Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence

Data Quality in Ontology-Based Data Access: The Case of Consistency

Marco Console, Maurizio Lenzerini
Dipartimento di Ingegneria Informatica, Automatica e Gestionale "Antonio Ruberti'
Sapienza Università di Roma, Roma, Italy
lastname@dis.uniroma1.it

Abstract

Ontology-based data access (OBDA) is a new paradigm aiming at accessing and managing data by means of an ontology, i.e., a conceptual representation of the domain of interest in the underlying information system. In the last years, this new paradigm has been used for providing users with abstract (independent from technological and system-oriented aspects), effective, and reasoning-intensive mechanisms for querying the data residing at the information system sources. In this paper we argue that OBDA, besides querying data, provides the right principles for devising a formal approach to data quality. In particular, we concentrate on one of the most important dimensions considered both in the literature and in the practice of data quality, namely consistency. We define a general framework for data consistency in OBDA, and present algorithms and complexity analysis for several relevant tasks related to the problem of checking data quality under this dimension, both at the extensional level (content of the data sources), and at the intensional level (schema of the data sources).

1 Introduction

Ontology-based data access (OBDA) is a new paradigm aiming at accessing and managing the data of an information system by means of an ontology (Lenzerini 2011). The ontology provides a conceptual representation of the domain of interest, and suitable mapping assertions relate the concepts and the roles in the ontology to the actual data stored at the sources. Depending on the process defining the OBDA specification, there are two types of OBDA systems, that we call simple and composite. A simple OBDA system models a top-down scenario in which the information system is designed starting from the ontology, and the data sources are defined with the goal of providing correct data structures for storing the instances of concepts and roles in the ontology. A composite OBDA system is built using a bottom-up approach, where the ontology is superimposed on a set of pre-existing and autonomous data sources, so as to improve the access to the data by the users, who can greatly benefit by an abstract representation of the information system.

In the last years, this new paradigm has been adopted for allowing the users to query the information system through the ontology (Rodriguez-Muro and Calvanese 2012a; 2012b; Calvanese et al. 2011; Kontchakov et al. 2011; Calì, Gottlob, and Pieris 2011; Bienvenu et al. 2013). In this paper we argue that OBDA, besides querying data, provides the right principles for devising a formal approach to data quality. Data quality (Batini and Scannapieco 2006; Fan and Geerts 2012) is a sub-discipline of Data Management whose aim is to investigate principles and techniques for ensuring high quality of information. Despite many studies and approaches, data quality is still a hot topic, and is currently gaining even more importance with the advent of the Big Data wave (Dong and Srivastava 2013). Quality of data is characterized by different dimensions, such as completeness, consistency, accuracy, and currency. In this first investigation, we concentrate on consistency, which is one of the most important dimensions considered both in the literature and in the practice of data quality. Consistency is the property of an information system of being free from conflicts and contradictions, such as conflicting information about the same underlying data object, or incoherence of data values across different data sets, or the lack of expected relationships between interdependent attributes, or the lack of conformance of source data and the source schema with the business rules of the organization.

Data quality, and in particular consistency, is crucial in OBDA, both for simple and composite systems. In simple systems, data consistency ensures that the designed data structures are suited for representing the instance level of the information system, without contradicting the domain knowledge. In composite systems, consistency is a fundamental property of data sources, and consistency checking is crucial for governing the information system, and maintaining, improving or re-designing its data structures.

In this paper, we present the first study of data quality issues in OBDA, and provide the following contributions.

- We present a formal framework for data consistency in OBDA (Section 3), addressing both the extensional level (content of the sources), and at the intensional level (schema of the sources) issues. At the intensional level, the framework is based on two notions, namely faithfulness and protection. Intuitively, a source schema is faithful to an OBDA specification if its integrity constraints do not block any data that is consistent with the ontology. On the contrary, a source schema protects an OBDA system from inconsistency if its integrity constraints block every

data that are in conflict with the ontology.

- We show that for unrestricted OBDA specifications, both faithfulness and protection are undecidable, and we introduce the notion of lightweight OBDA specification, with the goal of singling out OBDA specifications that are both reasonably expressive, and enable efficient quality checking tasks. We then present an extended notion of OBDA specification, where the data at the source are modelled as an incomplete database, and illustrate a new technique for checking satisfiability of such specifications.
- Based on this result, we devise algorithms for checking faithfulness (Section 4) and checking protection (Section 5) in lightweight OBDA systems, and characterize their computational complexity. We show that, for this class of OBDA systems, data quality is no more complex than reasoning about schema constraints in a traditional database setting. In other words, checking data quality in lightweight specifications behave exactly as querying, since in both querying and quality checking, the complexity of carrying out the task through the ontology does not increase with respect to traditional databases.
- When the source schema does not protect the OBDA system from inconsistency, satisfiability should be checked at run time. With the goal of minimizing the number of axioms that are needed to consider during such check, we introduce the notion of superfluous axiom, i.e., an axiom of the ontology that can be ignored without endangering the correctness of the satisfiability check. We present an algorithm for superfluousness in Section 6, and show that it has the same nice computational properties as faithfulness and protection.

Using ontologies for characterizing data quality is not a new idea (Wand and Wang 1996). However, we are not aware of any other formal approach aiming at devising principles, techniques, algorithms and formal analysis for data consistency in OBDA. We believe that the work presented here provides the basis for a novel investigation on the issue of data quality, grounded on the idea of using the ontology as a the fundamental tool for measuring the quality of the information system. Possible developments of our work in this direction are outlined in the final section of the paper.

2 Preliminaries

In this section we present some preliminary notions on databases and ontologies that are at the basis of the OBDA approach, and we illustrate what is an OBDA system.

Databases. We consider relational databases, and refer to (Abiteboul, Hull, and Vianu 1995) for a more detailed account of databases. A *schema* $\mathcal{S}$ is a pair $\langle \Sigma_\mathcal{S}, \mathcal{C}_\mathcal{S} \rangle$, where $\Sigma_\mathcal{S}$ is the alphabet of $\mathcal{S}$, and $\mathcal{C}_\mathcal{S}$ is the set of integrity constraints (or simply constraints) of $\mathcal{S}$, which are rules that each database conforming to the schema must obey. A *database* for $\mathcal{S}$, or simply a $\Sigma_\mathcal{S}$-database, is a finite set of ground atoms over the predicates in $\Sigma_\mathcal{S}$ and the constants in an alphabet Γ (constants are subject to the unique name assumption). A $\Sigma_\mathcal{S}$-database D is *legal* for $\mathcal{S}$, written $D \models \mathcal{S}$, if it satisfies all the integrity constraints in $\mathcal{C}_\mathcal{S}$, written $D \models \mathcal{C}_\mathcal{S}$. Also, we say that $D \cup \mathcal{C}_\mathcal{S}$ is satisfiable if there exists a $\Sigma_\mathcal{S}$-database D' such that $D \subseteq D'$ and $D' \models \mathcal{C}_\mathcal{S}$.

