Abstraction in Data Integration

Gianluca Cima, Marco Console, Maurizio Lenzerini, Antonella Poggi

Abstraction in Data Integration

(Invited Paper)

Gianluca Cima
CNRS & University of Bordeaux
gianluca.cima@u-bordeaux.fr

Marco Console
Sapienza University of Rome
console@diag.uniroma1.it

Maurizio Lenzerini
Sapienza University of Rome
lenzerini@diag.uniroma1.it

Antonella Poggi
Sapienza University of Rome
poggi@diag.uniroma1.it

***Abstract*—Data integration provides a unified and abstract view over a set of existing data sources. The typical architecture of a data integration system comprises the global schema, which is the structure for the unified view, the source schema, and the mapping, which is a formal account of how data at the sources relate to the global view. Most of the research work on data integration in the last decades deals with the problem of processing a query expressed on the global schema by computing a suitable query over the sources, and then evaluating the latter in order to derive the answers to the original query. Here, we address a novel issue in data integration: starting from a query expressed over the sources, the goal is to find an abstraction of such query, i.e., a query over the global schema that captures the original query, modulo the mapping. The goal of the paper is to provide an overview of the notion of abstraction in data integration, by presenting a formal framework, illustrating the results that have appeared in the recent literature, and discussing interesting directions for future research.**

I. Introduction

Data integration is the problem of providing a unified and reconciled view of the data stored in a set of autonomous and heterogeneous sources [1]. Issues related to integrating data have been studied for many years, but the problem is still relevant today, when the volume of data and the need to collect, link and share them explodes. The theoretical works on data integration systems have fostered a three-tier architecture [2] comprising the *data sources*, with the associated source schema, the *global view*, with the associated global schema, and the *mapping* from the sources to the global view.

In the declarative approach to integration, a data integration system is formalized as a triple $\mathcal{J} = \langle \mathcal{G}, \mathcal{S}, \mathcal{M} \rangle$, where $\mathcal{G}$ is the global schema, $\mathcal{S}$ is the source schema and $\mathcal{M}$ is the mapping. The global schema is often aimed at providing an abstract representation of the domain of interest, rather than a mere data structure capturing the source data. Following this idea, the role of the system is to provide different types of services based on such abstract representation, thus freeing the users from the implementation details of the data sources when accessing the information content of the system. Also, the mapping is specified through logical assertions on the source and the global schema, describing how the data at the sources relate to the elements of the global view. Note that, since such assertions are typically expressed as logical implication, given a particular configuration of the data at the sources (a source database), there may be many global databases satisfying the mapping. It follows that data integration naturally leads to the issue of dealing with incomplete information.

Several specializations of the general architecture have been explored in the last two decades with the goal of capturing different variants of the problem: in data exchange [3] the goal is to move data from one source to the global view, in view-based query processing [4] the sources are modeled as materialized views defined on the basis of the global schema, in data warehousing [5] the global view is materialized in a relational database with associated data marts, in ontology-based data management [6] the global view is modeled through an ontology, and in (virtual) knowledge graphs [7] the global view is rendered as a (virtual) graph database.

In all these contexts, the task that received most attention by the research community is the one of answering queries expressed on the global schema. This task is typically based on the certain answer semantics, where the tuples satisfying a query $q_{\mathcal{G}}$ posed to the global schema $\mathcal{G}$ of a data integration system $\mathcal{J}$ are the answers to $q_{\mathcal{G}}$ in all the global databases satisfying the mapping of $\mathcal{J}$ w.r.t. the current source database D. Note that, since the concrete data reside in the sources, a common approach to computing the certain answers to $q_{\mathcal{G}}$ is based on "rewriting" $q_{\mathcal{G}}$ into a query $q_{\mathcal{S}}$ over the source schema such that evaluating $q_{\mathcal{S}}$ over D yields exactly the certain answers to $q_{\mathcal{G}}$.

Recent works [8]–[12] address a novel issue in data integration: starting from a query $q_{\mathcal{S}}$ expressed over the sources, the goal is to find a query $q_{\mathcal{G}}$ over the global schema that captures the original query, modulo the mapping. The query $q_{\mathcal{G}}$ is called a *perfect abstraction* of the data service expressed by $q_{\mathcal{S}}$, because it provides a formulation of $q_{\mathcal{S}}$ on the basis of the abstract representation of the domain, obtained by leveraging on the mapping of the data integration system.

Obviously, the meaning of "capturing" a source query has to be made precise. Ideally, the query $q_{\mathcal{G}}$ that captures $q_{\mathcal{S}}$ at best is the one that, when it is evaluated over the incomplete global

978-1-6654-4895-6/21/$31.00

database corresponding to a source database D, returns the answers of q_S over D. In this case, we call q_G the perfect $\mathcal{J}$-abstraction of q_S. We will see that, when a perfect abstraction does not exist, it may be possible to compute (sound or complete) approximations of such query.

We use an example for informally introducing some of the notions that we will discuss in the rest of the paper. In the example, we assume that the evaluation of a query expressed over the global schema is based on the certain answer semantics.

Example 1. *Let $\mathcal{J} = \langle \mathcal{G}, \mathcal{S}, \mathcal{M} \rangle$ be a data integration system where the elements of the source schema $\mathcal{S}$ are the predicates (with associated arity) $\{s_1/1, s_2/2, s_3/1, s_4/1, s_5/2\}$, the elements of the global schema $\mathcal{G}$ are $\{g_1/2, g_2/1, g_3/2, g_4/2, g_5/1\}$, and $\mathcal{M}$ contains the following assertions (where the free variables are implicitly universally quantified):*

$m_1 : s_1(x) \rightarrow \exists y.g_1(x, y)$
$m_2 : s_2(x, y) \rightarrow g_1(x, y)$
$m_3 : s_3(x) \wedge s_4(x) \rightarrow g_2(x)$
$m_4 : s_3(x) \wedge s_2(x, y) \rightarrow g_3(x, y)$
$m_5 : \exists z.s_5(y, z) \wedge s_2(x, y) \rightarrow g_3(x, y)$
$m_6 : s_5(x, y) \rightarrow g_4(x, y)$
$m_7 : s_1(x) \wedge s_4(x) \rightarrow g_5(x)$

Consider the query $q^1_S = \{x, y \mid s_2(x, y)\}$. It is easy to see that, for every database D, the certain answers of $q^1_G = \{x, y \mid g_1(x, y)\}$ coincides with the answer to q^1_S w.r.t. D. It follows that the CQ $q^1_G = \{x, y \mid g_1(x, y)\}$ is a perfect $\mathcal{J}$-abstraction of q^1_S.

