Epistemic Integrity Constraints for Ontology-Based Data Management

Marco Console,¹ Maurizio Lenzerini²
¹University of Edinburgh
²Sapienza, University of Rome

Abstract
Ontology-based data management (OBDM) is a powerful knowledge-oriented paradigm for managing data spread over multiple heterogeneous sources. In OBDM, the data sources of an information system are handled through the reconciled view provided by an ontology, i.e., the conceptualization of the underlying domain of interest expressed in some formal language. In any information system where the basic knowledge resides in data sources, it is of paramount importance to specify the acceptable states of such information. Usually, this is done via integrity constraints, i.e., requirements that the data must satisfy formally expressed in some specific language. However, while the semantics of integrity constraints are clear in the context of databases, the presence of inferred information, typical of OBDM systems, considerably complicates the matter. In this paper, we establish a novel framework for integrity constraints in the OBDM scenarios, based on the notion of knowledge state of the information system. For integrity constraints in this framework, we define a language based on epistemic logic, and study decidability and complexity of both checking satisfaction and performing different forms of static analysis on them.

1 Introduction
Managing information spread over multiple heterogeneous data sources is a long-standing topic in the fields of data management and knowledge representation. A powerful tool to address this issue is the Ontology-based Data Management (OBDM) paradigm (Xiao et al. 2018), where a conceptual specification of the domain of interest, called ontology, is superimposed over a set of pre-existing data sources using suitable mapping assertions (Doan, Halevy, and Ives 2012). The resulting information systems, called Ontology-Based Data Management Systems, enable users to interact with the data using the vocabulary of the ontology, thus giving a high-level view of the information contained in the data sources. This high-level view of the data can be used to enhance query answering (Calvanese et al. 2011; Calì et al. 2010; Rodriguez-Muro and Calvanese 2012) and other data management tasks (Lenzerini 2018). In this paper, we focus on ontologies expressed in Description Logics (DL) (Baader et al. 2003). DLs are logical formalisms that represent the domain of interest in terms of concepts, i.e., classes of objects, and roles, i.e., binary relations between objects. Intuitively, the models of the logical theory constituted by the ontology, the mapping, and the data, i.e., the models of the system, represent all the possible worlds that are consistent with both the conceptual specification and the data. An example of OBDM system follows.

Example 1 Consider a set \( \mathcal{D} \) of data sources, storing HR data from different branches of a company. Branch A stores information about employees and their department in table \( \text{empA}(\text{Name}, \text{Code}, \text{Department}) \). The central HR office stores information about departments in table \( \text{dep}(\text{Code}, \text{Manager}) \) and employee codes in table \( \text{pers}(\text{Name}, \text{Code}) \). The following database, denoted by \( D \), represents a possible instance of \( \mathcal{D} \).

<table>
<thead>
<tr>
<th>empA</th>
<th>dep</th>
<th>pers</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bob</td>
<td>C</td>
<td>D</td>
</tr>
<tr>
<td>Jim</td>
<td>M</td>
<td>D</td>
</tr>
<tr>
<td></td>
<td></td>
<td>D1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Jim</td>
</tr>
</tbody>
</table>

We assume that the domain of interest can be formally described by a DL ontology \( T \) with concepts \( \text{Emp} \), for employees, and \( \text{Dep} \), for departments, and roles \( \text{hasDep} \), associating employees to their department, \( \text{hasCode} \), associating employees to their individual code, and \( \text{hasMan} \), associating departments to their manager. In this domain, every employee works for at least one department, as formalized by the following axioms in \( T \).

\[ \text{Emp} \sqsubseteq \exists \text{hasDep} \quad \exists \text{hasDep} \sqsubseteq \text{Dep} \]

The set \( M \) of mapping assertions (Figure 1) describes the relationship between the data in \( \mathcal{D} \) and the concepts and roles of \( T \). In the remainder of this section, we will denote by \( \mathcal{O} \) the OBDM system constituted by \( D \), \( T \), and \( M \).

In the context of data management, a question that arises naturally is whether the stored data conform with our understanding of the domain. During the years, this fundamental question gave rise to the concept of integrity constraints (Abiteboul, Hull, and Vianu 1995; Reiter 1988), ICs for short. Informally, ICs are a means to describe the acceptable states of a knowledge base, and are very popular in re-
Ople 1 reveals that any acceptable state of the OBDM system
1. Every employee has at most one code.

After a detailed analysis, the scenario in Example 3

ever, the need of defining the acceptable states of an OBDM
solutions are many. The consequence is that, after years of ef-
forts, there is still no general agreement on what formalism
semantics of integrity constraints is much more challenging.

Indeed, any mechanism that is able to validate the ICs in
this context requires a way to control the subtle interplay
between data, ontology axioms, mapping, and constraints.

Unfortunately, controlling the way these elements interact
is a task far from being straightforward, and the possible
solutions are many. The consequence is that, after years of ef-
fors, there is still no general agreement on what formalism
for ICs should be used in ontological reasoning and OBDM.
Despite the lack of a generally accepted formalism, how-
ever, the need of defining the acceptable states of an OBDM
system is very natural as the following example shows.


evaluation formaulae are true in the logical interpretation repre-
sented by the stored data. For this reason, defining the
states of the data. In this context, there is a general agree-
ment on languages and semantics suitable for ICs, typically
based on data dependencies, i.e., special first-order formulae
to be evaluated as queries over the database.

