Mapping Analysis in Ontology-based Data Access: Algorithms and Complexity

Domenico Lembo[1], Jose Mora[1], Riccardo Rosati[1],
Domenico Fabio Savo[1], Evgenij Thorstensen[2]

[1] Sapienza Università di Roma
lastname@dis.uniroma1.it

[2] University of Oslo
evgenit@ifi.uio.no

Abstract. Ontology-based data access (OBDA) is a recent paradigm for accessing data sources through an ontology that acts as a conceptual, integrated view of the data, and declarative mappings that connect the ontology to the data sources. We study the formal analysis of mappings in OBDA. Specifically, we focus on the problem of identifying mapping inconsistency and redundancy, two of the most important anomalies for mappings in OBDA. We consider a wide range of ontology languages that comprises OWL 2 and all its profiles, and examine mapping languages of different expressiveness over relational databases. We provide algorithms and establish tight complexity bounds for the decision problems associated with mapping inconsistency and redundancy. Our results prove that, in our general framework, such forms of mapping analysis enjoy nice computational properties, in the sense that they are not harder than standard reasoning tasks over the ontology or over the relational database schema.

1 Introduction

Ontology-based data access (OBDA) [18] is a recent paradigm for accessing *data sources* through an *ontology* (also called TBox) that acts as a conceptual, integrated view of the data, and declarative *mappings* that connect the ontology to the data sources. The framework of OBDA has received a lot of attention in the last years: many theoretical studies have paved the way for the construction of OBDA systems (e.g., [6, 19, 11] and the development of OBDA projects for enterprise data management in various domains [2, 15].

One important aspect in OBDA concerns the construction of a system specification, i.e., defining the ontology and the mappings over an existing set of data sources. Mappings are indeed the most complex part of an OBDA specification, since they have to capture the semantics of the data sources and express such semantics in terms of the ontology. More precisely, a mapping is a set of assertions, each one associating a query $\phi(\boldsymbol{x})$ over the source schema with a query $\psi(\boldsymbol{x})$ over the ontology. The intuitive meaning of a mapping assertion is that all the tuples satisfying the query $\phi(\boldsymbol{x})$ also satisfy the query $\psi(\boldsymbol{x})$. We write a mapping assertion as $\phi(\boldsymbol{x}) \rightsquigarrow \psi(\boldsymbol{x})$. As an example, consider $\mathtt{tabP}(x, y, z) \rightsquigarrow \mathsf{person}(x), \mathsf{name}(x, y)$, which maps the ontology predicates person and name to the database relation tabP, thus indicating how ontology instances can be constructed from the data retrieved at the sources.

The first experiences in the application of the OBDA framework in real-world scenarios (e.g., [2, 15]) have shown that the semantic distance between the conceptual and the data layer is often very large, because data sources are mostly application-oriented: this makes the definition, debugging, and maintenance of mappings a hard and complex task. Such experiences have clearly shown the need of tools for supporting the management of mappings.

However, no specific approach (with the exception of [17]) has explicitly dealt with the problem of mapping analysis in the context of OBDA. The work on *schema mappings in data exchange* has considered the problem of analyzing the formal properties of mappings, although in a different framework. Indeed, in data exchange the ontology is replaced by a relational schema, called target schema, possibly equipped with tuple-generating dependencies and equality-generating dependencies [10, 3]. Such kinds of dependencies are not able to capture arbitrary ontology languages, such as those considered in this paper. Also, in data exchange suitable conditions are imposed on the interaction among database dependencies to guarantee that *finite* instances for the target schema exist that are coherent with the database at the sources, the mapping, and the target dependencies. Such conditions are normally not imposed in OBDA, where the focus is not on moving data from the sources to the target, and indeed we do not adopt them. Among the works on data exchange, [12] is the closest to our approach: it proposes techniques for the optimization and normalization of schema mappings, in particular, finding a global, semantically equivalent transformation of a set of mappings that is optimal with respect to some minimality criterion.

In a recent paper [17], we started providing a theoretical basis for mapping management support in OBDA, focusing on the formal analysis of mappings in ontology-based data access. In particular, in that paper the two most important semantic anomalies of mappings have been analyzed: inconsistency and redundancy. Roughly speaking, an inconsistent mapping for an ontology and a source schema is a specification that gives rise to logical contradictions with the ontology and/or the source schema. Then, a mapping $\mathcal{M}$ is redundant with respect to an OBDA specification if adding $\mathcal{M}$ to the specification does not change its semantics. Verifying whether a mapping is affected by these anomalies is a crucial task in OBDA. A designer that is creating (or modifying) the mapping needs to know whether the new (or updated) mapping leads to an inconsistency. Given the complexity of the OBDA specification, this is very hard to check manually. Similarly, a redundant mapping is not wanted, since it is very difficult to maintain; furthermore, it may affect the performance of query answering [8].

The work presented in [17] has defined both a *local* notion of mapping inconsistency and redundancy, which focuses on single mapping assertions, and a *global* notion, where inconsistency and redundancy is considered with respect to a whole mapping specification (set of mapping assertions). In this paper, we study the computational properties of verifying both local and global mapping inconsistency and redundancy in an OBDA specification. We consider a wide range of ontology languages that comprises the description logics underlying OWL 2 and all its profiles (OWL 2 EL, OWL 2 QL, and OWL 2 RL),[1] and examine mapping languages of different expressiveness (the so-called GAV and GLAV mappings [9]) over sources corresponding to relational

[1] `http://www.w3.org/TR/owl2-profiles/`

databases. We provide algorithms and establish tight complexity bounds for the decision problems associated with both local and global mapping inconsistency and mapping redundancy, for both GAV mappings and a large class of GLAV mappings, and for both combined complexity and TBox complexity (which only considers the size of the TBox).

The outcome of our analysis is twofold. First, in our framework, it is possible to define *general and modular techniques* that are able to reduce the analysis of mappings to the composition of standard reasoning tasks over the ontology (inconsistency and instance checking, query answering) and over the data sources (query answering and containment). This is a non-trivial result, because mappings are formulas combining both ontology and data source elements. Moreover, the above forms of mapping analysis enjoy *nice computational properties*, in the sense that they are not harder than the above mentioned standard reasoning tasks over the ontology and the data sources (see Figure 1 and Figure 2 at the end of the paper).

The above results allow us to conclude that, in our OBDA framework, the formal analysis of mappings is feasible, at least for ontology languages enjoying nice computational properties, as in the case of the three OWL 2 profiles.

The paper is organized as follows. In Section 2 we recall OBDA specifications and the formal notions of mapping inconsistency and redundancy in OBDA. In Section 3 we study the complexity of checking local and global mapping inconsistency, while in Section 4 we study the complexity of verifying local and global mapping redundancy. We conclude the paper in Section 5.

2 Preliminaries

In the following, we assume to have three pairwise disjoint, countably infinite alphabets: an alphabet $\Gamma_{\mathcal{T}}$ of ontology predicates, an alphabet $\Gamma_{\mathcal{S}}$ of source schema predicates, and an alphabet $\Gamma_{\mathcal{C}}$ of constants.

Source schemas. A source schema $\mathcal{S}$ is a relational schema containing relations in $\Gamma_{\mathcal{S}}$, possibly equipped with integrity constraints (ICs). A *legal instance D for $\mathcal{S}$* is a database for $\mathcal{S}$ (i.e., a finite set of ground atoms over $\mathcal{S}$ and the constants in $\Gamma_{\mathcal{C}}$) that satisfies the ICs of $\mathcal{S}$. We denote by $Const(D)$ the set of constants occurring in D.

We consider *simple schemas*, i.e., relational schemas without ICs, and *FD schemas*, i.e., simple schemas with *functional dependencies* (FDs) [1]. We adopt standard notions for conjunctive queries (CQs) over relational schemas [1], and by a CQ over a source schema $\mathcal{S}$ we mean a CQ over the alphabet of $\mathcal{S}$. With $\phi(\boldsymbol{x})$ we denote a CQ with free variables $\boldsymbol{x}$. The number of variables in $\boldsymbol{x}$ is the arity of the query. A Boolean CQ is a CQ without free variables. Given a CQ q over $\mathcal{S}$ and a legal instance D for $\mathcal{S}$, $eval(q, D)$ denotes the evaluation of q over D. Throughout the paper $\mathcal{S}$ will always denote a source schema.