In general, constraints in $\mathcal{S}$ are expressed as first-order logic (FOL) sentences, or subclasses thereof. A popular subclass is constituted by *tuple-generating dependencies* (tgds) and *equality-generating dependencies* (egds) (Beeri and Vardi 1981; Calì et al. 2010). A tgd has the form $\forall \vec{x} \phi(\vec{x}) \rightarrow \exists \vec{y} \psi(\vec{x}, \vec{y})$, where both $\phi(\vec{x})$ and $\psi(\vec{x}, \vec{y})$ are conjunctive queries, i.e., conjunctions of atoms with free variables $\vec{x}$, and $\vec{x} \cup \vec{y}$, respectively. An egd has the form $\forall \vec{x} \phi(\vec{x}, \vec{y}) \rightarrow x_1 = x_2$, where x_1, x_2 are among the variables in $\vec{x}$. For technical reasons, we assume that the conjunctive queries used in tgds and egds do not contain constants. Note that, given a $\Sigma_\mathcal{S}$-database D and a set $\mathcal{C}_S$ of tgds and egds, checking whether $D \models \mathcal{C}_\mathcal{S}$ can be done in PSPACE, whereas checking whether $D \cup \mathcal{C}_\mathcal{S}$ is satisfiable, i.e., there exists a $\Sigma_\mathcal{S}$-database D' such that $D' \supseteq D$ and $D' \models \mathcal{C}_\mathcal{S}$, is undecidable (Abiteboul, Hull, and Vianu 1995).

A notable subclass of tgds and egds that is very relevant in practice, and has a decidable satisfiability problem is the class of *weakly-acyclic tgds and egds*. This class captures the most important relational database constraints, including keys, functional dependencies, foreign keys, and a large class of inclusion dependencies. If $\mathcal{C}_\mathcal{S}$ is a set of weakly-acyclic tgds and egds, then checking whether $D \cup \mathcal{C}_\mathcal{S}$ is satisfiable can be done by means of the *chase* algorithm. Informally, the chase is a procedure that, starting from D, repeatedly adds new facts (atoms possibly with variables) according to the tgds in $\mathcal{C}_\mathcal{S}$, and enforces equality of terms (constants and variables) according to the egds in $\mathcal{C}_\mathcal{S}$. The property of the chase is that $D \cup \mathcal{C}_\mathcal{S}$ is satisfiable if and only if the chase does not fail, where the chase is said to fail if the set computed by the procedure, denoted by $chase_{\mathcal{C}_\mathcal{S}}(D)$, contains an explicit contradiction, i.e., an equality of two different constants. For acyclic tgds and egds, $chase_{\mathcal{C}_\mathcal{S}}(D)$ is finite, and can be computed in EXPTIME. We refer to (Fagin et al. 2005; Kolaitis, Panttaja, and Tan 2006) for a formal account of weakly acyclic tdgs, egds, and the chase.

Description Logic Ontologies. An ontology is a conceptualization of a domain of interest expressed in terms of a formal language. Here, we consider logic-based languages, and, more specifically, Description Logics (DLs) (Baader et al. 2010). Generally speaking, a knowledge base expressed in a DL is a pair $\langle \mathcal{T}, \mathcal{A} \rangle$ where the *TBox* $\mathcal{T}$ is the ontology, i.e., a set of axioms specifying universal properties of the concepts and the roles that are relevant in the domain, and the *ABox* $\mathcal{A}$ contains axioms specifying the instances of concepts and roles.

In this paper, in particular in lightweight OBDA, we focus on ontologies expressed in *DL-Lite*$_\mathcal{A}$ (Calvanese et al. 2005; Poggi et al. 2008), a member of the *DL-Lite* family[1] of tractable Description Logics (DLs). For the sake of brevity, we provide only a short account of *DL-Lite*$_\mathcal{A}$ here.

The syntax of concept, role and attribute *expressions* in *DL-Lite*$_\mathcal{A}$ over an alphabet $\Sigma_\mathcal{T}$ is specified by means of the following grammar (where A, P, U are atomic concepts,

[1] Not to be confused with the set of DLs studied in (Artale et al. 2009), which form the *DL-Lite*$_{bool}$ family.

roles, and attributes, respectively, and $T_1, \ldots, T_n$ are unbounded pairwise disjoint predefined value-domains):

$$\begin{array}{ll} B \longrightarrow A \mid \exists Q \mid \delta(U) & E \longrightarrow \rho(U) \\ C \longrightarrow B \mid \neg B & F \longrightarrow T_1 \mid \cdots \mid T_n \\ Q \longrightarrow P \mid P^- & V \longrightarrow U \mid \neg U \\ R \longrightarrow Q \mid \neg Q & \end{array}$$

A *DL-Lite*$_\mathcal{A}$ TBox $\mathcal{T}$ over an alphabet $\Sigma_\mathcal{T}$ is constituted by:

- the set $\mathcal{T}^+$ of "positive" inclusion assertions between concepts, roles and attributes (e.g., $A \sqsubseteq \exists P^-$);
- the set $\mathcal{T}^-$ of "negative" assertions, partitioned into:
  - the set $\mathcal{T}^d$ of disjointness assertions between concepts, roles and attributes (e.g., $A \sqsubseteq \neg\exists P$);
  - the set T^f of functionality assertions, i.e., (funct Q), or (funct U).

In *DL-Lite*$_\mathcal{A}$ TBoxes we further impose that roles and attributes occurring in functionality assertions cannot be specialized, i.e., they cannot occur in the right-hand side of positive inclusions.

Note that checking *DL-Lite*$_\mathcal{A}$-KB for satisfiability, i.e., checking if $Mod(\langle \mathcal{T}, \mathcal{A}\rangle) = \{ \mathcal{I} \mid \mathcal{I}$ is an interpretation for $\Sigma_\mathcal{T}$ such that $\mathcal{I} \models \mathcal{T} \}$ is non-empty , can be done in AC_0 with respect to $\mathcal{A}$ and in PTIME with respect to $\mathcal{T}$.

Ontology-based Data Access. An OBDA system is constituted by an OBDA specification, the intensional level of the system, and a database, representing the data stored in the sources, i.e., the extensional level of the system.

An OBDA specification provides the characteristics of the three basic components of the system, as specified by the following definition.