Consider the query $q^2_S = \{x \mid \exists y.s_2(x, y)\}$. A natural candidate for the perfect $\mathcal{J}$-abstraction of q^2_S is $q^2_G = \{x \mid \exists y.g_1(x, y)\}$. Note, however, that the certain answers to q^2_G include tuples in s_1 that may not belong to s_2, and therefore q^2_G is not even a sound $\mathcal{J}$-abstraction of q^2_S (i.e., it does not retrieve only tuples of q^2_S). Indeed, it can be shown that no UCQ esists that is a perfect $\mathcal{J}$-abstraction of q^2_S. However, the query asking for those x such that $g_1(x, y)$ is known to be true, i.e., holds in every model of $\mathcal{J}$, *cannot exploit mapping m_1, and therefore avoids retrieving tuples from s_1. It follows that such query, which is not expressible as a UCQ, is a perfect $\mathcal{J}$-abstraction of q^2_S.*

Consider the query $q^3_S = \{x \mid s_1(x)\}$. Again, the natural candidate for the perfect $\mathcal{J}$-abstraction of q^3_S is clearly $q^2_G = \{x \mid \exists y.g_1(x, y)\}$. However, because of m_2, the certain answers to q^2_G also include the values in the first component of s_2, and this means that q^2_G is not a sound $\mathcal{J}$-abstraction of q^3_S, although it is a complete one (i.e., it retrieves all tuples of q^3_S). Another possible candidate is the query $q^3_G = \{x \mid \exists y.g_5(x)\}$. However, this query captures only the tuples occurring in s_1 which also occur in s_4. It follows that q^3_G is a sound $\mathcal{J}$-abstraction, although not a complete one. Actually, it can be shown that no perfect $\mathcal{J}$-abstraction of q^3_S exists in the class UCQ, but q^2_G and q^3_G are, respectively, the minimally complete and the maximally sound $\mathcal{J}$-abstraction of q^3_S in the class UCQ.

Consider now the query $q^4_S = \{() \mid \exists x, y.s_5(x, y) \wedge s_3(x)\}$, and assume that we aim at checking whether its perfect $\mathcal{J}$-abstraction can be expressed as a UCQ. We immediately observe that $\{() \mid \exists x, y.g_4(x, y) \wedge g_2(x)\}$ is a sound $\mathcal{J}$-abstraction of q^4_S. Also, we can easily verify that $\{() \mid \exists x, y, x_1.g_4(x, y) \wedge g_3(x, x_1) \wedge g_2(x_1)\}$ is also sound, and may retrieve tuples that are not retrieved by $\{() \mid \exists x, y.g_4(x, y) \wedge g_2(x)\}$. More generally, all queries of the form $\{() \mid \exists x, y, x_1, \ldots, x_n.g_4(x, y) \wedge g_3(x, x_1) \wedge \ldots \wedge g_3(x_{n-1} \wedge x_n) \wedge g_2(x_n)\}$, for $n \geq 1$, are pairwise incomparable sound $\mathcal{J}$-abstractions of q^4_S. Based on this observation, one can show that there exists no maximally sound $\mathcal{J}$-abstraction of q^4_S in the class UCQ. However, the following Datalog query (with goal Ans) is the maximally sound $\mathcal{J}$-abstraction of q^4_S in the whole class of monotone queries:

$$
\begin{array}{lcl}
g_3(x, y) & \rightarrow & t_1(x, y) \\
t_1(x, y) \wedge t_1(y, z) & \rightarrow & t_1(x, z) \\
g_4(x, y) \wedge g_2(x) & \rightarrow & Ans() \\
g_4(x, y) \wedge t_1(x, z) \wedge g_2(z) & \rightarrow & Ans()
\end{array}
$$

△

We argue that abstraction is relevant in a plethora of scenarios. We mention some of them here. Abstraction it can be used for a kind of reverse engineering task, namely to automatically produce a semantic characterization of a data service implemented at the level of data sources [10]. Also, following the ideas in [8], it can be shown that abstractions can provide the semantics of open datasets and open APIs published by organizations, which is a key aspect for unchaining all the potentials of open data [13]. In the context of ontology-based data management, [9] has shown that abstraction can be used to study the concept of realization of source queries, i.e., checking whether the mapping provides the right coverage for expressing the relevant data services at the global schema level. Finally, abstraction can be the basis for a semantic-based approach to source profiling [14], in particular for describing the structure and the content of a data source in terms of the business vocabulary.

The goal of this paper is to provide an overview of the notion of abstraction in data integration, by

- presenting a formal framework for abstraction,
- illustrating the results that have appeared in the recent literature about checking the existence of and computing abstractions, and
- discussing open problems and interesting directions for future research.

The paper is organized as follows. In Section II we recall the basic notions about data management that will be used in the paper. Section III presents the framework for abstraction in data integration, and Section IV discusses the relationship between this notion and view-based query processing. Sections V and VI illustrate recent technical results on computing abstractions belonging to specific classes of queries, namely unions of conjunctive queries, and monotone queries, respectively. Finally, Section VII concludes the paper by discussing possible future research on abstraction.

II. Preliminaries

We consider databases whose atomic values are from a denumerable set of constant symbols C, and we assume that every alphabet mentioned in this paper includes such symbols.

A *database schema* (or simply schema) $\mathcal{T}$ is a logical theory (set of axioms) over an alphabet $\mathcal{A}_\mathcal{T}$ (including a set of symbols called predicate symbols), and a *$\mathcal{T}$-database* is a finite structure[1] over $\mathcal{A}_\mathcal{T}$ that satisfies all axioms in $\mathcal{T}$. As usual, we often see a database as a set of facts, where each fact asserts that a tuple of constants satisfies a given predicate in $\mathcal{A}_\mathcal{T}$. In the usual meaning, a query over a schema $\mathcal{T}$, called $\mathcal{T}$-query, is a function associating to each $\mathcal{T}$-database a finite set of tuples of constants. A $\mathcal{T}$-query q is *monotone* if for each pair of $\mathcal{T}$-databases D_1, D_2 such that $D_1 \subseteq D_2$ we have $q^{D_1} \subseteq q^{D_2}$.