Ontologies. We consider ontologies expressed in some Description Logic (DL) language $\mathcal{L}_{\mathcal{O}}$ and use standard DL notions [16]. In particular, a DL ontology $\mathcal{O}$ is pair $\langle \mathcal{T}, A \rangle$, where $\mathcal{T}$ is the TBox and $\mathcal{A}$ is the ABox. $\mathcal{O}$, $\mathcal{T}$, and $\mathcal{A}$ will always have the

same meaning in the paper. As in the W3C standard OWL, we do not interpret ontologies under the Unique Name Assumption. We denote with $Mod(\mathcal{O})$ the set of models of $\mathcal{O}$, and with $\mathcal{O} \models \alpha$ the fact that $\mathcal{O}$ entails a sentence α. Also, by *ontology inconsistency* we mean the task of deciding whether $Mod(\mathcal{O}) = \emptyset$, and by *instance checking* the task of deciding whether $\mathcal{O} \models \beta$, where β is a ground atom. By *CQs over* $\mathcal{O}$ we mean CQs over the alphabet of the TBox of $\mathcal{O}$, and by *CQ entailment* the task of checking whether $\mathcal{O} \models q$, where q is a Boolean CQ. In the following, we consider DLs that are the logical basis of the W3C standard OWL and of its profiles, i.e., $\mathcal{SROIQ}$ [14], which underpins OWL, *DL-Lite*$_R$ [5], which is the basis of OWL 2 QL, RL [16], a simplified version of OWL 2 RL, and $\mathcal{EL}_\bot$, a slight extension of the DL $\mathcal{EL}$ [4], which is the basis of OWL 2 EL.

Mappings. A *mapping assertion* m from a source schema $\mathcal{S}$ to a TBox $\mathcal{T}$ has the form $\phi(\boldsymbol{x}) \rightsquigarrow \psi(\boldsymbol{x})$, where $\phi(\boldsymbol{x})$, called the *body of* m, and $\psi(\boldsymbol{x})$, called the *head of* m, are queries over $\mathcal{S}$ and $\mathcal{T}$, respectively, both with free variables $\boldsymbol{x}$, which are called the *frontier variables*. The number of variables in $\boldsymbol{x}$ is the *arity* of the mapping assertion. Given a mapping assertion m, we also use $FR(m)$ do denote the frontier variables $\boldsymbol{x}$, $head(m)$ to denote the query $\psi(\boldsymbol{x})$, and $body(m)$ to denote the query $\phi(\boldsymbol{x})$. We also remark that queries used in our mappings, besides variables, may contain constants from $\Gamma_\mathcal{C}$. A mapping $\mathcal{M}$ from $\mathcal{S}$ to $\mathcal{T}$ is a finite set of mapping assertions from $\mathcal{S}$ to $\mathcal{T}$. Hereinafter $\mathcal{M}$ will always denote a mapping.

In principle, $\phi(\boldsymbol{x})$ and $\psi(\boldsymbol{x})$ can be specified in generic query languages. The literature on data integration and OBDA has mainly considered $\phi(\boldsymbol{x})$ expressed in a (fragment of) first-order logic, and $\psi(\boldsymbol{x})$ expressed as a CQ [9, 17, 18]. In this paper, we focus on the notable cases in which $\phi(\boldsymbol{x})$ is a CQ over $\mathcal{S}$ and $\psi(\boldsymbol{x})$ is as follows:

- $\psi(\boldsymbol{x})$ is a CQ over $\mathcal{T}$. This is a powerful form of GLAV mapping [9], and is among the most expressive types of mappings studied in the literature. We refer to it simply as *GLAV*.
- $\psi(\boldsymbol{x})$ is a CQ with a bounded number of existential variables in the head. This is a practically relevant form of GLAV mappings, which we call *GLAV*$_{BE}$.
- $\psi(\boldsymbol{x})$ is a CQ without existential variables in the head. Such mappings are the most used in OBDA applications [13, 2], and are a special case of the W3C standard R2RML mappings [7]. According to the data integration literature, we call them *GAV*.

We say that a mapping assertion m *is active on a source instance* D if $eval(body(m), D)$ is a non-empty set of tuples of constants. A mapping $\mathcal{M}$ is active on D if all its mapping assertions $m \in \mathcal{M}$ are active on D.

Without loss of generality, we assume that different mapping assertions use different variable symbols. A *freeze* of a set of atoms Γ is a set of ground atoms obtained from Γ by replacing every variable with a *fresh* distinct constant. In this paper, the freeze is always used in the context of a mapping $\mathcal{M}$, so it suffices to assume that fresh constants do not appear in $\mathcal{M}$. Different freezes of the same set of atoms are equal up to renaming of constants. Thus, in the following we assume, without loss of generality, that the freeze of a set of atoms Γ is unique and is obtained by replacing each variable occurrence x with a fresh constant c_x, and we denote it by $freeze(\Gamma)$.

Given a mapping assertion m of arity n and an n-tuple of constants $\boldsymbol{t}$, we denote by $m(\boldsymbol{t})$ the mapping assertion obtained by replacing $FR(m)$ in m with the constants in $\boldsymbol{t}$.

OBDA specifications. An OBDA specification is a triple $\mathcal{J} = \langle \mathcal{T}, \mathcal{S}, \mathcal{M} \rangle$. The semantics of $\mathcal{J}$ is given with respect to a database instance D legal for $\mathcal{S}$: a model for $\mathcal{J}$ w.r.t. D is a FOL interpretation $\mathcal{I}$ over the alphabet $\Gamma_{\mathcal{T}} \cup \Gamma_{\mathcal{C}}$ that satisfies both $\mathcal{T}$ and $\mathcal{M}$. Formally, we say that $\mathcal{I}$ satisfies the mapping $\mathcal{M}$ if for each assertion $m \in \mathcal{M}$ and each tuple of constants $\boldsymbol{t}$ such that $\boldsymbol{t} \in \mathit{eval}(\mathit{body}(m), D)$ we have that $\mathcal{I} \models \mathit{head}(m(\boldsymbol{t}))$. The set of models of $\mathcal{J}$ w.r.t. D is denoted with $\mathit{Mod}(\mathcal{J}, D)$. Also, we use $(\mathcal{J}, D)$ to denote $\mathcal{J}$ with source instance D, say that $(\mathcal{J}, D)$ *is inconsistent* if $\mathit{Mod}(\mathcal{J}, D) = \emptyset$, and denote with $(\mathcal{J}, D) \models \alpha$ the entailment of a sentence α by $(\mathcal{J}, D)$.

Example 1. We consider a source schema $\mathcal{S}$ where the `plants` relation contains data on extraction facilities, while the `eZones` relation contains data on the areas used for oil and gas extraction. Below, the underlined attributes represent the keys of the relations, which can be expressed by FDs.

$$\texttt{plants}(\underline{\texttt{id\_pl}}, \texttt{pl\_typ}, \texttt{id\_zn}) \qquad \texttt{eZones}(\underline{\texttt{id\_zn}}, \texttt{zn\_typ})$$

The following RL TBox models a very small portion of the domain of oil and gas production extracted from an ontology developed within the Optique EU project[2]. In particular, the TBox focuses on the facilities (concept Facility) used in the oil and gas extraction and on the geographical areas (concept Area) in which they are located (role locatedIn). Facilities that are located in a marine area (concept MarArea) are platforms (concept Platform).

$$\begin{array}{lll} \mathcal{T} = \{\ \mathsf{Platform} \sqsubseteq \mathsf{Facility}, & \mathsf{MarArea} \sqsubseteq \mathsf{Area}, & \exists \mathsf{locatedIn} \sqsubseteq \mathsf{Facility}, \\ \quad \exists \mathsf{locatedIn}^- \sqsubseteq \mathsf{Area} & \mathsf{Facility} \sqcap \mathsf{Area} \sqsubseteq \bot & \exists \mathsf{locatedIn}.\mathsf{MarArea} \sqsubseteq \mathsf{Platform}\ \} \end{array}$$

An example of a GAV mapping $\mathcal{M}$ from $\mathcal{S}$ to $\mathcal{T}$ follows:

$$\begin{array}{lll} m_1: & \texttt{plants}(x, y, z) & \rightsquigarrow \mathsf{Facility}(x), \mathsf{locatedIn}(x, z) \\ m_2: & \texttt{plants}(x', \text{'pl'}, y') & \rightsquigarrow \mathsf{Platform}(x') \\ m_3: & \texttt{eZones}(z', \text{'mz'}) & \rightsquigarrow \mathsf{MarArea}(z'). \end{array}$$

■

Mapping inconsistency and redundancy. The following definitions are taken from [17].