Example 2 Refer to Example 1, and assume that dep is the reference
table for departments, meaning that every depart-
ment must appear in dep. The following data dependency,
also called foreign key constraint, captures this requirement:

\[
\chi : \forall x, y, z. \ empA(x, y, z) \rightarrow \exists w. \ dep(z, w)
\]

ICs have received attention also in the fields of DL knowl-
edge bases and ontological reasoning (Donini et al. 1998;
Motik, Horrocks, and Sattler 2009) and OBDM (Nikolaou et
al. 2019), where, analogously to the case of databases, their
goal is to define the acceptable states of an information sys-
tem. However, databases are often based on the closed-world
semantics. Therefore, checking satisfaction of constraints in
a database D amounts to verifying whether the correspond-
ing formulæ are true in the logical interpretation repre-
sented by D. In contrast, OBDM systems support both mod-
eling of incomplete information and inferences on the data.
This means that the semantics of an OBDM system com-
prise a set of possible logical models extending the world
represented by the stored data. For this reason, defining the
semantics of integrity constraints is much more challenging.
Indeed, any mechanism that is able to validate the ICs in
this context requires a way to control the subtle interplay
between data, ontology axioms, mapping, and constraints.

Unfortunately, controlling the way these elements interact

Requirements in Example 3 should not be confused with
general statements about the way the world behaves, i.e.,
these are not ontological axioms. Consider, e.g., Require-
ment 1. In this scenario, an employee may have multiple
codes, maybe relative to the different roles she covers. To
allow this possibility, the ontology of O has no axioms enta-
tailing that codes are unique for an employee. What we re-
quire with Requirement 1 is that, in the current state of the
system, no employee has more than one code. Statements of
this kind require some form of introspection, and hence they
cannot be expressed using ontological axioms. To express
these requirements, we need to use ICs. These observations
bring us to the following conclusions: while ontological ax-
ions shape the information contained in OBDM systems a-
priori, ICs should act a-posteriori on the system, aiming at
validating its current state.

Depending on how its current state is defined, there are
different techniques to check and validate the information an
OBDM system contains. Two methods proposed during the
years are the Entailment Semantics (ES), and the Minimal
Herbrand Model Semantics (MHMS). In the Entailment Semantics (Reiter 1982), each IC is
treated simply as a boolean query, with the idea that it should be evaluated to true in every model of the system. This
approach suffers from two main drawbacks. First, checking satisfaction of ICs under ES can very quickly become un-
decidable, depending on the form of ICs. Second, ES is too strict, and its behavior can become counterintuitive at times.
For instance, by referring to Example 1, observe that, since
the ontological axioms do not entail that codes are unique for
the various employees, there exist models of O in which the
same employee has more than one code. In turn, this proves
that, under ES, O violates Requirement 1 mentioned in Ex-
ample 3. However, if we inspect the data sources, there is
no evidence whatsoever that the requirement is violated, be-
cause for no employee in the system we have more than one
code. Something analogous happens with Requirement 3.
The case of Requirement 2 is different. Such requirement is
satisfied under ES due to the ontological axioms, although
we do not have any evidence in the data about the depart-
ment where Wim works.

Intuitively, the above observations tell us that, under ES,
the current state of the data plays a marginal role. On the
one hand, violations that are not directly supported by the
data in the sources may blame the information in the OBDM
system, as in the case of Requirement 1 and 3. On the other
hand, constraints may be validated by the ontology, although
no data coming from the sources can be used for this pur-
pose, as in the case of Requirement 2.

A possible approach to get closer to the information con-
tained in the sources is to use some notion of minimal in-
formation. This is the idea of the Minimal Herbrand Models
Semantics. Intuitively, the unique minimal Herbrand model
H of an OBDM system O is an interpretation of O with the
following property: for every existentially quantified con-
junction of atomic formulæ \( \phi \), O entails \( \phi \) if and only if

### Example 1

1. Every employee has at most one code.

2. Every employee works in at least one department.

3. Every department has at least one manager.
\( \phi \) is true in \( \mathcal{H} \). Under MHMS, an OBDM system satisfies an IC if and only if the minimal Herbrand model of the system does. This notion was first advocated for ontological reasoning in (Motik, Horrocks, and Sattler 2009) and recently extended to OBDM systems in (Nikolaou et al. 2019). Interestingly, checking satisfaction of constraints under MHMS is decidable in many relevant cases.

However, as the definition of MHMS heavily relies on the minimal Herbrand Model, the violation or satisfaction of an IC under MHMS is still only loosely related to the content of the data sources. In our running example, it is easy to see that the minimal Herbrand model \( \mathcal{O} \) validates Requirements 1 and 2. However, observe that, although \( \mathcal{O} \) satisfies \( \exists x . \text{hasDep}(\text{Wim}, x) \), \( \mathcal{O} \) does not satisfy \( \exists x, y . \text{hasDep}(\text{Wim}, x) \land \text{hasMan}(x, y) \). Therefore, \( \mathcal{O} \) does not satisfy Requirement 3 under MHMS.

If the goal is to manage a set of data sources using an ontology, we argue that the violation or the satisfaction of an IC should be solidly grounded on the knowledge that the data in the sources allow the system to achieve. For this reason, we think that it is worth exploring a new semantics of ICs for OBDM systems, epistemic in nature. Under this semantics, ICs are validated by what the system knows about, i.e., by what the system knows, rather than by the possible worlds described by the system, or by the minimal information contained in such worlds. We call this approach to ICs for OBDM systems Knowledge Semantics (KS). In our current example, the employees known by \( \mathcal{O} \) are \( \text{Tim}, \text{Jim}, \text{Wim} \) and \( \text{Bob} \). For each of them, \( \mathcal{O} \) knows exactly one code and therefore, under KS, Requirement 1 is satisfied. In other words, KS reads Requirement 1 as follows: for each known employee, the system knows exactly one code. Similarly, Requirement 3 is satisfied under KS, since the departments known by \( \mathcal{O} \) are \( D_1 \) and \( D_0 \), and the system knows the manager of each of them. With regard to Requirement 2, we consider two possible readings of the condition. The first reading is the one imposing that all known employees work in a department. If we consider such reading, it is easy to see that the system satisfies the condition under KS. In the second reading the condition states that for each known employee, the department she works for is known. In this case, \( \mathcal{O} \) does not satisfy Requirement 2 under KS, because, for employee \( \text{Wim} \), we know that a department exists but we do not know its exact identity.

