2\}$ (since $Q(a, a)$ alone is already a violation of δ_2). A non-redundant representation of this policy would be $\{\delta'_1, \delta_2\}$, where $\delta'_1 = \forall x, y.Q(x, y) \wedge C(y) \wedge \neg(x = y) \rightarrow \bot$. Our algorithm to compute a non-redundant policy $\mathcal{P}'$, called policyRefine, takes into account also this situation, applying a variant of the saturate method used in [16] to solve a similar problem in the context of consistent query answering over ontologies.

Hereinafter, we assume that $\mathcal{P}$ has been *expanded* w.r.t. $\mathcal{T}$, that is, $\mathcal{P}$ contains every denial δ such that $\mathcal{P} \cup \mathcal{T} \models \delta$. In this way, to establish non-redundancy we can look only at $\mathcal{P}$, getting rid of $\mathcal{T}$. To expand the policy, we use the rewriting algorithm perfectRef of [7] to reformulate (the premise of) denials in $\mathcal{P}$ with respect to the assertions in $\mathcal{T}$.

Example 4. Consider the same PPOBDA specification $\mathcal{E}$ of Example 1. By rewriting each denial in $\mathcal{P}$ w.r.t. $\mathcal{T}$ through perfectRef[6], we obtain the following set of denials.

d_1: $\forall x, y.OilComp(x) \wedge IssuesLic(x, y) \wedge Comp(y) \rightarrow \bot$
d_2: $\forall x, y.PipeOp(x, y) \wedge OilComp(y) \rightarrow \bot$
d_3: $\forall x, y.OilComp(x) \wedge IssuesLic(x, y) \wedge OilComp(y) \rightarrow \bot$
d_4: $\forall x, y.OilComp(x) \wedge IssuesLic(x, y) \rightarrow \bot$
d_5: $\forall x, y, z.OilComp(x) \wedge IssuesLic(x, y) \wedge PipeOp(z, y) \rightarrow \bot$

Intuitively, perfectRef adds to the original denials d_1 and d_2 the new denials d_3, d_4 and d_5, obtained by rewriting the atom $Comp(y)$ in d_1 according to

[6] For details on perfectRef, we refer the reader to [7].

Algorithm 1: PolicyEmbed

input: a *DL-Lite*$_\mathcal{R}$ TBox $\mathcal{T}$, a mapping $\mathcal{M}$, a policy $\mathcal{P}$;
output: a mapping $\mathcal{M}'$;
1) let $\hat{\mathcal{P}}$ be the expansion of the policy $\mathcal{P}$ w.r.t $\mathcal{T}$;
2) $\mathcal{P}' \rightarrow \mathsf{policyRefine}(\hat{P})$;
3) $\mathcal{M}' \leftarrow \emptyset$;
4) **for each** atomic concept C **do**
5) $\quad \psi \leftarrow \mathsf{addPolicyConditions}(C(x), \mathcal{P}')$;
6) $\quad \phi_p \leftarrow \mathsf{expand}(C(x),\ \mathcal{T})$;
7) $\quad \phi_n \leftarrow \mathsf{expand}(\psi,\ \mathcal{T})$;
8) $\quad \mathcal{M}' \leftarrow \mathcal{M}' \cup \{\mathsf{unfold}(\phi_p \wedge \neg\phi_n, \mathcal{M}) \rightsquigarrow C(x)\}$
9) **for each** atomic role Q **do**
10) $\quad \psi \leftarrow \mathsf{addPolicyConditions}(Q(x,y), \mathcal{P}')$;
11) $\quad \phi_p \leftarrow \mathsf{expand}(Q(x,y),\ \mathcal{T})$;
12) $\quad \phi_n \leftarrow \mathsf{expand}(\psi,\ \mathcal{T})$;
13) $\quad \mathcal{M}' \leftarrow \mathcal{M}' \cup \{\mathsf{unfold}(\phi_p \wedge \neg\phi_n, \mathcal{M})) \rightsquigarrow Q(x,y)\}$
14) **return** $\mathcal{M}'$;

the inclusions $OilComp \sqsubseteq Comp$, $\exists IssuesLic^- \sqsubseteq Comp$, and $\exists PipeOp^- \sqsubseteq Comp$, respectively (for d_4, perfectRef also unifies two atoms having $IssuesLic$ as predicate). It is easy to verify that d_1, d_3 and d_5 are implied by d_4, and thus must be discarded. So, the non-redundant policy $\mathcal{P}'$ contains only d_2 and d_4. □