In brief, a mapping assertion m from $\mathcal{S}$ to $\mathcal{T}$ is head-inconsistent or body-inconsistent if $\mathit{head}(m)$ or $\mathit{body}(m)$ have certainly an empty evaluation in every model for $\mathcal{T}$ or legal instance for $\mathcal{S}$, respectively.

Definition 1 (mapping head-inconsistency). *Let $\mathcal{T}$ be a TBox, $\mathcal{S}$ a source schema, and $m : \phi(\boldsymbol{x}) \rightsquigarrow \psi(\boldsymbol{x})$ a mapping assertion from $\mathcal{S}$ to $\mathcal{T}$. We say that m* is head-inconsistent for $\mathcal{T}$ *if $\mathcal{T} \models \forall \boldsymbol{x}.(\neg\psi(\boldsymbol{x}))$.*

Example 2. Let $\mathcal{T}$ and $\mathcal{S}$ be as in Example 1. Consider the following mapping assertion:

$$m: \texttt{plants}(x, \text{'pl'}, z) \rightsquigarrow \mathsf{Platform}(x), \mathsf{MarArea}(x)$$

[2] `http://www.optique-project.eu/`

Then, m is head-inconsistent for $\mathcal{T}$ since $\mathcal{T} \models \mathsf{Platform} \sqcap \mathsf{MarArea} \sqsubseteq \bot$. ∎

Definition 2 (mapping body-inconsistency). *Let $\mathcal{T}$ be a TBox, $\mathcal{S}$ a source schema, and $m : \phi(\boldsymbol{x}) \rightsquigarrow \psi(\boldsymbol{x})$ a mapping assertion from $\mathcal{S}$ to $\mathcal{T}$. We say that m* is body-inconsistent for *$\mathcal{S}$ if $\mathcal{S} \models \forall \boldsymbol{x}.(\neg\phi(\boldsymbol{x}))$.*

Example 3. Let $\mathcal{T}$ and $\mathcal{S}$ be as in Example 1. Then, the following mapping assertion is body-inconsistent for $\mathcal{S}$.

$m : \texttt{plants}(x, \text{'pl'}, z), \texttt{plants}(x, \text{'ref'}, k) \rightsquigarrow \mathsf{Facility}(x)$ ∎

We extend the inconsistency notions to whole mapping assertions and whole mappings.

Definition 3 (local mapping inconsistency). *Let $\mathcal{T}$ be a TBox, $\mathcal{S}$ a source schema, and $m : \phi(\boldsymbol{x}) \rightsquigarrow \psi(\boldsymbol{x})$ a mapping assertion from $\mathcal{S}$ to $\mathcal{T}$. We say that m* is inconsistent for *$\langle \mathcal{T}, \mathcal{S} \rangle$ if m is head-inconsistent for $\mathcal{T}$ or body-inconsistent for $\mathcal{S}$.*

Definition 4 (global mapping inconsistency). *Let $\mathcal{J} = \langle \mathcal{T}, \mathcal{S}, \mathcal{M} \rangle$ be an OBDA specification. We say that $\mathcal{M}$* is globally inconsistent for *$\langle \mathcal{T}, \mathcal{S} \rangle$ if there does not exist a source instance D legal for $\mathcal{S}$ such that $\mathcal{M}$ is active on D and $Mod(\mathcal{J}, D) \neq \emptyset$.*

Intuitively, it is impossible to consistently activate all the assertions of a globally inconsistent mapping simultaneously.

Example 4. Let $\mathcal{J} = \langle \mathcal{T}, \mathcal{S}, \mathcal{M} \rangle$ be an OBDA specification where $\mathcal{T}$ and $\mathcal{S}$ are as in Example 1. Suppose that the mapping $\mathcal{M}$ contains the following mapping assertions:

$m_1 : \texttt{plants}(x, y, z) \rightsquigarrow \mathsf{Area}(x)$
$m_2 : \texttt{plants}(x', \text{'pl'}, z') \rightsquigarrow \mathsf{Platform}(x'), \mathsf{locatedIn}(x', z')$

It is easy to see that $\mathcal{M}$ is globally inconsistent for $\langle \mathcal{T}, \mathcal{S} \rangle$, because $\mathcal{T} \models \mathsf{Platform} \sqcap \mathsf{Area} \sqsubseteq \bot$ and every activation of m_2 also activates m_1, thus implying $\mathsf{Platform}(x)$ and $\mathsf{Area}(x)$ for the same individual x. ∎

Then, we recall the notion of global mapping redundancy.

Definition 5 (global mapping redundancy). *Let $\mathcal{J} = \langle \mathcal{T}, \mathcal{S}, \mathcal{M} \rangle$ be an OBDA specification and let $\mathcal{M}'$ be a mapping from $\mathcal{S}$ to $\mathcal{T}$. We say that $\mathcal{M}'$* is globally redundant for *$\mathcal{J}$ if, for every source instance D that is legal for $\mathcal{S}$, $Mod(\langle \mathcal{T}, \mathcal{S}, \mathcal{M} \rangle, D) = Mod(\langle \mathcal{T}, \mathcal{S}, \mathcal{M} \cup \mathcal{M}' \rangle, D)$.*

Informally, a mapping $\mathcal{M}'$ is redundant for an OBDA specification $\mathcal{J}$ if adding $\mathcal{M}'$ to $\mathcal{J}$ produces a specification equivalent to $\mathcal{J}$.

Example 5. Let $\langle \mathcal{T}, \mathcal{S}, \mathcal{M} \rangle$ be an OBDA specification, where $\mathcal{T}$ and $\mathcal{S}$ are as in Example 1, and $\mathcal{M}$ is as follows:

$m_1 : \texttt{plants}(x, y, z), \texttt{eZones}(z, \text{'mz'}) \rightsquigarrow \mathsf{locatedIn}(x, z)$
$m_2 : \texttt{eZones}(x', \text{'mz'}) \rightsquigarrow \mathsf{MarArea}(x')$
$m_3 : \texttt{plants}(y', \text{'pl'}, z'), \texttt{eZones}(z', \text{'mz'}) \rightsquigarrow \mathsf{Platform}(y')$

Then, $\{m_3\}$ is globally redundant for $\langle \mathcal{T}, \mathcal{S}, \{m_1, m_2\} \rangle$. ∎

Finally, *local mapping redundancy* is a special case of global mapping redundancy in which the mappings $\mathcal{M}$ and $\mathcal{M}'$ are both composed of a single assertion.

3 Complexity of mapping inconsistency

We now study local and global mapping inconsistency and show that, for every DL $\mathcal{L}_\mathcal{O}$, both problems have the same TBox complexity as ontology inconsistency in $\mathcal{L}_\mathcal{O}$. We also establish combined complexity results for the DLs considered in this paper.

We start with some auxiliary definitions.

