# Monotone Abstractions in Ontology-based Data Management

**Gianluca Cima,**[1] **Marco Console,**[2] **Maurizio Lenzerini,**[2] **Antonella Poggi**[2]

[1]CNRS & University of Bordeaux
[2]Sapienza University of Rome
gianluca.cima@u-bordeaux.fr, {console, lenzerini, poggi}@diag.uniroma1.it,

**Abstract**

In Ontology-based Data Management (OBDM), an abstraction of a source query q is a query over the ontology capturing the semantics of q in terms of the concepts and the relations available in the ontology. Since a perfect characterisation of a source query may not exist, the notions of best sound and complete approximations of an abstraction have been introduced and studied in the typical OBDM context, i.e., in the case where the ontology is expressed in DL-Lite, and source queries are expressed as unions of conjunctive queries (UCQs). Interestingly, if we restrict our attention to abstractions expressed as UCQs, even best approximations of abstractions are not guaranteed to exist. Thus, a natural question to ask is whether such limitations affect even larger classes of queries. In this paper, we answer this fundamental question for an essential class of queries, namely the class of monotone queries. We define a monotone query language based on disjunctive Datalog enriched with an epistemic operator, and show that its expressive power suffices for expressing the best approximations of monotone abstractions of UCQs.

## 1 Introduction

In *Ontology-based Data Management (OBDM)* (Poggi et al. 2008), an ontology, i.e., a formal, logic-based representation of a domain of interest, is used to provide a high-level conceptual tool for accessing and managing the data sources of an information system. Suitable mappings declaratively specify the relationship between the data at the sources and the elements in the ontology, and this enables the user to carry out many relevant tasks on data through the lens of the ontology (Lenzerini 2018; De Giacomo et al. 2018).

In the last decade, there have been extensive investigations on the problem of answering queries expressed over the ontology by means of algorithms that, taking into account both the ontology and the mapping, produce suitable queries that extract the correct data from the sources. The problem is complicated by the need of reasoning about all components of the system, so as not to miss any answer logically implied by the axioms expressing both the ontology and the mapping (see the surveys in (Bienvenu 2016; Xiao et al. 2018; Ortiz 2018)).

Recent papers (Cima 2017; Lutz, Marti, and Sabellek 2018; Cima, Lenzerini, and Poggi 2019, 2020b; Cima et al. 2021) address a different issue in OBDM: starting from a query $q_S$ expressed over the sources, the goal is to find a so-called *abstraction* of $q_S$ (Cima 2020), i.e., an ontology-based characterization of $q_S$ expressed in terms of the ontology elements, and whose answers coincide with the answer to the original query, modulo the ontology and the mapping. We encountered the need of abstraction during a joint project with a public statistical research institute. The institute's departments must publish subsets of the data they gather in the form of semantically described linked open data. To compute the content of the datasets the departments execute suitable queries over the data sources mapped to a shared ontology. Notably, when the dataset is published, it must be documented through a SPARQL query expressed in terms of the ontology. This task is currently done manually. The notion of abstraction perfectly captures this scenario and provides the formal tool for automating the process: given the query over the sources computing the content of the dataset, the abstraction of such query with respect to the mapping and the ontology is exactly the SPARQL query to be associated to the open dataset. Besides the above use case, abstraction can be the appropriate tool in various scenarios. For additional insights we refer to the references mentioned above.

The first investigations on abstraction appear in (Cima 2017; Lutz, Marti, and Sabellek 2018). Both papers point out that the "perfect" abstraction of a union of conjunctive queries (UCQ) expressed over the data source not always exists, and present algorithms for computing such abstraction in the case where it both exists, and can be expressed as a UCQ over the ontology. In (Cima 2017; Cima, Lenzerini, and Poggi 2019) the notion of (sound and complete) approximations of the perfect abstraction is introduced, exactly to cope with situations in which perfectness cannot be achieved. Moreover, both papers make it clear that, for a given class of queries $C$, one is probably interested in two specific forms of approximations, called *$C$-minimally complete* and *$C$-maximally sound* abstractions. Based on these notions, (Cima, Lenzerini, and Poggi 2019) presents a thorough analysis of the verification problem (check whether a given query is a complete or sound abstraction) and the computation problem of UCQ-minimally complete and UCQ-maximally sound abstractions of UCQ source queries in

OBDM systems based on *DL-Lite*. In (Cima, Lenzerini, and Poggi 2020b) the computation problem is studied in the context of a specific class of non-monotone queries for expressing abstractions, and it is shown that this class can provide abstractions that are better than the one in the UCQ class.

Thus, with the exception of (Cima, Lenzerini, and Poggi 2020b), all the results on abstractions have been obtained under the assumptions that abstractions are expressed as UCQs over the ontology, and many of them originate from the observations that best approximations in the UCQ class are not guaranteed to exist. Thus, a natural issue to investigate is whether such limitations affect even larger classes of queries for expressing abstractions. The main goal of this paper is to address the following question: do approximations of perfect abstraction that are best in a fundamental class of queries, namely the class of monotone queries, always exist? Obviously, a related goal is to derive algorithms for computing approximations of abstractions that are best in the class of monotone queries, if they exist. Note that the class of monotone queries includes queries expressible in First-Order Logic and is therefore extremely important.

In this paper we answer positively to the above-mentioned fundamental question. More specifically, we provide the following contributions.

1. We present a general framework for abstraction in OBDM, based on the definition of queries as functions from the logical models of OBDM systems to sets of tuples. This is important for the goal of making our investigation, and in particular the notion of monotone queries, to be independent from any particular syntactic form. The usual concept of queries whose semantics is based on certain answers is a special case of our general notion.
2. The framework includes a new monotone query language, called $\mathsf{Datalog}^{\vee}_{\mathbf{K}}$, based on disjunctive Datalog enriched with inequalities and an epistemic operator.
3. We consider a scenario where the OBDM specification $J$ is based on *DL-Lite*$_{RDFS}$ (i.e., the fragment of `RDFS` expressible in Description Logic), and present algorithms that, given a source query $q_{\mathcal{S}}$ expressed as UCQ, compute the best (sound or complete) approximations of the $J$-abstraction of $q_{\mathcal{S}}$ in the class of monotone queries, expressed in $\mathsf{Datalog}^{\vee}_{\mathbf{K}}$.
4. The above algorithms provide the proof that, in the considered scenario, for any UCQ $q_{\mathcal{S}}$, the best (sound or complete) approximations of the $J$-abstraction of $q_{\mathcal{S}}$ in the class of monotone queries always exists. As a consequence, if the perfect abstraction exists and is in the class of monotone queries, then it can be expressed in $\mathsf{Datalog}^{\vee}_{\mathbf{K}}$.

The paper is organized as follows. In Section 2 we recall some preliminary notions. In Section 3 we describe the formal framework for abstraction, including $\mathsf{Datalog}^{\vee}_{\mathbf{K}}$. Starting from Section 4, we focus on the above-mentioned scenario, and present the algorithms for computing the best complete (Section 4) and the best sound (Section 5) abstractions of source UCQs. In Section 6 we discuss perfect monotone abstractions. Finally, Section 7 concludes the paper by mentioning future works in the context of abstraction.

## 2 Preliminaries

A relational schema $R$ is a finite set of predicate symbols (each one with a fixed arity) and an $R$-database $D$ is a finite set of facts using predicate symbols in $R$ and constants from a countable infinite set $Const$. A query $q$ of arity $n$ over a schema $R$ is a function mapping each $R$-database to $n$-tuples of constants occurring in $Const$. We denote by $q^D$ the set of tuples obtained by applying $q$ to the $R$-database $D$. For two queries $q_1$ and $q_2$ over a schema $\mathcal{S}$, we write $q_1 \sqsubseteq_R q_2$ if $q_1^D \subseteq q_2^D$ for each $R$-database $D$.

Let $Var$ be a countable infinite set of symbols, called variables, disjoint from $Const$. A relational atom over $R$ is an expression constituted by a predicate symbol in $R$ applied to as many arguments (constants or variables) as its arity. We also consider inequality atoms, where the predicate symbol is inequality ($\neq$). For a set of atoms $\mathcal{C}$, we let $Dom(\mathcal{C}) = Var(\mathcal{C}) \cup Const(\mathcal{C})$, where $Var(\mathcal{C})$ and $Const(\mathcal{C})$ are the set of all variables and all constants, respectively, occurring in $\mathcal{C}$.