The epistemic approach to ICs was first proposed by Reiter in the context of incomplete databases (Reiter 1992; 1988) and then applied to ontological reasoning in (Donini et al. 1998). Also, in (Calvanese et al. 2007) the authors presented several query languages enhanced by an epistemic operator and suggest the use of queries in these languages as integrity constraints for OBDM systems. In this paper, we investigate this epistemic approach to ICs further, we give a formal definition of what an OBDM system knows about the real world, and we develop a new formalism for ICs based on this notion. The epistemic nature of this language allows us to express ICs that cannot be expressed in other formalisms and distinguish between different readings of the integrity constraints, such as the ones mentioned in the example above. For ICs expressed in this formalism, we study the computational complexity of both checking satisfaction and performing different tasks of static analysis on ICs.

Before concluding this section, we want to stress out that our attempt is not to define the single right way to express integrity constraints in OBDM systems. Defining the acceptable states of information systems is a widespread necessity and the possible different scenarios are many. However, we observe that, if satisfied, ICs under KS do not alter the semantics of OBDM systems. For this reason, we believe that our formalism can be used either as it or as a powerful complement to other forms of constraints.

The remainder of this paper is organized as follows. Section 2 contains preliminary definitions used throughout the paper. Section 3 introduces our framework for ICs in OBDM systems. Section 4 presents computational complexity results relative to this framework. Section 6 concludes the paper.

### 2 Preliminary Definitions

In this section, we briefly review the concepts used in the technical development of this paper. In what follows, we assume basic familiarity with the standard notions of computational complexity and first-order logic and refer the reader to (Arora and Barak 2009) for a detailed account.

**Databases.** We assume relational databases over a countably infinite set of constants \( \Delta \) and refer to (Abiteboul, Hull, and Vianu 1995) for a detailed account on the topic. A database schema \( S \) is a pair \( \langle \Sigma_S, C_S \rangle \) where \( \Sigma_S \) is a relational signature, and \( C_S \) is a set of integrity constraints. A \( \Sigma_S \)-database \( D \) is a relational structure in the signature \( \Sigma_S \); \( D \) is also consistent with \( S \) (alternatively \( S \)-database) if it satisfies all the constraints in \( C_S \), written \( D \models C_S \). In general, integrity constraints take the form of sentences in some well-behaved fragment of first-order logic (FOL). A popular such fragment is Data Dependencies, i.e., formulae \( \forall \bar{x} . \phi(\bar{x}) \rightarrow \exists \bar{y} . \psi(\bar{x}, \bar{y}) \), where \( \phi \) and \( \psi \) are conjunctions of relational and equality atoms. In this paper, we focus on the following widely accepted classes of Data Dependencies: tuple-generating dependencies (tgds), equality-generating dependencies (egds), and denial constraints (dens). In tgds \( \psi \) is a conjunction of only positive relational atoms; in egds \( \psi \) is a conjunction of equality atoms; in dens \( \psi \) is the symbol False (\( \perp \)). Unfortunately, the unrestricted interaction between tgds, egds, and dens leads easily to the undecidability of many fundamental decision problems. To ensure decidability, sets of tgds, egds, and dens are usually subject to some syntactic requirement. A widely-accepted requirement, common in data exchange and integration as well as in data management, is weak-acyclicity (Fagin et al. 2005). For sets of weak-acyclic tgds, egds, and dens, many reasoning tasks are decidable due to the termination of the chase algorithm (Onet 2013).

Besides checking the satisfaction of integrity constraints, a fundamental task to perform with databases is Query Answering. Given a database \( D \) and a query \( q(\bar{x}) \), i.e., a first-order formula with free variables \( \bar{x} \), Query Answering asks for the set \( \text{ans}(q, D) \) of tuples of constants \( \bar{c} \) such that
Ontology-Based Data Management. An OBDM specification $B$ is a triple $(\mathcal{T}, \mathcal{M}, \Sigma)$ where $\mathcal{T}$ is a description logic ontology, $\Sigma$ is a relational signature, and $\mathcal{M}$ is a set of mapping assertions from $\Sigma$ to $\mathcal{T}$, i.e., FOL sentences of the form $\forall \bar{x}. \phi(\bar{x}) \rightarrow \exists \bar{y}. \psi(\bar{x}, \bar{y})$, where $\phi$ and $\psi$ are first-order formulae on $\Sigma$ and on the alphabet of $\mathcal{T}$, respectively. In this paper, we focus on two classes of OBDM specifications: lightweight and trivial. In a lightweight OBDM specification, $\mathcal{T}$ is a DL-Lite$_A$ TBox and $\mathcal{M}$ is a set of conjunctive GAV mapping assertions, i.e., formulae of the form $\forall \bar{x}. \phi(\bar{x}) \rightarrow \psi(\bar{x})$, where both $\phi$ and $\psi$ are conjunctions of atoms (Lenzerini 2002). In trivial OBDM specifications, $\mathcal{T}$ is an ontology on alphabet $\Sigma$ and empty set of axioms, and $\mathcal{M}$ is the identity mapping. Clearly, trivial specifications are also lightweight.