In this paper we extend the notion of query to functions that are applied to sets of databases: a *$\mathcal{T}$-query* of arity n is a function associating to each set of $\mathcal{T}$-databases a finite set of n-tuples of constants in C. Note that, if the set is a singleton, we obtain the usual notion. For a $\mathcal{T}$-query q and a set of $\mathcal{T}$-databases Σ, q^Σ denotes the *answers of q for Σ*, i.e., the set of tuples obtained by applying q to Σ, and we simply write q^D instead of $q^{\{D\}}$. We observe that a common method for defining a query q for a nonempty set Σ, based on the semantics of q for a single database, is through the certain answer semantics: the answers of q for Σ is the set of *certain answers*, defined as $\bigcap_{D\in\Sigma} q^D$. For two $\mathcal{T}$-queries q_1 and q_2, we write $q_1 \sqsubseteq q_2$ if $q_1^\Sigma \subseteq q_2^\Sigma$ for each set Σ of $\mathcal{T}$-databases, and we write $q_1 \equiv q_2$ if both $q_1 \sqsubseteq q_2$ and $q_2 \sqsubseteq q_1$.

Queries are often specified in terms of expressions in a particular language. In this paper, when we talk about a query language $\mathcal{L}$, we mean that the class of queries defined by such language are formulated in terms of expressions belonging to $\mathcal{L}$. A *query language is monotone* if all queries in such language are monotone. Notable monotone query languages correspond to the classes of conjunctive queries and unions of conjunctive queries. A *conjunctive query (CQ)* q over schema $\mathcal{T}$ 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, all appearing in ϕ, $\bar{y}$ is a tuple of variables, called the *existential variables* of q, and $\phi(\bar{x},\bar{y})$ is a finite conjunction of atoms over $\mathcal{T}$, where an atom is a predicate symbol applied to variables. We often write $\{\bar{x} \mid \exists\bar{y}.\phi(\bar{x},\bar{y})\}$ simply as $\phi(\bar{x})$, and we assume that we can form atoms of any arity using $\top$ as predicate symbol, where such atoms are always interpreted true. Given a CQ $q = \{\bar{x} \mid \exists\bar{y}.\phi(\bar{x},\bar{y})\}$, we say that an existential variable $y \in \bar{y}$ is a *join existential variable* of q if it occurs more than once in the atoms of $\phi(\bar{x},\bar{y})$. In what follows, we say that a CQ q is a *conjunctive query with join-free existential variables (CQJFE)* if there is no join existential variable occurring in q.

The arity of q is the arity of $\bar{x}$. A *union of conjunctive queries (UCQ)* (resp., *union of conjunctive queries with join-free existential variables (UCQJFE)*) is a finite union of CQs (resp., CQJFEs) with same arity, called its *disjuncts*. Note that, for a UCQ q and a $\mathcal{T}$-database D that can be seen as a First Order Logic (FOL) interpretation over $\mathcal{A}_\mathcal{T}$, q^D is obtained by evaluating the FOL formula expressing q over D. Other classes of queries considered in this paper, such as Datalog, Disjunctive Datalog, and Disjunctive Datalog with inequalities (denoted by DD$^\neq$), are generalizations of UCQs. For CQs, UCQs, and the queries of the above mentioned classes, we assume that, when they are evaluated over a set of databases, the certain answer semantics is used. Observe that all these languages are monotone query languages.

View-based query processing is a general term denoting several tasks related to the presence of views in databases. A set of views $\mathcal{V}$ over a schema $\mathcal{T}$ is constituted by a finite set of view predicate symbols, where each $V \in \mathcal{V}$ has a specific arity, and an associated *view definition* $V_\mathcal{T}$, i.e., a query over $\mathcal{T}$ of the same arity of V. An extension of a view V is simply a set of facts for V, and a *$\mathcal{V}$-extension* $\mathcal{E}$ is constituted by an extension for each view in $\mathcal{V}$. Given a $\mathcal{T}$-database D, we denote by $\mathcal{V}(D)$ the $\mathcal{V}$-extension $\{V(\bar{c}) \mid V \in \mathcal{V} \text{ and } \bar{c} \in V_\mathcal{T}^D\}$. In what follows, we use the term $\mathcal{L}$ views to indicate a set of views in which all view definitions are queries expressed in the query language $\mathcal{L}$.

Two particular notions have been subject to extensive investigations in the view-based processing literature, namely *view-based query rewriting* and *view-based query answering* [15], [16].

In the former notion, originated in [17], we are given a query $q_\mathcal{T}$ over a schema $\mathcal{T}$ and a set of views $\mathcal{V}$ over $\mathcal{T}$, and the goal is to reformulate $q_\mathcal{T}$ into a query $q_\mathcal{V}$, called a $\mathcal{V}$-rewriting, in terms of the view predicate symbols of $\mathcal{V}$. We obtain different variants of $\mathcal{V}$-rewritings depending on the relationship between $q_\mathcal{T}$ and $q_\mathcal{V}$ we aim at. We call $q_\mathcal{V}$ (i) a *$\mathcal{V}$-rewriting of $q_\mathcal{T}$ under exact views*, or simply *$\mathcal{V}$-rewriting of $q_\mathcal{T}$*, if for every $\mathcal{T}$-database D it holds that $q_\mathcal{V}^{\mathcal{V}(D)} \subseteq q_\mathcal{T}^D$, (ii) an *exact $\mathcal{V}$-rewriting of $q_\mathcal{T}$* if for every $\mathcal{T}$-database D it holds that $q_\mathcal{V}^{\mathcal{V}(D)} = q_\mathcal{T}^D$. Note that, if we fix a specific query language $\mathcal{L}_\mathcal{V}$ for expressing $\mathcal{V}$-rewritings, we might lose power in expressing $\mathcal{V}$-rewritings. In this case, a reasonable goal is to compute $\mathcal{V}$-rewritings expressible in $\mathcal{L}_\mathcal{V}$ that are "maximal" in the class $\mathcal{L}_\mathcal{V}$. Formally, we say that a query $q_\mathcal{V} \in \mathcal{L}_\mathcal{V}$ is an *$\mathcal{L}_\mathcal{V}$-maximal $\mathcal{V}$-rewriting of $q_\mathcal{T}$*, if (i) $q_\mathcal{V}$ is a $\mathcal{V}$-rewriting of $q_\mathcal{T}$; and (ii) there is no $q_1 \in \mathcal{L}_\mathcal{V}$ such that (a) q_1 is a $\mathcal{V}$-rewriting of $q_\mathcal{T}$, (b) $q_\mathcal{V}^{\mathcal{V}(D)} \subseteq q_1^{\mathcal{V}(D)}$ for each $\mathcal{T}$-database D, and (c) there is a $\mathcal{T}$-database D for which $q_\mathcal{V}^{\mathcal{V}(D)} \subsetneq q_1^{\mathcal{V}(D)}$.