A *conjunctive query (CQ)* $q$ over a schema $R$ is a query expressible as $\{\bar{x} \mid \exists \bar{y}.\phi(\bar{x}, \bar{y})\}$, where $\bar{x}$ is a tuple of variables, called the *distinguished variables* of $q$, $\bar{y}$ is a tuple of variables, called the *existential variables* of $q$, and $\phi(\bar{x}, \bar{y})$ is the *body* of $q$, i.e., a finite conjunction of relational atoms over $R$ containing all the distinguished and existential variables of $q$. The arity of $q$ is the arity of $\bar{x}$. A *union of conjunctive queries (UCQ)* is a finite union of CQs with same arity, called its *disjuncts*. A (U)CQ with inequality ((U)CQ$^{\neq}$) is simply a (U)CQ allowing inequality atoms in its body.

Let $\mathcal{C}_1$ and $\mathcal{C}_2$ be two sets of atoms. A *homomorphism* $h$ from $\mathcal{C}_1$ to $\mathcal{C}_2$ (written $h : \mathcal{C}_1 \to \mathcal{C}_2$) is a mapping from $Dom(\mathcal{C}_1)$ to $Dom(\mathcal{C}_2)$ such that: (*i*) $h(\alpha) \in \mathcal{C}_2$, for each relational atom $\alpha \in \mathcal{C}_1$; (*ii*) $h(c) = c$, for each constant $c \in Const(\mathcal{C}_1)$; and (*iii*) $h(x) \neq h(y)$, for each inequality atom $x \neq y$ occurring in $\mathcal{C}_1$.

Given a signature of unary and binary predicates, a *Description Logic (DL)* ontology is simply a TBox (set of axioms) expressed in a specific DL (Baader et al. 2003). In the technical sections, we are interested in ontologies expressed in *DL-Lite*$_{RDFS}$ (Cuenca Grau 2004; Rosati 2007; Cima, Lenzerini, and Poggi 2020a). Such language corresponds to the fragment of `RDFS` (Brickley and Guha 2014) expressible in *DL-Lite*. Specifically, a *DL-Lite*$_{RDFS}$ ontology is a finite set of assertions of the form $B \sqsubseteq A$ or $R_1 \sqsubseteq R_2$, where $B$ denotes a basic concept, i.e., an expression of the form $A$, $\exists P$, or $\exists P^-$, with $A$ and $P$ denoting an atomic concept and an atomic role, respectively, and $R_1$ and $R_2$ denote basic roles, i.e., expressions of the form $P$, or $P^-$.

An *OBDM specification* $J$ is a triple $\langle \mathcal{O}, \mathcal{S}, \mathcal{M} \rangle$, where $\mathcal{O}$ is a DL ontology, $\mathcal{S}$ is a relational schema, called the source schema of $J$, and $\mathcal{M}$ is a *Global-and-Local-As-View (GLAV)* mapping, i.e., a set of GLAV mapping assertions of the form:

$$\forall \bar{x}.\forall \bar{y}.\phi(\bar{x}, \bar{y}) \to \exists \bar{z}.\psi(\bar{x}, \bar{z}),$$

where $\bar{x}$ are called the *frontier variables* of $m$, $\phi(\bar{x}, \bar{y})$, called the *body* of $m$, is a conjunction of relational atoms over $\mathcal{S}$, and $\psi(\bar{x}, \bar{z})$, called the *head* of $m$, is a conjunction of atoms over $\mathcal{O}$.

An *OBDM system* is a pair $\Sigma = \langle J, D\rangle$, where $J = \langle \mathcal{O}, \mathcal{S}, \mathcal{M}\rangle$ is an OBDM specification, and $D$ is an $\mathcal{S}$-database. For a mapping $\mathcal{M}$ relating a source schema $\mathcal{S}$ to an ontology $\mathcal{O}$ and an $\mathcal{S}$-database $D$, we denote by $\mathcal{M}(D)$ the set of atoms over $\mathcal{O}$ obtained by *chasing* $D$ with respect to $\mathcal{M}$ as described in (Fagin et al. 2005).

The semantics of an OBDM system is given in terms of interpretations for $\mathcal{O}$ following the so-called *standard names assumption* (Calvanese et al. 2007a). This means that the domain of every interpretation is $Const$, with every constant in the signature interpreted as itself. Note that this also implies the *unique name assumption*. We say that an interpretation $\mathcal{I}$ for $\mathcal{O}$ is a model of an OBDM system $\Sigma = \langle\langle \mathcal{O}, \mathcal{S}, \mathcal{M}\rangle, D\rangle$ if (*i*) $\mathcal{I} \models \mathcal{O}$, and (*ii*) $(D, \mathcal{I}) \models \mathcal{M}$, where $(D, \mathcal{I})$ is the interpretation obtained from $\mathcal{I}$ by adding the set of facts corresponding to $D$. We denote by $mod(\langle J, D\rangle)$ the set of models of an OBDM system $\langle J, D\rangle$, and when $mod(\langle J, D\rangle) \neq \emptyset$ we say that $D$ is *consistent* with $J$. We next provide an example of OBDM specification.

**Example 1.** *Let $J = \langle \mathcal{O}, \mathcal{S}, \mathcal{M}\rangle$ be an OBDM specification such that $\mathcal{O} = \emptyset$, $\mathcal{S} = \{s_1/2, s_2/1, s_3/1\}$ and $\mathcal{M} = \{\boldsymbol{m}_1, \boldsymbol{m}_2, \boldsymbol{m}_3, \boldsymbol{m}_4\}$, where:*

$$\begin{array}{lll}
\boldsymbol{m}_1: & \forall x_1, x_2. s_1(x_1, x_2) & \rightarrow \quad \mathsf{E}(x_1, x_2) \\
\boldsymbol{m}_2: & \forall x. s_1(x, x) & \rightarrow \quad \mathsf{SN}(x) \\
\boldsymbol{m}_3: & \forall x. s_2(x) & \rightarrow \quad \exists z. \mathsf{E}(x, z) \\
\boldsymbol{m}_4: & \forall x. s_3(x) & \rightarrow \quad \mathsf{E}(x, x)
\end{array}$$

*The mapping $\mathcal{M}$ of $J$ establishes how predicates in the schema $\mathcal{S}$ relate to the ontology predicates* $\mathsf{E}$ *(which stands for edge) and* $\mathsf{SN}$ *(which stands for special node).* $\triangle$

A query $q$ of arity $n$ over the signature of an ontology $\mathcal{O}$ is a function whose domain is the power set of the set of all the interpretations for the ontology, and whose range is the power set of the set of all $n$-tuples of constants in $Const$. A query $q$ over $J = \langle \mathcal{O}, \mathcal{S}, \mathcal{M}\rangle$ is a query over the signature of the ontology $\mathcal{O}$ that, for every $\mathcal{S}$-database $D$, associates to $mod(\langle J, D\rangle)$ a set of tuples of constants. The answers to $q$ with respect to the OBDM system $\langle J, D\rangle$, denoted as $q^{J,D}$, is obtained by applying the function $q$ to the set of models $mod(\langle J, D\rangle)$. Note that, although our framework is general enough to accept any kind of function as query, a very common method for defining $q^{J,D}$ when $q$ is expressed as a (U)CQ is by sanctioning that the function $q$ applied to $mod(\langle J, D\rangle)$ returns the set of *certain answers* to the conjunctive query, where a certain answer is a tuple of constants in $Const$ that, once assigned to the distinguished variables of $q$, satisfies the body of $q$ in every model in $mod(\langle J, D\rangle)$.

For an OBDM specification $J = \langle \mathcal{O}, \mathcal{S}, \mathcal{M}\rangle$, an $\mathcal{S}$-database $D$, and a query $q$ of arity $n$ over $J$, with a slight abuse of notation, we denote by $Dom(J, D, q)$ the set of all constants occurring in $J$, $D$, and $q$. Moreover, given a set of constants $\Lambda \subseteq Const$, we denote by $q^{J,D}_{|\Lambda}$ the answers to $q$ w.r.t. $\langle J, D\rangle$ *restricted to the constants of* $\Lambda$, i.e., $q^{J,D}_{|\Lambda} = q^{J,D} \cap \Lambda^n$.

The class of monotone queries is particularly important in this paper. In the context of an OBDM specification $J = \langle \mathcal{O}, \mathcal{S}, \mathcal{M}\rangle$, a query $q$ over $J$ is *monotone* if, for every pairs of $\mathcal{S}$-databases $D, D'$, $mod(\langle J, D\rangle) \subseteq mod(\langle J, D'\rangle)$ implies $q^{J,D'}_{\Lambda(D,D')} \subseteq q^{J,D}_{\Lambda(D,D')}$, where $\Lambda(D, D') = Dom(J, D, q) \cap Dom(J, D', q)$. This captures the usual intuition of monotonicity of queries in logic: when the knowledge possessed by the formal system increases (i.e., the set of models decreases), the answers to a query increase. Note that the restriction on the constants occurring in both the *active domains* of $D$ and $D'$ (other than in $J$ and $q$) is imposed for ensuring that the property of monotonicity looks only for those constants occurring in both databases. We use $\mathfrak{M}^J$ to denote the class of all monotone queries in the context of $J$, and when $J$ is understood, we simply use $\mathfrak{M}$. Note that, for each OBDM specification $J$, $\mathfrak{M}^J$ includes the whole class of first-order queries evaluated under the certain answer semantics, that is arguably the most studied class in both the KR and the DB literature.

