Data Accuracy as Knowledge in Ontology Based Data Access (preliminary report)

Citation for published version:
Console, M 2016, Data Accuracy as Knowledge in Ontology Based Data Access (preliminary report). in Proceedings of the 29th International Workshop on Description Logics Cape Town, South Africa, April 22-25, 2016.. CEUR Workshop Proceedings (CEUR-WS.org), 29th International Workshop on Description Logics, Cape Town, South Africa, 22/04/16.

Link:
Link to publication record in Edinburgh Research Explorer

Document Version:
Publisher's PDF, also known as Version of record

Published In:
Proceedings of the 29th International Workshop on Description Logics Cape Town, South Africa, April 22-25, 2016.

Data accuracy as knowledge in ontology based data access (preliminary report)

Marco Console

Dipartimento di Ing. Informatica, Automatica e Gestionale “Antonio Ruberti”
SAPIENZA Università di Roma
Via Ariosto 25, I-00186 Roma, Italy
`console@dis.uniroma1.it`

Abstract. In the context of Ontology Based Data Access (OBDA), consistency of data ensures that the data sources are coherent with the rules of the domain of interest represented by the ontology.
However, even when consistency holds, the data underlying an OBDA system can still be in a state that users perceive of poor quality, according to some intuitive requirements. In many of these cases, the mechanism currently used to specify an OBDA system seems to lack of the ability to express such requirements.
In this work, we argue that those requirements are often not about the world that the ontology represents, but about the knowledge that the system possesses on the world. Thus, with the aim of formalizing data quality specifications in the OBDA context, we propose the usage of a language of modal constraints, and show how they can be used in practice to capture cases of poor data quality.
For this novel class of assertions, and for OBDA systems where the ontology is expressed in DL-Lite, we present algorithms and complexity results for the problem of checking the accuracy of the knowledge that the system posses, i.e., whether the system respects the modal constraints in the specification.

1 Introduction

The problem of assessing the quality of data is a well-known topic, that has always received a lot of attention from different branches of the scientific community, in particular in statistics and data management (see [3]). Despite many studies and approaches, the problem is still a hot topic today, also because of the advent of the big data wave ([11]).

Assessing the quality of data is a manifold process, that consists of many different tasks. To ease its overall complexity, a largely accepted practice is the adoption of *dimensions*. A data quality dimension is a single aspect onto which the analysis is focused. Unfortunately, such dimensions often lack of a formal definition

A notable attempt to tackle this problem has been proposed by Y. Wand, and R. Y. Wang in [28]. In that work, the authors propose to build data quality dimensions starting from a formal framework, inspired to the notion of ontology, i.e. a formal description of a portion of the real world, made in terms of its states and laws. Despite being a very promising intuition, one of the first attempts to use it in practice has been proposed only recently in [10].

In this work, we go along the same line, and use the ontology-based data access paradigm (OBDA for short) to assess the *accuracy* dimension, i.e. the extent to which the information stored inside a database is accurate.

OBDA is a novel paradigm, aiming at accessing and managing the data contained in a database by means of a formal specification, made of an ontology and a mapping ([6, 26, 15, 5, 4]). The ontology provides a formalization for the domain of interest, in terms of formal axioms, while the mapping describes its relation with the data. Pairing an OBDA specification with a database gives rise to an OBDA system. In what follows, we shall consider OBDA systems built using *Description Logic* ontologies.

The advantages of using an OBDA system to assess the quality of a database are manifold. The high level of abstraction achieved by the OBDA paradigm, in fact, is very useful when considering the overall quality of the data scattered through the tables of real data sources. Furthermore, OBDA systems can be used to get rid of all the superfluous information, such as index attributes and reference tables, and focus on the real data quality issues.

However, differently from the dimension of *consistency* studied in [10], assessing the accuracy of data is an elusive task, that is closely related to the data stored in the database. In this sense, ontological languages currently used in OBDA specifications are not well suited to specify data accuracy requirements. The following example clarifies this intuition.

Example 1 (Consistent but inaccurate data). The following is the database schema $\mathcal{S}$ of the system used by a company to store data about orders and customers.

$$\begin{aligned}\mathcal{S} = \{ & PERSONS(code, SSN, isGoldCustomer),\\ & CENSUS(SSN, Address, Name, Surname, DateOfBirth),\\ & PRODUCTS(ProductCode, Type),\\ & ORDERS(OrderCode, PersonCode, ProductCode, hasBeenPaid)\}\end{aligned}$$

Consider the following instance D of the schema.

$$\begin{aligned}D = \{ & PERSONS(P1, SSN1, true), PERSONS(P2, SSN2, true)\\ & CENSUS(SSN1, A1, Name1, Surname1, DOB1),\\ & ORDERS(O01, P1, Pr1, true), ORDERS(O03, P3, Pr3, false)\}\end{aligned}$$

The company recognizes the status of *golden customer* only to those individuals who have already paid an order. No constraints in the database schema, however, enforce this requirement. The OBDA specification $\mathcal{B}$, defined as follows, formally describes this scenario.