Definition 1 *An* OBDA specification $\mathcal{B}$ *is a triple* $\langle \mathcal{T}, \mathcal{M}, \mathcal{S}\rangle$*, where*

- $\mathcal{T}$ *is a TBox, called the ontology of* $\mathcal{B}$*, with alphabet* $\Sigma_\mathcal{T}$*;*
- $\mathcal{S} = \langle \Sigma_\mathcal{S}, \mathcal{C}_\mathcal{S}\rangle$ *is a database schema, called the source schema of* $\mathcal{B}$*;*
- $\mathcal{M}$ *is a finite set of mapping assertions between* $\mathcal{S}$ *and* $\mathcal{T}$*, called the* mapping *of* $\mathcal{B}$*, where each mapping assertion is of the form* $\forall \vec{x} \phi(\vec{x}) \rightarrow \exists \vec{y} \psi(\vec{x}, \vec{y})$*, where* $\phi(\vec{x})$ *is a conjunctive query over* $\Sigma_\mathcal{S}$ *with free variables* $\vec{x}$*, and* $\psi(\vec{x}, \vec{y})$ *is a conjunctive query over the alphabet* $\Sigma_\mathcal{T}$ *with free variables* $\vec{x} \cup \vec{y}$*.*

In what follows, we will refer to a specific form of mappings, called GAV, extensively studied in the database literature (Halevy 2001; Lenzerini 2002). A *GAV (Global-as-view) mapping assertion* is a mapping assertion in which no existential variable appears in ψ, and can therefore be written as $\forall \vec{x} \phi(\vec{x}) \rightarrow \psi(\vec{z})$, where all the variables in $\vec{z}$ appear also in $\vec{x}$.

As we said before, when we pair an OBDA specification $\mathcal{B} = \langle \mathcal{T}, \mathcal{M}, \mathcal{S}\rangle$ with a $\Sigma_\mathcal{S}$-database D, we obtain an OBDA system. We define the semantics of an OBDA system by specifying which are the models of $\mathcal{B}$ relative to D, denoted by $Mod_D(\mathcal{B})$. Intuitively, if D is not legal with respect to $\mathcal{S}$, such models form the empty set. Otherwise, such models are the interpretations $\mathcal{I}$ for $\Sigma_\mathcal{T}$ that satisfy $\mathcal{T}$, and such that the pair $(D, \mathcal{I})$ satisfy all mapping assertions in $\mathcal{M}$, written $(D, \mathcal{I}) \models \mathcal{M}$.

Definition 2 *Let* $\mathcal{B} = \langle \mathcal{T}, \mathcal{M}, \mathcal{S}\rangle$ *be an OBDA specification, and let* D *be a* $\Sigma_\mathcal{S}$*-database. Then* $Mod_D(\mathcal{B}) = \{ \mathcal{I} \mid \mathcal{I} \models \mathcal{T},\ (D, \mathcal{I}) \models \mathcal{M},$ and $D \models \mathcal{C}_\mathcal{S} \}$.

Checking whether an OBDA system constituted by $\mathcal{B}$ and D is satisfiable amounts to checking whether $Mod_D(\mathcal{B}) \neq \emptyset$. If the system is managed by suitable software components including a database management system ensuring that $D \models \mathcal{C}_\mathcal{S}$, then the satisfiability checking reduces to verifying whether there exists an interpretation $\mathcal{I}$ for $\Sigma_\mathcal{T}$ that satisfies $\mathcal{T}$, and such that the pair $(D, \mathcal{I})$ satisfies all mapping assertions in $\mathcal{M}$.

3 Foundations

In this section, we introduce the notions we use for characterizing data quality in OBDA, in particular with respect to the consistency dimension.

We start by considering the quality of the content of the data source, i.e., we discuss data consistency at the extensional level. What we formalize here is the simple idea that the level of quality of a database D at the extensional level is related to the consistency of D with respect to $\mathcal{T}$ and $\mathcal{M}$.

Definition 3 *Let* $\mathcal{B} = \langle \mathcal{T}, \mathcal{M}, \mathcal{S}\rangle$ *be an OBDA specification, and let* D *be a* $\Sigma_\mathcal{S}$*-database. Then* D *is said to be* consistent with respect to $\mathcal{T}$ and $\mathcal{M}$ *in* $\mathcal{B}$*, or simply* consistent in $\mathcal{B}$*, if* $Mod_D(\langle \mathcal{T}, \mathcal{M}, \langle \Sigma_\mathcal{S}, \emptyset\rangle\rangle) \neq \emptyset$*.*

Note that we are not reducing consistency of D to checking satisfiability of the OBDA system constituted by $\mathcal{B}$ and D. Indeed, satisfiability of $\mathcal{B}$ and D requires that D satisfies $\mathcal{C}_\mathcal{S}$, while we interpret data consistency at the extensional level as checking consistency of the data source with respect to the TBox and the mapping of the OBDA specification, independently from the constraints in $\mathcal{C}_\mathcal{S}$. Since this problem has already been studied (see, for example, (Poggi et al. 2008)), we do not address it any further here. We only note that the check can be done in AC_0 with respect to D, in PTIME with respect to $\mathcal{T}$, and in coNP with respect to $\mathcal{M}$.

We now turn our attention to the quality of the source schema with respect to the consistency dimension. We introduce two notions for characterizing such quality at the intensional level, namely faithfulness and protection.

Definition 4 *Let* $\mathcal{B} = \langle \mathcal{T}, \mathcal{M}, \mathcal{S}\rangle$ *be an OBDA specification.* $\mathcal{S}$ *is said to be faithful to* $\mathcal{T}$ *and* $\mathcal{M}$ *in* $\mathcal{B}$ *with respect to consistency, or simply* faithful to $\mathcal{B}$*, if for all* $\Sigma_\mathcal{S}$*-database* D *such that* $Mod_D(\langle \mathcal{T}, \mathcal{M}, \langle \Sigma_\mathcal{S}, \emptyset\rangle\rangle) \neq \emptyset$*, we have that* $D \models \mathcal{C}_\mathcal{S}$*.*

Intuitively, the schema $\mathcal{S}$ is faithful to $\mathcal{B}$ if it does not constrain the source in such a way to filter out data that would not cause the OBDA system to fall into inconsistency, i.e., if every $\Sigma_\mathcal{S}$-database D that does not cause any inconsistencies to $\mathcal{T}$ and $\mathcal{M}$, does not violate any constraint in $\mathcal{S}$.

We now turn our attention to protection, and provide its formal definition.

Definition 5 *Let* $\mathcal{B} = \langle \mathcal{T}, \mathcal{M}, \mathcal{S}\rangle$ *be an OBDA specification.* $\mathcal{S}$ *is said to protect* $\mathcal{T}$ *and* $\mathcal{M}$ *from inconsistency, or simply* protect $\mathcal{B}$*, if for all* $\Sigma_\mathcal{S}$*-database* D *such that* $Mod_D(\langle \mathcal{T}, \mathcal{M}, \langle \Sigma_\mathcal{S}, \emptyset\rangle\rangle) = \emptyset$*, we have that* $D \not\models \mathcal{C}_\mathcal{S}$*.*

Intuitively, the schema $\mathcal{S}$ protects $\mathcal{B}$ if it does constrain the data in such a way to prevent the system from falling into inconsistencies, i.e., if every $\Sigma_{\mathcal{S}}$-database that, when paired to $\mathcal{T}$ through $\mathcal{M}$, causes some inconsistency, violates at least one of the constraints in $\mathcal{S}$.

The three aspects of data quality that we have discussed can be formally evaluated by resorting to three corresponding decision problems, that we now formalize.