For two queries $q_1$ and $q_2$ over $J$, we write $q_1 \sqsubseteq_J q_2$ if $q_1^{J,D} \subseteq q_2^{J,D}$ for each $\mathcal{S}$-database $D$, and write $q_1 \sqsubset_J q_2$ if $q_1 \sqsubseteq_J q_2$ and $q_1^{J,D} \subset q_2^{J,D}$ for at least one $\mathcal{S}$-database $D$. We say that $q_1$ and $q_2$ are $J$-equivalent if both $q_1 \sqsubseteq_J q_2$ and $q_2 \sqsubseteq_J q_1$ hold.

## 3 Framework

In what follows, we implicitly refer to an OBDM specification $J = \langle \mathcal{O}, \mathcal{S}, \mathcal{M}\rangle$, and when we denote a query by $q_{\mathcal{O}}$ (resp., $q_{\mathcal{S}}$) we mean that the query is a query for $J$ (resp., a source query), i.e., is over the signature of the ontology $\mathcal{O}$ (resp., the schema $\mathcal{S}$). We follow (Cima, Lenzerini, and Poggi 2019) for the basic definitions related to abstraction.

We say that $q_{\mathcal{O}}$ is a *perfect $J$-abstraction* of $q_{\mathcal{S}}$ if $q_{\mathcal{O}}^{J,D} = q_{\mathcal{S}}^{D}$, for each $\mathcal{S}$-database $D$ consistent with $J$.

As the condition for an ontology query to be a perfect abstraction of a source query is a strong one, it might be very well the case that a perfect abstraction of a source query does not exist. It is then reasonable to consider weaker notions, such as sound or complete approximations, of perfectness.

We say that $q_{\mathcal{O}}$ is a *complete (resp. sound) $J$-abstraction* of $q_{\mathcal{S}}$ if $q_{\mathcal{S}}^{D} \subseteq q_{\mathcal{O}}^{J,D}$ (resp. $q_{\mathcal{O}}^{J,D} \subseteq q_{\mathcal{S}}^{D}$), for each $\mathcal{S}$-database $D$ consistent with $J$.

Obviously, we might be interested in complete or sound abstractions that approximate $q_{\mathcal{S}}$ at best, at least in the context of a specific class of queries. If $\mathcal{L}_{\mathcal{O}}$ is a class of queries, we say that a query $q_{\mathcal{O}} \in \mathcal{L}_{\mathcal{O}}$ is an *$\mathcal{L}_{\mathcal{O}}$-minimally complete* (resp., *$\mathcal{L}_{\mathcal{O}}$-maximally sound*) $J$-abstraction of $q_{\mathcal{S}}$ if $q_{\mathcal{O}}$ is a complete (resp., sound) $J$-abstraction of $q_{\mathcal{S}}$ and there is no query $q'_{\mathcal{O}} \in \mathcal{L}_{\mathcal{O}}$ such that $q'_{\mathcal{O}}$ is a complete (resp., sound) $J$-abstraction of $q_{\mathcal{S}}$ and $q'_{\mathcal{O}} \sqsubset_J q_{\mathcal{O}}$ (resp., $q_{\mathcal{O}} \sqsubset_J q'_{\mathcal{O}}$).

We now introduce a new language, called $\mathsf{Datalog}^{\vee}_{\mathbf{K}}$, for expressing queries over OBDM specifications. The language is based on disjunctive Datalog, and is used in this paper for expressing abstractions. The basic component of a $\mathsf{Datalog}^{\vee}_{\mathbf{K}}$ query is a rule. Assume two disjoint and countably infinite sets of predicates $\mathcal{E}$ and $\mathcal{P}$, called extensional and intensional, respectively. A $\mathsf{Datalog}^{\vee}_{\mathbf{K}}$ rule has one of the following forms:

- The typical form of disjunctive Datalog, i.e.,

$$\gamma(\bar{x}) \rightarrow \alpha_1(\bar{x}_1) \vee \ldots \vee \alpha_n(\bar{x}_n) \qquad (1)$$

where $\gamma(\bar{x})$ is a conjunction of relational atoms on the predicates of $\mathcal{P}$ with $\vec{x}$ as variables, and for $i = 1, \dots, n$, $\alpha_i(\bar{x}_i)$ is a single relational atom whose predicate is in $\mathcal{P}$ such that $\bar{x}_i \subseteq \bar{x}$,

- A new form specified as follows

$$\mathbf{K}(\exists \bar{z}.\phi(\bar{x}, \bar{z}) \wedge \xi(\bar{x})) \rightarrow \psi_1(\bar{x}_1) \vee \ldots \vee \psi_n(\bar{x}_n) \quad (2)$$

where $\phi$ is a conjunction of relational atoms over $\mathcal{E}$, $\xi(\bar{x})$ is a conjunction of inequality atoms involving only variables from $\bar{x}$ and for $j = 1, \dots, n$, $\psi_j = \exists \bar{y}_j.\gamma_j(\bar{x}_j, \bar{y}_j)$, where $\gamma_j$ is a conjunction of relational atoms on $\mathcal{P}$. When $\bar{x}_j$ contains only variables $\bar{x}$ occurring in $\phi(\bar{x}, \bar{z})$, for $j = 1, \dots, n$, we say that the rule is *safe*.

An $n$-ary $\mathsf{Datalog}^{\vee}_{\mathbf{K}}$ query $q_{\mathcal{O}}$ over an OBDM specification $J$ is a finite set of $\mathsf{Datalog}^{\vee}_{\mathbf{K}}$ rules whose extensional predicates coincide with the alphabet of $\mathcal{O}$, and whose intensional predicates include a special $n$-ary predicate $Ans$. We say that $q_{\mathcal{O}}$ is *safe* if all of its rules are safe. The semantics of $q_{\mathcal{O}}$ is provided relative to an OBDM system. Given an OBDM system $\langle J, D\rangle$, an interpretation for $q_{\mathcal{O}}$ is a pair $I = (mod(\langle J, D\rangle), f)$, where $f$ is a first-order interpretation (with domain $Const$) for the predicates in $\mathcal{P}$. As usual, we may also see $f$ as the set of facts $\{p(\bar{c}) \mid \bar{c} \in p^f\}$. We now define when $I$ satisfies a $\mathsf{Datalog}^{\vee}_{\mathbf{K}}$ rule.

- $I$ satisfies a rule of the form (1) if the first-order formula $\forall \bar{x}.\gamma(\bar{x}) \rightarrow \alpha_1(\bar{x}_1) \vee \ldots \vee \alpha_n(\bar{x}_n)$ is true in $f$,
- $I$ satisfies a rule of the form (2) if for all tuples $\bar{c}$ of constants in $Const$, the fact that the first-order formula $\exists \bar{z}.\phi(\bar{c}, \bar{z}) \wedge \xi(\bar{c})$ is satisfied by every model in $mod(\langle J, D\rangle)$ implies that $f$ satisfies the first-order formula $\exists \bar{y}_j.\gamma_j(\bar{c}_j, \bar{y}_j)$, for some $j = 1, \dots, n$. Observe that, if the rule is unsafe, $\bar{c}_j$ may contain constants of $Const$ that do not occur in $\bar{c}$.

An interpretation $I$ for $q_{\mathcal{O}}$ is called a *model* of $q_{\mathcal{O}}$ if all the rules of $q_{\mathcal{O}}$ are satisfied by $I$. Finally, we define the notion of answers to an $n$-ary $\mathsf{Datalog}^{\vee}_{\mathbf{K}}$ query $q_{\mathcal{O}}$ w.r.t. an OBDM system $\langle J, D\rangle$, denoted by $q_{\mathcal{O}}^{J,D}$, as follows: $\{\bar{c} \in Dom(J, D, q_{\mathcal{O}})^n \mid \bar{c} \in Ans^f$ for each model $(mod(\langle J, D\rangle), f)$ of $q_{\mathcal{O}}\}$.