As argued in [18], given $q_\mathcal{T}$ and $\mathcal{V}$, the problem of checking whether there exists an exact $\mathcal{V}$-rewriting of $q_\mathcal{T}$ (called *losslessness with respect to rewriting* [16]) is equivalent to the problem, called *view determinacy* [18], of checking whether $q_\mathcal{T}$ is determined by $\mathcal{V}$, denoted $\mathcal{V} \twoheadrightarrow q_\mathcal{T}$, i.e., whether $\mathcal{V}(D_1) = \mathcal{V}(D_2)$ implies $q_\mathcal{T}^{D_1} = q_\mathcal{T}^{D_2}$ for each pair of $\mathcal{T}$-databases D_1 and D_2. Indeed, on the one hand, if $\mathcal{V} \twoheadrightarrow q_\mathcal{T}$, then the function $q_\mathcal{V}$ associating to each $\mathcal{V}(D)$ the tuples $q_\mathcal{T}^D$, for each $\mathcal{T}$-database D, is an exact $\mathcal{V}$-rewriting of $q_\mathcal{T}$, on the other hand, if $\mathcal{V} \not\twoheadrightarrow q_\mathcal{T}$, then such $q_\mathcal{V}$ is not a function, and

[1] In principle, we could also consider databases that are infinite structures.

hence an exact $\mathcal{V}$-rewriting of $q_\mathcal{S}$ cannot exist.

In the latter approach, originated in [19], besides $q_\mathcal{T}$ and $\mathcal{V}$ we are also given a $\mathcal{V}$-extension $\mathcal{E}$, and the goal is to compute the so-called *certain answers of $q_\mathcal{T}$ w.r.t. $\mathcal{V}$ and $\mathcal{E}$*, denoted by $cert^{\mathcal{E}}_{q_\mathcal{T},\mathcal{V}}$, which are those tuples of constants $\bar{c}$ such that $\bar{c} \in q^D_\mathcal{T}$ for each $\mathcal{S}$-database D satisfying $\mathcal{E} \subseteq \mathcal{V}(D)$. We denote by $cert_{q_\mathcal{T},\mathcal{V}}$ the query over $\mathcal{V}$ that, for every $\mathcal{V}$-extension $\mathcal{E}$, computes the certain answers of $q_\mathcal{T}$ w.r.t. $\mathcal{V}$ and $\mathcal{E}$, and we call $cert_{q_\mathcal{T},\mathcal{V}}$ the *perfect $\mathcal{V}$-rewriting of $q_\mathcal{T}$ under sound views*, or simply *perfect $\mathcal{V}$-rewriting of $q_\mathcal{T}$*.

III. Framework

A *data integration system* $\mathcal{J}$ is specified by a triple $\langle \mathcal{G}, \mathcal{S}, \mathcal{M} \rangle$, where $\mathcal{G}$, the global schema, is a schema over an alphabet $\mathcal{A}_\mathcal{G}$, $\mathcal{S}$, the source schema, is a schema over an alphabet $\mathcal{A}_\mathcal{S}$ (disjoint from $\mathcal{A}_\mathcal{G}$, except for the set C), and $\mathcal{M}$ is a mapping relating $\mathcal{S}$ to $\mathcal{G}$. Specifically, $\mathcal{M}$ is a finite set of assertions of the form $q_\mathcal{S} \rightarrow q_\mathcal{G}$, where $q_\mathcal{S}$ is an $\mathcal{S}$-query and $q_\mathcal{G}$ is a $\mathcal{G}$-query of the same arity as $q_\mathcal{S}$.

The semantics of $\mathcal{J}$ is defined relative to an $\mathcal{S}$-database D, and, intuitively, is the set of all interpretations for $\mathcal{J}$ over the domain C that satisfy both the axioms of $\mathcal{G}$ and the mapping $\mathcal{M}$ with respect to D. An interpretation for $\mathcal{G}$ is actually a $\mathcal{G}$-database, and one such database B satisfies $\mathcal{M}$ with respect to D, denoted by $(D, B) \models \mathcal{M}$, if it satisfies all assertions in $\mathcal{M}$, where B satisfies $(q_\mathcal{S} \rightarrow q_\mathcal{G}) \in \mathcal{M}$ with respect to D if $q^D_\mathcal{S} \subseteq q^B_\mathcal{G}$.

Formally, the semantics of $\mathcal{J}$ relative to D, denoted as $mod(\mathcal{J}, D)$, is defined as $\{B \mid B \text{ is a } \mathcal{G}\text{-database such that } (D, B) \models \mathcal{M}\}$. We say that D is consistent with $\mathcal{J}$ if $mod(\mathcal{J}, D) \neq \emptyset$. The answers to a $\mathcal{G}$-query q w.r.t. a data integration system $\mathcal{J} = \langle \mathcal{G}, \mathcal{S}, \mathcal{M} \rangle$ and an $\mathcal{S}$-database D is simply $q^{mod(\mathcal{J},D)}$, that we often write simply as $q^{\mathcal{J},D}$.