Given an OBDM specification $B = (\mathcal{T}, \mathcal{M}, \Sigma)$ and a $\Sigma$-database $D$, the pair $\mathcal{O} = (B, D)$ is called OBDM system. Semantics of $\mathcal{O}$ are defined as follows: a first-order interpretation $\mathcal{I}$ is a model for the OBDM system $\mathcal{O}$ if $\mathcal{I}$ is a model for $\mathcal{T}$, and the pair $(D, \mathcal{I})$ satisfies the mapping $\mathcal{M}$. As customary, constants from $D$ are assumed to satisfy the Unique Name Assumption, i.e., each constant is equal only to itself. The set of models of an OBDM system $\mathcal{O}$ will be denoted by $\text{Mod}(\mathcal{O})$; if $\text{Mod}(\mathcal{O}) \neq \emptyset$ we will say that $\mathcal{O}$ is satisfiable, unsatisfiable otherwise. If $B$ is lightweight, we call $\mathcal{O} = (B, D)$ lightweight OBDM system. For every lightweight OBDM system $\mathcal{O}$, there exists a first-order sentence $\pi(\mathcal{O})$ such that $\mathcal{I} \in \text{Mod}(\mathcal{O})$ if and only if $\mathcal{I}$ is a model for $\pi(\mathcal{O})$ (Poggi et al. 2008).

In the context of OBDM systems, two fundamental reasoning tasks are Satisfiability and Query Answering. Satisfiability is the problem of checking whether $\text{Mod}(\mathcal{O}) \neq \emptyset$, for input OBDM system $\mathcal{O}$. If $B$ is lightweight, there exists a UCQ with inequalities $v_B$, called violation query for $B$, such that $\mathcal{O}$ is satisfiable if and only if $\text{ans}(v_B, D) = \emptyset$. The query $v_B$ can be computed in time polynomial w.r.t. $B$, so the problem of checking whether $\mathcal{O}$ is satisfiable is in $\text{NP}$, and in $\text{AC}^0$ if $B$ is fixed. Given an OBDM system $\mathcal{O} = (B, D)$ and a conjunctive query $q(\bar{x})$, Query Answering asks for the set $\text{cert}(\mathcal{O})$ of tuples of constants $\bar{c}$ such that the sentence $q(\bar{c})$ is true in every model of $\mathcal{O}$. If $B$ is lightweight, for every $q \in \text{UCQ}$ there exists a UCQ $\text{Rew}(q, B)$ such that $\bar{c} \in \text{cert}(\mathcal{O})$ if and only if $\bar{c} \in \text{ans}(\text{Rew}(q, B), D)$. In general, Query Answering is $\text{NP}$-Complete, and it can be solved in $\text{AC}^0$ if we fix the query and the specification.

Epistemic Logics. In what follows, we use the logic $\mathcal{OL}$ defined in (Lakemeyer and Levesque 2012). Formulae in $\mathcal{OL}$ are defined as follows: atomic formulae and the symbol $\bot$ are in $\mathcal{OL}$, if $\phi$ and $\psi$ are in $\mathcal{OL}$ then so are $\phi \land \psi$, $\phi \lor \psi$, $\neg \phi$, $\exists x. \phi(x)$, $\forall x. \phi(x)$, $\mathcal{K}(\phi)$, and $\mathcal{O}(\phi)$. A formula $\phi \in \mathcal{OL}$ is objective if no occurrence of $\mathcal{K}$ and $\mathcal{O}$ appears in $\phi$. To interpret formulae in $\mathcal{OL}$, we use epistemic interpretations over a domain $\Delta$. An epistemic interpretation is a pair $(\mathcal{W}, w)$ where $W$ is a set of first-order interpretations with domain $\Delta$ and $w$ is an interpretation in $W$. Let $\phi$ be a formula in $\mathcal{OL}$ and let $\mathcal{E} = (W, w)$ be an epistemic interpretation, $\mathcal{E}$ satisfies $\phi$, written $\mathcal{E} \models \phi$, if the following conditions hold:

- $\phi$ is an atomic formula and $w \models \phi$. $D$ satisfies the formula $q(\bar{c})$. Important classes of queries are Conjunctive Queries (CQs), i.e., existentially quantified conjunctions of relational atoms, and Union of Conjunctive Queries, i.e., disjunctions of CQs sharing the same tuple of free variables. Computing $\text{ans}(q, D)$ for $q \in \text{UCQ}$ can be done in $\text{NP}$, and in $\text{AC}^0$ if we fix the query $q$. When dealing with queries, a fundamental question is whether a query $q$ is contained in a query $q'$ under a database schema $\mathcal{S}$, i.e., whether $\text{ans}(q, D)$ is contained in $\text{ans}(q', D)$ for every $\mathcal{S}$-database $D$. In the literature, this problem is called Query Containment Under Constraints. If $\mathcal{S}$ contains only weakly-acyclic tgds, egds, and dans, checking whether an UCQ $q$ is contained in an UCQ $q'$ under $\mathcal{S}$ is $2\text{EXPTIME}$-complete (Cali, Gottlob, and Pieris 2010).