Definition 6 *Given an OBDA specification $\mathcal{B} = \langle \mathcal{T}, \mathcal{M}, \mathcal{S} \rangle$,*
- *Data consistency is the following decision problem: given a $\Sigma_{\mathcal{S}}$-database D, check whether D is consistent in $\mathcal{B}$.*
- *Faithfulness is the following decision problem: check whether $\mathcal{S}$ is faithful to $\mathcal{B}$.*
- *Protection is the following decision problem: check whether $\mathcal{S}$ protects $\mathcal{B}$.*

As we said, we do not deal in detail with data consistency at the extensional level in this paper. As for the other problems, we observe that, in characterizing their computational complexity, the expressive power of the language used to specify $\mathcal{C}_{\mathcal{S}}$ plays an important role. In particular, if $\mathcal{T}$ and $\mathcal{M}$ are such that for every $\Sigma_{\mathcal{S}}$-database D, $Mod_D(\langle \mathcal{T}, \mathcal{M}, \langle \Sigma_{\mathcal{S}}, \emptyset \rangle \rangle) \neq \emptyset$, then $\mathcal{S}$ is faithful to $\mathcal{B}$ if and only if $\mathcal{C}_{\mathcal{S}}$ is valid. Analogously, if $\mathcal{T}$ and $\mathcal{M}$ are such that for every $\Sigma_{\mathcal{S}}$-database D, $Mod_D(\langle \mathcal{T}, \mathcal{M}, \langle \Sigma_{\mathcal{S}}, \emptyset \rangle \rangle) = \emptyset$, then $\mathcal{S}$ protects $\mathcal{B}$ if and only if $\mathcal{S}$ is unsatisfiable.

Since in unrestricted OBDA specifications, $\mathcal{C}_{\mathcal{S}}$ can be any set of FOL sentences, from the above observations we can easily derive the following theorem.

Theorem 1 *For unrestricted OBDA specifications, both protection and faithfulness are undecidable.*

This suggests considering suitable limitations in the various components of the OBDA specification. To capture this, we introduce the notion of lightweight OBDA specifications.

Definition 7 *$\mathcal{B} = \langle \mathcal{T}, \mathcal{M}, \mathcal{S} \rangle$ is a* lightweight OBDA specification *if (i) $\mathcal{T}$ is a DL-Lite$_{\mathcal{A}}$ TBox, (ii) $\mathcal{M}$ is a GAV mapping, and (iii) the integrity constraints $\mathcal{C}_{\mathcal{S}}$ of $\mathcal{S}$ form a set of weakly acyclic tgds and egds.*

Lightweight OBDA systems enjoy several desirable properties that we cannot discuss here for lack of space. We only briefly discuss how we can answer queries in a satisfiable lightweight OBDA system, i.e., compute the tuples satisfying the query in all the models of the system. Let $\mathcal{B} = \langle \mathcal{T}, \mathcal{M}, \mathcal{S} \rangle$ be a lightweight OBDA system, and D a $\Sigma_{\mathcal{S}}$-database. The results in (Poggi et al. 2008) show that, in order to answer a union of conjunctive query q (expressed in the alphabet $\Sigma_{\mathcal{T}}$) posed to $\mathcal{B} = \langle \mathcal{T}, \mathcal{M}, \mathcal{S} \rangle$ and D, we can compute the *perfect rewriting* $\mathcal{R}_{\mathcal{T},\mathcal{M}}$ of q with respect to $\mathcal{T}$ and $\mathcal{M}$, which is a union of conjunctive query over the alphabet $\Sigma_{\mathcal{S}}$, and then evaluate $\mathcal{R}_{\mathcal{T},\mathcal{M}}$ over D. We will make use of this result in the following.

In the rest of the paper, we address faithfulness and protection in lightweight OBDA specifications. For both tasks, we will face the problem of checking whether an incomplete database is consistent with respect to a lightweight OBDA specification. Thus, we now precisely define this problem, and illustrate how to solve it.

Definition 8 *Let $\mathcal{S} = \langle \Sigma_{\mathcal{S}}, \mathcal{C}_{\mathcal{S}} \rangle$ be a schema, and let V be an alphabet of variables. An incomplete $\Sigma_{\mathcal{S}}$-database is a $\Sigma_{\mathcal{S}}$-database where variables from V can appear in the arguments of relations.*

Intuitively, an incomplete database F represents all the databases that are a subset of the set of facts obtained by instantiating the variables of F with constants. Formally, we say that a (complete) database D is an instance of F, written $D \preceq F$, if there is a homomorphism from F to D. The semantics of an OBDA specification with respect to an incomplete database is then defined as follows.

Definition 9 *Let $\mathcal{B} = \langle \mathcal{T}, \mathcal{M}, \mathcal{S} \rangle$ be an OBDA specification, and let F be an incomplete $\Sigma_{\mathcal{S}}$-database. Then $Mod_F(\mathcal{B}) = \{ \mathcal{I} \mid$ there exists a $\Sigma_{\mathcal{S}}$-database D such that $D \preceq F$, and $Mod_D(\mathcal{B}) \neq \emptyset \}$.*

Observe that, in general, for a lightweight OBDA specification $\mathcal{B} = \langle \mathcal{T}, \mathcal{M}, \mathcal{S} \rangle$, $\mathcal{C}_{\mathcal{S}} \cup \mathcal{M} \cup \mathcal{T}$ cannot be expressed as a set of acyclic tgds and egds. However, $\mathcal{C}_{\mathcal{S}} \cup \mathcal{M} \cup \mathcal{T}^f$ can indeed be expressed as s a set of weakly acyclic tdgs and egds, and therefore $chase_{\mathcal{C}_{\mathcal{S}} \cup \mathcal{M} \cup \mathcal{T}^f}(F)$ is a finite structure that can be computed in a finite amount of time. Also, it is immediate to verify that from $chase_{\mathcal{C}_{\mathcal{S}} \cup \mathcal{M} \cup \mathcal{T}^f}(F)$ one can obtain an ABox over the alphabet of $\mathcal{T}$, simply by keeping only the atoms over the alphabet $\Sigma_{\mathcal{T}}$, and substituting every variable in such atoms with a fresh constant. We denote such an ABox by $\mathcal{A}_{chase_{\mathcal{C}_{\mathcal{S}} \cup \mathcal{M} \cup \mathcal{T}^f}(F)}$.