Algorithm 1 shows our overall procedure, called PolicyEmbed. Steps 1 expands the input policy $\mathcal{P}$ into the policy $\hat{\mathcal{P}}$ by using $\mathsf{perfectRef}(\mathcal{P}, \mathcal{T})$. Step 2 produces the non-redundant policy $\mathcal{P}'$ by means of $\mathsf{policyRefine}(\hat{\mathcal{P}})$. Then, the algorithm constructs one mapping assertion for each ontology predicate. We discuss steps 4-8 for concepts (steps 9-13 for roles are analogous).

The algorithm $\mathsf{addPolicyConditions}(C(x), \mathcal{P}')$ constructs an FO query ψ expressing the disjunction of all BCQs corresponding to the premise of a denial $\delta \in \mathcal{P}'$ such that $C(x)$ unifies with an atom of δ. For instance, if $\mathcal{P}'$ contains $\forall x.C(x) \wedge D(x) \rightarrow \bot$ and $\forall x, y.C(x) \wedge Q(x,y) \wedge E(y) \rightarrow \bot$, $\mathsf{addPolicyConditions}(C(x), \mathcal{P}')$ returns $((\exists x.C(x) \wedge D(x)) \vee (\exists x, y.C(x) \wedge Q(x,y) \wedge E(y)))$. This is actually the union of all the conditions that lead to the generation of dangerous facts for C.

Then, the algorithm $\mathsf{expand}(\varphi, \mathcal{T})$ rewrites every positive atom α occurring in the formula φ according to the TBox $\mathcal{T}$. More precisely, the expansion $\mathsf{expand}(C(x), \mathcal{T})$ of a positive concept atom is the disjunction of the atoms of the form $A(x)$ (resp. $\exists y.Q(x,y)$, $\exists y.Q(y,x)$), where A is an atomic concept (resp. Q is an atomic role), such that $\mathcal{T} \models A \sqsubseteq C$ (resp. $\mathcal{T} \models \exists Q \sqsubseteq C$, $\mathcal{T} \models \exists Q^- \sqsubseteq C$). For example, if $\mathcal{T}$ infers $A \sqsubseteq C$ and $\exists Q \sqsubseteq C$, then $\mathsf{expand}(C(x), \mathcal{T})$ returns $C(x) \vee A(x) \vee \exists y.Q(x,y)$. The expansion $\mathsf{expand}(Q(x,y), \mathcal{T})$ of a role atom is defined analogously. Finally, the expansion $\mathsf{expand}(\varphi, \mathcal{T})$ of an arbitrary formula φ is obtained by replacing each occurrence of a positive atom α in φ with the formula $\mathsf{expand}(\alpha, \mathcal{T})$.

At step 8, the mapping is incremented with the mapping assertion for C. The function unfold realizes a typical unfolding for GAV mapping [23]. The presence of (the expansion of) the subformula ψ in $\neg\phi_n$ guarantees that no fact causing a violation of a denial involving C is retrieved.

Example 5. In our ongoing example, $\mathsf{PolicyEmbed}(\mathcal{T}, \mathcal{M}, \mathcal{P})$ returns