**Example 2.** *Consider the OBDM specification $J$ illustrated in Example 1. The following safe* $\mathsf{Datalog}^{\vee}_{\mathbf{K}}$ *query $q_{\mathcal{O}}$ over $J$ returns all the pairs $(v_1, v_2)$ of special nodes that are* known *to be distinct and such that there is a path from $v_1$ to $v_2$ passing only for nodes* known *to be special:*

$$\begin{array}{lcl}
\mathbf{K}(\mathsf{E}(x_1, x_2) \wedge \mathsf{SN}(x_1) \wedge \mathsf{SN}(x_2)) & \rightarrow & T_1(x_1, x_2) \\
\mathbf{K}(\mathsf{SN}(x_1) \wedge \mathsf{SN}(x_2) \wedge x_1 \neq x_2) & \rightarrow & T_2(x_1, x_2) \\
T_1(x_1, y) \wedge T_1(y, x_2) & \rightarrow & T_1(x_1, x_2) \\
T_1(x_1, x_2) \wedge T_2(x_1, x_2) & \rightarrow & Ans(x_1, x_2) \quad \triangle
\end{array}$$

The following proposition shows that $\mathsf{Datalog}^{\vee}_{\mathbf{K}}$ is a monotone query language.

**Proposition 1.** *Every* $\mathsf{Datalog}^{\vee}_{\mathbf{K}}$ *query over $J$ is in $\mathfrak{M}^J$, for every OBDM specification $J$.*

The semantics should make it clear that $\mathbf{K}$ is the knowledge operator in the S5 epistemic logic: the formula $\mathbf{K}A$ should be read as "$A$ is known (i.e., logically implied) by the system" (Levesque and Lakemeyer 2001). Therefore, when accessing the information modeled by $\langle J, D\rangle$, a $\mathsf{Datalog}^{\vee}_{\mathbf{K}}$ query extracts what is known by the system, and this characteristic is crucial for not falling into undecidability resulting from using Datalog rules jointly with Description Logics (see (Levy and Rousset 1998; Calvanese and Rosati 2003)), as stated in the following proposition.

**Proposition 2.** *Let $\Sigma$ be an OBDM system. Answering safe* $\mathsf{Datalog}^{\vee}_{\mathbf{K}}$ *queries w.r.t. $\Sigma$ is decidable if and only if answering CQs w.r.t. $\Sigma$ is decidable*[1].

Although our framework is general enough to consider any DL for expressing ontologies and any query language for expressing source queries, in the rest of this paper we will carry out our investigation in the following setting: $(i)$ ontologies are expressed in *DL-Lite*$_{RDFS}$, and $(ii)$ source queries are expressed as UCQs.

At this point, one may wonder whether $\mathsf{Datalog}^{\vee}_{\mathbf{K}}$ is the right language to express monotone abstractions in this setting. While a thorough analysis of the language is outside the scope of the present paper, the following proposition provides a positive answer to this question, at least from the computational point of view.

**Proposition 3.** *In our setting, $(i)$ answering safe* $\mathsf{Datalog}^{\vee}_{\mathbf{K}}$ *queries is in coNP in data complexity, and $(ii)$ there exists an OBDM specification $J$ and a CQ $q_{\mathcal{S}}$ such that, given an $\mathcal{S}$-database $D$, answering the $\mathfrak{M}$-maximally sound $J$-abstraction of $q_{\mathcal{S}}$ is coNP-hard in data complexity.*

To ease the presentation, from now on we assume that mappings and source queries do not mention constants. However, all our results can be straightforwardly adapted to the case where constants are allowed.

## 4 Minimally Complete Abstractions

We now prove that $\mathfrak{M}$-minimally complete $J$-abstractions of UCQs always exist and can be expressed in $\mathsf{Datalog}^{\vee}_{\mathbf{K}}$. Actually, a fragment of $\mathsf{Datalog}^{\vee}_{\mathbf{K}}$ suffices for this purpose, namely the one using only rules of form (2) without disjunction. We start with some useful subroutines.

Given a CQ $q_{\mathcal{S}} = \{\bar{x} \mid \exists \bar{y}.\phi(\bar{x}, \bar{y})\}$, the subroutine $\mathsf{SaturateQ}(q_{\mathcal{S}})$ computes a $\mathsf{UCQ}^{\neq}$ in the following way: for each possible unifier $\mu$ on the variables in $\bar{x} \cup \bar{y}$ such that $\mu(x) \in \bar{x}$ for each $x \in \bar{x}$, $\mathsf{SaturateQ}(q_{\mathcal{S}})$ contains a query obtained from $\mu(q_{\mathcal{S}})$ by adding the inequality atom $t_1 \neq t_2$ for each pair of distinct variables $t_1, t_2$ occurring in $\mu(q_{\mathcal{S}})$. For a UCQ $q_{\mathcal{S}}$, we denote by $\mathsf{SaturateQ}(q_{\mathcal{S}})$ the $\mathsf{UCQ}^{\neq}$ obtained by applying $\mathsf{SaturateQ}(q)$ to each disjunct $q$ of $q_{\mathcal{S}}$. We write each $\mathsf{CQ}^{\neq}$ $q$ generated by $\mathsf{SaturateQ}(q_{\mathcal{S}})$ as $q = \{\bar{x} \mid \exists \bar{y}.\phi(\bar{x}, \bar{y}) \wedge \xi(\bar{x}, \bar{y})\}$, where $\phi(\bar{x}, \bar{y})$ and $\xi(\bar{x}, \bar{y})$ are the conjunctions of the relational atoms over $\mathcal{S}$ and of the inequality atoms, respectively, occurring in the body of $q$.

Moreover, for an OBDM specification $J = \langle \mathcal{O}, \mathcal{S}, \mathcal{M}\rangle$ and a $\mathsf{CQ}^{\neq}$ $q = \{\bar{x} \mid \exists \bar{y}.\phi(\bar{x}, \bar{y}) \wedge \xi(\bar{x}, \bar{y})\}$ over $\mathcal{S}$, we denote by $r_q$ the following $\mathsf{Datalog}^{\vee}_{\mathbf{K}}$ rule of form (2):

$$r_q = \mathbf{K}(\exists \bar{z}.\mathcal{M}(q) \wedge \xi(\bar{x}, \bar{y})) \rightarrow Ans(\bar{x}),$$

[1]With answering we implicitly refer to the associated *recognition problem*, i.e., check whether a tuple is in the answer to a query.

where (*i*) $\mathcal{M}(q)$ is computed by simply ignoring the inequality atoms and chasing the set of relational atoms occurring in the body of $q$; (*ii*) $\bar{y} \subseteq \bar{y}$ is the subset of the existential variables of $q$ occurring in $\mathcal{M}(q)$; (*iii*) $\bar{z}$ are the fresh variables introduced when computing $\mathcal{M}(q)$; and (*iv*) $\xi(\bar{x}, \bar{y})$ is the conjunction of the inequality atoms obtained from $\xi(\bar{x}, \bar{y})$ by removing all those atoms of the form $y \neq t$ and $t \neq y$ in which $y$ is an existential variable occurring in $\bar{y}$ but not in $\bar{y}$ (i.e., not in $\mathcal{M}(q)$) and $t$ is any other possible variable. Observe that the epistemic operator is exploited to bind the existential variables coming from $q$. This is achieved by pushing the subset $\bar{y}$ of the existential variables $\bar{y}$ of $q$ occurring in $\mathcal{M}(q)$ inside the $\mathbf{K}$ operator.

We are now ready to present the algorithm $\mathfrak{M}$-MinComplete for computing the $\mathfrak{M}$-minimally complete $J$-abstractions. Given an OBDM specification $J = \langle \mathcal{O}, \mathcal{S}, \mathcal{M} \rangle$ and a UCQ $q_{\mathcal{S}}$ over $\mathcal{S}$ such that $\mathsf{SaturateQ}(q_{\mathcal{S}}) = q_1 \cup \ldots \cup q_n$, $\mathfrak{M}$-MinComplete$(J, q_{\mathcal{S}})$ outputs the $\mathsf{Datalog}^{\vee}_{\mathbf{K}}$ query $q_{\mathcal{O}} = \{r_{q_1}, \ldots, r_{q_n}\}$ over $J$.