Definition 6 (minimal instance activating a mapping). *Let $\mathcal{M}$ be a mapping and let $\mathcal{S}$ be a source schema. A* minimal instance for $\mathcal{S}$ that activates $\mathcal{M}$ *is a source instance D legal for $\mathcal{S}$ such that $\mathcal{M}$ is active on D and, for every source instance D' legal for $\mathcal{S}$ such that $\mathcal{M}$ is active on D', there exists a homomorphism h from $Const(D)$ to $Const(D')$ that maps constants occurring in $\mathcal{M}$ to themselves and is such that $h(D) \subseteq D'$, where $h(D) = \{r(h(c_1), \ldots, h(c_n)) \mid r(c_1, \ldots, c_n) \in D\}$.*

Given a GLAV mapping assertion m of arity n, we denote by $cvars(m)$ the sequence of frontier variables occurring together with an existential variable in an atom of the head of m. Moreover, given an n-tuple of constants $\boldsymbol{t}$, we denote by $cvars(m)[\boldsymbol{t}]$ the tuple of constants obtained from $cvars(m)$ by replacing each occurrence of a frontier variable with the corresponding constant of $\boldsymbol{t}$. For instance, if m is the assertion $\phi(x, w) \rightsquigarrow R(x, y), S(y, z), T(z, w), R(x, z), S(w, x)$, then $cvars(m)$ is the tuple of variables $\langle x, w, x\rangle$, and if $\boldsymbol{t} = \langle a, b\rangle$ then $cvars(m)[\boldsymbol{t}]$ is the tuple of constants $\langle a, b, a\rangle$.

Definition 7 (retrieved ABox). *Given a mapping $\mathcal{M}$ from $\mathcal{S}$ to $\mathcal{T}$ and an instance D legal for $\mathcal{S}$, the* ABox retrieved by $\mathcal{M}$ from D, *denoted by $Retr(\mathcal{M}, D)$, is the ABox defined as follows: $Retr(\mathcal{M}, D) = \{freezeH(head(m(\boldsymbol{t}))) \mid \boldsymbol{t} \in eval(body(m), D)\}$, where $freezeH(head(m(\boldsymbol{t})))$ is the set of atoms obtained from $head(m(\boldsymbol{t}))$ by replacing each occurrence of a (existential) variable x with the fresh constant $c_{x,cvars(m)[\boldsymbol{t}]}$.*

3.1 Local mapping inconsistency

We start from the following property (whose proof is trivial) for the problem of head inconsistency.

Lemma 1. *Let m be a GLAV mapping assertion, $\mathcal{T}$ a TBox, and let $\mathcal{A} = freeze(head(m))$. Then, m is head-inconsistent for $\mathcal{T}$ iff $\langle \mathcal{T}, \mathcal{A}\rangle$ is inconsistent.*

Conversely, inconsistency of an ontology $\langle \mathcal{T}, \mathcal{A}\rangle$ can be immediately reduced to head inconsistency, considering $\mathcal{T}$ as the TBox of the OBDA specification, and constructing a GAV mapping assertion m (with no frontier variables) whose head is the conjunction of the ABox assertions in $\mathcal{A}$. Consequently, the following property holds.

Lemma 2. *For both GAV and GLAV mappings and for every ontology language $\mathcal{L}_\mathcal{O}$, the combined (resp., TBox) complexity of mapping head inconsistency is the same as the combined (resp., TBox) complexity of ontology inconsistency in $\mathcal{L}_\mathcal{O}$.*

Now, from the definition of local mapping inconsistency, it follows that the TBox complexity of local mapping inconsistency is the same as the TBox complexity of mapping head inconsistency. Therefore:

Theorem 1. *For both GAV and GLAV mappings and for every ontology language $\mathcal{L}_\mathcal{O}$, the TBox complexity of local mapping inconsistency is the same as the TBox complexity of ontology inconsistency in $\mathcal{L}_\mathcal{O}$.*

The above theorem implies row 1 in Figure 1.

Moreover, from the definition of local mapping inconsistency, it follows that, for simple source schemas, local mapping inconsistency corresponds to mapping head inconsistency (since all mapping assertions are trivially body-consistent). Therefore:

Corollary 1. *For simple source schemas, for both GAV and GLAV mappings, and for every ontology language $\mathcal{L}_\mathcal{O}$, the combined complexity of local mapping inconsistency is the same as the combined complexity of ontology inconsistency in $\mathcal{L}_\mathcal{O}$.*

The above result is summarized in row 1 in Figure 2.

Then, we analyze the case of FD schemas. We start by defining the algorithm $\mathsf{freezeFD}(\mathcal{M}, \mathcal{S})$, which takes as input a mapping $\mathcal{M}$ and a source schema $\mathcal{S}$, and applies the *chase* procedure [1] to the database instance $D = \bigcup_{m \in \mathcal{M}} \mathit{freeze}(\mathit{body}(m))$ using the functional dependencies of $\mathcal{S}$, and considering the constants occurring in D but not occurring in $\mathcal{M}$ as unifiable terms (since they act as "soft constants" differently from the constants occurring in $\mathcal{M}$). Such a chase procedure runs in PTIME and may end up in two ways: (i) it fails, i.e., it derives that two constants occurring in $\mathcal{M}$ should be equal (which violates the Unique Name Assumption of databases); (ii) it returns a database D' that is obtained from D by unifying constants occurring in D but not occurring in $\mathcal{M}$ according to the equalities induced by the functional dependencies.

We are now able to show the following lemma.

Lemma 3. *Let $\mathcal{S}$ be a source schema and let $\mathcal{M}$ be a mapping. Deciding whether there exists a minimal instance D for $\mathcal{S}$ that activates $\mathcal{M}$, and computing such a D if it exists, can be done: (i) in linear time, if $\mathcal{S}$ is a simple schema; (ii) in PTIME, if $\mathcal{S}$ is an FD schema.*

Proof. The proof easily follows from the fact that the algorithm $\mathsf{freezeFD}(\mathcal{M}, \mathcal{S})$ runs in PTIME, and computes a minimal instance D for $\mathcal{S}$ that activates $\mathcal{M}$ iff such an instance exists. In particular, for property (i), it is easy to verify that, if $\mathcal{S}$ is a simple schema, then $\bigcup_{m \in \mathcal{M}} \mathit{freeze}(\mathit{body}(m))$ is a minimal instance for $\mathcal{S}$ that activates $\mathcal{M}$. For property (ii), in the case when $\mathcal{S}$ is an FD schema, if the algorithm $\mathsf{freezeFD}(\mathcal{M}, \mathcal{S})$ fails, then there exists no legal instance for $\mathcal{S}$ that activates $\mathcal{M}$; otherwise, the algorithm returns a database D' that corresponds to the application of the equalities induced by the functional dependencies over the constants occurring in D but not occurring in $\mathcal{M}$. Therefore, there exists a endomorphism h of the constants in D that is the identity for the constants of $\mathcal{M}$ and is such that $h(D) = D'$. Due to the property of the chase, it follows that such an instance D' is a minimal instance for $\mathcal{S}$ that activates $\mathcal{M}$. □

We can now prove the following property.

Theorem 2. *For both GAV and GLAV mappings, and for FD schemas, the combined complexity of local mapping inconsistency is PTIME-complete for DL-Lite$_R$, RL, and $\mathcal{EL}_\bot$, and is N2EXPTIME-complete for $\mathcal{SROIQ}$.*

Proof. To decide local mapping consistency of m, besides head inconsistency we also have to check body inconsistency of m. This corresponds to decide whether there exists a minimal instance for $\mathcal{S}$ that activates the mapping $\{m\}$. By Lemma 3, this can be done in PTIME in the case of FD schemas. Moreover, consistency of a database D with respect to an FD schema $\mathcal{S}$ can be immediately reduced to mapping body inconsistency, by creating a GAV mapping assertion whose body contains the conjunction of the facts in D. In the case of FD schemas, this provides a PTIME lower bound for body inconsistency, and hence for local mapping inconsistency. The lower bound in the case of $\mathcal{SROIQ}$ follows from the lower bound for head inconsistency. □

The above results are summarized in row 1 in Figure 2.

3.2 Global mapping inconsistency

To define a technique for global mapping inconsistency, we start by showing the following property.