Theorem 2 *Let $\mathcal{B} = \langle \mathcal{T}, \mathcal{M}, \mathcal{S} \rangle$ be a lightweight OBDA specification, and let F be an incomplete $\Sigma_{\mathcal{S}}$-database. Then $Mod_F(\mathcal{B}) = \emptyset$ if and only if $chase_{\mathcal{C}_{\mathcal{S}} \cup \mathcal{M} \cup \mathcal{T}^f}(F)$ fails, or the DL-Lite$_{\mathcal{A}}$-KB $\langle \mathcal{T}^+ \cup \mathcal{T}^d, \mathcal{A}_{chase_{\mathcal{C}_{\mathcal{S}} \cup \mathcal{M} \cup \mathcal{T}^f}(F)} \rangle$ is unsatisfiable.*

The above theorem directly suggests the following algorithm for checking $\mathcal{B} \cup F$ for satisfiability, i.e., for checking whether $Mod_F(\mathcal{B}) \neq \emptyset$:

1. if $chase_{\mathcal{C}_{\mathcal{S}} \cup \mathcal{M} \cup \mathcal{T}^f}(F)$ fails, then return false;
2. otherwise, from $chase_{\mathcal{C}_{\mathcal{S}} \cup \mathcal{M} \cup \mathcal{T}^f}(F)$ build the ABox $\mathcal{A}_{chase_{\mathcal{C}_{\mathcal{S}} \cup \mathcal{M} \cup \mathcal{T}^f}(F)}$, and return the result of checking whether $\langle \mathcal{T}^+ \cup \mathcal{T}^d, \mathcal{A} \rangle$ is satisfiable, by relying on the algorithm for satisfiability checking in *DL-Lite*-KBs.

From the correctness of the above algorithm, we can then derive the following result.

Theorem 3 *If $\mathcal{B} = \langle \mathcal{T}, \mathcal{M}, \mathcal{S} \rangle$ is a lightweight OBDA specification, and F an incomplete $\Sigma_{\mathcal{S}}$-database, then checking whether $Mod_F(\mathcal{B}) = \emptyset$ can be done in EXPTIME with respect to $\mathcal{S}$ and $\mathcal{M}$, and in PTIME with respect to F and $\mathcal{T}$.*

4 Faithfulness in lightweight OBDA

In this section we study the problem of checking whether a source schema is faithful to a lightweight OBDA specification. The technique we present is based on several notions, that we now introduce.

The first notion concerns the idea of expressing the existence of a violation of a constraint in $\mathcal{C}_{\mathcal{S}}$ by means of a suitable conjunctive query in the alphabet $\Sigma_{\mathcal{S}}$. To this end, we define the function $\mathcal{V}$ as follows.

Definition 10 *If $\langle \Sigma_S, \mathcal{C}_S \rangle$ is a source schema, and $\beta \in \mathcal{C}_S$, then $\mathcal{V}(\beta)$ is defined as follows:*
- *if β is of the form $\forall \vec{x} \phi(\vec{x}) \rightarrow x_1 = x_2$, then $\mathcal{V}(\beta)$ is the query $\exists \vec{x} \phi(\vec{x}) \wedge (x_1 \neq x_2)$*
- *if β is of the form $\forall \vec{x} \phi(\vec{x}) \rightarrow \exists \vec{y} \psi(\vec{x}, \vec{y})$, then $\mathcal{V}(\beta)$ is the query $\exists \vec{x} \phi(\vec{x})$.*

We call trivial an egd of the form $\forall \vec{x} \phi(\vec{x}) \rightarrow x = x$. Analogously, we call trivial a tgd $\forall \vec{x} \phi(\vec{x}) \rightarrow \exists \vec{y} \psi(\vec{x}, \vec{y})$ whenever the query $\phi(\vec{x})$ is contained in the query $\exists \vec{y} \psi(\vec{x}, \vec{y})$. The next lemma shows that the query $\mathcal{V}(\beta)$ really captures the violations to β.

Lemma 1 *Let D be a Σ_S-database and let $\beta \in \mathcal{C}_S$ be not trivial. If $D \not\models \beta$, the $D \models \mathcal{V}(\beta)$.*

We now define a function, called $\mathcal{F}$, that, given a conjunctive query in the alphabet Σ_S, possibly with one inequality, returns an incomplete Σ_S-database satisfying the query.

Definition 11 *If γ is conjunctive query with at most one inequality in the alphabet Σ_S, then $\mathcal{F}(\gamma)$ is defined as follows.*
- *if γ is of the form $\exists \vec{x} \phi(\vec{x}, \vec{y}) \wedge (x_1 \neq x_2)$, then $\mathcal{F}(\gamma)$ is the incomplete Σ_S-database that is obtained from γ by choosing two fresh constants c_1 and c_2 not appearing in γ, and then substituting x_1 with c_1 and c_2 for x_2 in γ.*
- *if γ is a conjunctive query without inequalities, then $\mathcal{F}(\gamma)$ is the incomplete Σ_S-database coinciding with γ.*

Lemma 2 *Let $\mathcal{B} = \langle \mathcal{T}, \mathcal{M}, \langle \Sigma_S, \mathcal{C}_S \rangle \rangle$ be a lightweight OBDA specification, let $\beta \in \mathcal{C}_S$, and let D be a Σ_S-database D. Then if $D \not\models \beta$ then D is an instance of $\mathcal{F}(\mathcal{V}(\beta))$.*

Theorem 4 *Let $\mathcal{B} = \langle \mathcal{T}, \mathcal{M}, \mathcal{S} \rangle$ be a lightweight OBDA specification, and let $\mathcal{S} = \langle \Sigma_S, \mathcal{C}_S \rangle$. Then $\mathcal{S}$ is faithful to $\mathcal{B}$ if and only if for each $\beta \in \mathcal{C}_S$ which is not trivial, we have that $Mod_{\mathcal{F}(\mathcal{V}(\beta))}(\langle \mathcal{T}, \mathcal{M}, \langle \Sigma_S, \emptyset \rangle \rangle) = \emptyset$.*

Proof. (Sketch) *If-part.* Suppose that for each $\beta \in \mathcal{C}_S$ which is not trivial, we have that $Mod_{\mathcal{F}(V(\beta))}(\langle \mathcal{T}, \mathcal{M}, \langle \Sigma_S, \emptyset \rangle \rangle) = \emptyset$, and $\mathcal{S}$ is not faithful to $\mathcal{B}$ with respect to consistency. Let D be a database such that $Mod_D(\langle \mathcal{T}, \mathcal{M}, \langle \Sigma_S, \emptyset \rangle \rangle) \neq \emptyset$, and $D \not\models \mathcal{C}_S$. Since $D \not\models \mathcal{C}_S$, it follows that there is a $\beta \in \mathcal{C}_S$ such that $D \not\models \beta$. But $D \not\models \beta$ implies that β is not trivial, and $D \models \mathcal{V}(\beta)$. By exploiting Lemma 2, we can show that this implies that D is an instance of $\mathcal{F}(\mathcal{V}(\beta))$, and this in turn implies that $Mod_{\mathcal{F}(\mathcal{V}(\beta))}(\langle \mathcal{T}, \mathcal{M}, \langle \Sigma_S, \emptyset \rangle \rangle) \neq \emptyset$, which is a contradiction.