$$\begin{aligned}
\mathcal{M}' = \{\ & m_1\colon \exists y.\mathsf{company}(x,y) \rightsquigarrow \mathit{Comp}(x),\\
& m_1'\colon \mathsf{company}(x,\text{`}oil\text{'}) \rightsquigarrow \mathit{Comp}(x),\\
& m_1''\colon \exists x.\mathsf{license}(x,y) \rightsquigarrow \mathit{Comp}(y),\\
& m_1'''\colon \exists x.\mathsf{operator}(x,y) \rightsquigarrow \mathit{Comp}(y),\\
& m_2'\colon \mathsf{company}(x,\text{`}oil\text{'}) \wedge \neg((\exists y.\mathsf{company}(x,\text{`}oil\text{'}) \wedge \mathsf{license}(x,y)) \vee\\
& \qquad (\exists z.\mathsf{operator}(z,x) \wedge \mathsf{company}(x,\text{`}oil\text{'}))) \rightsquigarrow \mathit{OilComp}(x),\\
& m_3'\colon \mathsf{license}(x,y) \wedge \neg(\mathsf{license}(x,y) \wedge \mathsf{company}(x,\text{`}oil\text{'})) \rightsquigarrow \mathit{IssuesLic}(x,y)\\
& m_4'\colon \mathsf{operator}(x,y) \wedge \neg(\mathsf{operator}(x,y) \wedge \mathsf{company}(x,\text{`}oil\text{'})) \rightsquigarrow \mathit{PipeOp}(x,y)\\
& m_5'\colon \exists y.\mathsf{operator}(x,y) \rightsquigarrow \mathit{Pipeline}(x)\ \}
\end{aligned}$$

For the database instance D for $\mathcal{S}$ provided in Example 3, one can verify that $\mathsf{cens}_{IGA}(D) = \mathit{ret}(\langle \mathcal{T}, \mathcal{S}, \mathcal{M}' \rangle, D)$. □

PolicyEmbed can be used to realize a PPOBDA-OBDA transformation.

Theorem 3. *Let $\mathcal{E} = \langle \mathcal{T}, \mathcal{S}, \mathcal{M}, \mathcal{P} \rangle$ be a DL-Lite$_\mathcal{R}$ PPOBDA specification, and let $\mathcal{J}$ be the OBDA specification $\langle \mathcal{T}, \mathcal{S}, \mathcal{M}' \rangle$, where $\mathcal{M}'$ is the mapping returned by $\mathsf{PolicyEmbed}(\mathcal{T}, \mathcal{M}, \mathcal{P})$. Then, $\mathcal{J}$ is query-equivalent to $\mathcal{E}$ under IGA censor.*

Proof. Let D be a source database for $\mathcal{S}$. We prove the theorem by showing that $\mathit{ret}(\mathcal{J}, D)$ is equal to $\mathsf{cens}_{IGA}(D)$, where $\mathsf{cens}_{IGA}(\cdot)$ is the IGA censor for $\mathcal{E}$.

We start by showing a lemma that is crucial for this proof. From now on, we denote by $\mathcal{A}$ the ABox $\mathit{ret}(\langle \mathcal{T}, \mathcal{S}, \mathcal{M} \rangle, D)$, i.e., the ABox retrieved from D through the initial mapping $\mathcal{M}$. Moreover, we denote by $\mathcal{A}''$ the ABox $\mathit{ret}(\langle \mathcal{T}, \mathcal{S}, \mathcal{M}'' \rangle, D)$, where $\mathcal{M}''$ is the mapping obtained from the algorithm by discarding the formulas ϕ_n, i.e., when $\mathsf{unfold}(\phi_p \wedge \neg\phi_n, \mathcal{M})$ is replaced with $\mathsf{unfold}(\phi_p, \mathcal{M})$ in steps 8 and 13 of the algorithm.

The next lemma follows easily from the definition of the algorithm expand:

Lemma 1. $\mathcal{A}'' = \mathit{cl}_{\mathbf{GA}}(\mathcal{T} \cup \mathcal{A})$.

Informally, the lemma states that the "positive" part of the mapping computed by the algorithm retrieves from D exactly the set of ground atoms derivable by the TBox $\mathcal{T}$ from the ABox $\mathcal{A}$ retrieved from D through $\mathcal{M}$.

In the following, we prove that every concept assertion $C(a)$ belongs to $\mathit{ret}(\mathcal{J}, D)$ iff $C(a)$ belongs to $\mathsf{cens}_{IGA}(D)$ (the proof for role assertions is analogous). From now on, ϕ_p and ϕ_n denote the formula computed for $C(x)$ at step 6 and step 7 of the algorithm, respectively.