Theorem 3. *Let $\mathcal{J} = \langle \mathcal{T}, \mathcal{S}, \mathcal{M} \rangle$ be an OBDA specification. Then, $\mathcal{M}$ is globally inconsistent for $\langle \mathcal{T}, \mathcal{S} \rangle$ iff either* $\mathsf{freezeFD}(\mathcal{M}, \mathcal{S})$ *fails or the instance D returned by* $\mathsf{freezeFD}(\mathcal{M}, \mathcal{S})$ *is such that $(\mathcal{J}, D)$ is inconsistent.*

Proof. The proof of the only-if part is trivial. For the if part, we will prove the contrapositive: If $\mathcal{M}$ is not globally inconsistent for $\langle \mathcal{T}, \mathcal{S} \rangle$, then $\mathsf{freezeFD}(\mathcal{M}, \mathcal{S})$ returns an instance D such that $(\mathcal{J}, D)$ is consistent.

Let D' be a source instance legal for $\mathcal{S}$ such that $\mathcal{M}$ is active on D', and let $\mathcal{I}$ be a model of $(\mathcal{J}, D')$. Then, $\mathsf{freezeFD}(\mathcal{M}, \mathcal{S})$ does not fail and returns an instance D. Since D is minimal, Definition 6 implies that there exists a homomorphism h from the constants of D to the constants of D' such that $h(D) \subseteq D'$. Now let $\mathcal{I}'$ be the interpretation obtained from $\mathcal{I}$ by changing the interpretation of constants as follows: If c occurs in D then $c^{\mathcal{I}'} = h(c)^{\mathcal{I}}$, otherwise $c^{\mathcal{I}'} = c^{\mathcal{I}}$. It is immediate to verify that $\mathcal{I}'$ is a model for $\mathcal{J}$ w.r.t. D. □

The above theorem immediately implies the following algorithm for deciding the global inconsistency of a GLAV mapping $\mathcal{M}$ for a TBox $\mathcal{T}$ and a source schema $\mathcal{S}$.

Algorithm GlobalInconsistency:
Input: OBDA specification $\langle \mathcal{T}, \mathcal{S}, \mathcal{M} \rangle$
  if (a) algorithm $\mathsf{freezeFD}(\mathcal{M}, \mathcal{S})$ fails
  then return true
  else
    let D be the instance returned by $\mathsf{freezeFD}(\mathcal{M}, \mathcal{S})$;
    if (b) $(\langle \mathcal{T}, \mathcal{S}, \mathcal{M} \rangle, D)$ is inconsistent
    then return true **else** return false

The complexity of step (a) of the algorithm, i.e., deciding the existence and computing a minimal instance for $\mathcal{S}$ that activates $\mathcal{M}$, has been established by Lemma 3. It remains to analyze the complexity of checking inconsistency of $(\langle \mathcal{T}, \mathcal{S}, \mathcal{M} \rangle, D)$. To this aim, we present two techniques for deciding the inconsistency of $(\langle \mathcal{T}, \mathcal{S}, \mathcal{M} \rangle, D)$. First, we use the following property, whose proof easily follows from Definition 7.

Lemma 4. *For every model $\mathcal{I}$ of $(\langle \mathcal{T}, \mathcal{S}, \mathcal{M} \rangle, D)$ there exists a model $\mathcal{I}'$ of $\langle \mathcal{T}, Retr(\mathcal{M}, D) \rangle$ such that $\mathcal{I}$ and $\mathcal{I}'$ coincide except for the interpretation of the constants in $Const(Retr(\mathcal{M}, D)) \setminus Const(D)$. The converse also holds.*

From the above lemma, to decide inconsistency of $(\langle \mathcal{T}, \mathcal{S}, \mathcal{M} \rangle, D)$, we can compute the ABox $\mathcal{A} = Retr(\mathcal{M}, D)$ and then check inconsistency of $\langle \mathcal{T}, \mathcal{A} \rangle$.

Example 6. Let $\mathcal{J} = \langle \mathcal{T}, \mathcal{S}, \mathcal{M} \rangle$ be the OBDA specification of Example 4. We show how algorithm GlobalInconsistency runs on $\mathcal{J}$. First, the algorithm computes a minimal instance D for $\mathcal{S}$ by means of the algorithm freezeFD (cf. Lemma 3). In our example, this actually coincides with computing $freeze(body(m))$ for each mapping $m \in \mathcal{M}$. Hence, we have that $D = \{\texttt{plants}(\mathsf{c}_x, \mathsf{c}_y, \mathsf{c}_z), \texttt{plants}(\mathsf{c}_{x'}, \mathsf{pl}, \mathsf{c}_{z'})\}$. The second step consists in checking if $\langle \mathcal{J}, D \rangle$ is consistent. To this end, one can exploit Lemma 4 and: (i) compute the ABox $\mathcal{A} = Retr(\mathcal{M}, D)$, which is $\{\mathsf{Area}(\mathsf{c}_x), \mathsf{Area}(\mathsf{c}_{x'}), \mathsf{Platform}(\mathsf{c}_{x'}), \mathsf{Platform}(\mathsf{c}_x), \mathsf{locatedIn}(\mathsf{c}_{x'}, \mathsf{c}_{z'}), \mathsf{locatedIn}(\mathsf{c}_x, \mathsf{c}_z)\}$ and (ii) check the consistency of the ontology $\langle \mathcal{T}, \mathcal{A} \rangle$. Since, e.g., both $\mathsf{Area}(\mathsf{c}_{x'})$ and $\mathsf{Platform}(\mathsf{c}_{x'})$ belong to $\mathcal{A}$, and since $\mathcal{T} \models \mathsf{Platform} \sqcap \mathsf{Area} \sqsubseteq \bot$, the ontology $\langle \mathcal{T}, \mathcal{A} \rangle$ is inconsistent. Hence, the algorithm returns true. ∎

Now, observe that the cost of computing $Retr(\mathcal{M}, D)$ does not depend on the size of the TBox. This implies that, with respect to TBox complexity, the complexity of ontology inconsistency is an upper bound for global mapping inconsistency. Conversely, ontology inconsistency can be easily reduced to global mapping inconsistency, by creating a GAV mapping assertion (with no frontier variables) whose head is the conjunction of the ABox assertions in $\mathcal{A}$. Consequently:

Theorem 4. *For both simple and FD schemas, for both GAV and GLAV mappings, and for every ontology language $\mathcal{L}_\mathcal{O}$, the TBox complexity of global mapping inconsistency is the same as the TBox complexity of ontology inconsistency in $\mathcal{L}_\mathcal{O}$.*

The above theorem implies row 2 in Figure 1.

To establish combined complexity, we define a second way to decide inconsistency of $(\langle \mathcal{T}, \mathcal{S}, \mathcal{M} \rangle, D)$. We start from the following property.

Lemma 5. *Let $\mathcal{M}$ be a GLAV$_{BE}$ mapping, and let D be a source instance. Then, the size of $Retr(\mathcal{M}, D)$ is polynomial with respect to the size of $\mathcal{M}$ and D.*

Proof. When $\mathcal{M}$ is a GAV mapping, from Definition 7 it follows that the number of assertions in $Retr(\mathcal{M}, D)$ is bounded by $(n_c \cdot n_v) + (n_r \cdot n_v^2)$, where n_c is the number of concepts, n_r is the number of roles, and n_v is the number of constants occurring in D and $\mathcal{M}$. When $\mathcal{M}$ is a GLAV$_{\text{BE}}$ mapping, observe that, by Definition 7, the number of fresh constants n_f occurring in $Retr(\mathcal{M}, D)$ is not greater than $m \cdot k \cdot n^k$, where m is the number of mapping assertions in $\mathcal{M}$, n is the number of constants in D, and k is the maximum number of occurrences of existential variables in the head of a mapping assertion (observe that k is the maximum length of $cvars(m)$ in the definition of $Retr(\mathcal{M}, D)$). Since k is bounded in GLAV$_{\text{BE}}$ mappings, we derive that such a number of constants n_f is polynomially bounded. And since the number of assertions in $Retr(\mathcal{M}, D)$ is bounded by $(n_c \cdot n_w) + (n_r \cdot n_w^2)$, where n_c is the number of concepts, n_r is the number of roles, and $n_w = n_v + n_f$, the thesis follows. □

Notice that the above property does not hold for arbitrary GLAV mappings (for which $Retr(\mathcal{M}, D)$ may be of exponential size), so in the rest of this section we focus on GLAV_{BE} mappings. Notice also that the above lemma does not imply that for GLAV_{BE} mappings $Retr(\mathcal{M}, D)$ can be computed in polynomial time with respect to the size of $\mathcal{M}$ and D: conversely, it is immediate to verify that deciding whether an ABox assertion belongs to $Retr(\mathcal{M}, D)$ is an NP-hard problem.