**Example 3.** *Consider the OBDM specification $J$ illustrated in Example 1 and the CQ $q_{\mathcal{S}} = \{(x) \mid \exists y. s_1(x, y)\}$ over $\mathcal{S}$. One can verify that $\mathfrak{M}$-MinComplete$(J, q_{\mathcal{S}})$ returns the following safe $\mathsf{Datalog}^{\vee}_{\mathbf{K}}$ query $q_{\mathcal{O}}$ over $J$ asking for all those nodes $v$ such that either $v$ is connected to a node $v'$ known to be different from $v$ or $v$ is a special node with a self-loop:*

$$\begin{array}{lcl}\mathbf{K}(\mathsf{E}(x,y) \land x \neq y) & \to & Ans(x)\\ \mathbf{K}(\mathsf{E}(x,x) \land \mathsf{SN}(x)) & \to & Ans(x)\end{array}$$

*Note that $q_{\mathcal{O}}$ is a better complete approximation than the query $\{(x) \mid \exists y. \mathsf{E}(x, y)\}$, which is the UCQ-minimally complete $J$-abstraction of $q_{\mathcal{S}}$ (Cima, Lenzerini, and Poggi 2019).* $\triangle$

**Theorem 1.** $\mathfrak{M}$-MinComplete$(J, q_{\mathcal{S}})$ *terminates and returns the unique (up to $J$-equivalence) $\mathfrak{M}$-minimally complete $J$-abstraction of $q_{\mathcal{S}}$.*

Furthermore, we observe that the result of $\mathfrak{M}$-MinComplete$(J, q_{\mathcal{S}})$ is independent from the assertions occurring in the ontology of the OBDM specification $J$. Similar results can be obtained for OBDM specifications based on more expressive Horn DL ontologies.

Before concluding this section, we observe that $\mathfrak{M}$-MinComplete$(J, q_{\mathcal{S}})$ may return unsafe $\mathsf{Datalog}^{\vee}_{\mathbf{K}}$ queries. Nevertheless, these queries enjoy nice computational properties in our setting.

**Proposition 4.** *Let $q_{\mathcal{O}}$ be a $\mathsf{Datalog}^{\vee}_{\mathbf{K}}$ query with only rules of the form $\mathbf{K}(\exists \bar{z}. \phi(\bar{x}, \bar{z}) \land \xi(\bar{x})) \to Ans(\bar{x}_a)$. In our setting, (*i*) answering $q_{\mathcal{O}}$ is in* PTIME *in data complexity, and (*ii*) if $q_{\mathcal{O}}$ is safe, then it is possible to compute a UCQ$^{\neq}$ $q_r$ over $\mathcal{S}$ such that $q_{\mathcal{O}}^{J,D} = q_r^D$, for each $\mathcal{S}$-database $D$.*

## 5 Maximally Sound Abstractions

We present the algorithm $\mathfrak{M}$-MaxSound for the computation of $\mathfrak{M}$-maximally sound $J$-abstractions of UCQs. The output of $\mathfrak{M}$-MaxSound is a safe $\mathsf{Datalog}^{\vee}_{\mathbf{K}}$ query, thus proving that such abstractions always exist and can be expressed in $\mathsf{Datalog}^{\vee}_{\mathbf{K}}$. To present our algorithm, we first need to introduce the notion of *inverse mappings*, which extends the one introduced in (Duschka and Genesereth 1998) for the problem of rewriting queries using *disjunctive views*.[2]

To ease the presentation of our techniques, we now introduce a technical tool to encode a *DL-Lite$_{RDFS}$* ontology into a set of GLAV mapping assertions. Given a conjunction of atoms $\psi$ over a *DL-Lite$_{RDFS}$* ontology $\mathcal{O}$, we denote by $\mathcal{O}(\psi)$ the conjunction of atoms over $\mathcal{O}$ obtained by chasing the atoms in $\psi$ w.r.t. $\mathcal{O}$ as described in (Calvanese et al. 2007b). Note that, since $\mathcal{O}$ is a *DL-Lite$_{RDFS}$* ontology, $\mathcal{O}(\psi)$ is always finite and do not introduce further existential variables. Starting from an OBDM specification $J = \langle \mathcal{O}, \mathcal{S}, \mathcal{M} \rangle$, we can generate an equivalent OBDM specification $J' = \langle \emptyset, \mathcal{S}, \mathcal{M}' \rangle$ as follows. For each mapping assertion $m \in \mathcal{M}$ of the form $\forall \bar{x}. \forall \bar{y}. \phi(\bar{x}, \bar{y}) \to \exists \bar{z}. \psi(\bar{x}, \bar{z})$, the mapping $\mathcal{M}'$ contains the mapping assertion $\forall \bar{x}. \forall \bar{y}. \phi(\bar{x}, \bar{y}) \to \exists \bar{z}. \mathcal{O}(\psi(\bar{x}, \bar{y}))$. From now on, without loss of generality, we assume OBDM specifications with an empty set of ontological assertions.

### 5.1 Inverse Mapping

Our algorithm for the computation of $\mathfrak{M}$-maximally sound abstractions is based on the notion of inverse mappings. Intuitively, given a GLAV mapping $\mathcal{M}$, the inverse of $m \in \mathcal{M}$ is a safe $\mathsf{Datalog}^{\vee}_{\mathbf{K}}$ rule that characterizes those that may give rise to an homomorphic image of the head of $m$ when chased with $\mathcal{M}$. To compute inverse mappings, we will define the algorithm $\mathsf{InvMap}$. In order to define such algorithm, we need to introduce several additional notions.

**Splitting.** Given a set of relational atoms $A$ and a set of variables $U$ occurring in $A$, we define the $U$-graph $G^U_A$ of $A$ as follows: (*i*) each atom $\alpha$ in $A$ is a node of $G^U_A$, and (*ii*) there is an edge between nodes $\alpha_1, \alpha_2 \in A$ if and only if there exists a variable $z \in U$ such that $z$ occurs in both $\alpha_1$ and $\alpha_2$. The *splitting of $A$ around $U$* (written $Split(A, U)$) is the set of all those $C \subseteq A$ such that the atoms in $C$ form a connected component of $G^U_A$, i.e., $C$ is a maximal subset of $A$ s.t. there exists a path from $\alpha$ to $\alpha'$ in $G^U_A$, for each $\alpha, \alpha' \in C$. Whenever necessary, we will assume that sets in $Split(A, U)$ are presented as a tuple.

**Saturate.** Given a GLAV mapping $\mathcal{M}$, the algorithm $\mathsf{SaturateM}(\mathcal{M})$ produces an equivalent set of rules with explicit inequalities on the frontier variables. Given a GLAV mapping assertion $m = \forall \bar{x}. \forall \bar{y}. \phi(\bar{x}, \bar{y}) \to \exists \bar{z}. \psi(\bar{x}, \bar{z})$, the algorithm $\mathsf{SaturateM}(m)$ computes a set of rules in the following way: for each possible unifier $\mu$ between the variables in $\bar{x}$, $\mathsf{SaturateM}(m)$ contains a rule obtained from $\mu(m)$ by adding the inequality atom $x_1 \neq x_2$ in the body of $\mu(m)$, for each pair of distinct variables $x_1, x_2 \in \mu(\bar{x})$. For a GLAV mapping $\mathcal{M}$, we denote by $\mathsf{SaturateM}(\mathcal{M})$ the set of rules obtained by applying $\mathsf{SaturateM}(\mathcal{M})$ to each mapping assertion $m$ of $\mathcal{M}$. It is easy to see that $(D, \mathcal{I}) \models \mathcal{M}$ if and only if $(D, \mathcal{I}) \models \mathsf{SaturateM}(\mathcal{M})$. We call each assertion in $\mathsf{SaturateM}(\mathcal{M})$ a saturated GLAV mapping assertion and $\mathsf{SaturateM}(\mathcal{M})$ a saturated GLAV mapping. We write each

[2]We refer the interested reader to (Cima et al. 2021) for the relation between abstraction and the problem of rewriting queries using disjunctive views.

rule $m \in \mathsf{SaturateM}(\mathcal{M})$ as $\phi(\bar{x},\bar{y}) \wedge \xi(\bar{x}) \to \exists \bar{z}.\psi(\bar{x},\bar{z})$, where $\phi(\bar{x},\bar{y})$ and $\psi(\bar{x},\bar{z})$ are conjunctions of relational atoms over $\mathcal{S}$ and over $\mathcal{O}$, respectively, whereas $\xi(\bar{x})$ is a conjunction of inequality atoms. We write $r\text{-}body(m)$ for the set of relational atoms in $\phi(\bar{x},\bar{y})$, and, as in GLAV mappings, $head(m)$ for the set of relational atoms in $\psi(\bar{x},\bar{z})$.