First, assume that the concept assertion $C(a)$ belongs to $\mathit{ret}(\mathcal{J}, D)$ but does not belong to $\mathsf{cens}_{IGA}(D)$. Then, there exists a censor $\mathsf{cens}'(\cdot)$ in **GA** for $\mathcal{E}$ such that $C(a) \notin \mathsf{cens}'(D)$. Now, there are two possible cases:

(i) $C(a) \notin \mathsf{cl}_{\mathbf{GA}}(\mathcal{T} \cup \mathcal{A})$. In this case, by Lemma 1 it follows that $C(a) \notin \mathcal{A}''$, hence $\mathsf{unfold}(\phi_p, \mathcal{M})$ (that is, the positive part of the mapping for the concept C in $\mathcal{M}'$) is false in D for $x = a$, and therefore $C(a)$ does not belong to $ret(\mathcal{J}, D)$;
(ii) $C(a)$ belongs to a minimal violation of $\mathcal{P}$ in $\mathsf{cl}_{\mathbf{GA}}(\mathcal{T} \cup \mathcal{A})$: then, from Definition 6 it follows that there exists a denial δ in $\mathcal{P}'$ such that $C(a)$ belongs to a minimal violation of δ in $\mathsf{cl}_{\mathbf{GA}}(\mathcal{T} \cup \mathcal{A})$. Consequently, from the definition of the algorithms addPolicyConditions and expand it follows that $\mathsf{unfold}(\phi_n, \mathcal{M})$ (that is, the negative part of the mapping for the concept C in $\mathcal{M}'$) is true in D for $x = a$, and therefore $C(a)$ does not belong to $ret(\mathcal{J}, D)$.

Conversely, assume that the concept assertion $C(a)$ belongs to $\mathsf{cens}_{IGA}(D)$ but does not belong to $ret(\mathcal{J}, D)$. Then, the mapping for the concept C in $\mathcal{M}'$ is false for $x = a$. Now, there are two possible cases:

(i) $\mathsf{unfold}(\phi_p, \mathcal{M})$ is false in D for $x = a$. This immediately implies by Lemma 1 that $C(a) \notin \mathsf{cl}_{\mathbf{GA}}(\mathcal{T} \cup \mathcal{A})$: hence, in every censor cens' in $\mathbf{GA}$ for $\mathcal{E}$, $C(a) \notin \mathsf{cens}'(D)$, and therefore $C(a) \notin \mathsf{cens}_{IGA}(D)$;
(ii) $\mathsf{unfold}(\phi_n, \mathcal{M})$ is true in D for $x = a$. From the definition of the algorithms addPolicyConditions and expand, this immediately implies that there exists $\delta \in \mathcal{P}'$ such that $C(a)$ belongs to a minimal violation of δ in $\mathsf{cl}_{\mathbf{GA}}(\mathcal{T} \cup \mathcal{A})$: then, from Definition 6 it follows that $C(a)$ belongs to a minimal violation of $\mathcal{P}$ in $\mathsf{cl}_{\mathbf{GA}}(\mathcal{T} \cup \mathcal{A})$. Consequently, there exists a censor cens' in $\mathbf{GA}$ for $\mathcal{E}$ such that $C(a) \notin \mathsf{cens}'(D)$, and therefore $C(a) \notin \mathsf{cens}_{IGA}(D)$. □

7 Experiments

In this section, we report the results of the experimentation we carried out using the NPD benchmark for OBDA [15]. The benchmark is based on real data coming from the oil industry: the Norwegian Petroleum Directorate (NPD) FactPages. It provides an OWL 2 ontology, the NPD database, the mapping between the ontology and the database, an RDF file specifying the instances of the ontology predicates, i.e., the retrieved ABox of the OBDA setting, and a set of 31 SPARQL queries. We remark that we tested non-Boolean CQs adapted from this set (details later on).

For our experimentation, we produced an approximationin OWL 2 QL of the OWL 2 benchmark ontology. Moreover, we made use of the benchmark RDF file containing the retrieved ABox to populate a relational database constituted by unary and binary tables (a unary table for each concept of the ontology and a binary table for each role and each attribute). Finally, we specified a mapping between the ontology and this database. In this case, the mapping in simply a set of one-to-one mapping assertions, i.e., every ontology predicate is mapped to the database table containing its instances. This kind of OBDA specification, with the simplest possible form of mapping assertions, allowed us to verify the feasibility of our technique for data protection, leaving aside the impact of more complex queries in the mapping.