Description Logics Ontologies. An ontology is a conceptualization of a domain of interest expressed in terms of a formal language. Hereafter, we assume ontologies expressed in Description Logics (DLs). A DL knowledge base is a pair $(\mathcal{T}, \mathcal{A})$ 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 logical assertions (called ABox assertions) specifying the instances of concepts and roles. In this paper, we focus on ontologies expressed in DL-Lite$_A$, a member of the DL-Lite family of DLs. In what follows, we give only a brief account on DL-Lite$_A$ and refer the reader to (Poggi et al. 2008) for a thorough introduction. The syntax of concept, roles and attributes expressions in DL-Lite$_A$ is specified by the following grammar, where $\Delta$ are atomic expressions in $\mathcal{T}$, $\mathcal{M}$ are set of models of $\langle \mathcal{T}, \mathcal{A} \rangle$ and denote the set of models of $\langle \mathcal{T}, \mathcal{A} \rangle$ by $\text{Mod}(\langle \mathcal{T}, \mathcal{A} \rangle)$.
Definition 4 formally discussed in the introduction, these semantics are

If \( \phi \) means for \( O \), then \( O \) captures many of the intuitions presented in the introduction, we now present

An epistemic interpretation \( E \) is an epistemic state of an OBDM system \( O \) if \( E \models O(\pi(O)) \).

The set of epistemic states of an OBDM system \( O \) will be denoted by \( \mathcal{E}(O) \). Intuitively, in every epistemic state \( \langle W, w \rangle \) of \( O \), the set \( W \) represents what \( O \) knows about the world while \( w \) represents what \( O \) believes to be true. To define our formalism for integrity constraints we focus on the former and observe that, while the epistemic states of \( O \) may be infinitely many, the knowledge of \( O \) is uniquely defined, as the following proposition shows.

Proposition 1 If \( O \) is a lightweight OBDM system, every pair \( \langle W, w \rangle, \langle W', w' \rangle \in \mathcal{E}(O) \) is such that \( W = W' \).

Proposition 1 is not the only interesting characteristic of knowledge states and, in fact, they capture many of the intuitions discussed in the introduction. To prove this statement, we need to establish a proper framework for ICs in OBDM systems. To this end, we start by augmenting OBDM specifications by a set of \( OL \) formulae that will be interpreted as integrity constraints.

Definition 2 An OBDM specification with constraints \( B_c \) is a tuple \( \langle T, M, \Sigma, C \rangle \) where \( B = \langle T, M, \Sigma \rangle \) is an OBDM specification and \( C \) is a set of \( OL \) sentences.

In what follows, given an OBDM specification with constraints \( B_c \), we denote by \( B \) the underlying OBDM specification (without constraints). Similarly to the standard case, OBDM systems with constraints are pairs \( \langle B_c, D \rangle \) where \( D \) is a database and \( B_c \) is an OBDM specification with constraints. In the following definition, we formalize what it means for \( \langle B_c, D \rangle \) to satisfy the set of constraints \( C \).

Definition 3 An OBDM system with constraints \( \langle B_c, D \rangle \) satisfies \( \chi \in OL \) if \( E \models \chi \) for every \( E \in \mathcal{E}(\langle B, D \rangle) \).

If \( O \) satisfies every \( \chi \in C \), we will say that \( O \) satisfies its constraints. With this notion in place, we are ready to define semantics for OBDM systems with constraints. As informally discussed in the introduction, these semantics are shaped by the ontology and the data and validated by the integrity constraints.

Definition 4 Let \( O = \langle B_c, D \rangle \). The set \( \text{Mod}(O) \) is equal to \( \text{Mod}(\langle B, D \rangle) \) if \( O \) satisfies \( C \), \( \text{Mod}(O) = \emptyset \) otherwise.

If \( \text{Mod}(O) \neq \emptyset \), we say that \( O \) is satisfiable, unsatisfiable otherwise. Definition 4 ensures that the semantics of satisfiable systems are unaltered by the constraints.

With this notion of semantics in place, we can focus on the definition of a fragment of \( OL \) suited to express ICs. From a computational standpoint, the use of a restricted fragment of \( OL \) is necessary, as the following proposition shows.

Proposition 2 Let \( B \) be a trivial OBDM specification and let \( C \) be a set of \( OL \) formulae. It is undecidable to check whether a system \( \langle B_c, D \rangle \) is satisfiable.

To prove the claim, observe that if \( C \) contains an objective formula \( \phi \), then \( O \) is satisfiable if and only if \( \phi \) is true in every model of \( O \). Reduction from Validity follows straightforwardly. Undecidability, however, is not the only reason why only specific fragments of \( OL \) are suitable as ICs. Consider, e.g., Proposition 2. In the informal proof given, we can see how the use of an objective sentence \( \phi \) as integrity constraint for a system \( O \) is equivalent to requiring that \( \phi \) is entailed by \( O \). In turn, this corresponds to assuming for \( \phi \) the Entailment Semantics. With the goal of having a language that is computationally well-behaved and soundly grounded on the intuitions presented in the introduction, we now present Epistemic Dependencies (EDs).

Definition 5 An epistemic dependency is an \( OL \) formula

\[
\forall \bar{x}, \exists \bar{y}, \exists \bar{z}. \bigwedge_i \phi_i(\bar{x}, \bar{z}) \rightarrow \exists \bar{y}. K(\exists \bar{w}. \psi(\bar{x}, \bar{y}, \bar{w}))
\]

where \( \bar{x}, \bar{y}, \bar{z} \) are disjoint tuples of variables, each \( \phi_i \) is a relational atom, and \( \psi \) is either:

- a conjunction of relational atoms,
- a conjunction of equality atoms over \( \bar{x} \), or
- the symbol \( \bot \).

While EDs may not be the only fragment of \( OL \) that can express suitable ICs, we observe that they satisfy many of our desiderata. In the following example, we show how the Knowledge Semantics of all the requirements in Example 3 can be faithfully translated into epistemic dependencies.