$$\begin{aligned}\mathcal{T} = \{ & GoldenCustomer \sqsubseteq \exists hasPaid, GoldenCustomer \sqsubseteq Customer,\\ & Order \sqsubseteq \exists hasOrdered^{-}, \exists hasOrdered^{-} \sqsubseteq Order,\\ & Customer \sqsubseteq \exists hasOrdered, \exists hasOrdered \sqsubseteq Customer,\\ & Customer \sqsubseteq \exists hasAddress, hasPaid \sqsubseteq hasOrdered\}\\ \mathcal{M} = \{ & PERSONS(x, z_1, \text{`}true\text{'}) \to GoldenCustomer(x),\\ & PERSONS(x, z_1, z_2) \wedge CENSUS(z_1, y, z_3, z_4, z_5) \to hasAddress(x, y),\\ & ORDERS(x, y, z_1, z_2) \to hasOrdered(y, x),\\ & ORDERS(x, y, z_1, \text{`}true\text{'}) \to hasPaid(y, x),\\ & ORDERS(z_1, x, z_2, z_3) \to Customer(x)\}\end{aligned}$$

Intuitively, the logical theory formed by $\mathcal{B}$ and D (i.e. the OBDA system $\langle \mathcal{B}, D \rangle$) is consistent, in the sense that no rule of the ontology is violated. However, the data stored in D are inaccurately considering the customer associated with $P2$ as a golden customer, in spite of the fact that no paid order is connected to that customer code.

In order to represent such constraint, and in fact many others, we can resort to what the OBDA system $\langle \mathcal{B}, D \rangle$ is sure about, or, in other terms, what the OBDA system *knows*.

Example 2. (Enforcing knowledge) Consider again the scenario described in Example 1, and suppose to impose to the system an additional constraint k, whose intuitive meaning is the following: each known golden customer has a known paid order associated to her.

Now consider the following. The golden customers known by the system are, intuitively, $P1$ and $P2$. However, no known *paid* order is associated to $P2$. Thus, we can conclude that the constraint k is not satisfied by the OBDA system $\langle \mathcal{B}, D \rangle$.

In the rest of this paper, we expand on the topic of using constraints of the kind presented in Example 2, with the aim of expressing requirements about the accuracy of data. To this aim, we shall present the class of *epistemic dependencies*, i.e. a modal derivative of the well known class of embedded dependencies (see e.g. [1]), and study their interaction with OBDA systems based on *DL-Lite*$_\mathcal{A}$ ontologies.

In fact, the languages currently used to specify OBDA systems are, in general, unable to express requirements about knowledge, as shown in the previous example. Furthermore, the language presented in [12] is not well suited to specify ontologies for the OBDA scenario, while constraints presented in [22] cannot directly express knowledge requirements.

In order to give to our framework formal semantic, we shall study the problem of constraint satisfaction using the well known modal logic of knowledge and belief $\mathcal{OL}$, due to Levesque ([19]).

Notice that we don't claim any novelty on the notion of epistemic state of an ontology, see e.g. [12], nor on the idea of using modal formulas to express constraints over knowledge-bases, which has been proposed for the first time by Reiter in [24]. However, to the best of our knowledge, the idea of using OBDA systems in conjunction with epistemic constraints to assess the accuracy of the information stored in a set of data sources has not been investigated yet.

The paper is organized as follows. In Section 2 we present preliminary notions that we use throughout this work. In Section 3 we detail the notion of what is known to an OBDA system, which we use in Section 4 to present our class of constraints. The complexity of handling such constraints is presented in Section 6. The notion of data accuracy in the OBDA scenario is investigated in Section 5. In Section 7 we conclude.

2 Preliminary definitions

In this section we present technical notions that we use in the remainder of this work.

Formulas and interpretations. For our logical formulas, we consider predicate logic, and sometimes allow the presence of modal operators. For the syntax of formulas, we assume the standard inductive definition (see e.g. [1] and [20]), and sometimes allow the symbol $\bot$, meaning the empty set. Formulas in which appears no modal operator are called *objective*. A logical theory is a set of formulas.

A *query* is an open formula, i.e. a formula in which some variables don't appear in the scope of any quantifier. Two notable classes of queries are the class of conjunctive queries, and union thereof. A conjunctive query (CQs) is a conjunction of existentially quantified atoms, possible containing free variables. For union of conjunctive queries (UCQs), we further require that the free variables in each disjunct are the same. To better point out the set of free variables of a query q, we sometimes use the set theoretic notation, i.e. $\{\bar{x} \mid q(\bar{x})\}$, where $\bar{x}$ is a vector of variables.

Objective formulas shall be interpreted over *first order interpretations*, with their usual semantics (see e.g. [1]), and $\bot$ will always have empty extension. Semantics for the modalities are given later.