In the resulting OBDA setting, the TBox comprises 1377 axioms over 321 atomic concepts, 135 roles, and 233 attributes. There are in total 2 millions of instances circa, which are stored in a MySQL database of 689 tables.

We specified a policy $\mathcal{P}$ constituted by the following denials:

d_1: $\forall d, l.DevelopmentWellbore(d) \wedge developmentWellboreForLicence(d, l) \wedge ProductionLicence(l) \rightarrow \bot$

d_2: $\forall d, t, w, b, q, f.Discovery(d) \wedge dateIncludedInField(d, t) \wedge containsWellbore(b, w) \wedge wellboreForDiscovery(w, d) \wedge ExplorationWellbore(w) \wedge quadrantLocation(b, q) \wedge explorationWellboreForField(w, f) \rightarrow \bot$

d_3: $\forall c, w.WellboreCore(c) \wedge coreForWellbore(c, w) \wedge DevelopmentWellbore(w) \rightarrow \bot$

d_4: $\forall c, f, d.Company(c) \wedge currentFieldOperator(f, c) \wedge Field(f) \wedge includedInField(d, f) \wedge Discovery(d) \rightarrow \bot$

d_5: $\forall w, e, f, l.belongsToWell(w, e) \wedge wellboreAgeHc(w, l) \wedge drillingFacility(w, f) \wedge ExplorationWellbore(w) \rightarrow \bot$

d_6: $\forall f, p, l.Field(f) \wedge currentFieldOwner(f, p) \wedge ProductionLicence(p) \wedge licenseeForLicence(l, p) \rightarrow \bot$.

As queries, we considered nine (non-Boolean) CQs from the ones provided with the NPD benchmark. Strictly speaking, some of these queries in the benchmark are not CQs, since they use aggregation operators, but we extracted from them their conjunctive subqueries. The resulting queries are reported below.

q_3 : $\exists li.ProductionLicence(li) \wedge name(li, ln) \wedge dateLicenceGranted(li, d) \wedge isActive(li, a) \wedge licensingActivityName(li, an)$

q_4 : $\exists li, w.ProductionLicence(li) \wedge name(li, n) explorationWellboreForLicence(w, li) \wedge dateWellboreEntry(w, e)$

q_5 : $\exists le, li, c.licenseeForLicence(le, li) \wedge ProductionLicence(li) \wedge name(li, ln) \wedge licenceLicensee(le, c) \wedge name(c, n) \wedge dateLicenseeValidFrom(le, d)$

q_9 : $\exists li, w.ProductionLicence(li) \wedge name(li, n) \wedge belongsToWell(w, we) \wedge explorationWellboreForLicence(w, li) \wedge name(we, wn)$

q_{12} : $\exists w, lu, c.wellboreStratumTopDepth(w, st) \wedge wellboreStratumBottomDepth(w, sb) \wedge stratumForWellbore(w, u) \wedge name(u, n) \wedge inLithostratigraphicUnit(w, lu) \wedge name(lu, un) \wedge WellboreCore(c) \wedge coreForWellbore(c, u) \wedge coreIntervalTop(c, ct) \wedge coreIntervalBottom(c, cb)$

q_{13} : $\exists wc, we, c.WellboreCore(wc) \wedge coreForWellbore(wc, we) \wedge name(we, wn) \wedge Wellbore(we) \wedge wellboreCompletionYear(we, y) \wedge drillingOperatorCompany(we, c) \wedge name(c, cn)$

q_{14} : $\exists we, c.Wellbore(we) \wedge name(we, n) \wedge wellboreCompletionYear(we, y) \wedge drillingOperatorCompany(we, c) \wedge name(c, cn)$

q_{18} : $\exists p, m, f, op.productionYear(p, \text{`2010'}) \wedge productionMonth(p, m) \wedge producedGas(p, g) \wedge producedOil(p, o) \wedge productionForField(p, f) \wedge name(f, fn) \wedge currentFieldOperator(f, op) \wedge Field(f) \wedge shortName(op, \text{`statoil petroleum as'})$

q_{44} : $\exists y, f, c.wellboreAgeTD(w, a) \wedge explorationWellboreForField(w, f) \wedge wellboreEntryYear(w, y) \wedge Field(f) \wedge name(f, fn) \wedge coreForWellbore(c, w)$