From the above lemma and from Lemma 4, it follows that, in the case of GLAV_{BE} mappings, inconsistency of $(\langle \mathcal{T}, \mathcal{S}, \mathcal{M} \rangle, D)$ can be decided by checking the existence of a polynomial subset $\mathcal{A}'$ of $Retr(\mathcal{M}, D)$ such that $\langle \mathcal{T}, \mathcal{A}' \rangle$ is inconsistent.

Given a mapping assertion m, a *grounding for* m is the mapping assertion obtained from m by replacing every variable in m with a constant symbol. A grounding for a mapping $\mathcal{M}$ is a set $\{m_g \mid \exists\ m \in \mathcal{M}$ s.t. m_g is a grounding for $m\}$. Now let D be a source instance. A grounding $\mathcal{G}$ for $\mathcal{M}$ is *generated by* D if, for every $m_g \in \mathcal{G}$, every atom in $body(m_g)$ occurs in D. Given a grounding $\mathcal{G}$ for $\mathcal{M}$, the *ABox induced by* $\mathcal{G}$, denoted as $\mathcal{A}(\mathcal{G})$, is defined as the set of atoms occurring in the heads of the mapping assertions of $\mathcal{G}$.

Lemma 6. *Let $\mathcal{M}$ be a GLAV$_{BE}$ mapping and let D be a source instance. Then: (i) for every grounding $\mathcal{G}$ for $\mathcal{M}$ that is generated by D, if $\langle \mathcal{T}, \mathcal{A}(\mathcal{G}) \rangle$ is inconsistent, then $\langle \mathcal{T}, Retr(\mathcal{M}, D) \rangle$ is inconsistent; (ii) there exists a grounding $\mathcal{G}$ for $\mathcal{M}$ that is generated by D such that $\mathcal{G}$ has polynomial size with respect to $\mathcal{M}$ and D, and $\langle \mathcal{T}, \mathcal{A}(\mathcal{G}) \rangle$ is inconsistent iff $\langle \mathcal{T}, Retr(\mathcal{M}, D) \rangle$ is inconsistent.*

Proof. The proof of (i) follows from the fact that there exists a homomorphism h from $Const(\mathcal{A}(\mathcal{G})) \setminus Const(D)$ to $Const(Retr(\mathcal{M}, D))$ such that $h(\mathcal{A}(\mathcal{G})) \subseteq Retr(\mathcal{M}, D)$. Consequently, if $\mathcal{I}$ is a model for $\langle \mathcal{T}, Retr(\mathcal{M}, D) \rangle$, we can immediately derive a model $\mathcal{I}'$ for $\langle \mathcal{T}, \mathcal{A}(\mathcal{G})) \rangle$ from $\mathcal{I}$ by just changing the interpretation of the constants, defining $c^{\mathcal{I}'} = h(c)^{\mathcal{I}}$ for every $c \in Const(\mathcal{A}(\mathcal{G})) \setminus Const(D)$, and $c^{\mathcal{I}'} = c^{\mathcal{I}}$ otherwise. Then, the proof of (ii) easily follows from (i), Lemma 5 and the fact that, by definition of $Retr(\mathcal{M}, D)$, there exists a grounding $\mathcal{G}$ for $\mathcal{M}$ such that $\mathcal{A}(\mathcal{G})$ is equal to $Retr(\mathcal{M}, D)$. □

Consequently, the following algorithm is able to decide inconsistency of $(\langle \mathcal{T}, \mathcal{S}, \mathcal{M} \rangle, D)$.

Algorithm OBDAInconsistency:
Input: OBDA specification $\langle \mathcal{T}, \mathcal{S}, \mathcal{M} \rangle$ with $\mathcal{M}$ GLAV_{BE} mapping, source instance D
if there exists a polynomial grounding $\mathcal{G}$ for $\mathcal{M}$
such that $\mathcal{G}$ is generated by D **and** the ontology $\langle \mathcal{T}, \mathcal{A}(\mathcal{G}) \rangle$ is inconsistent
then return true **else** return false

We are now able to analyze the combined complexity of the algorithm GlobalInconsistency when step (b) is executed through the algorithm OBDAInconsistency. As shown by Lemma 3, step (a) can always be executed in polynomial time. Then, if the ontology inconsistency check is in PTIME, check (b) can be executed in nondeterministic polynomial time. Consequently, the algorithm GlobalInconsistency provides an NP upper bound for *DL-Lite*$_R$, *RL*, and $\mathcal{EL}_\bot$, while it provides a N2EXPTIME upper bound for $\mathcal{SROIQ}$.

Concerning the lower bounds, the one for $\mathcal{SROIQ}$ is trivial, while the NP bound for the other three cases can be proved by an easy reduction of conjunctive query containment in relational databases. Consequently:

Theorem 5. *For both simple and FD schemas, and for both GAV and GLAV$_{BE}$ mappings: (i) if the ontology language is DL-Lite$_R$, RL, or $\mathcal{EL}_\bot$, then the combined complexity of global mapping inconsistency is NP-complete; (ii) if the ontology language is $\mathcal{SROIQ}$, then the combined complexity of global mapping inconsistency is N2EXPTIME-complete.*

The above results are summarized in row 2 in Fig. 2.

4 Complexity of mapping redundancy

We now show that local and global mapping redundancy have the same TBox complexity as instance checking for GAV mappings and CQ entailment over an ontology for GLAV mappings. We also study the combined complexity for the DLs considered in this paper. We focus on the global case only, since as we said, the local redundancy is a special case of the global one. Also, observe that a mapping $\mathcal{M}'$ is globally redundant for an OBDA specification iff each subset of $\mathcal{M}'$ is redundant. We thus consider only the case in which $\mathcal{M}' = \{m\}$, and with a slight abuse of notation, we call such case global redundancy of a mapping assertion m for $\mathcal{J}$.

From now on, we do not consider the trivial case when m is body-inconsistent for $\mathcal{S}$. Under this assumption, a minimal instance for $\mathcal{S}$ that activates $\{m\}$ always exists (and the algorithm freezeFD does never fail for every mapping $\mathcal{M}$ and source schema $\mathcal{S}$ as input). We notice, however, that all the complexity results of this section also hold without this assumption.

Theorem 6. *Let $\mathcal{J} = \langle \mathcal{T}, \mathcal{S}, \mathcal{M} \rangle$ be an OBDA specification and m a mapping assertion. Then, m is globally redundant for $\mathcal{J}$ iff there exists a minimal instance D for $\mathcal{S}$ that activates $\{m\}$ such that $Mod(\mathcal{J}, D) = Mod(\langle \mathcal{T}, \mathcal{S}, \mathcal{M} \cup \{m\} \rangle, D)$.*

Proof (sketch). The proof of the only-if part is trivial. As for the if part, since a minimal instance has a homomorphism to every other instance, the fact that the models for a minimal instance are the same can be used to show that, for every legal instance D' for $\mathcal{S}$, a model for $(\langle \mathcal{T}, \mathcal{S}, \mathcal{M} \rangle, D')$ has to be a model for $(\langle \mathcal{T}, \mathcal{S}, \mathcal{M} \cup \{m\} \rangle, D')$ too. □

Based on the above theorem, below we provide an algorithm that establishes whether m is globally redundant for $\mathcal{J}$ by checking whether a suitable Boolean CQ is entailed by $\mathcal{J}$ coupled with the minimal instance that activates $\{m\}$ returned by the algorithm freezeFD$(\mathcal{M}, \mathcal{S})$ (cf. Lemma 3). In the following, with a little abuse of notation, we denote with $freeze(FR(m))$ the tuple obtained by freezing the frontier variables of m.