For two $\mathcal{G}$-queries q_1 and q_2, we write $q_1 \sqsubseteq_\mathcal{J} q_2$ if $q_1^{\mathcal{J},D} \subseteq q_2^{\mathcal{J},D}$ for each $\mathcal{S}$-database D; $q_1 \sqsubset_\mathcal{J} q_2$ and $\mathcal{J}$-equivalence are defined accordingly.

Specific classes of mappings considered in the literature are GAV, LAV, GLAV, PGAV and SPGAV. We introduce them under the assumption that the queries appearing in mapping assertions are conjunctive queries or restricted forms thereof.

A GLAV mapping is a set of assertions of the form $q_S(\vec{x}) \rightarrow q_\mathcal{G}(\vec{x})$, where both $q_\mathcal{S}$ and $q_\mathcal{G}$ are conjunctive queries over $\mathcal{S}$ and $\mathcal{G}$ respectively, with distinguished variables $\vec{x}$, implicitly universally quantified.

A GAV mapping is a special case of GLAV, constituted by a set of assertions of the form $q_\mathcal{S}(\vec{x}) \rightarrow A(\vec{x})$, where (i) $q_\mathcal{S}$ is a conjunctive query over $\mathcal{S}$, (ii) $A \in \mathcal{A}_\mathcal{G}$, and (iii) the arity of A is the same as the arity $\vec{x}$. A pure GAV mapping (PGAV) is a GAV mapping in which each assertion $q_\mathcal{S}(\vec{x}) \rightarrow A(\vec{x})$ is such that no repeated variables appear in $\vec{x}$. A PGAV mapping is called SPGAV (PGAV with single assertion per predicate) if it does not contain a pair of assertions with the same predicate symbol in the head.

A LAV mapping is a special case of GLAV, constituted by a set of assertions of the form $A(\vec{x}) \rightarrow q_\mathcal{G}(\vec{x})$, where (i) $A \in \mathcal{A}_\mathcal{S}$, (ii) $q_\mathcal{G}$ is a conjunctive query over $\mathcal{G}$ with distinguished variables $\vec{x}$, and (iii) the arity of A is the same as the arity $\vec{x}$.

In what follows, we implicitly refer to a data integration system $\mathcal{J} = \langle \mathcal{G}, \mathcal{S}, \mathcal{M} \rangle$, and when we denote a query by $q_\mathcal{G}$ (resp., $q_\mathcal{S}$) we mean that the query is a $\mathcal{G}$-query (resp., $\mathcal{S}$-query). We follow [10] for the basic definitions related to abstraction. We say that $q_\mathcal{G}$ is a *perfect $\mathcal{J}$-abstraction of $q_\mathcal{S}$* if $q_\mathcal{G}^{\mathcal{J},D} = q_\mathcal{S}^D$, for each $\mathcal{S}$-database D consistent with $\mathcal{J}$. Clearly, if a perfect $\mathcal{J}$-abstraction of $q_\mathcal{S}$ exists, then it is unique up to $\mathcal{J}$-equivalence, and therefore in the following we will talk about *the* perfect $\mathcal{J}$-abstraction of $q_\mathcal{S}$.

Borrowing the above mentioned idea on view determinacy [18], it is easy to prove the following characterization theorem for the existence of a perfect $\mathcal{J}$-abstraction for $q_\mathcal{S}$.

Theorem 1. *There exists a perfect $\mathcal{J}$-abstraction of $q_\mathcal{S}$ if and only if for all pair D, D' of $\mathcal{S}$-databases, $mod(\mathcal{J}, D) = mod(\mathcal{J}, D')$ implies $q_\mathcal{S}^D = q_\mathcal{S}^{D'}$.*

As the condition for a global schema query to be a perfect $\mathcal{J}$-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 $q_\mathcal{G}$ is a *complete* (resp., *sound*) *$\mathcal{J}$-abstraction of $q_\mathcal{S}$* if $q_\mathcal{S}^D \subseteq q_\mathcal{G}^{\mathcal{J},D}$ (resp., $q_\mathcal{G}^{\mathcal{J},D} \subseteq q_\mathcal{S}^D$), for each $\mathcal{S}$-database D consistent with $\mathcal{J}$.

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 $\mathcal{L}_\mathcal{G}$ is a class of queries, we say that a global schema query $q_\mathcal{G} \in \mathcal{L}_\mathcal{G}$ is an *$\mathcal{L}_\mathcal{G}$-minimally complete* (resp., *$\mathcal{L}_\mathcal{G}$-maximally sound*) *$\mathcal{J}$-abstraction of $q_\mathcal{S}$* if $q_\mathcal{G}$ is a complete (resp., sound) $\mathcal{J}$-abstraction of $q_\mathcal{S}$ and there is no global schema query $q'_\mathcal{G} \in \mathcal{L}_\mathcal{G}$ such that $q'_\mathcal{G}$ is a complete (resp., sound) $\mathcal{J}$-abstraction of $q_\mathcal{S}$ and $q'_\mathcal{G} \sqsubset_\mathcal{J} q_\mathcal{G}$ (resp., $q_\mathcal{G} \sqsubset_\mathcal{J} q'_\mathcal{G}$).

The class of monotone queries in the context of a data integration system is particularly important in this paper. A $\mathcal{G}$-query q is *monotone in the context of a data integration system $\mathcal{J} = \langle \mathcal{G}, \mathcal{S}, \mathcal{M} \rangle$* if for every pair D, D' of $\mathcal{S}$-databases, $mod(\mathcal{J}, D) \subseteq mod(\mathcal{J}, D')$ implies $q^{\mathcal{J},D'} \subseteq q^{\mathcal{J},D}$. This captures the intuition of monotonicity of queries in logic: when the knowledge possessed by the system increases (i.e., the set of models shrinks), the answers to a query does not decrease. Note that the class of monotone queries includes the whole class of First Order queries, that is arguably the most studied class in both the Knowledge Representation and the Data Management literature. In the following, we use $\mathfrak{M}^J$ to denote the class of monotone queries in the context of $\mathcal{J} = \langle \mathcal{G}, \mathcal{S}, \mathcal{M} \rangle$, and when $\mathcal{J}$ is understood, we simply use $\mathfrak{M}$.