We executed each query in seven different settings, in each of which we considered an incremental number of denials in the policy among those given above. For each setting, we computed a new mapping through a Java implementation of the algorithm illustrated in Section 6. So, in the first setting, we used the mapping computed by considering the empty policy $\mathcal{P}_\emptyset$; in the second one, we

	q_3 [5]		q_4 [4]		q_5 [6]		q_9 [5]		q_{12} [10]		q_{13} [7]		q_{14} [5]		q_{18} [9]		q_{44} [6]	
Policy	**res**	**time**	**res**	**time**	**res**	**time**	**res**	**time**	**res**	**time**	**res**	**time**	**res**	**time**	**res**	**time**	**res**	**time**
$\mathcal{P}_\emptyset$	910	4789	1558	4625	17254	4545	1566	4648	96671	7368	22541	6410	141439	20150	339	6933	5078	4179
$\mathcal{P}_1$	910	3871	1558	4111	17254	4782	1566	4401	96671	7133	22541	6886	130341	15544	339	6128	5078	4078
$\mathcal{P}_2$	910	4154	880	4078	17254	4628	888	4204	96671	6852	22541	5007	126679	16566	339	5887	12	4413
$\mathcal{P}_3$	910	4080	880	4189	17254	4902	888	3953	96641	7746	15340	5623	124248	16807	339	5873	12	4653
$\mathcal{P}_4$	910	4419	880	4089	17254	5015	888	4487	96641	7836	15340	6011	124248	17393	339	6893	12	4318
$\mathcal{P}_5$	910	5548	880	4373	17254	6224	888	4422	96641	8683	15340	6499	123816	20116	339	7201	12	4491
$\mathcal{P}_6$	910	4309	880	4029	14797	5189	888	4785	96641	8297	15340	6796	123816	17513	339	6176	12	4475

Table 1: CQE test results. The "res" columns contain the size of the results while the "time" columns contain the query evaluation times in milliseconds.

considered the policy $\mathcal{P}_1$ containing only the denial d_1; in the third one, we considered the policy $\mathcal{P}_2$ containing the denials d_1 and d_2; and so on. For each query, we report in Table 1 the size of the result and the query evaluation time, columns "res" and columns "time" in the table, respectively. The number in square brackets near each query name indicates the length of the query.

For our experiments, we used the OBDA Mastro system, and a standard laptop with Intel i5 @1.6Ghz processor and 8Gb of RAM.

Values in Table 1 show the effect of the policy on the size of the result of the queries. Specifically, we have that the queries q_0, q_3, and q_{18} are not censored in any of the considered settings. The answers to the queries q_4, q_9, and q_{44} are affected by the introduction of the denial d_2 in the policy, while the denial d_3 alters the answers of the queries q_{12} and q_{13}. Some answers to the query q_5 are cut away by the introduction of the denial d_6 in the policy. Moreover, the query q_{14} is affected by the denials d_1, d_2, d_3, and d_5. Finally, the denial d_4 alters no queries. Notably, although the policy alters the query results, one can see that the execution time is only slightly affected. This suggests that our proposed technique can be effectively used for protecting data in OBDA setting.

8 Conclusions

Our current research is mainly focused on modifying the user model formalized in our framework to capture richer data protection scenarios. In particular, the user model we adopted (which we inherited from previous works on CQE over ontologies) assumes that an attacker has only the ability of making standard inference reasoning on the ontology and the query answers. Under these assumptions, data declared as confidential are certainly protected in our framework. We are also investigating more expressive forms of policy. Finally, while our experimental evaluation clearly shows the practical feasibility of our approach, we still have to consider the issue of optimization of our algorithms and implementation.

References

1. F. Baader, D. Calvanese, D. McGuinness, D. Nardi, and P. F. Patel-Schneider, editors. *The Description Logic Handbook: Theory, Implementation and Applications*. Cambridge University Press, 2nd edition, 2007.