For queries, we are interested in computing their *certain answers*, i.e. the set of substitutions for their free variables that make the formula true in every model of a given logical theory. We denote the set of certain answers for a given query q, under the logical theory $\mathcal{L}$, as $ans(q, \mathcal{L})$. We assume $ans(\bot, \mathcal{L}) = \varnothing$, whenever $\mathcal{L}$ has at least one model.

Databases. In this work, we consider *relational databases*, and refer the reader to [1] for a more detailed account.

A database *schema* $\mathcal{S} = \langle \Sigma_\mathcal{S}, \mathcal{C}_\mathcal{S} \rangle$ is a pair where $\Sigma_\mathcal{S}$ is an alphabet of predicate symbols, while $\mathcal{C}_\mathcal{S}$ is a set of formulas in the alphabet $\Sigma_\mathcal{S}$. A set of ground facts, built using the alphabet $\Sigma_\mathcal{S}$ of a schema $\mathcal{S}$, is called $\mathcal{S}$-database. Whenever an $\mathcal{S}$-database D also respects the constraints in $\mathcal{C}_\mathcal{S}$, we say that it is *legal* w.r.t. $\mathcal{S}$ (written $D \models \mathcal{S}$), or simply legal. In this work, we don't impose any particular restriction on the class of constraints that can appear in a database schema.

Description Logic ontologies. An ontology is the conceptualization of a domain of interest, expressed in terms of a formal language. In this work, we consider ontologies expressed using Description Logic, and focus our attention on the language *DL-Lite*$_\mathcal{A}$ [7, 23], a member of the *DL-Lite* family[1] of tractable Description Logics. 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, roles, and attributes, respectively, and $T_1, \ldots, T_n$ are unbounded pairwise disjoint predefined value-domains):

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

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

- Inclusion assertions in the form: $B \sqsubseteq C$, $Q \sqsubseteq R$, $U \sqsubseteq V$, $E \sqsubseteq F$.
- Functionality assertions in the form: $(\mathsf{funct}\ Q)$, $(\mathsf{funct}\ U)$.

[1] Not to be confused with the set of DLs studied in [2], which form the *DL-Lite*$_{bool}$ family.

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 (see [23] for a detailed account on *DL-Lite*$_{\mathcal{A}}$).

Note that checking *DL-Lite*$_{\mathcal{A}}$-KB for satisfiability, i.e., checking whether $Mod(\langle \mathcal{T}, \mathcal{A} \rangle) = \{ \mathcal{I} \mid \mathcal{I}$ is an interpretation for $\Sigma_{\mathcal{T}}$ such that $\mathcal{I} \models \langle \mathcal{T}, \mathcal{A} \rangle \}$ 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 intentional component, that we call OBDA specification, and an extensional component, represented by a database.

An OBDA specification is in turn constituted by three main components , i.e. the ontology, the mapping, and the database schema.

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

- $\mathcal{T}$ *is a DL-Lite*$_{\mathcal{A}}$ *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* $\forall \bar{x} \phi(\bar{x}) \rightarrow \exists \bar{y} \psi(\bar{x}, \bar{y})$*, where* $\phi(\bar{x})$ *is a conjunctive query over* $\Sigma_{\mathcal{S}}$ *with free variables* $\bar{x}$*, and* $\psi(\bar{x}, \bar{y})$ *is a conjunctive query over the alphabet* $\Sigma_{\mathcal{T}}$ *with free variables* $\bar{x} \cup \bar{y}$*.*

In what follows, we will refer to a specific form of mappings, called GAV, extensively studied in the database literature [14, 18]. 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 $\forall \bar{x} \phi(\bar{x}) \rightarrow \psi(\bar{z})$, where all the variables in $\bar{z}$ appear also in $\bar{x}$.

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, they are all those interpretations $\mathcal{I}$ for $\Sigma_{\mathcal{T}}$ that satisfy $\mathcal{T}$, and such that the pair $\langle D, \mathcal{I} \rangle$ satisfy all mapping assertions in $\mathcal{M}$, written $\langle D, \mathcal{I} \rangle \models \mathcal{M}$.

Definition 2 (OBDA system semantics). *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 $D \models \mathcal{S} \}$.

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 $\langle D, \mathcal{I} \rangle$ satisfies all mapping assertions in $\mathcal{M}$.

Query answering and rewriting. OBDA systems defined above enjoy several desirable properties, that we cannot discuss here for lack of space. We only briefly discuss how we can compute certain answers for queries in a satisfiable OBDA system. Let $\mathcal{B} = \langle \mathcal{T}, \mathcal{M}, \mathcal{S} \rangle$ be a OBDA system, and D a $\Sigma_{\mathcal{S}}$-database. The results presented in [23] 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* of

q with respect to $\mathcal{T}$ and $\mathcal{M}$, written $\mathcal{R}_{\mathcal{T},\mathcal{M}}(q)$, 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. Answering queries under an OBDA system is in $PTime$ w.r.t. the size of the ontology, and in AC^0 w.r.t. the size of the database.

3 What an OBDA system knows

In our study of data accuracy, a central part is played by the knowledge that an OBDA system possesses on the real world. In this section, we discuss how we can formally represent this knowledge.