Algorithm mapRedundancy:
Input: OBDA specification $\langle \mathcal{T}, \mathcal{S}, \mathcal{M} \rangle$, mapping assertion m
(a) $D \leftarrow$ freezeFD$(\{m\}, \mathcal{S})$;

let σ be the substitution derived by $\mathsf{freezeFD}(\{m\}, \mathcal{S})$;
$\boldsymbol{t}_F \leftarrow \sigma(\mathit{freeze}(FR(m)))$;
if (b) $(\mathcal{J}, D) \models \mathit{head}(m(\boldsymbol{t}_F))$
then return true **else** return false

In the algorithm, σ denotes the substitution of terms derived by the application of $\mathsf{freezeFD}(\{m\}, \mathcal{S})$, i.e., $\sigma = \{x_1 \rightarrow y_1, \ldots, x_n \rightarrow y_n\}$ where each y_i is a constant (either fresh or non-fresh) and each x_i is a fresh constant in $\mathit{freeze}(\mathit{body}(m))$; σ is applied to the tuple obtained by freezing the frontier variables of m, in order to propagate the term substitutions derived by the chase to such a tuple. Notice that, for simple source schemas, σ is the identity and thus it has no effect. Finally, mapRedundancy verifies whether the Boolean query corresponding to the head of the mapping m whose frontier variables are substituted with $\boldsymbol{t}_F$ is entailed by $(\mathcal{J}, D)$.

The following theorem states that mapRedundancy is sound and complete with respect to the problem of establishing global mapping redundancy (termination of the algorithm is straightforward).

Theorem 7. *Let $\mathcal{J} = \langle \mathcal{T}, \mathcal{S}, \mathcal{M} \rangle$ be an OBDA specification and m a mapping assertion. Then, m is globally redundant for $\mathcal{J}$ iff* mapRedundancy*$(\mathcal{J}, m)$ returns true.*

As shown in Section 3, step (a) can be executed in polynomial time for both simple schemas and FD schemas. As for step (b), the first technique we present is tailored to establish TBox complexity of global mapping redundancy. We first give the following lemma.

Lemma 7. *Let $\mathcal{J} = \langle \mathcal{T}, \mathcal{S}, \mathcal{M} \rangle$ be and OBDA specification, D a minimal instance for $\mathcal{S}$ that activates $\mathcal{M}$, and q a Boolean CQ. Then, $(\mathcal{J}, D) \models q$ iff $\langle \mathcal{T}, \mathit{Retr}(\mathcal{M}, D) \rangle \models q$.*

According to the above result, step (b) of mapRedundancy can be performed by first computing the ABox $\mathit{Retr}(\mathcal{M}, D)$, and then checking whether $(\mathcal{T}, \mathit{Retr}(\mathcal{M}, D)) \models \mathit{head}(m(\sigma(\mathit{freeze}(FR(m)))))$.

Example 7. Consider the OBDA specification $\mathcal{J} = \langle \mathcal{T}, \mathcal{S}, \mathcal{M} \rangle$, where $\mathcal{T}$ and $\mathcal{S}$ are as in Example 1, and $\mathcal{M}$ is as follows:

m_1 : $\mathtt{plants}(x, y, z), \mathtt{eZones}(z, \text{'mz'}) \rightsquigarrow \mathsf{locatedIn}(x, z)$
m_2 : $\mathtt{eZones}(x', \text{'mz'}) \rightsquigarrow \mathsf{MarArea}(x')$.

Moreover, consider the following mapping assertion:

m_3 : $\mathtt{plants}(y', \text{'pl'}, z'), \mathtt{eZones}(z', \text{'mz'}) \rightsquigarrow \mathsf{Platform}(y')$.

The algorithm mapRedundancy first computes $D = \mathsf{freezeFD}(\{m\}, \mathcal{S}) = \{\mathtt{plants}(\mathsf{c}_{y'}, \text{'pl'}, \mathsf{c}_{z'}), \mathtt{eZones}(\mathsf{c}_{z'}, \text{'mz'})\}$. Then, it produces the Boolean CQ $q_{m_3} = \mathit{head}(m(\boldsymbol{t}_F)) = \mathsf{Platform}(\mathsf{c}_{y'})$. To check whether $(\mathcal{J}, D) \models q_{m_3}$ the algorithm computes $\mathit{Retr}(\mathcal{M}, D) = \{\mathsf{locatedIn}(\mathsf{c}_{y'}, \mathsf{c}_{z'}), \mathsf{MarArea}(\mathsf{c}_{z'})\}$. Since $\mathsf{locatedIn}.\mathsf{MarArea} \sqsubseteq \mathsf{Platform} \in \mathcal{T}$, we have that $\langle \mathcal{T}, \mathit{Retr}(\mathcal{M}, D) \rangle \models q_{m_3}$, and thus mapRedundancy returns true (i.e., m_3 is globally redundant for $\mathcal{J}$). ∎

For TBox complexity, we notice that in mapRedundancy both step (a) and the size of $Retr(\mathcal{M}, D)$ do not depend on the TBox $\mathcal{T}$. In particular, we have that:

- In the case of GAV mappings, the check in step (b) corresponds to a linear number (in the size of $head(m)$) of instance checking tasks in the language $\mathcal{L}_\mathcal{O}$ used for $\mathcal{T}$.
- In the case of GLAV mappings, the check in step (b) corresponds to a single Boolean CQ entailment task in $\mathcal{L}_\mathcal{O}$.

Thus, mapRedundancy together with the techniques for step (a) and (b) discussed above allows us to obtain upper bounds for the TBox complexity of global mapping redundancy. More precisely, the complexity of instance checking in $\mathcal{L}_\mathcal{O}$ is an upper bound for GAV mappings, while the complexity of CQ entailment in $\mathcal{L}_\mathcal{O}$ is an upper bound for GLAV.

As for lower bounds, we notice that both instance checking and CQ entailment in $\mathcal{L}_\mathcal{O}$ can be easily reduced to *local* mapping redundancy for GAV and GLAV mappings, respectively, with a technique similar to the one we used for Lemma 2.

The following theorem sums up the above results.

Theorem 8. *For both simple and FD schemas, and for every ontology language $\mathcal{L}_\mathcal{O}$, the TBox complexity of both local and global mapping redundancy for GAV and GLAV mappings is the same as the TBox complexity of instance checking in $\mathcal{L}_\mathcal{O}$ and TBox complexity of CQ entailment in $\mathcal{L}_\mathcal{O}$, respectively.*

The above theorem implies rows 3 and 4 in Figure 1.

Similarly to the case of global mapping inconsistency, since executing step (b) by computing the retrieved ABox $Retr(\mathcal{M}, D)$ requires exponential time in combined complexity, to establish combined complexity of the overall problem we need to resort to a different strategy for step (b). To this aim, we exploit a property that generalizes Lemma 6 (which focuses on inconsistency) to query entailment. From this property, it follows that, for every CQ q that does not mention constants occurring in $Const(Retr(\mathcal{M}, D)) \setminus Const(D)$, and for every GLAV_{BE} mapping $\mathcal{M}$, $\langle \mathcal{T}, Retr(\mathcal{M}, D) \rangle \models q$ can be decided by checking the existence of a polynomial grounding $\mathcal{G}$ for $\mathcal{M}$ that is generated by D such that $\langle \mathcal{T}, \mathcal{A}(\mathcal{G}) \rangle \models q$. Therefore, the following algorithm for checking CQ entailment over an OBDA specification $\mathcal{J}$ and a source instance D follows.

Algorithm CQEntailment:
Input: OBDA specification $\langle \mathcal{T}, \mathcal{S}, \mathcal{M} \rangle$ with $\mathcal{M}$ GLAV_{BE} mapping, source instance D, CQ q
if there exists a polynomial grounding $\mathcal{G}$ for $\mathcal{M}$
such that $\mathcal{G}$ is generated by D **and** $\langle \mathcal{T}, \mathcal{A}(\mathcal{G}) \rangle \models q$
then return true **else** return false

Then, in the case of GLAV_{BE} mappings we can perform step (b) of mapRedundancy by executing CQEntailment$(\mathcal{J}, D, head(m(freeze(FR(m)))))$.