It is easy to see that if an $\mathfrak{M}$-maximally sound $\mathcal{J}$-abstraction of $q_\mathcal{S}$ exists, then it is unique up to $\mathcal{J}$-equivalence, and the same holds for $\mathfrak{M}$-minimally complete $\mathcal{J}$-abstractions. Analogously, if a UCQ-maximally sound (resp., UCQ-minimally complete) $\mathcal{J}$-abstraction of $q_\mathcal{S}$ exists, then it is unique up to

$\mathcal{J}$-equivalence. Thus, in the following, we simply talk about *the* $\mathfrak{M}$-maximally (resp., UCQ-maximally) sound and *the* $\mathfrak{M}$-minimally (resp., UCQ-minimally) complete $\mathcal{J}$-abstraction of $q_{\mathcal{S}}$.

In the next sections, we will deal with data integration systems of a specific form, namely where (i) the mapping is of type GLAV or special cases of GLAV, and (ii) if not otherwise stated, the set of axioms of both the global schema and the source schema is empty. Also, we will limit our analysis to abstractions of UCQ source queries.

IV. Comparison with View-based Query Processing

It is well-known that there is a relationship between data integration and view-based query processing, based on the idea that the sources of a LAV data integration systems can be considered as views over the global schema, in particular sound views [2]. In this section, we take another approach and establish a relationship between GAV data integration systems and views, based on the idea that the elements of the global schema can be considered sound views over source databases.

We start by describing how to obtain, from any data integration system $\mathcal{J}$ with PGAV mapping, a suitable set of UCQ views[2] $\mathcal{V}_{\mathcal{J}}$, and, viceversa, from any set of UCQ views $\mathcal{V}$, a suitable data integration system $\mathcal{J}_{\mathcal{V}}$ with PGAV mapping.

For a data integration system $\mathcal{J} = \langle \mathcal{G}, \mathcal{S}, \mathcal{M} \rangle$ with $\mathcal{M} \in$ PGAV, the set of UCQ views $\mathcal{V}_{\mathcal{J}}$ is such that (i) the set of view symbols coincides with $\mathcal{A}_{\mathcal{G}}$, and (ii) for each view symbol g, the associated view definition $g_{\mathcal{S}}$ is the following UCQ over $\mathcal{S}$:

$$\{\bar{x_1} \mid \exists \bar{y_1}.\phi^1_{\mathcal{S}}(\bar{x_1}, \bar{y_1})\} \cup \ldots \cup \{\bar{x_l} \mid \exists \bar{y_l}.\phi^l_{\mathcal{S}}(\bar{x_l}, \bar{y_l})\},$$

where we have one disjunct $\exists \bar{y_i}.\phi^i_{\mathcal{S}}(\bar{x_i}, \bar{y_i})$ for each mapping assertion in $\mathcal{M}$ of the form $\exists \bar{y_i}.\phi^i_{\mathcal{S}}(\bar{x_i}, \bar{y_i}) \rightarrow g(\bar{x_i})$. Note that, if $\mathcal{M} \in$ SPGAV, then all view definitions in $\mathcal{V}_{\mathcal{J}}$ are CQs.

Example 2. *Let $\mathcal{J} = \langle \mathcal{G}, \mathcal{S}, \mathcal{M} \rangle$ be a data integration system such that $\mathcal{M} = \{m_1, m_2, m_3\}$ with:*

$$m_1 : \exists y_1, y_2.s_1(y_1, x, x) \wedge s_2(x, y_2, y_2) \rightarrow g_1(x)$$
$$m_2 : \exists y_1, y_2, y_3.s_1(y_1, x_1, x_2) \wedge s_2(x_2, y_2, y_3) \rightarrow g_2(x_1, x_2)$$
$$m_3 : \exists y_1.s_3(x_1, x_2, y_1) \rightarrow g_2(x_1, x_2)$$

Then, the UCQ views $\mathcal{V}_{\mathcal{J}}$ over $\mathcal{S}$ is $\mathcal{V}_{\mathcal{J}} = \{g_1, g_2\}$, where $g_{1\mathcal{S}} = \{x \mid \exists y_1, y_2.s_1(y_1, x, x) \wedge s_2(x, y_2, y_2)\}$ and $g_{2\mathcal{S}} = \{x_1, x_2 \mid \exists y_1, y_2, y_3.s_1(y_1, x_1, x_2) \wedge s_2(x_2, y_2, y_3)\} \cup \{x_1, x_2 \mid \exists y_1.s_3(x_1, x_2, y_1)\}$. △

For a set of UCQ views $\mathcal{V}$ over a schema $\mathcal{S}$, the data integration system $\mathcal{J}_{\mathcal{V}} = \langle \mathcal{G}, \mathcal{S}, \mathcal{M} \rangle$ is such that (i) $\mathcal{A}_{\mathcal{G}}$ coincides with the view predicate symbols in $\mathcal{V}$, (ii) $\mathcal{G}$ has no axiom, and (iii) $\mathcal{M}$ is defined as follows: for each view symbol $V \in \mathcal{V}$ and for each CQ $\{\bar{x} \mid \exists \bar{y}.\phi_{\mathcal{S}}(\bar{x}, \bar{y})\}$ that is a disjunct in the UCQ $V_{\mathcal{S}}$, the mapping $\mathcal{M}$ includes a mapping assertion of the form: $\exists \bar{y}.\phi_{\mathcal{S}}(\bar{x}, \bar{y}) \rightarrow V(\bar{x})$. Note that, in general, $\mathcal{M} \in$ PGAV. However, if $\mathcal{V}$ is a set of CQ views, then $\mathcal{M} \in$ SPGAV.