**Generators and Supports.** Assume a saturated GLAV mapping $\mathcal{M}$ from a source schema $\mathcal{S}$ to an ontology $\mathcal{O}$. Without loss of generality, we assume that mapping assertions in $\mathcal{M}$ use distinct variables. Given a set of atoms $A$ over $\mathcal{O}$, a *generator of $A$ w.r.t. $\mathcal{M}$* is a pair $g = \langle h, m\rangle$, where $h$ is a homomorphism from $A$ to the head of $m$ and $m \in \mathcal{M}$. A generator $g$ is $V$-distinguished, for some $V \subseteq Var(A)$, if $h(v)$ is a frontier variable of $m$, for each $v \in V$, and $h(u)$ is not a frontier variable of $m$, for each $u \notin V$. We write $Gens(A,\mathcal{M},V)$ for the set of $V$-distinguished generators of $A$ w.r.t. $\mathcal{M}$. Intuitively, a generator $g = \langle h, m\rangle$ of $A$ w.r.t. $\mathcal{M}$ represents a possible way to obtain a homomorphic copy $B \subseteq \mathcal{M}(D)$ of $A$ by applying $m$ to an $\mathcal{S}$-database $D$. If such generator is $V$-distinguished, then the only constants occurring in $B$ are images of the variables in $V$.

**Example 4.** *Let $\mathcal{M} = \{\boldsymbol{m}_A, \boldsymbol{m}_B, \boldsymbol{m}_C, \boldsymbol{m}_D\}$, where:*

$$\boldsymbol{m}_A : s_1(x_1,x_2) \wedge x_1 \neq x_2 \to \exists Z.P_1(x_1,Z) \wedge P_2(Z,x_2),$$
$$\boldsymbol{m}_B : s(x_3,x_4) \wedge x_3 \neq x_4 \to \exists Z'.P_1(x_3,Z') \wedge P_2(Z',x_4),$$
$$\boldsymbol{m}_C : s_3(x_5,y_1) \wedge s_4(y_1,y_2) \to P_1(x_5,x_5),$$
$$\boldsymbol{m}_D : s_5(x_6,x_7) \wedge x_6 \neq x_7 \to P_1(x_6,x_7) \wedge P_2(x_7,x_6).$$

*For the set of atoms $A_1 = \{P_1(x_1,Z), P_2(Z,x_2)\}$, we have that $Gens(A_1,\mathcal{M},\{x_1,x_2\}) = \{g_1,g_2\}$ with $g_1 = \langle h_1, \boldsymbol{m}_A\rangle$ and $g_2 = \langle h_2, \boldsymbol{m}_B\rangle$ where $h_1$ is the identity function, whereas $h_2(x_1) = x_3$, $h_2(x_2) = x_4$, and $h_2(Z) = Z'$.*

*For the set of atoms $A_2 = \{P_1(x_1,Z)\}$, we have that $Gens(A_2,\mathcal{M},\{x_1,Z\}) = \{g_3,g_4\}$ with $g_3 = \langle h_3, \boldsymbol{m}_C\rangle$ and $g_4 = \langle h_4, \boldsymbol{m}_D\rangle$ where $h_3(x_1) = x_5$ and $h_3(Z) = x_5$, whereas $h_4(x_1) = x_6$ and $h_4(Z) = x_7$.*

*For the set of atoms $A_3 = \{P_2(Z,x_2)\}$, we have that $Gens(A_3,\mathcal{M},\{Z,x_2\}) = \{g_5\}$ with $g_5 = \langle h_5, \boldsymbol{m}_D\rangle$ where $h_5(Z) = x_7$ and $h_5(x_2) = x_6$.* $\triangle$

We now extend the notion of generators to the case of multiple applications of mapping assertions. Given a tuple of sets of atoms $\mathcal{A} = \langle A_1,\ldots,A_n\rangle$ and a set of variables $V \subseteq Var(A_1) \cup \ldots \cup Var(A_n)$, the $V$-distinguished generators of $\mathcal{A}$ w.r.t. $\mathcal{M}$ are all the tuples $G = \langle g_1,\ldots,g_n\rangle$ such that $g_i \in Gens(A_i,\mathcal{M},V)$, for each $i = 1,\ldots,n$. With a slight abuse of notation, we use $Gens(\mathcal{A},\mathcal{M},V)$ to denote the set of all tuples of $V$-distinguished generators of elements of $\mathcal{A}$ w.r.t. $\mathcal{M}$ and we call *generator of $\mathcal{A}$* every such tuple. Intuitively, each $G \in Gens(\mathcal{A},\mathcal{M},V)$ represents a possible way to obtain a homomorphic copy of $\mathcal{A}$ using a single assertion of $\mathcal{M}$ for each $A \in \mathcal{A}$.

**Example 5.** *Refer to Example 4. For $\mathcal{A} = \langle A_1\rangle$ and $\mathcal{A}' = \langle A_2, A_3\rangle$, we have $Gens(\mathcal{A},\mathcal{M},\{x_1,x_2\}) = \{G_1,G_2\}$ and $Gens(\mathcal{A}',\mathcal{M},\{x_1,x_2,Z\}) = \{G_3,G_4\}$, resp., where $G_1 = \langle g_1\rangle$, $G_2 = \langle g_2\rangle$, $G_3 = \langle g_3, g_5\rangle$, and $G_4 = \langle g_4, g_5\rangle$.* $\triangle$

We now characterize those databases $D$ for which $\mathcal{M}(D)$ contains an homomorphic copy of $\mathcal{A}$ generated using the the mappings constituting some $G \in Gens(\mathcal{A},\mathcal{M},V)$. To this end, we define a formula $\sigma_G$ that we call the *support* of $G \in Gens(\mathcal{A},\mathcal{M},V)$. Given $G = \langle g_1,\ldots,g_n\rangle$ in $Gens(\mathcal{A},\mathcal{M},V)$ with $g_i = \langle h_i, m_i\rangle$ for each $i = 1,\ldots,n$, let $\langle \rho_1,\ldots,\rho_n\rangle$ be a tuple of renaming for the variables of $\mathcal{M}$ whose images are pairwise disjoint. For $i = 1,\ldots,n$, we define $\beta_i = \rho_i(r\text{-}body(m_i))$ and $\sigma_G = \bigwedge_i \rho_i(r\text{-}body(m_i))$. Intuitively, each $\beta_i$ represents a set of relational source atoms that is needed to generate an homomorphic image of $h_i(A_i)$ using $m_i$.

**Example 6.** *Refer to Example 5. We have $\sigma_{G_1} = s_1(x_1',x_2')$, $\sigma_{G_2} = s(x_3',x_4')$, $\sigma_{G_3} = s_3(x_5',y_1') \wedge s_4(y_1',y_2') \wedge s_5(x_6'',x_7'')$, and $\sigma_{G_4} = s_5(x_6',x_7') \wedge s_5(x_6'',x_7'')$.* $\triangle$

In order to connect together the different $h_i(A_i)$ to form an image of $\mathcal{A}$, we need to impose further restrictions on the variables of $\sigma_G$. These restrictions will take the form of equalities. Let $\eta_G$ denote binary relation over $Var(\sigma_G)$ defined as follows: for each $i,j = 1,\ldots,n$, and $v \in V$, $(\rho_i(h_i(v)), \rho_j(h_j(v))) \in \eta_G$. Moreover, given a set $\bar{x} \subseteq V$, let $\eta^{[\bar{x}]}$ denote the binary relation over $Var(\sigma_G) \cup \bar{x}$ defined as follows: for each $i = 1,\ldots,n$, and $x \in \bar{x}$, $(x, \rho_i(h_i(x))) \in \eta^{[\bar{x}]}$. We use $\eta_G^{[\bar{x}]}$ to denote the reflexive and transitive closure of $\eta_G \cup \eta^{[\bar{x}]}$.

**Example 7.** *Refer to Example 6. We have that $\eta_{G_1}^{[\{x_1,x_2\}]}$, $\eta_{G_2}^{[\{x_1,x_2\}]}$, $\eta_{G_3}^{[\{x_1,x_2\}]}$, and $\eta_{G_4}^{[\{x_1,x_2\}]}$ are the reflexive and transitive closure of $\{(x_1,x_1'),(x_2,x_2')\}$, $\{(x_1,x_3'),(x_2,x_4')\}$, $\{(x_5',x_7''),(x_1,x_5'),(x_2,x_6'')\}$, $\{(x_7',x_7''),(x_1,x_6'),(x_2,x_6'')\}$, respectively.* $\triangle$

As customary, for each equivalence class $E$ of $\eta_G^{[\bar{x}]}$, we select an element $s(E) \in E$ as representative; if $E$ contains elements of $\bar{x}$, we require $s(E) \in \bar{x}$. Moreover, given $v \in Var(\sigma_G)$, we use $v_{\sim G,\bar{x}}$ to denote the equivalence class of $\eta_G^{[\bar{x}]}$ that contains $v$. Finally, we define the *$\bar{x}$-support $\sigma_G^{\bar{x}}$ of $G$* as the formula $\exists \bar{w}.\eta_G^{[\bar{x}]}(\sigma_G)$, where $\eta_G^{[\bar{x}]}(\sigma_G)$ is obtained from $\sigma_G$ by replacing each variable $v \in Var(\sigma_G)$ with $s(v_{\sim G,\bar{x}})$ and $\bar{w}$ are the variables of $\eta_G^{[\bar{x}]}(\sigma_G)$ not occurring in $\bar{x}$.