To this aim, we resort to the predicate version of the well known modal logic $\mathcal{OL}$ of knowledge and belief, due to Levesque ([19]). Through the years, $\mathcal{OL}$ has been extensively used in many different flavors ([19, 16, 17]), of which we use the version presented in [17]. This logic comes equipped with two modal operators: the standard modality $\mathbf{K}$, meaning that something is *known* to be true, and the modality $\mathbf{O}$, whose intuitive meaning is *only-know*. We consider $\mathcal{OL}$ formulas to be standard first-order formulas, in which we further allow the appearance of $\mathbf{K}$ and $\mathbf{O}$.

We give the semantics for the $\mathcal{OL}$ logic using the notion of *epistemic state*. An epistemic state $\mathcal{E}$ is simply a set of first order interpretations, called worlds. For worlds inside the same epistemic state $\mathcal{E}$, we require that they share a single (countably infinite) domain of discourse, $\Delta_{\mathcal{E}}$, called the set of *standard names* for $\mathcal{E}$. Notice how this assumption implies the Barcan axiom ([16]). The next definition formalize the semantics of $\mathcal{OL}$, over epistemic states.

Definition 3. *Let α be a generic $\mathcal{OL}$ formula, $\mathcal{E}$ an epistemic state, and w a world in $\mathcal{E}$. Then α is* true *in $\mathcal{E}, w$, or $\mathcal{E}, w \models \alpha$, if the following conditions hold.*

- *α is an atomic formula, and $w \models \alpha$.*
- *$\alpha = \neg\alpha'$, and $\mathcal{E}, w \not\models \alpha'$*
- *$\alpha = \alpha' \wedge \alpha''$, and both $\mathcal{E}, w \models \alpha'$ and $\mathcal{E}, w \models \alpha''$ hold.*
- *$\alpha = \exists\bar{x}.\alpha'(\bar{x})$, and for some tuple $\bar{c}$ of parameters, we have that $\mathcal{E}, w \models \alpha'(\bar{c})$.*
- *$\alpha = \mathbf{K}(\alpha')$, and $w' \in W \Longrightarrow \mathcal{E}, w' \models \alpha'$.*
- *$\alpha = \mathbf{O}(\alpha')$, and $w' \in W \Longleftrightarrow \mathcal{E}, w' \models \alpha'$.*

Abbreviations $\vee$, $\rightarrow$, and $\forall$ shall be considered with the usual meaning.

Intuitively, if an OBDA system *knows* that some formula is true, then it is true in all the possible worlds represented by the ontological specification. On the other hand, if a formula is what an OBDA system *only-knows* about the real world, all and only the wolds in which that formula is true are models of the system.

In light of this, the knowledge possessed by an OBDA system is closely related to the data stored inside its database. In fact, from the notion of semantics given in Definition 2, something can be true in all the models of an OBDA system only if it is grounded into its extensional part.

Notice however, that the standard definition of semantics for OBDA systems cannot be directly used in the $\mathcal{OL}$ context. In order to gather all the knowledge that a system possesses on the real world, we make use of the $\mathbf{O}$ operator, and interact with the gathered knowledge by means of $\mathbf{K}$.

In fact, as pointed out in ([19]), the presence of the $\mathbf{O}$ operator allows $\mathcal{OL}$ to formalize the notion of what an agent knows about its own knowledge, i.e. its auto-epistemic state. Whenever $\mathbf{O}$ is not present, this notion needs to be formalized by means of some meta-logical operator, such as the stable expansion (see e.g. [21]), or the $\models\!\approx$ operator ([25]).

We use the notion of *knowledge state*, presented below, to formalize this concept in the OBDA scenario.

Definition 4. *Let $\mathcal{O}$ be an ontology based data access system, then the epistemic state $\mathcal{E}$ is said to be a* knowledge state *of $\mathcal{O}$ if and only if $\mathcal{E} \models \mathbf{O}(\mathcal{O})$.*

Our notion of knowledge state of an OBDA system is similar to the notion of epistemic model of an ontology given in [12]. However, the use of the modal operator $\mathbf{O}$ allows us to omit any further condition of maximality.

From Definition 4 comes the following.

Proposition 1. *Let $\mathcal{E}$, $\mathcal{E}'$ be two knowledge states for the OBDA system $\mathcal{O}$. Then $\mathcal{E}$ and $\mathcal{E}'$ are equivalent, up to isomoprhisms on the set of standard names.*

The property presented in Proposition 1 comes from the fact that $\mathcal{O}$ is a first order theory. In fact, once fixed the set of standard names, the knowledge state of an OBDA system must contain all and only the first order interpretations that are models of the system.

In the OBDA systems defined in Section 2, knowledge state is the product of the general truth about the real world, expressed in the ontology, and the data stored in the database. In light of this, OBDA specifications currently lack of the ability to directly express requirements on knowledge. The language of constraints presented in Section 4 can be used to express requirements over epistemic states, and hence enforce, or assess, the knowledge possessed by an OBDA system.