Example 3. *Let $\mathcal{V} = \{V_1, V_2\}$ be a set of UCQ views over $\mathcal{S}$ such that: $V_{1\mathcal{S}} = \{x_1, x_2, x_3 \mid s_3(x_1, x_2, x_3)\} \cup \{x_1, x_2, x_3 \mid \exists y.s_1(x_1, y) \wedge s_2(y, x_2, x_3)\} \cup \{x_1, x_2, x_3 \mid \exists y_1, y_2.s_1(x_1, y_1) \wedge s_4(y_2, x_2, x_3)\}$ and $V_{2\mathcal{S}} = \{x_1, x_2, x_3, x_4 \mid \exists y.s_1(x_1, x_2, y) \wedge s_3(y, x_3, x_4)\}$.*

Then, the data integration system is $\mathcal{J}_{\mathcal{V}} = \langle \mathcal{G}, \mathcal{S}, \mathcal{M} \rangle$, where $\mathcal{A}_{\mathcal{G}} = \{V_1, V_2\}$ and $\mathcal{M} = \{m_1, m_2, m_3, m_4\}$ with:

$$m_1 : s_3(x_1, x_2, x_3) \rightarrow V_1(x_1, x_2, x_3),$$
$$m_2 : \exists y.s_1(x_1, y) \wedge s_2(y, x_2, x_3) \rightarrow V_1(x_1, x_2, x_3),$$
$$m_3 : \exists y_1, y_2.s_1(x_1, y_1) \wedge s_4(y_2, x_2, x_3) \rightarrow V_1(x_1, x_2, x_3),$$
$$m_4 : \exists y.s_2(x_1, x_2, y) \wedge s_4(y, x_3, x_4) \rightarrow V_2(x_1, x_2, x_3, x_4).$$

△

For a data integration system $\mathcal{J}$ with PGAV mapping and a set of UCQ views $\mathcal{V}$, the pair $(\mathcal{J}, \mathcal{V})$ is said to be *coherent* if (i) the schema over which the set of views $\mathcal{V}$ is defined and the source of $\mathcal{J}$ coincide, and (ii) $\mathcal{J} = \mathcal{J}_{\mathcal{V}}$ or $\mathcal{V} = \mathcal{V}_{\mathcal{J}}$. In what follows, when we talk about a coherent pair $(\mathcal{J}, \mathcal{V})$, we use $\mathcal{S}$ to denote the common schema between $\mathcal{J}$ and $\mathcal{V}$.

Based on the relationship between $\mathcal{J}_{\mathcal{V}}$ and $\mathcal{V}_{\mathcal{J}}$, the following proposition provides a connection between existence of perfect abstractions and existence of exact rewritings.

Proposition 1. *If $(\mathcal{J}, \mathcal{V})$ is a coherent pair and $q_{\mathcal{S}}$ is an $\mathcal{S}$-query, then there exists a perfect $\mathcal{J}$-abstraction of $q_{\mathcal{S}}$ if and only if there exists an exact $\mathcal{V}$-rewriting of $q_{\mathcal{S}}$.*

Proof. We recall that there exists an exact $\mathcal{V}$-rewriting of $q_{\mathcal{S}}$ if and only if $\mathcal{V} \twoheadrightarrow q_{\mathcal{S}}$, i.e., $\mathcal{V}(D_1) = \mathcal{V}(D_2)$ implies $q_{\mathcal{S}}^{D_1} = q_{\mathcal{S}}^{D_2}$ for each pair of $\mathcal{S}$-databases D_1, D_2.

Let $\mathcal{J} = \langle \mathcal{G}, \mathcal{S}, \mathcal{M} \rangle$ with $\mathcal{M} \in$ PGAV. Due to Theorem 1, there exists a perfect $\mathcal{J}$-abstraction of $q_{\mathcal{S}}$ if and only if $mod(\mathcal{J}, D_1) = mod(\mathcal{J}, D_2)$ implies $q_{\mathcal{S}}^{D_1} = q_{\mathcal{S}}^{D_2}$ for each pair of $\mathcal{S}$-databases D_1, D_2. Since $\mathcal{G}$ has no axiom, every $\mathcal{S}$-database D is consistent with $\mathcal{J}$, and, if D is an $\mathcal{S}$-database and B is a $\mathcal{G}$-database, then $B \in mod(\mathcal{J}, D)$ if and only if $\mathcal{M}(D) \subseteq B$. It follows that there exists a perfect $\mathcal{J}$-abstraction of $q_{\mathcal{S}}$ if and only $\mathcal{M}(D_1) = \mathcal{M}(D_2)$ implies $q_{\mathcal{S}}^{D_1} = q_{\mathcal{S}}^{D_2}$ for each pair $\mathcal{S}$-databases D_1, D_2.

Then the claim trivially follows from the fact that, since $(\mathcal{J}, \mathcal{V})$ is a coherent pair, by construction $\mathcal{M}(D) = \mathcal{V}(D)$ holds for every $\mathcal{S}$-database D. □

In the rest of this section, when we use $\mathcal{L}$, we refer to a sublanguage of DD$^{\neq}$ (therefore, $\mathcal{L}$ is a monotone query language). By exploiting well-known results, we provide connections between the notion of $\mathcal{J}$-abstractions and $\mathcal{V}$-rewritings in the context of monotone query languages. To this end, we first introduce some terminology.

Given a mapping $\mathcal{M} \in$ PGAV relating $\mathcal{S}$ to $\mathcal{G}$ and a $\mathcal{G}$-query q in a certain query language $\mathcal{L}$, the *$\mathcal{M}$-unfolding of q* [2], denoted by $unf_{\mathcal{M}}(q)$, is the $\mathcal{S}$-query obtained by replacing each atom α occurring in the expression corresponding to q by

[2]When we refer to UCQ views, we in fact assume that view definitions are UCQs without *repeated variables* in the target list. We refer to [20] for the complications that can arise when this assumption is removed.

the logical disjunction of all the left-hand sides of the mapping assertions in $\mathcal{M}$ having the predicate symbol of α in the right-hand side (being careful to use unique variables in place of those variables that appear in the left-hand side of the mapping assertions but not in the right-hand side of those).

Given a set of UCQ views $\mathcal{V}$ over $\mathcal{S}$ and a $\mathcal{V}$-query q in a certain query language $\mathcal{L}$, the *$\mathcal{V}$-expansion of q* [17], denoted by $exp_{\mathcal{V}}(q)$, is the $\mathcal{S}$-query obtained by replacing each atom α occurring in in the expression corresponding to q by the view definition associated to the view predicate name of α (again, being careful to use unique variables in place of those variables that appear in the bodies of the view but not in the heads of those).