Only-if-part. Suppose that $\mathcal{S}$ is faithful to $\mathcal{B}$ with respect to consistency, and there is $\beta \in \mathcal{C}_S$ which is not trivial such that $Mod_{\mathcal{F}(\mathcal{V}(\beta))}(\langle \mathcal{T}, \mathcal{M}, \langle \Sigma_S, \emptyset \rangle \rangle) \neq \emptyset$. By construction we have that, starting from $\mathcal{F}(\mathcal{V}(\beta))$ we can easily build a Σ_S-database D that is an instance of $\mathcal{F}(\mathcal{V}(\beta))$ such that $Mod_D(\langle \mathcal{T}, \mathcal{M}, \langle \Sigma_S, \emptyset \rangle \rangle) \neq \emptyset$, and $D \not\models \mathcal{C}_S$, proving that $\mathcal{S}$ is not faithful to $\mathcal{B}$ with respect to consistency, and therefore leading to a contradiction. □

The theorem directly provides an algorithm for checking faithfulness. The algorithm considers each β in $\mathcal{C}_S$ in turn, and checks whether it is not trivial, and is such that $Mod_{\mathcal{F}(\mathcal{V}(\beta))}(\langle \mathcal{T}, \mathcal{M}, \langle \Sigma_S, \emptyset \rangle \rangle) = \emptyset$. To verify the condition $Mod_{\mathcal{F}(\mathcal{V}(\beta))}(\langle \mathcal{T}, \mathcal{M}, \langle \Sigma_S, \emptyset \rangle \rangle) = \emptyset$, the algorithm resorts to the technique described in Section 3 (note that in this case $\mathcal{C}_S$ is not considered). From the characteristics of this algorithm, we can derive the following result.

Theorem 5 *Faithfulness in lightweight OBDA specifications is in PTIME with respect to $\mathcal{S}$ and $\mathcal{T}$, and in EXPTIME with respect to $\mathcal{M}$.*

5 Protection in lightweight OBDA

In this section we address the problem of checking protection in lightweight OBDA specifications. The technique we present is based on the idea of carrying out the protection check on the basis of an axiom-by-axiom method. To formalize this idea, we now introduce the notion of a source schema protecting an OBDA specification from inconsistencies arising *due to a single axiom* in the ontology.

Definition 12 *Let $\mathcal{B} = \langle \mathcal{T}, \mathcal{M}, \mathcal{S} \rangle$ be an OBDA specification, and let $\alpha \in \mathcal{T}^-$. Then $\mathcal{S}$ protects $\mathcal{B}$ from α-inconsistencies if for all Σ_S-database D, if $Mod_D(\langle \mathcal{T}^+ \cup \{\alpha\}, \mathcal{M}, \langle \Sigma_S, \emptyset \rangle \rangle) = \emptyset$, then $D \not\models \mathcal{C}_S$.*

With this notion in place, we can now show that the problem of checking whether a schema $\mathcal{S}$ protects a lightweight OBDA specification $\mathcal{B}$ can be reduced to the problem of checking whether $\mathcal{S}$ α-protects $\mathcal{B}$, for each $\alpha \in \mathcal{T}^-$.

Theorem 6 *Let $\mathcal{B} = \langle \mathcal{T}, \mathcal{M}, \mathcal{S} \rangle$ be a lightweight OBDA specification. Then $\mathcal{S}$ protects $\mathcal{B}$ if and only if for each $\alpha \in \mathcal{T}^-$, $\mathcal{S}$ protects $\mathcal{B}$ from α-inconsistencies.*

We are now left with the problem of checking whether $\mathcal{S}$ protects $\mathcal{B}$ from α-inconsistencies, for $\alpha \in \mathcal{T}^-$. The key observation for coming up with a technique solving this problem is that the existence of a violation of $\alpha \in \mathcal{T}^-$ can be encoded as a conjunctive query over the alphabet of $\mathcal{T}$. In order to define the conjunctive query associated to an axiom in $\mathcal{T}^-$, we extend the function $\mathcal{V}$ described in the previous section, as follows.

Definition 13 *If $\mathcal{T}$ is a DL-Lite$_\mathcal{A}$ TBox, and $\alpha \in \mathcal{T}^-$, then $\mathcal{V}(\alpha)$ is defined as follows:*
- *if $\alpha = (\mathsf{funct}\ R)$ then*
  then $\mathcal{V}(\alpha) = R(x, y) \wedge R(x, z) \wedge y \neq z$
- *if $\alpha = (\mathsf{funct}\ R^-)$ then*
  then $\mathcal{V}(\alpha) = R(y, x) \wedge R(z, x) \wedge y \neq z$
- *if $\alpha = A \sqsubseteq \neg B$ then then $\mathcal{V}(\alpha) = A(x) \wedge B(x)$*

Given an interpretation $\mathcal{I}$ for $\mathcal{T}$, and $\alpha \in \mathcal{T}^-$, it is easy to prove that $\mathcal{I} \models \neg\alpha$ if and only if $I \models \mathcal{V}(\alpha)$. By relying on this notion, we can state the main result of this section: in a lightweight OBDA specification, checking weather $\mathcal{S}$ protects $\mathcal{B}$ from α-inconsistencies can be reformulated as a satisfiability problem. This is shown by the next theorem.

Theorem 7 *Let $\mathcal{B} = \langle \mathcal{T}, \mathcal{M}, \mathcal{S} \rangle$ a lightweight OBDA specification, and let $\alpha \in \mathcal{T}^-$. Then $\mathcal{S}$ protects $\mathcal{B}$ from α-inconsistencies if and only if for each $q \in \mathcal{R}_{\mathcal{T},\mathcal{M}}(\mathcal{V}(\alpha)))$, $Mod_{\mathcal{F}(q)}(\langle \emptyset, \emptyset, \mathcal{S} \rangle) = \emptyset$.*

Proof. (Sketch) *If-part* Let $q \in \mathcal{R}_{\mathcal{T},\mathcal{M}}(\mathcal{V}(\alpha)))$ be such that $Mod_{\mathcal{F}(q)}(\langle \emptyset, \emptyset, \mathcal{S} \rangle) \neq \emptyset$. Further, let D be one of the instances of $\mathcal{F}(q)$ such that $D \models \mathcal{S}$. For such definite database we have that $D \models \mathcal{R}_{\mathcal{T},\mathcal{M}}(\mathcal{V}(\alpha))$, being the latter a union of conjunctive queries. Since $\mathcal{T}$ is a *DL-Lite$_\mathcal{A}$*

TBox, $Mod_D(\langle \mathcal{T}^+ \cup \{\alpha\}, \mathcal{M}, \langle \Sigma_\mathcal{S}, \emptyset \rangle \rangle) = \emptyset$, and D is legal with respect to $\mathcal{C}_\mathcal{S}$, we conclude that $\mathcal{S}$ does not protect $\mathcal{B}$ from α-inconsistencies.