As for combined complexity, in the following we consider simple source schemas for the lower bounds and FD source schemas for the upper bounds. First, step (b) can be executed through the nondeterministic algorithm CQEntailment. Consequently, this algorithm provides an NP upper bound for the case of GLAV_{BE} mappings if, for the

	GAV				GLAV			
task	*DL-Lite*$_R$	RL	$\mathcal{EL}_\bot$	$\mathcal{SROIQ}$	*DL-Lite*$_R$	RL	$\mathcal{EL}_\bot$	$\mathcal{SROIQ}$
local inc.	=NLOGSPACE	=P	=P	=N2EXPTIME	=NLOGSPACE	=P	=P	=N2EXPTIME
global inc.	=NLOGSPACE	=P	=P	=N2EXPTIME	=NLOGSPACE	=P	=P	=N2EXPTIME
local red.	=NLOGSPACE	=P	=P	=N2EXPTIME	=NP	=NP	=NP	open
global red.	=NLOGSPACE	=P	=P	=N2EXPTIME	=NP	=NP	=NP	open

Fig. 1. TBox compl. of mapping inconsistency and redundancy (for both simple and FD schemas).

	GAV				GLAV$_{\text{BE}}$			
task	*DL-Lite*$_R$	RL	$\mathcal{EL}_\bot$	$\mathcal{SROIQ}$	*DL-Lite*$_R$	RL	$\mathcal{EL}_\bot$	$\mathcal{SROIQ}$
local inc.	=NLOGSPACE (SI) =P (FD)	=P	=P	=N2EXPTIME	=NLOGSPACE (SI)* =P (FD)*	=P*	=P*	=N2EXPTIME*
global inc.	=NP	=NP	=NP	=N2EXPTIME	=NP	=NP	=NP	=N2EXPTIME
local red.	=NP	=NP	=NP	=N2EXPTIME	=NP	=NP	=NP	open
global red.	=NP	=NP	=NP	=N2EXPTIME	=NP	=NP	=NP	open

Fig. 2. Combined compl. of mapping inconsistency and redundancy (SI = simple schemas, FD = FD schemas). * The result also holds for arbitrary GLAV mappings.

ontology language $\mathcal{L}_\mathcal{O}$, CQ entailment is in NP, i.e., for *DL-Lite*$_R$, RL, and $\mathcal{EL}_\bot$. The matching NP lower bounds can be proved already for GAV mappings, by an easy reduction of conjunctive query containment in relational databases. In the case of $\mathcal{SROIQ}$, for GLAV$_{\text{BE}}$ mappings we are not able to even prove decidability of global mapping redundancy (since decidability of CQ entailment in this language is currently an open problem too), while for the GAV case we can easily derive a N2EXPTIME exact bound.

Theorem 9. *For both simple and FD source schemas, global and local mapping redundancy are: (i) NP-complete w.r.t. combined complexity for both GAV and GLAV$_{BE}$ mappings, in the case of DL-Lite$_R$, RL, or $\mathcal{EL}_\bot$; (ii) N2EXPTIME-complete w.r.t. combined complexity for GAV mappings, in the case of $\mathcal{SROIQ}$.*

The above theorem implies rows 3 and 4 in Figure 2.

5 Conclusions

The tables in Fig. 1 and Fig. 2 report the results presented in Sec. 3 and 4. These results clarify the complexity of the fundamental mapping analysis tasks studied in this paper.

The analysis presented in this paper can be extended in different directions. First, it would be interesting to establish tight combined complexity bounds for general GLAV mappings, and extend our study to other forms of mappings (beyond GLAV), admitting, for instance, forms of negation in the source queries. Then, it would be interesting to extend our analysis beyond the OWL framework, considering, e.g., DLs interpreted under the Unique Name Assumption, or languages of the Datalog+/- family. Finally, we believe that the problems and techniques studied in this paper may constitute the core of practical tools for the crucial task of constructing, debugging, and maintaining an OBDA specification. So, an important direction for future work is the implementation and practical evaluation of techniques for mapping analysis in OBDA.

Acknowledgments. This research has been partially supported by the EU under FP7 project Optique (n. FP7-318338), and by the RCN under project DOIL (n. 213115).

References

1. S. Abiteboul, R. Hull, and V. Vianu. *Foundations of Databases*. Addison Wesley Publ. Co., 1995.
2. N. Antonioli, F. Castanò, S. Coletta, S. Grossi, D. Lembo, M. Lenzerini, A. Poggi, E. Virardi, and P. Castracane. Ontology-based data management for the italian public debt. In *Proc. of FOIS*, pages 372–385, 2014.
3. M. Arenas, P. Barceló, L. Libkin, and F. Murlak. *Foundations of Data Exchange*. Cambridge University Press, 2014.
4. F. Baader, S. Brandt, and C. Lutz. Pushing the $\mathcal{EL}$ envelope. In *Proc. of IJCAI*, pages 364–369, 2005.
5. 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. *JAR*, 39(3):385–429, 2007.
6. C. Civili, M. Console, G. De Giacomo, D. Lembo, M. Lenzerini, L. Lepore, R. Mancini, A. Poggi, R. Rosati, M. Ruzzi, V. Santarelli, and D. F. Savo. MASTRO STUDIO: Managing ontology-based data access applications. *PVLDB*, 6:1314–1317, 2013.
7. S. Das, S. Sundara, and R. Cyganiak. R2RML: RDB to RDF Mapping Language. *W3C RDB2RDF Working Group*, W3C recommendation, Sept. 2012.
8. F. Di Pinto, D. Lembo, M. Lenzerini, R. Mancini, A. Poggi, R. Rosati, M. Ruzzi, and D. F. Savo. Optimizing query rewriting in ontology-based data access. In *Proc. of EDBT*, pages 561–572. ACM Press, 2013.
9. A. Doan, A. Y. Halevy, and Z. G. Ives. *Principles of Data Integration*. Morgan Kaufmann, 2012.
10. R. Fagin, P. G. Kolaitis, R. J. Miller, and L. Popa. Data exchange: Semantics and query answering. *TCS*, 336(1):89–124, 2005.
11. M. Giese, A. Soylu, G. Vega-Gorgojo, A. Waaler, P. Haase, E. Jiménez-Ruiz, D. Lanti, M. Rezk, G. Xiao, Ö. L. Özçep, and R. Rosati. Optique: Zooming in on big data. *IEEE Computer*, 48(3):60–67, 2015.
12. G. Gottlob, R. Pichler, and V. Savenkov. Normalization and optimization of schema mappings. *VLDBJ*, 20(2):277–302, 2011.
13. P. Haase et al. Optique system: Towards ontology and mapping management in OBDA solutions. In *Proc. of WoDOOM*, pages 21–32, 2013.
14. I. Horrocks, O. Kutz, and U. Sattler. The even more irresistible $\mathcal{SROIQ}$. In *Proc. of KR*, pages 57–67, 2006.
15. E. Kharlamov et al. Optique 1.0: Semantic access to big data: The case of norwegian petroleum directorate's factpages. In *Proc. of ISWC*, pages 65–68, 2013.
16. R. Kontchakov and M. Zakharyaschev. An introduction to description logics and query rewriting. In *RW Tutorial Lectures*, number 8714 in LNCS, pages 195–244. Springer, 2014.
17. D. Lembo, J. Mora, R. Rosati, D. F. Savo, and E. Thorstensen. Towards mapping analysis in ontology-based data access. In *Proc. of RR*, pages 108–123, 2014.
18. 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.
19. M. Rodriguez-Muro, R. Kontchakov, and M. Zakharyaschev. Ontology-based data access: Ontop of databases. In *Proc. of ISWC*, volume 8218 of *LNCS*, pages 558–573. Springer, 2013.