4 A language to enforce knowledge

To describe our language of constraints for knowledge states, we start from the considerations about integrity constraints presented by Reiter in [24].

In that work, the author defines the class of *pure* KFOPCE sentences being, essentially, the class of all $\mathbf{K}$-modal sentences in which predicates appear only in the scope of the modal operator $\mathbf{K}$. This class of formulas is then used to express a more meaningful form of integrity constraints for databases, through the notion of the entailment operator $\models\!\approx$.

Later, in [25], the author establishes an equivalence between such operator and the validity of formulas of the form: $\mathbf{O}(\mathcal{T}) \rightarrow \mathbf{K}(\alpha)$, where $\mathcal{T}$ is a first order non modal theory, and α is a *pure* KFOPCE formula.

A similar notion is used in [8] as the basic building block of the query language $\mathcal{EQL}$. Such language has been devised with the aim of expressing ontological queries, using a syntactically restricted form of the epistemic operator $\mathbf{K}$.

To refer to this concept, and avoid any discrepancy in the notation, we shall call these formulas simply *subjective*.

For our language of constraints, we make use of this same intuition but, unlike our predecessors, we restrict ourselves to the well known class of embedded dependencies ([1]), defining their subjective version as follows.

Definition 5. *A formula is a subjective disjunctive dependency if and only if it has the form*

$$\forall \bar{x}, \bar{y}.\mathbf{K}(\exists \bar{b}.\phi(\bar{x}, \bar{y}, \bar{b})) \rightarrow \Psi$$

such that the following conditions hold.

- $\phi(\bar{x}, \bar{y})$ *is a formula of the form* $\bigwedge_i C_i(\bar{x}_i, \bar{y}_i, \bar{b}_i)$*, where each* $C_i(\bar{x}_i, \bar{y}_i, \bar{b}_i)$ *is a relational atom in the variables* $\bar{x}_i \cup \bar{y}_i \cup \bar{b}_i$, $\bar{x} = \bigcup_i \bar{x}_i$, $\bar{y} = \bigcup_i \bar{y}_i$, *and* $\bar{b} = \bigcup_i \bar{b}_i$
- Ψ *is either:*
  1. *a formula* $\bigvee_s \psi_s(\bar{x}, \bar{z})$*, where each* $\psi_s(\bar{x}, \bar{z})$ *has the form:*

     $$\exists \bar{z}.\mathbf{K}(\exists \bar{b}' \bigwedge_j C_j(\bar{x}_j, \bar{z}_j, \bar{b}_j))$$

     and each $C_j(\bar{x}_j, \bar{z}_j, \bar{b}_j)$ *is a relational atom in the variable* $\bar{x}_j \cup \bar{z}_j \cup \bar{b}_j$*, and* $\bar{x} = \bigcup_j \bar{x}_j$, $\bar{z} = \bigcup_j \bar{z}_j$, $\bar{b}' = \bigcup_j \bar{b}_j$, *or*
  2. *a conjunction of equality atoms in the form* $\bigwedge_e t_e = t'_e$*, where each* t_e *and* t'_e *are variables from* $\bar{x}$*, or*
  3. *the symbol* $\perp$.

For the sake of brevity, from now on we shall refer to the class of purely subjective embedded dependencies simply as *epistemic dependencies* (EDs for short), and consider this as our class of constraints. Notice how the explicit usage of the operator $\mathbf{K}$ makes EDs different from the constraints presented in [27].

As already pointed out in Section 3, the notion of semantics given in Definition 2 is not well suited for the interaction with EDs. The next definition presents a novel notion of satisfaction, suitable for EDs in the OBDA scenario.

Definition 6. *Let* $\mathcal{O}$ *be an OBDA system, and* k *an epistemic dependency. Then we say that* $\mathcal{O}$ *satisfies* k *if and only if* $\mathbf{O}(\mathcal{O}) \rightarrow \mathbf{K}(k)$.

EDs can be used to enhance OBDA specifications, adding further constraints about knowledge. We now present a new definition for OBDA specifications, augmented by a set of EDs.

Definition 7. *Let* $\mathcal{K}$ *be a set of epistemic dependencies, and* $\mathcal{B}$ *an OBDA specification. Then the constrained OBDA specification* $\mathcal{B}_{\mathcal{K}}$ *is the pair* $\langle \mathcal{B}, \mathcal{K} \rangle$.

A constrained OBDA specification can be paired with a database to form a *constrained OBDA system*, in the usual way. The following is the definition of semantics for constrained OBDA systems.

Definition 8. *Let $\mathcal{B}_{\mathcal{K}} = \langle \mathcal{B}, \mathcal{K} \rangle$ be a constrained OBDA specification, and let D be a database. Then a first order interpretation $\mathcal{I}$ is a model for the constrained OBDA system $\mathcal{O}_{\mathcal{K}} = \langle \mathcal{B}_{\mathcal{K}}, D \rangle$ if and only if:*

1. *$\mathcal{I}$ is a model for the OBDA system $\mathcal{O} = \langle \mathcal{B}, D \rangle$*
2. *$\mathcal{O}$ satisfies k, for each $k \in \mathcal{K}$*

Intuitively, condition 2 requires that a given set of constraints is satisfied by the current epistemic state of the system. In this sense, the condition is on the whole system, and the single model doesn't play any part in it.