Only-if-part Assume $\mathcal{S}$ does not protect $\mathcal{B}$ from α-inconsistencies. Then for at least one database D we have that $Mod_D(\langle \mathcal{T}^+ \cup \{\alpha\}, \mathcal{M}, \langle \Sigma_\mathcal{S}, \emptyset \rangle \rangle) = \emptyset$ but $D \models \mathcal{C}_\mathcal{S}$. Since $\mathcal{T}$ is a *DL-Lite* TBox, the violation of α can be represented by $\mathcal{R}_{\mathcal{T},\mathcal{M}}(\mathcal{V}(\alpha))$ over $\Sigma_\mathcal{S}$. Let $q \in \mathcal{R}_{\mathcal{T},\mathcal{M}}(\mathcal{V}(\alpha))$ be one of the conjunctive queries evaluating true over D. This means that $D \models \mathcal{C}_\mathcal{S} \cup \{q\}$, and therefore, by the definition of $\mathcal{F}$, $D \models \mathcal{F}(q) \cup \mathcal{C}_\mathcal{S}$ too, thus proving that $Mod_{\mathcal{F}(q)}(\langle \emptyset, \emptyset, \mathcal{S} \rangle) \neq \emptyset$. □

The above theorem directly provides an algorithm for checking whether $\mathcal{S}$ protects $\mathcal{B}$. The algorithm considers each $\alpha \in \mathcal{T}^-$ in turn, and check whether $\mathcal{S}$ protects $\mathcal{B}$ from α-inconsistencies by applying the method suggested by theorem 7. The result will be false if and only there is a specific $\alpha \in \mathcal{T}^-$ such that $Mod_{\mathcal{F}(q)}(\langle \emptyset, \emptyset, \mathcal{S} \rangle) \neq \emptyset$ for some $q \in \mathcal{R}_{\mathcal{T},\mathcal{M}}(\mathcal{V}(\alpha)))$. Obviously, to check whether $Mod_{\mathcal{F}(q)}(\langle \emptyset, \emptyset, \mathcal{S} \rangle) = \emptyset$, we rely on the technique presented in Section 3.

By analysing the computational complexity of this algorithm, it is immediate to derive the following result.

Theorem 8 *Protection in lightweight OBDA specifications is in PTIME with respect to $\mathcal{T}$ and $\mathcal{M}$, and in EXPTIME with respect to $\mathcal{S}$.*

6 Superfluousness

In an OBDA specification $\mathcal{B} = \langle \mathcal{T}, \mathcal{M}, \mathcal{S} \rangle$ where the source schema $\mathcal{S}$ does not protect $\mathcal{B}$ from inconsistency, satisfiability should be checked at run time: given the $\Sigma_\mathcal{S}$-database D at hand, we should check whether $Mod_D(\langle \mathcal{T}, \mathcal{M}, \mathcal{S} \rangle) \neq \emptyset$. In general, this is costly. Since in lightweight OBDA specifications checking whether $Mod_D(\langle \mathcal{T}, \mathcal{M}, \mathcal{S} \rangle) \neq \emptyset$ means checking whether for each $\alpha \in \mathcal{T}^-$, $Mod_D(\langle \mathcal{T}^+ \cup \{\alpha\}, \mathcal{M}, \mathcal{S} \rangle) \neq \emptyset$, the cost of the satisfiability check grows with the number of axioms in $\mathcal{T}^-$. Thus, one might wonder if there is a way to minimize the number of axioms that are needed to consider during such check. To come up with a precise method toward this goal, we introduce the notion of superfluousness.

Definition 14 *Let $\mathcal{B} = \langle \mathcal{T}, \mathcal{M}, \mathcal{S} \rangle$ be an OBDA specification, and let $\alpha \in \mathcal{T}^-$. Then α is said to be superfluous in $\mathcal{B}$ with respect to satisfiability, or simply* superfluous in $\mathcal{B}$*, if for all $\Sigma_\mathcal{S}$-database D such that $D \models \mathcal{S}$, $Mod_D(\langle \mathcal{T}, \mathcal{M}, \emptyset \rangle) = \emptyset$ implies $Mod_D(\langle \mathcal{T} \setminus \{\alpha\}, \mathcal{M}, \emptyset \rangle) = \emptyset$.*

Note that the above definition really captures the notion that we are looking for: in checking $\mathcal{B}$ for satisfiability with respect to a database D, we can ignore an axiom in $\mathcal{T}^-$ exactly when such axiom is superfluous in $\mathcal{B}$.

The next theorem states that protection of $\mathcal{B}$ from α-inconsistencies is sufficient for superfluousness of α in $\mathcal{B}$.

Theorem 9 *Let $\mathcal{B} = \langle \mathcal{T}, \mathcal{M}, \mathcal{S} \rangle$ be an OBDA specification, and let $\alpha \in \mathcal{T}^-$. If $\mathcal{S}$ protects $\mathcal{B}$ from α-inconsistencies, then α is superfluous in $\mathcal{B}$.*

On the other hand, there are cases where $\mathcal{S}$ does not protect $\mathcal{B}$ from α-inconsistencies, but α is still superfluous in $\mathcal{B}$. The next theorem provides a complete characterization of superfluousness in lightweight OBDA specifications.

Theorem 10 *Let $\mathcal{B} = \langle \mathcal{T}, \mathcal{M}, \mathcal{S} \rangle$ be a lightweight OBDA specification. Then $\alpha \in \mathcal{T}^-$ is superfluous in $\mathcal{B}$ if and only if for each $q \in \mathcal{R}_{\mathcal{T},\mathcal{M}}(\mathcal{V}(\alpha))$, $Mod_{\mathcal{F}(q)}(\langle \mathcal{T} \setminus \{\alpha\}, \mathcal{M}, \mathcal{S} \rangle = \emptyset$.*

Proof. (Sketch) *If-part.* Suppose that for each $q \in \mathcal{R}_{\mathcal{T},\mathcal{M}}(\mathcal{V}(\alpha))$, $Mod_{\mathcal{F}(q)}(\langle \mathcal{T} \setminus \{\alpha\}, \mathcal{M}, \mathcal{S} \rangle) = \emptyset$. We consider a (definite) $\Sigma_\mathcal{S}$-database D such that $Mod_D(\langle \mathcal{T}, \mathcal{M}, \mathcal{S} \rangle) = \emptyset$, and take into account two cases. In the first case, there is a $q \in \mathcal{R}_{\mathcal{T},\mathcal{M}}(\mathcal{V}(\alpha))$ and a homomorphism from $\mathcal{F}(q)$ to D; since $Mod_{\mathcal{F}(q)}(\langle \mathcal{T} \setminus \{\alpha\}, \mathcal{M}, \mathcal{S} \rangle) = \emptyset$, it holds that $Mod_D(\langle \mathcal{T} \setminus \{\alpha\}, \mathcal{M}, \mathcal{S} \rangle) = \emptyset$. In the second case, for no $q \in \mathcal{R}_{\mathcal{T},\mathcal{M}}(\mathcal{V}(\alpha))$, we have a homomorphism from $\mathcal{F}(q)$ to D. Since $Mod_D(\langle \mathcal{T}, \mathcal{M}, \mathcal{S} \rangle) = \emptyset$, and $\mathcal{B}$ is a lightweight OBDA specification, it holds that there is a $\alpha' \in T^-$ different from α such that $Mod_D(\langle \mathcal{T}^+ \cup \{\alpha'\}, \mathcal{M}, \mathcal{S} \rangle) = \emptyset$, which implies that $Mod_D(\langle \mathcal{T} \setminus \{\alpha\}, \mathcal{M}, \mathcal{S} \rangle) = \emptyset$. So, we have proved that for all $\Sigma_\mathcal{S}$-database D, $Mod_D(\langle \mathcal{T}, \mathcal{M}, \mathcal{S} \rangle) = \emptyset$ implies $Mod_D(\langle \mathcal{T} \setminus \{\alpha\}, \mathcal{M}, \mathcal{S} \rangle) = \emptyset$, i.e., α is superfluous in $\mathcal{B}$.