2. M. Benedikt, P. Bourhis, L. Jachiet, and M. Thomazo. Reasoning about disclosure in data integration in the presence of source constraints. In *Proc. of IJCAI*, pages 1551–1557, 2019.
3. M. Benedikt, B. Cuenca Grau, and E. V. Kostylev. Logical foundations of information disclosure in ontology-based data integration. *AIJ*, 262:52–95, 2018.
4. J. Biskup and P. A. Bonatti. Controlled query evaluation for known policies by combining lying and refusal. *AMAI*, 40(1-2):37–62, 2004.
5. J. Biskup and T. Weibert. Keeping secrets in incomplete databases. *Int. J. of Information Security*, 7(3):199–217, 2008.
6. P. A. Bonatti and L. Sauro. A confidentiality model for ontologies. In *Proc. of ISWC*, volume 8218 of *LNCS*, pages 17–32, 2013.
7. D. Calvanese, G. De Giacomo, D. Lembo, M. Lenzerini, and R. Rosati. Tractable reasoning and efficient query answering in description logics: The *DL-Lite* family. *J. of Automated Reasoning*, 39(3):385–429, 2007.
8. R. Chirkova and T. Yu. Exact detection of information leakage: Decidability and complexity. *Trans. Large Scale Data Knowl. Centered Syst.*, 32:1–23, 2017.
9. G. Cima, D. Lembo, R. Rosati, and D. F. Savo. Controlled query evaluation in description logics through instance indistinguishability. In *Proc. of IJCAI*, pages 1791–1797, 2020.
10. B. Cuenca Grau and I. Horrocks. Privacy-preserving query answering in logic-based information systems. In *Proc. of ECAI*, pages 40–44, 2008.
11. B. Cuenca Grau, E. Kharlamov, E. V. Kostylev, and D. Zheleznyakov. Controlled query evaluation over OWL 2 RL ontologies. In *Proc. of ISWC*, pages 49–65, 2013.
12. B. Cuenca Grau, E. Kharlamov, E. V. Kostylev, and D. Zheleznyakov. Controlled query evaluation for datalog and OWL 2 profile ontologies. In *Proc. of IJCAI*, pages 2883–2889, 2015.
13. S. Das, S. Sundara, and R. Cyganiak. R2RML: RDB to RDF mapping language. W3C Recommendation, W3C, Sept. 2012.
14. A. Doan, A. Y. Halevy, and Z. G. Ives. *Principles of Data Integration*. Morgan Kaufmann, 2012.
15. D. Lanti, M. Rezk, G. Xiao, and D. Calvanese. The NPD benchmark: Reality check for OBDA systems. In *Proc. of EDBT*, pages 617–628, 2015.
16. D. Lembo, M. Lenzerini, R. Rosati, M. Ruzzi, and D. F. Savo. Inconsistency-tolerant query answering in ontology-based data access. *J. of Web Semantics*, 33:3–29, 2015.
17. D. Lembo, R. Rosati, and D. F. Savo. Revisiting controlled query evaluation in description logics. In *Proc. of IJCAI*, pages 1786–1792, 2019.
18. B. Motik, B. Cuenca Grau, I. Horrocks, Z. Wu, A. Fokoue, and C. Lutz. OWL 2 Web Ontology Language profiles (second edition). W3C Recommendation, W3C, Dec. 2012.
19. A. Nash and A. Deutsch. Privacy in GLAV information integration. In *Proc. of ICDT*, pages 89–103, 2007.
20. A. Poggi, D. Lembo, D. Calvanese, G. De Giacomo, M. Lenzerini, and R. Rosati. Linking data to ontologies. *J. on Data Semantics*, X:133–173, 2008.
21. G. L. Sicherman, W. de Jonge, and R. P. van de Riet. Answering queries without revealing secrets. *ACM Trans. Database Syst.*, 8(1):41–59, 1983.
22. P. Stouppa and T. Studer. Data privacy for $\mathcal{ALC}$ knowledge bases. In *Proc. of LFCS*, pages 409–421, 2009.
23. G. Xiao, D. Calvanese, R. Kontchakov, D. Lembo, A. Poggi, R. Rosati, and M. Zakharyaschev. Ontology-based data access: A survey. In *Proc. of IJCAI*, pages 5511–5519, 2018.