As usual, a constrained OBDA system with no model, is said to be unsatisfiable, satisfiable otherwise. In the same way, we call a database D consistent with respect to a constrained OBDA specification $\mathcal{B}_{\mathcal{K}}$, whenever the system $\langle \mathcal{B}_{\mathcal{K}}, D \rangle$ is satisfiable; inconsistent otherwise.

The complexity of checking whether an OBDA system satisfies a given ED (Definition 6), and hence the complexity of checking whether a constrained OBDA system is satisfiable (Definition 8), is analyzed in Section 6.

5 Modelling accuracy

With the aim of drawing a parallel between data accuracy and OBDA, we start this section by reminding the reader of the standard definition of *accuracy*. A database D is said to be accurate whenever it accurately describes the reality of interest, with respect to the tasks at hand (see e.g. [3, 13]).

When the database is paired to an OBDA specification, we can use the language of the ontology to express accuracy requirements, by means of a set of EDs. In fact, as already mentioned, something is known to an OBDA system only if it is grounded into the data stored into its database. Next definition formalizes this intuition.

Definition 9. *Let $\mathcal{B}_{\mathcal{K}} = \langle \mathcal{B}, \mathcal{K} \rangle$ be a constrained OBDA specification, and let D be a database. Then D is said to be of poor quality with respect to accuracy, according to $\mathcal{B}_{\mathcal{K}}$, if and only if $\mathcal{D}$ is consistent with $\mathcal{B}$ but it is not consistent with $\mathcal{B}_{\mathcal{K}}$.*

One may ask whether EDs posses the expressiveness required to specify meaningful constraints. In the remainder of this section, we argue that this is the case, by showing some practical example of what EDs may express, using the scenario presented in Example 1.

Example 3 (Information accuracy). Consider again the constraint described in Example 2. It's very straightforward to see how the following ED fully captures the intended meaning.

$$k_1 : \forall x.\mathbf{K}(Customer(x)) \rightarrow \exists y.\mathbf{K}(hasPaid(x, y))$$

As expected, the constraint k_1 is violeted for $x/P2$.

Example 4 (Completeness of the known information). The problem of assessing data completeness is a complex topic, behind the scope of this work, and probably outside the expressiveness of our constraints.

In spite of this, EDs can still be used to impose completeness on the known information. The following constraint checks whether the address of each customer that has paid an order is known to the system.

$$k_2 : \forall x.\mathbf{K}(\exists y.Customer(x) \land hasPaid(x, y)) \rightarrow \exists z.\mathbf{K}(hasAddress(x, z))$$

Notice how k_2 doesn't require any *known* order. The constraint is violated for $x/P2$.

Example 5 (Disjunctions). We believe that one of the most interesting features of EDs is the possibility of using disjunctions. For example, we can impose that only golden customers can delay the payment of known orders, i.e. each known order has been either paid, or placed by a golden customer.

$$\begin{aligned} k_3 : \forall x.\mathbf{K}(Order(x)) \rightarrow\ & \mathbf{K}(\exists y.hasPaid(y, x)) \lor \\ & \mathbf{K}(\exists z.hasOrdered(z, x) \land GoldenCustomer(z)) \end{aligned}$$

Constraint k_3 is violated for $x/O03$.

6 An algorithm to check constraints

In this section we present an algorithm to check whether an OBDA system satisfies a given ED, and show its complexity. In what follows, we shall often transform subjective formulas into first order queries of a specific form. The following definition details this transformation.

Definition 10. *Let $\phi : \mathbf{K}(\exists \bar{y} \bigwedge_i C_i(\bar{x}_i, \bar{y}_i))$ be a formula where $\bar{x}$ is either free or quantified outside the scope of $\mathbf{K}$. Then $q_\phi(\bar{x})$ is the following conjunctive query:*

$$\{\bar{x} \mid \exists \bar{y} \bigwedge_i C_i(\bar{x}_i, \bar{y}_i)\}$$

We now present the complexity results for the problem of checking satisfaction of EDs. To this aim, we start by formalizing the decision problem associated to the notion of satisfaction.

Definition 11. *Let $\mathcal{O}$ be an OBDA system, and let k be an epistemic dependency. Then $\mathcal{K}$-Satisfaction is the following decision problem: check whether $\mathcal{O}$ satisfies k.*

For OBDA systems defined in Section 2, $\mathcal{K}$-Satisfaction can be reduced to query answering. To this aim, we start by stating the following lemma.