**Example 8.** *Refer to the previous example. We have that $\sigma_{G_1}^{\{x_1,x_2\}} = s_1(x_1,x_2)$, $\sigma_{G_2}^{\{x_1,x_2\}} = s(x_1,x_2)$, $\sigma_{G_3}^{\{x_1,x_2\}} = \exists y_1',y_2'.s_3(x_1,y_1') \wedge s_4(y_1',y_2') \wedge s_5(x_2,x_1)$, and $\sigma_{G_4}^{\{x_1,x_2\}} = \exists x_7'.s_5(x_1,x_7') \wedge s_5(x_2,x_7')$.* $\triangle$

**The $\mathsf{InvMap}$ Algorithm.** We are now ready to formally define the $\mathsf{InvMap}$ algorithm. In the algorithm, we use $Partitions(\bar{z})$, where $\bar{z}$ is a set of variables, to denote the set of all 2-tuples of pairwise disjoint sets $\langle U, V\rangle$ such that $U \cup V = \bar{z}$. Intuitively, given an OBDM specification $J = \langle \mathcal{O},\mathcal{S},\mathcal{M}\rangle$, $\mathsf{InvMap}(J)$ returns a set of safe $\mathsf{Datalog}_{\mathbf{K}}^{\vee}$ rules that represents a *sound inverse of $\mathcal{M}$ w.r.t. $\mathcal{O}$*, i.e., a set of formulae that, given an $\mathcal{S}$-database $D$, characterize those databases $D'$ such that $mod(J,D) \supseteq mod(J,D')$. Equivalently, given an $\mathcal{S}$-database $D$, $\mathsf{InvMap}(J)$ characterizes all those databases $D'$ such that $\mathcal{M}(D')$ is an homomorphic copy of $\mathcal{M}(D)$.

In order to define a sound inverse of $\mathcal{M}$ w.r.t. $\mathcal{O}$, $\mathsf{InvMap}(J)$ relies on the notions of supports and genera-

Algorithm 1: InvMap

```
Input: OBDM specification $\langle \emptyset, \mathcal{S}, \mathcal{M} \rangle$
Output: Set $\mathcal{R}$ of safe $\mathsf{Datalog}^{\vee}_{\mathbf{K}}$ rules
 1: $\mathcal{M}' = \mathsf{SaturateM}(\mathcal{M})$
 2: $\mathcal{R} = \emptyset$
 3: for all $m : \phi(\bar{x}, \bar{y}) \wedge \xi(\bar{x}) \rightarrow \exists \bar{z}.\psi(\bar{x}, \bar{z}) \in \mathcal{M}'$ do
 4:   $\Gamma = \emptyset$
 5:   for all $\langle U, V \rangle \in Partitions(\bar{z})$ do
 6:     Let $S_\psi = Split(\psi, U)$
 7:     for all $G \in Gens(S_\psi, \mathcal{M}', V \cup \bar{x})$ do
 8:       if $(x, x') \notin \eta_G^{[\bar{x}]}$ for each $x, x' \in \bar{x}$ s.t. $x \neq x'$ then
 9:         Let $\sigma_G^{\bar{x}}$ be the $\bar{x}$-support of $G$
10:         $\Gamma = \Gamma \cup \{\sigma_G^{\bar{x}}\}$
11:       end if
12:     end for
13:   end for
14:   $inv(m) = \mathbf{K}(\exists \bar{z}.\psi(\bar{x}, \bar{z}) \wedge \xi(\bar{x})) \rightarrow \bigvee_{\gamma \in \Gamma} \gamma$
15:   $\mathcal{R} = \mathcal{R} \cup \{inv(m)\}$
16: end for
17: return $\mathcal{R}$
```

tors. More specifically, for every saturated mapping assertion $\phi(\bar{x}, \bar{y}) \wedge \xi(\bar{x}) \rightarrow \exists \bar{z}.\psi(\bar{x}, \bar{z})$, $\mathsf{InvMap}(J)$ contains a safe $\mathsf{Datalog}^{\vee}_{\mathbf{K}}$ rule of the form $\mathbf{K}(\exists \bar{z}.\psi(\bar{x}, \bar{z}) \wedge \xi(\bar{x})) \rightarrow \bigvee_{\gamma \in \Gamma} \gamma$, where $\Gamma$ is the set of all $\bar{x}$-supports of the generators of all possible splittings of $\psi$. Intuitively, the latter guarantees that, if $(mod(\langle J, D \rangle), D')$ satisfies $\mathsf{InvMap}(J)$, for every atom $\alpha \in \mathcal{M}(D)$, there exists an assertion $m \in \mathcal{M}$ such that $h(body(m)) \subseteq D'$ and $\ell(\alpha) \in h(head(m))$, where $h$ and $\ell$ are homomorphisms.

**Example 9.** *Consider the OBDM specification J illustrated in Example 1. One can verify that* $\mathsf{InvMap}(J)$ *returns the following set* $\mathcal{R}$ *of safe* $\mathsf{Datalog}^{\vee}_{\mathbf{K}}$ *rules:*

$$\begin{array}{lll}
\mathbf{K}(\mathsf{E}(x_1, x_2) \wedge x_1 \neq x_2) & \rightarrow & s_1(x_1, x_2) \\
\mathbf{K}(\mathsf{E}(x, x)) & \rightarrow & s_3(x) \vee s_1(x, x) \\
\mathbf{K}(\exists z.\mathsf{E}(x, z)) & \rightarrow & s_2(x) \vee \exists y.s_1(x, y) \vee s_3(x) \\
\mathbf{K}(\mathsf{SN}(x)) & \rightarrow & s_1(x, x)
\end{array}$$

$\triangle$

In the reminder of this section, we introduce two crucial properties of the InvMap algorithm. Assume an OBDM specification $J$ and an $\mathcal{S}$-database $D$. One can show that $(mod(\langle J, D \rangle), D)$ satisfies $\mathsf{InvMap}(J)$.

**Lemma 1.** *For every $\mathcal{S}$-database D, $(mod(\langle J, D \rangle), D) \models$* $\mathsf{InvMap}(J)$.

As it will turn out, Lemma 1 guarantees that our approximation is sound. Moreover, one can show that, whenever $(mod(\langle J, D \rangle), D') \models \mathsf{InvMap}(J)$, we have that $\mathcal{M}(D)$ can be homomorphically mapped into $\mathcal{M}(D')$. Informally, this property guarantees that our approximation is maximal.

**Lemma 2.** *For every pair of $\mathcal{S}$-databases $D, D'$ such that* $(mod(\langle J, D \rangle), D') \models \mathsf{InvMap}(J)$*, there exists a homomorphism from $\mathcal{M}(D)$ to $\mathcal{M}(D')$.*

### 5.2 Maximally Sound Abstraction Algorithm

With the InvMap algorithm and its properties at hand, we are now ready to focus on the problem of computing the $\mathfrak{M}$-maximally sound $J$-abstractions. First, let us introduce some additional notations. For an OBDM specification $J = \langle \mathcal{O}, \mathcal{S}, \mathcal{M} \rangle$, the *standard predicate renaming of* $J$, denoted by $\mathsf{ren}_J$, is the renaming such that $\mathsf{ren}_J(s) = s'$, for every $s \in \mathcal{S}$, where $s'$ is an IDB predicate with $ar(s) = ar(s')$. Furthermore, with a slight abuse of notation, given a set of formulae $\Psi$ over $\mathcal{S} \cup \mathcal{O}$, we write $\mathsf{ren}_J(\Psi)$ for the set of formulae obtained from $\Psi$ by replacing each occurrence of $s$ in $\Psi$ with $\mathsf{ren}_J(s)$, for each $s \in \mathcal{S}$.

We can now present the algorithm $\mathfrak{M}$-MaxSound for computing the $\mathfrak{M}$-maximally sound $J$-abstractions. Given an OBDM specification $J = \langle \mathcal{O}, \mathcal{S}, \mathcal{M} \rangle$ and a UCQ $q_{\mathcal{S}} = q^1_{\mathcal{S}} \cup \ldots \cup q^n_{\mathcal{S}}$ over $\mathcal{S}$ such that $q^i_{\mathcal{S}} = \{\bar{x}_i \mid \exists \bar{y}_i.\phi_i(\bar{x}_i, \bar{y}_i)\}$ for each $i = 1, \ldots, n$, $\mathfrak{M}$-$\mathsf{MaxSound}(J, q_{\mathcal{S}})$ simply outputs the safe $\mathsf{Datalog}^{\vee}_{\mathbf{K}}$ query $q_{\mathcal{O}} = \mathsf{ren}_J(\Psi)$, where $\Psi = \mathsf{InvMap}(J) \cup \{\phi_1(\bar{x}_1, \bar{y}_1) \rightarrow Ans(\bar{x}_1)\} \cup \ldots \cup \{\phi_n(\bar{x}_n, \bar{y}_n) \rightarrow Ans(\bar{x}_n)\}$.