Example 4 The following EDs capture Requirement 1 – 3 in Example 3 under Knowledge Semantics.

1. \( \forall x, y, y'. K(\text{hasCode}(x, y) \land \text{hasCode}(x, y')) \rightarrow y = y' \)
2. \( \forall x. K(\text{Emp}(x)) \rightarrow \exists y. K(\text{hasDep}(x, y)) \)
3. \( \forall x. K(\text{Dep}(x)) \rightarrow \exists y. K(\text{hasMan}(x, y)) \)

Consider now ED 1 – 3 in the OBDM system \( O \) defined in Example 1. For each \( E \in \mathcal{E}(O) \), we have:

1. \( E \models K(\text{hasCode}(x, y)) \) if and only if \( x = \text{Bob} \) and \( y = 1 B \), or \( x = \text{Jim} \) and \( y = 2 J \), or \( x = \text{Wim} \) and \( y = 3 W \). Hence, \( O \) satisfies ED 1.
2. \( E \models K(\text{Emp}(\text{Wim})) \) but \( E \nvdash K(\text{hasDep}(\text{Wim}, c)) \) for any constant \( c \). Hence, \( O \) does not satisfy ED 2.
3. \( E \models K(\text{Dep}(x)) \) if and only if \( x = \text{D0} \) or \( x = \text{D1} \), and \( E \models K(\text{hasMan}(x, y)) \) if \( x = \text{D0} \) and \( y = \text{Tim} \), or \( x = \text{D1} \) and \( y = \text{Jim} \). Hence, \( O \) satisfies ED 3.

Example 4, shows how the violation or satisfaction of an epistemic dependency is really grounded on the data contained in the sources. Consider, e.g., ED 2. While the OBDM system \( O \) entails the axiom \( \text{Emp} \subseteq \exists \text{hasDep} \), \( O \) violates
In the reminder of this section, we discuss standard reasoning tasks in the context of OBDM specifications and systems with constraints. The first of these tasks is checking satisfiability, i.e., given an OBDM system with constraints \( O \), check whether \( \text{Mod}(O) \neq \emptyset \). From Definition 4, it follows that checking whether \( \langle B_C, D \rangle \) is satisfiable can be done in two steps: first, check whether \( \langle B, D \rangle \) alone is satisfiable, second, check whether \( \langle B_C, D \rangle \) satisfies its constraints. While the complexity of the former is well-known for many ontological languages, we present the complexity of checking whether a lightweight OBDM system satisfies a set of epistemic dependencies in the following section.

Another important task that can be performed with OBDM systems with constraints is query answering. Definition 4 ensures that, after checking whether a system \( \langle B_C, D \rangle \) is satisfiable, the set of constraints \( C \) does not affect query answering and can be safely disregarded. This observation is formalized in the following proposition.

**Proposition 3** Given a satisfiable OBDM system with constraints \( \langle B_C, D \rangle \), a tuple \( \bar{c} \) is in \( \text{cert}(\phi(\bar{x}), \langle B_C, D \rangle) \) if and only if \( \bar{c} \in \text{cert}(\phi(\bar{x}), \langle B, D \rangle) \).

In light of Proposition 3, in what follows we will not discuss query answering any further. When dealing with standard ICs, an important static analysis task is checking whether a constraint \( \chi \) is redundant with respect to a set of constraints \( C \), i.e., check whether \( \chi \) is satisfied whenever all the constraints in \( C \) are. In OBDM specifications, redundancy can be translated as follows: given an OBDM specification \( B_C \) and an epistemic dependency \( \kappa \), check whether every satisfiable system \( \langle B_C, D \rangle \) also satisfies \( \kappa \). As we will show in the following section, checking redundancy of epistemic dependencies is undecidable even in the case of trivial OBDM specification.

Other interesting static analysis are checking faithfulness and checking protection (Console and Lenzerini 2014a). Faithfulness and protection are defined as follows.

**Definition 6** Let \( B_C = \langle T, M, \Sigma_S, C \rangle \) be an OBDM specification with constraints and let \( S = \langle \Sigma_S, C_S \rangle \) denote a database schema. Then:

- \( S \) is faithful to \( B_C \) if, for every \( \Sigma_S \)-database \( D \), \( D \) is consistent with \( S \) if \( \text{Mod}(\langle B_C, D \rangle) \neq \emptyset \);
- \( S \) protects \( B_C \) if, for every \( \Sigma_S \)-database \( D \), \( \text{Mod}(\langle B_C, D \rangle) \neq \emptyset \) if \( D \) is consistent with \( S \).

Informally, faithfulness and protection are a way to evaluate a given data source schema at the conceptual level, via the OBDM specification. In this context, faithfulness and protection are relevant schema-level measures of the quality of data, as well as useful tools for the OBDM paradigm. For example, when OBDM specifications are paired with data sources that evolve through time, checking satisfiability may become a frequent task. If the source schema protects the specification, however, satisfiability can be safely delegated to the systems managing the data sources (Console and Lenzerini 2014b). In the same spirit, protection and faithfulness can be used as effectual designing tools for data source schemas. The following example illustrates protection.

**Example 5** Consider the OBDM specification \( B \) defined in Example 1, the database schema \( S \) in Example 2, and the epistemic dependencies in Example 4. The schema \( S \) protects \( O \) from ED 3. Informally, this is because the departments known by a system \( O = \langle B, D \rangle \) are those stored in tables \( \text{dep} \) and \( \text{empA} \). For the former, \( O \) always knows at least one manager while, for the latter, it may not. However, due to the integrity constraints in \( S \), the departments stored in table \( \text{empA} \) are a subset of those stored in \( \text{dep} \). In turn, this implies that every known department is associated to a manager that is also known.