Only-if-part. Suppose that there is a $q \in \mathcal{R}_{\mathcal{T},\mathcal{M}}(\mathcal{V}(\alpha))$ such that $Mod_{\mathcal{F}(q)}(\langle \mathcal{T} \setminus \{\alpha\}, \mathcal{M}, \mathcal{S} \rangle \neq \emptyset$. From $\mathcal{F}(q)$ we can easily obtain a $\Sigma_\mathcal{S}$-database D such that $Mod_D(\langle \mathcal{T} \setminus \{\alpha\}, \mathcal{M}, \mathcal{S} \rangle) = \emptyset$, but $Mod_D(\langle \{\alpha\}, \mathcal{M}, \mathcal{S} \rangle) = \emptyset$, implying that there is a $\Sigma_\mathcal{S}$-database D such that $Mod_D(\langle \mathcal{T}, \mathcal{M}, \mathcal{S} \rangle) = \emptyset$, and $Mod_D(\langle \mathcal{T} \setminus \{\alpha\}, \mathcal{M}, \mathcal{S} \rangle) = \emptyset$, i.e., α is not superfluous in $\mathcal{B}$. □

The above theorem directly suggests an algorithm for checking $\alpha \in \mathcal{T}^-$ for superfluousness in lightweight OBDA specifications. Once again, the algorithm exploits the technique presented before for checking satisfiability with respect to an incomplete database (in this case, the incomplete database $\mathcal{F}(q)$). From this observation, it is easy to derive the complexity of the superfluousness algorithm.

Theorem 11 *Superfluousness in lightweight OBDA specifications is in PTIME with respect to $\mathcal{T}$, and in EXPTIME with respect to $\mathcal{M}$ and $\mathcal{S}$.*

7 Conclusions

We have presented the first investigation on the issue of data quality in OBDA systems. We have proposed a formal framework, and illustrated several results for the case of data consistency in lightweight OBDA specifications. Our approach shows that basing data quality on the conceptual model of the domain represented by the ontology sheds new light to the long-standing problem of ensuring data quality in information systems.

We will continue our work along different directions. In particular, while in this paper we have studied the class of weakly acyclic tdgs and egds, we plan to consider other source constraints that are relevant in practice, such as cyclic foreign keys. Also, we aim at broadening our investigation so as to address other data quality dimensions beyond consistency, in particular completeness and accuracy.

Acknowledgements

Work partially supported by the EU under FP7, project Optique (Scalable End-user Access to Big Data), grant n. FP7-318338.

References

Abiteboul, S.; Hull, R.; and Vianu, V. 1995. *Foundations of Databases*. Addison Wesley Publ. Co.

Artale, A.; Calvanese, D.; Kontchakov, R.; and Zakharyaschev, M. 2009. The *DL-Lite* family and relations. *J. of Artificial Intelligence Research* 36:1–69.

Baader, F.; Calvanese, D.; McGuinness, D.; Nardi, D.; and Patel-Schneider, P. F., eds. 2010. *The Description Logic Handbook: Theory, Implementation, and Applications*. Cambridge University Press. Paperback edition.

Batini, C., and Scannapieco, M. 2006. *Data Quality: Concepts, Methodologies and Techniques*. Data-Centric Systems and Applications. Springer.

Beeri, C., and Vardi, M. Y. 1981. The implication problem for data dependencies. In *Proc. of ICALP'81*, volume 115 of *LNCS*, 73–85. Springer.

Bienvenu, M.; ten Cate, B.; Lutz, C.; and Wolter, F. 2013. Ontology-based data access: A study through disjunctive datalog, CSP, and MMSNP. In *Proc. of PODS 2013*. ACM Press.

Calì, A.; Gottlob, G.; Lukasiewicz, T.; Marnette, B.; and Pieris, A. 2010. Datalog+/-: A family of logical knowledge representation and query languages for new applications. In *Proc. of LICS 2010*, 228–242.

Calì, A.; Gottlob, G.; and Pieris, A. 2011. New expressive languages for ontological query answering. In *Proc. of AAAI 2011*, 1541–1546.

Calvanese, D.; De Giacomo, G.; Lembo, D.; Lenzerini, M.; and Rosati, R. 2005. *DL-Lite*: Tractable description logics for ontologies. In *Proc. of AAAI 2005*, 602–607.

Calvanese, D.; De Giacomo, G.; Lembo, D.; Lenzerini, M.; Poggi, A.; Rodriguez-Muro, M.; Rosati, R.; Ruzzi, M.; and Savo, D. F. 2011. The Mastro system for ontology-based data access. *Semantic Web J.* 2(1):43–53.

Dong, X. L., and Srivastava, D. 2013. Big data integration. *PVLDB* 6(11):1188–1189.

Fagin, R.; Kolaitis, P. G.; Miller, R. J.; and Popa, L. 2005. Data exchange: Semantics and query answering. *Theoretical Computer Science* 336(1):89–124.

Fan, W., and Geerts, F. 2012. *Foundations of Data Quality Management*. Synthesis Lectures on Data Management. Morgan & Claypool Publishers.

Halevy, A. Y. 2001. Answering queries using views: A survey. *VLDB Journal* 10(4):270–294.

Kolaitis, P. G.; Panttaja, J.; and Tan, W. C. 2006. The complexity of data exchange. In *Proc. of PODS 2006*, 30–39.

Kontchakov, R.; Lutz, C.; Toman, D.; Wolter, F.; and Zakharyaschev, M. 2011. The combined approach to ontology-based data access. In *Proc. of IJCAI 2011*, 2656–2661.

Lenzerini, M. 2002. Data integration: A theoretical perspective. In *Proc. of PODS 2002*, 233–246.

Lenzerini, M. 2011. Ontology-based data management. In *Proc. of CIKM 2011*, 5–6.

Poggi, A.; Lembo, D.; Calvanese, D.; De Giacomo, G.; Lenzerini, M.; and Rosati, R. 2008. Linking data to ontologies. *J. on Data Semantics* X:133–173.

Rodriguez-Muro, M., and Calvanese, D. 2012a. High performance query answering over *DL-Lite* ontologies. In *Proc. of KR 2012*, 308–318.

Rodriguez-Muro, M., and Calvanese, D. 2012b. Quest, an OWL 2 QL reasoner for ontology-based data access. In *Proc. of OWLED 2012*, volume 849 of *CEUR,* ceur-ws.org.

Wand, Y., and Wang, R. Y. 1996. Anchoring data quality dimensions in ontological foundations. *Commun. ACM* 39(11):86–95.