Lemma 1. *Let $\phi = \mathbf{K}(\exists y.C(\bar{x}, \bar{y}))$, where C is a conjunction of propositional atoms, and let $\mathcal{O} = \langle \mathcal{B}, D \rangle$ be a satisfiable OBDA system. Then $\mathbf{O}(\mathcal{O}) \rightarrow \mathbf{K}(\exists \bar{y}.\phi(\bar{c}, \bar{y}))$, being $\bar{c}$ a tuple of standard names, if and only if $\bar{c} \in ans(q_\phi(\bar{x}), \mathcal{O})$.*

Lemma 1 gives us a direct method for computing $\mathcal{K}$-Satisfaction under a given OBDA system. Intuitively, we can compute the answer of the conjunctive query representing each part of an ED, and then compare the results.

Theorem 1. *Let $\mathcal{B}$ be an OBDA specification, D a database, and $\mathcal{O} = \langle \mathcal{B}, D \rangle$ a satisfiable OBDA system. Furthermore, let $k : \phi \rightarrow \bigvee_s \psi_s$ be an epistemic dependency in the form of Definition 5.*

Then $\mathcal{O}$ satisfies k if and only if, for each vector of standard names $\bar{c}$, such that $ans(q_\phi(\bar{c}, \bar{y}), \mathcal{O})$ is non-empty, we have that also $ans(q_{\psi_s}(\bar{c}, \bar{z}), \mathcal{O})$ is non-empty for some ψ_s.

Theorem 1 directly suggests the algorithm presented in Figure 1. Intuitively, the algorithm goes as follows: it computes the perfect rewriting of the queries representing each part of an ED, and issues a suitable first order query to the underlying database system to check whether a violating tuple exists.

```
Algorithm check-dependency.
Let $k : \phi \rightarrow \psi$ be an ED in the form of Definition 5, and $\mathcal{O} = \langle \mathcal{T}, \mathcal{M}, \mathcal{S} \rangle$.
check-dependency($k$, $\mathcal{O}$, $D$) do:
        $q_b(\bar{x}, \bar{y}) := \mathcal{R}_{\mathcal{T},\mathcal{M}}(q_\phi(\bar{x}, \bar{y}))$;
        if ($\Psi = \bigvee_i \psi_i(\bar{x}, \bar{z})$) then $q_h(\bar{x}, \bar{z}) := \bigvee_i \mathcal{R}_{\mathcal{T},\mathcal{M}}(\psi_i(\bar{x}, \bar{z}))$;
        else $q_h := \Psi$;
        $\phi_k := \exists \bar{x}, \bar{y}, \bar{z}. q_b(\bar{x}, \bar{y}) \wedge \neg q_h(\bar{x}, \bar{z})$;
        return $\neg ans(\phi_k, D)$;
```

Fig. 1. Algorithm to check whether $\mathcal{O}$ satisfies k.

Theorem 2. *Let $\mathcal{B} = \langle \mathcal{T}, \mathcal{M}, \mathcal{S} \rangle$ be an OBDA specification, k an epistemic dependency, and D a database. Then, $\mathcal{K}$-Satisfaction problem for k under $\mathcal{B}$ is in AC^0 in the size of D, in $PTime$ in the size of $\mathcal{T}$ and $\mathcal{M}$.*

Complexities shown in Theorem 2 are a direct consequence of the complexity of query answering under an OBDA systems (see e.g. [23, 9]).

7 Conclusions and future works

In this work we tackled the problem of assessing the accuracy of the data stored inside a database system, using the OBDA paradigm. Our techniques exploit the great level of abstraction of OBDA systems to thoroughly examine the information stored in the database, and makes use of modal formulas to interact with the real data.

We plan to extend this work by studying techniques to assess the accuracy at the schema level, i.e. to check whether a given database schema enforces sufficient accuracy requirements.

Acknowledgments I would like to thank Maurizio Lenzerini for the many invaluable discussions on the topics of this paper. Work partially supported by the EU under FP7, project Optique (Scalable End-user Access to Big Data), grant n. FP7-318338.