### 4 Decidability and Complexity

In this section, we study the complexity of performing the reasoning tasks presented in Section 3. To this end, we start by defining relevant decisions problems.

- **Satisfiability**: given an OBDM system with constraints \( O \), check whether \( \text{Mod}(O) \neq \emptyset \).
- **Protection**: given an OBDM specification with constraints \( B_C \) and a database schema \( S \), check whether \( S \) protects \( B_C \).
- **Constraint Implication**: given an OBDM specification with constraints \( B_C \) and a formula \( \chi \in \bigodot L \), check whether every satisfiable system \( \langle B_C, D \rangle \) satisfies \( \chi \).
- **Faithfulness**: given an OBDM specification with constraints \( B_C \) and a database schema \( S \), check whether \( S \) is faithful to \( B_C \).

For our complexity analysis, we will assume integrity constraints expressed as epistemic dependencies and lightweight OBDM specifications. This assumption will lead to decidability for Protection and even tractability, if the specification is fixed, for Satisfiability. Unfortunately, Constraint Implication and Faithfulness for epistemic dependencies are inherently undecidable. To show that undecidability only depends on the constraints, for Constraint Implication and Faithfulness we will present undecidability results using trivial OBDM specifications.

**Satisfiability.** To present our complexity results, we need to prove several preliminary lemmas. First, we show that the satisfaction of epistemic formulae in OBDM systems is closely related to the problem of computing certain answers to queries. This is formalized in the following lemma.

**Lemma 1** Let \( O \) be a satisfiable OBDM system and let \( \phi(\bar{x}) \) be a objective \( \bigodot L \) formula. The formula \( K(\phi(\bar{c})) \), for some tuple of constants \( \bar{c} \), is true in the knowledge states of \( O \) if and only if \( \bar{c} \in \text{cert}(\phi(\bar{x}), O) \).

Using Lemma 1, we can check whether \( O \) satisfies its constraints by encoding epistemic dependencies as queries.

**Definition 7** Let \( \kappa \) be the epistemic dependency

\[ \forall \bar{x}. K(\exists \bar{z}. \bigwedge_i \phi_i(\bar{x}, \bar{z})) \rightarrow \exists \bar{y}. K(\exists \bar{w}. \psi(\bar{x}, \bar{y}, \bar{w})) \]\

The queries \( b_\kappa(\bar{x}) \) and \( h_\kappa(\bar{x}) \) are defined as follows:
\( b_k(x) = \exists z \land_1 \phi_1(x, z) \)

- If \( \psi \) is a conjunction of relational atoms, then \( h_\kappa(x, y) = \exists \bar{w}, \psi(x, y, \bar{w}) \)
- If \( \psi \) is a conjunction of equality atoms \( \land_1(x_1 = x_2) \), then \( h_\kappa(x) = b_k(x) \land \land_1(x_1 = x_2) \)
- If \( \psi(x) = \bot \), then \( h_\kappa(x) = \bot \)

Given an epistemic dependency, \( b_k^B(x) \) will denote the UCQ \( Rew(b_k(x), B) \) and \( h_k^B(x) \) will denote the UCQ \( \exists \bar{y}. Rew(h_\kappa(x, \bar{y})) \). Due to Lemma 1, \( b_k^B(x) \) and \( h_k^B(x) \) can be used to test the satisfaction of \( \kappa \) over \( B \). We formalize this claim in the following lemma.

**Lemma 2** Let \( B \) be a lightweight OBDM specification. A satisfiable OBDM system with constraints \( \langle B_C, D \rangle \) satisfies an epistemic dependency \( \kappa \) if and only if \( \text{ans}(b_k^B(x), D) \) is contained in \( \text{ans}(h_k^B(x), D) \).

With Lemma 2 in place, we are now ready to prove the complexity of checking satisfiability of lightweight OBDM systems with constraints. We present this result in the following theorem.

**Theorem 1** Let \( B \) be a lightweight OBDM specification and let \( C \) be a set of epistemic dependencies. For input \( \langle B_C, D \rangle \), Satisfiability is \( \Pi_2^B \)-complete. If \( B_C \) is fixed and the only input is \( D \), Satisfiability is in \( AC^0 \).

To prove membership in \( \Pi_2^B \), we can show a nondeterministic algorithm that guesses a tuple \( \bar{c} \) that satisfies a disjunct in \( b_k^B(x) \) and checks whether \( \bar{c} \) does not satisfy any of the disjuncts in \( h_k^B(x) \). For hardness, we can show a reduction from the problem of checking the satisfaction of tgd (Pichler and Skritek 2011). To prove membership in \( AC^0 \), observe that \( \forall \bar{x}. \exists \bar{y}. b_k^B(x) \land h_k^B(x, \bar{y}) \) is a first-order formula that can be evaluated directly over \( D \).

**Protection.** First, we observe that, given a specification \( B_C \), protection can be tested separately over \( B \) and \( C \). To formalize this intuition, we need to introduce some additional notation. Given an OBDM specification \( B_C = \langle T, M, \Sigma_S, C \rangle \) and a database schema \( S \), we say that \( S \) protects \( B_C \) from \( S \) if \( S \) protects the specification \( \langle T, M, \Sigma_S, \emptyset \rangle \). Moreover, we say that \( S \) protects \( B_C \) from \( \chi \in C \) if \( S \) protects the specification \( \langle T, M, \Sigma_S, \{ \chi \} \rangle \). With these definitions in place, we can prove the following.