We conclude this section by establishing termination and correctness of the $\mathfrak{M}$-MaxSound algorithm.

**Theorem 2.** $\mathfrak{M}$-$\mathsf{MaxSound}(J, q_{\mathcal{S}})$ *terminates and returns the unique (up to J-equivalence) $\mathfrak{M}$-maximally sound J-abstraction of $q_{\mathcal{S}}$.*

## 6 Perfect Abstractions

It follows from Theorem 1 that either the perfect abstraction of a source UCQ can be expressed in $\mathsf{Datalog}^{\vee}_{\mathbf{K}}$, or it cannot be expressed as a monotone query (if it exists at all). We now present an algorithm that, given an OBDM specification $J$ and a source UCQ $q_{\mathcal{S}}$, returns the perfect $J$-abstraction of $q_{\mathcal{S}}$, if and only if it exists and is in $\mathfrak{M}$. To this aim, we make use of Proposition 4, and refer to the $q_r$ defined in that proposition as the *rewriting of $q_{\mathcal{O}}$ w.r.t. J*.

Our algorithm, that we call $\mathfrak{M}$-Perfect, goes as follows. Given an OBDM specification $J = \langle \mathcal{O}, \mathcal{S}, \mathcal{M} \rangle$ and a UCQ $q_{\mathcal{S}}$ over $\mathcal{S}$ as input: if (*i*) $q_{\mathcal{O}} = \mathfrak{M}$-$\mathsf{MinComplete}(J, q_{\mathcal{S}})$ is safe and (*ii*) $q_r \sqsubseteq_{\mathcal{S}} q_{\mathcal{S}}$; then **return** $q_{\mathcal{O}}$; otherwise, **report** "no perfect $J$-abstraction of $q_{\mathcal{S}}$ is in $\mathfrak{M}$".

**Example 10.** *In Example 3,* $\mathfrak{M}$-$\mathsf{Perfect}(J, q_{\mathcal{S}})$ *returns $q_{\mathcal{O}}$, which is the perfect J-abstraction of $q_{\mathcal{S}}$.* $\triangle$

We conclude this section by establishing termination and correctness of the $\mathfrak{M}$-Perfect algorithm.

**Theorem 3.** $\mathfrak{M}$-$\mathsf{Perfect}(J, q_{\mathcal{S}})$ *terminates and returns the unique (up to J-equivalence) perfect J-abstraction of $q_{\mathcal{S}}$ if and only if such an abstraction can be expressed in $\mathfrak{M}$.*

## 7 Conclusion

We presented a thorough study of monotone abstractions of UCQs in OBDM systems. We proved that best approximations of such abstractions always exist and introduced a query language, $\mathsf{Datalog}^{\vee}_{\mathbf{K}}$, that captures them. Directions for future work are many. In the context of monotone abstractions, we would like to investigate the case of more expressive ontology languages, e.g., *DL-Lite*$_{\mathcal{R}}$, as well as more expressive source query languages, e.g., unions of conjunctive queries with inequalities and disjunctive Datalog. Finally, the problem of checking whether given best approximations are expressible in simpler and more user friendly languages remains open.

## Acknowledgements

This work has been partially supported by the ANR AI Chair INTENDED (ANR-19-CHIA-0014), by MIUR under the PRIN 2017 project "HOPE" (prot. 2017MMJJRE), and by the EU under the H2020-EU.2.1.1 project TAILOR, grant id. 952215.

## References

Baader, F.; Calvanese, D.; McGuinness, D.; Nardi, D.; and Patel-Schneider, P. F., eds. 2003. *The Description Logic Handbook: Theory, Implementation and Applications*. Cambridge University Press.

Bienvenu, M. 2016. Ontology-Mediated Query Answering: Harnessing Knowledge to Get More from Data. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence (IJCAI 2016)*, 4058–4061.

Brickley, D.; and Guha, R. V. 2014. RDF Schema 1.1. W3C Recommendation, World Wide Web Consortium. Available at https://www.w3.org/TR/2014/REC-rdf-schema-20140225/.

Calvanese, D.; De Giacomo, G.; Lembo, D.; Lenzerini, M.; and Rosati, R. 2007a. EQL-Lite: Effective First-Order Query Processing in Description Logics. In *Proceedings of the Twentieth International Joint Conference on Artificial Intelligence (IJCAI 2007)*, 274–279.

Calvanese, D.; De Giacomo, G.; Lembo, D.; Lenzerini, M.; and Rosati, R. 2007b. Tractable Reasoning and Efficient Query Answering in Description Logics: The *DL-Lite* Family. *Journal of Automated Reasoning*, 39(3): 385–429.

Calvanese, D.; and Rosati, R. 2003. Answering recursive queries under keys and foreign keys is undecidable. In *Proceedings of the Tenth International Workshop on Knowledge Representation meets Databases (KRDB 2003)*, volume 79 of *CEUR Electronic Workshop Proceedings*.

Cima, G. 2017. Preliminary Results on Ontology-based Open Data Publishing. In *Proceedings of the Thirtieth International Workshop on Description Logics (DL 2017)*, volume 1879 of *CEUR Electronic Workshop Proceedings*.

Cima, G. 2020. *Abstraction in Ontology-based Data Management*. Ph.D. thesis, Sapienza University of Rome.

Cima, G.; Console, M.; Lenzerini, M.; and Poggi, A. 2021. Abstraction in Data Integration. In *Proceedings of the Thirty-Sixth Annual ACM/IEEE Symposium on Logic in Computer Science (LICS 2021)*, 1–11.

Cima, G.; Lenzerini, M.; and Poggi, A. 2019. Semantic Characterization of Data Services through Ontologies. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence (IJCAI 2019)*, 1647–1653.

Cima, G.; Lenzerini, M.; and Poggi, A. 2020a. Answering Conjunctive Queries with Inequalities in *DL-Lite*$_{\mathcal{R}}$. In *Proceedings of the Thirty-Fourth AAAI Conference on Artificial Intelligence (AAAI 2020)*, 2782–2789.

Cima, G.; Lenzerini, M.; and Poggi, A. 2020b. Non-Monotonic Ontology-based Abstractions of Data Services. In *Proceedings of the Seventeenth International Conference on Principles of Knowledge Representation and Reasoning (KR 2020)*, 243–252.

Cuenca Grau, B. 2004. A possible simplification of the semantic web architecture. In *Proceedings of the Thirteenth International World Wide Web Conference (WWW 2004)*, 704–713.

De Giacomo, G.; Lembo, D.; Lenzerini, M.; Poggi, A.; and Rosati, R. 2018. Using Ontologies for Semantic Data Integration. In *A Comprehensive Guide Through the Italian Database Research Over the Last 25 Years*, 187–202.

Duschka, O. M.; and Genesereth, M. R. 1998. Query Planning with Disjunctive Sources. In *Proceedings of the AAAI-98 Workshop on AI and Information Integration*.

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.

Lenzerini, M. 2018. Managing Data through the Lens of an Ontology. *AI Magazine*, 39(2): 65–74.

Levesque, H. J.; and Lakemeyer, G. 2001. *The Logic of Knowledge Bases*. The MIT Press.

Levy, A. Y.; and Rousset, M.-C. 1998. Combining Horn Rules and Description Logics in CARIN. *Artificial Intelligence*, 104(1–2): 165–209.

Lutz, C.; Marti, J.; and Sabellek, L. 2018. Query Expressibility and Verification in Ontology-based Data Access. In *Proceedings of the Sixteenth International Conference on the Principles of Knowledge Representation and Reasoning (KR 2018)*, 389–398.

Ortiz, M. 2018. Improving Data Management using Domain Knowledge. In *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence (IJCAI 2018)*, 5709–5713.

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

Rosati, R. 2007. The Limits of Querying Ontologies. In *Proceedings of the Eleventh International Conference on Database Theory (ICDT 2007)*, volume 4353 of *Lecture Notes in Computer Science*, 164–178. Springer.

Xiao, G.; Calvanese, D.; Kontchakov, R.; Lembo, D.; Poggi, A.; Rosati, R.; and Zakharyaschev, M. 2018. Ontology-Based Data Access: A Survey. In *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence (IJCAI 2018)*, 5511–5519.