References

1. Abiteboul, S., Hull, R., Vianu, V.: Foundations of Databases. Addison-Wesley (1995)
2. Artale, A., Calvanese, D., Kontchakov, R., Zakharyaschev, M.: The dl-lite family and relations. J. Artif. Intell. Res. (JAIR) 36, 1–69 (2009), http://dx.doi.org/10.1613/jair.2820
3. Batini, C., Scannapieco, M.: Data Quality: Concepts, Methodologies and Techniques. Data-Centric Systems and Applications, Springer (2006)
4. Bienvenu, M., ten Cate, B., Lutz, C., Wolter, F.: Ontology-based data access: A study through disjunctive datalog, csp, and MMSNP. ACM Trans. Database Syst. 39(4), 33:1–33:44 (2014), http://doi.acm.org/10.1145/2661643
5. Calì, A., Gottlob, G., Pieris, A.: New expressive languages for ontological query answering. In: Proceedings of the Twenty-Fifth AAAI Conference on Artificial Intelligence, AAAI 2011, San Francisco, California, USA, August 7-11, 2011 (2011)
6. Calvanese, D., De Giacomo, G., Lembo, D., Lenzerini, M., Poggi, A., Rodriguez-Muro, M., Rosati, R., Ruzzi, M., Savo, D.F.: The MASTRO system for ontology-based data access. Semantic Web 2(1), 43–53 (2011)
7. Calvanese, D., De Giacomo, G., Lembo, D., Lenzerini, M., Rosati, R.: Dl-lite: Tractable description logics for ontologies. In: Proceedings, The Twentieth National Conference on Artificial Intelligence and the Seventeenth Innovative Applications of Artificial Intelligence Conference, July 9-13, 2005, Pittsburgh, Pennsylvania, USA (2005)
8. Calvanese, D., De Giacomo, G., Lembo, D., Lenzerini, M., Rosati, R.: Eql-lite: Effective first-order query processing in description logics. In: IJCAI 2007, Proceedings of the 20th International Joint Conference on Artificial Intelligence, Hyderabad, India, January 6-12, 2007 (2007)
9. Calvanese, D., De Giacomo, G., Lembo, D., Lenzerini, M., Rosati, R.: Tractable reasoning and efficient query answering in description logics: The *DL-Lite* family. J. Autom. Reasoning 39(3), 385–429 (2007)
10. Console, M., Lenzerini, M.: Data quality in ontology-based data access: The case of consistency. In: Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence, July 27 -31, 2014, Québec City, Québec, Canada. pp. 1020–1026 (2014)
11. Dong, X.L., Srivastava, D.: Big data integration. PVLDB 6(11), 1188–1189 (2013)
12. Donini, F.M., Lenzerini, M., Nardi, D., Nutt, W., Schaerf, A.: An epistemic operator for description logics. Artif. Intell. 100(1-2), 225–274 (1998)
13. Fan, W.: Data quality: From theory to practice. SIGMOD Record 44(3), 7–18 (2015), http://doi.acm.org/10.1145/2854006.2854008
14. Halevy, A.Y.: Answering queries using views: A survey. VLDB J. 10(4), 270–294 (2001)
15. Kontchakov, R., Lutz, C., Toman, D., Wolter, F., Zakharyaschev, M.: The combined approach to ontology-based data access. In: IJCAI 2011, Proceedings of the 22nd International Joint Conference on Artificial Intelligence, Barcelona, Catalonia, Spain, July 16-22, 2011. pp. 2656–2661 (2011)
16. Lakemeyer, G., Levesque, H.J.: Only-knowing: Taking it beyond autoepistemic reasoning. In: Proceedings, The Twentieth National Conference on Artificial Intelligence and the Seventeenth Innovative Applications of Artificial Intelligence Conference, July 9-13, 2005, Pittsburgh, Pennsylvania, USA (2005)
17. Lakemeyer, G., Levesque, H.J.: Only-knowing meets nonmonotonic modal logic. In: Principles of Knowledge Representation and Reasoning: Proceedings of the Thirteenth International Conference, KR 2012, Rome, Italy, June 10-14, 2012 (2012)
18. Lenzerini, M.: Data integration: A theoretical perspective. In: Proceedings of the Twenty-first ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems, June 3-5, Madison, Wisconsin, USA. pp. 233–246 (2002)

19. Levesque, H.J.: All I know: A study in autoepistemic logic. Artif. Intell. 42(2-3), 263–309 (1990)
20. Levesque, H.J., Lakemeyer, G.: The logic of knowledge bases. MIT Press (2000)
21. Moore, R.C.: Semantical considerations on nonmonotonic logic. Artif. Intell. 25(1), 75–94 (1985)
22. Motik, B., Horrocks, I., Sattler, U.: Bridging the gap between OWL and relational databases. J. Web Sem. 7(2), 74–89 (2009), http://dx.doi.org/10.1016/j.websem.2009.02.001
23. Poggi, A., Lembo, D., Calvanese, D., De Giacomo, G., Lenzerini, M., Rosati, R.: Linking data to ontologies. J. Data Semantics 10, 133–173 (2008)
24. Reiter, R.: On integrity constraints. In: Proceedings of the 2nd Conference on Theoretical Aspects of Reasoning about Knowledge, Pacific Grove, CA, March 1988. pp. 97–111 (1988)
25. Reiter, R.: What should a database know? J. Log. Program. 14(1&2), 127–153 (1992)
26. Rodriguez-Muro, M., Calvanese, D.: High performance query answering over dl-lite ontologies. In: Principles of Knowledge Representation and Reasoning: Proceedings of the Thirteenth International Conference, KR 2012, Rome, Italy, June 10-14, 2012 (2012)
27. Tao, J., Sirin, E., Bao, J., McGuinness, D.L.: Integrity constraints in OWL. In: Proceedings of the Twenty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2010, Atlanta, Georgia, USA, July 11-15, 2010 (2010), http://www.aaai.org/ocs/index.php/AAAI/AAAI10/paper/view/1931
28. Wand, Y., Wang, R.Y.: Anchoring data quality dimensions in ontological foundations. Commun. ACM 39(11), 86–95 (1996)