**Lemma 3** A database schema \( S \) protects an OBDM specification \( B_C \) if and only if \( S \) protects \( B_C \) from \( B \), and \( S \) protects \( B_C \) from \( \chi \), for every \( \chi \in C \).

Lemma 3 will be the basic building block of our algorithm for Protection. Assume a specification \( B_C \), where \( B \) is lightweight and \( C \) is a set of epistemic dependencies. Informally, while a technique to check whether \( S \) protects \( B_C \) from \( B \) is known in the literature (Console and Lenzerini 2014), checking whether \( S \) protects \( B_C \) from \( \kappa \) can be done via query containment. This intuition is formalized in the following lemma.

**Lemma 4** A database schema \( S = \langle \Sigma_S, C_S \rangle \) protects the lightweight OBDM specification \( B_C \) from the epistemic dependency \( \kappa \in C \) if and only if the following hold:
- \( S \) protects \( B_C \) from \( B \), and
- \( \text{ans}(b_k^S(x), D) \subseteq \text{ans}(h_k^S(x), D), \) for every \( D \models C_S \).

With Lemma 4 in place, we can finally prove the complexity of Protection for lightweight OBDM specifications and epistemic dependencies.

**Theorem 2** For input \( B_C \) and \( S \), where \( B \) is a lightweight OBDM specifications, \( C \) is a set of epistemic dependencies, and \( S \) is a database schema with weakly acyclic tgd, egds, and dens, Protection is \( 2EXPTIME \)-complete.

The proof of Theorem 2 relies on the finiteness of the chase for the class of constraints considered.

**Constraint Implication and Faithfulness** As we briefly discussed at the beginning of this section, both Constraint Implication and Faithfulness are undecidable even for the class of OBDM specifications that we called trivial. To prove these claims, we need to introduce some additional notation and some preliminary results. A k-tgd is an epistemic dependency of the form \( \forall \bar{x}. K(\phi(\bar{x})) \rightarrow \exists \bar{y} K(\psi(\bar{x}, \bar{y})) \), where \( \psi \) is a conjunction of relational atoms. Given a k-tgd \( \kappa \), by \( \kappa_\perp \) denote the tgd obtained by removing from \( \kappa \) every occurrence of \( K \), i.e., \( \kappa_\perp = \forall \bar{x}. \phi(\bar{x}) \rightarrow \exists \bar{y} \psi(\bar{x}, \bar{y}) \). Intuitively, \( \kappa_\perp \) can be used to simulate \( \kappa_{\perp, \perp} \), as the following lemma shows.

**Lemma 5** Let \( B \) be a trivial OBDM specification and let \( C \) be a set of k-tgds. The OBDM system \( \langle B_C, D \rangle \) satisfies its constraints if and only if \( D \) satisfies \( \kappa_{\perp, \perp} \) for every \( \kappa \in C \).

Informally, Lemma 5 shows that epistemic dependencies can simulate binary database dependencies. To prove undecidability of Constraint Implication, we can use this result to show a reduction from the Finite Implication problem, i.e., given a schema \( S = \langle \Sigma_S, C_S \rangle \) and a tgd \( \tau \), check whether every \( S \)-database satisfies \( \tau \). Undecidability of Finite Implication was first proved in (Vardi 1981) even for binary tuple-generating dependencies. With this observation in place, we are ready to prove the following.

**Theorem 3** Constraint Implication is undecidable for trivial OBDM specifications and epistemic dependencies.

To prove Theorem 3, assume a binary tgd \( \kappa_{\perp, \perp} \), and let \( \kappa \) be the associated k-tgd. Given an instance of Finite Implication \( \langle S, \kappa_{\perp, \perp} \rangle \), we can construct a specification \( B_C \), where \( B \) is trivial, such that \( B_C \) implies \( \kappa \) if and only if \( S \) finally entails \( \kappa_{\perp, \perp} \). A similar construction proves the following.

**Theorem 4** With input \( B_C \) and \( \tau \), where \( B \) is a trivial OBDM specification, \( C \) is a set of epistemic dependencies, and \( \tau \) is a tgd, Faithfulness is undecidable.

## 5 Complexity of Other Semantics

Handling EDs under KS is an inherently complex task and one may wonder whether other semantics may yield better results. Unfortunately, the situation does not change dramatically in both ES and MHMS. Satisfiability under ES amounts to checking validity of first-order formulae, and therefore it is often undecidable. The situation improves only marginally under MHMS. A non-elementary upperbound for Satisfiability under MHMS is established in (Motik, Horrocks, and Sattler 2009) for a very expressive
ontological language (\textit{ACCHT}) and arbitrary first-order formulae as constraints. In (Nikolaou et al. 2019), the authors study a restriction of \textit{DL-Lite}_R without negative axioms and disjunctive tgds as constraints. In this setting, Satisfiability is \Pi_2^p-complete and \textit{LOGSPACE} if the specification is fixed. The same paper studies Protection and Faithfulness under MHMS. For Protection, the authors establish decidability in the case of source constraints for which Finite Entailment is decidable. For Faithfulness, they establish decidability for ontologies consisting of full \textit{DL-Lite}_R axioms and restricted mapping, source constraints, and ontology constraints. All upper-bound presented, however, are superpolynomial.

6 Conclusions

We provided a framework for integrity constraints in OBDM systems based on the notion of what such systems know and should know about the real world. In this framework, we defined a language for constraints and studied the complexity of satisfaction and different forms of static analysis. As future directions, we would like to study the decidability and complexity of different languages for ontologies, mappings, and constraints.

7 Acknowledgements

This work was partly supported by EPSRC (grants M025268 and N023056) and by MIUR (PRIN 2017 project “HOPE”).

References