Proposition 2. *If $(\mathcal{J}, \mathcal{V})$ is a coherent pair, $q_{\mathcal{S}}$ is an $\mathcal{S}$-query in $\mathcal{L}$, and q is a query in $\mathcal{L}$, then q is a sound (resp., perfect) $\mathcal{J}$-abstraction of $q_{\mathcal{S}}$ if and only if q is a $\mathcal{V}$-rewriting (resp., exact $\mathcal{V}$-rewriting) of $q_{\mathcal{S}}$.*

Proof. For data integration systems $\mathcal{J} = \langle \mathcal{G}, \mathcal{S}, \mathcal{M} \rangle$ with $\mathcal{M} \in \mathsf{PGAV}$ and monotone query languages $\mathcal{L}$, it is well-known that $q^{\mathcal{J},D} = unf_{\mathcal{M}}(q)^D$ for every $q \in \mathcal{L}$ and $\mathcal{S}$-database D [2]. So, by definition, q is a sound (resp., perfect) $\mathcal{J}$-abstraction of $q_{\mathcal{S}}$ if and only if $unf_{\mathcal{M}}(q) \sqsubseteq q_{\mathcal{S}}$ (resp., $unf_{\mathcal{M}}(q) \equiv q_{\mathcal{S}}$), for every pair of queries $q_{\mathcal{S}}, q \in \mathcal{L}$. On the other hand, for UCQ views $\mathcal{V}$ and monotone query languages $\mathcal{L}$, it is easy to see that q is a $\mathcal{V}$-rewriting (resp., exact $\mathcal{V}$-rewriting) of $q_{\mathcal{S}}$ if and only if $exp_{\mathcal{V}}(q) \sqsubseteq q_{\mathcal{S}}$ (resp., $exp_{\mathcal{V}}(q) \equiv q_{\mathcal{S}}$), for every pair of queries $q_{\mathcal{S}}, q \in \mathcal{L}$.

Let $\mathcal{J} = \langle \mathcal{G}, \mathcal{S}, \mathcal{M} \rangle$. The claim holds because the fact that $(\mathcal{J}, \mathcal{V})$ is a coherent pair implies $unf_{\mathcal{M}}(q) = exp_{\mathcal{V}}(q)$ for every q in $\mathcal{L}$. □

Actually, as shown in [21, Lemma 1], if $\mathcal{L}$ allows for the union operator, then for any pair of UCQ views $\mathcal{V}$ over $\mathcal{S}$ and query $q_{\mathcal{S}} \in \mathcal{L}$ over $\mathcal{S}$, if an $\mathcal{L}$-maximal $\mathcal{V}$-rewriting of $q_{\mathcal{S}}$ exists, then it is unique up to $\mathcal{V}$-equivalence, and, moreover, it coincides with the perfect $\mathcal{V}$-rewriting of $q_{\mathcal{S}}$[3]. From Proposition 2 and the above observation, we can derive the following result.

Corollary 1. *If $(\mathcal{J}, \mathcal{V})$ is a coherent pair and $\mathcal{L}$ allows for the union operator, then for every pair of queries $q_{\mathcal{S}}, q \in \mathcal{L}$, we have that q is the $\mathcal{L}$-maximally sound $\mathcal{J}$-abstraction of $q_{\mathcal{S}}$ if and only if q is the perfect $\mathcal{V}$-rewriting of $q_{\mathcal{S}}$.*

By exploiting the above provided relationships, we are now ready to investigate how results and techniques from the view-based processing literature can be directly translated into results and techniques in the context of abstraction, and viceversa.

A. Novel results on abstraction

By combining Proposition 1 with a well-known undecidability result about view determinacy, we can derive a novel, negative result about an arguably fundamental problem for the notion of abstraction, namely the *existence problem* (with no restrictions on the query language to express perfect abstractions) of perfect abstractions, even in very restricted settings.

Theorem 2. *Given a data integration system $\mathcal{J} = \langle \mathcal{G}, \mathcal{S}, \mathcal{M} \rangle$ with $\mathcal{M} \in \mathsf{SPGAV}$ and a CQ $\mathcal{S}$-query $q_{\mathcal{S}}$, checking whether there exists a perfect $\mathcal{J}$-abstraction of $q_{\mathcal{S}}$ is undecidable.*

Proof. Proposition 1 provides a reduction from the view determinacy problem in the case of CQ views and CQs $q_{\mathcal{S}}$ to our problem. Specifically, given a set of CQ views $\mathcal{V}$ over a schema $\mathcal{S}$ and a CQ $\mathcal{S}$-query $q_{\mathcal{S}}$, we have that $\mathcal{V} \twoheadrightarrow q_{\mathcal{S}}$ (i.e., there exists an exact $\mathcal{V}$-rewriting of $q_{\mathcal{S}}$) if and only if there exists a perfect $\mathcal{J}_{\mathcal{V}}$-abstraction of $q_{\mathcal{S}}$, where $\mathcal{J}_{\mathcal{V}} = \langle \mathcal{G}, \mathcal{S}, \mathcal{M} \rangle$ with $\mathcal{M} \in \mathsf{SPGAV}$. Since the view determinacy problem in the case of CQ views $\mathcal{V}$ and CQs $q_{\mathcal{S}}$ is undecidable [23], it follows that checking whether there exists a perfect $\mathcal{J}$-abstraction of $q_{\mathcal{S}}$ is undecidable as well. □

By exploiting Corollary 1, we now illustrate how to use off-the-shelf algorithms for rewriting queries in the presence of views as algorithms for computing abstractions. By results of [17], for CQ views $\mathcal{V}$, perfect $\mathcal{V}$-rewritings of UCQs $q_{\mathcal{S}}$ can be always expressed as UCQs, and can be always computed (e.g., by means of the *bucket* algorithm [24] or the *MiniCon* algorithm [25]). Thus Corollary 1 implies that, given a data integration system $\mathcal{J} = \langle \mathcal{G}, \mathcal{S}, \mathcal{M} \rangle$ with $\mathcal{M} \in \mathsf{SPGAV}$ and a UCQ $\mathcal{S}$-query $q_{\mathcal{S}}$, we can compute the UCQ-maximally sound $\mathcal{J}$-abstraction of $q_{\mathcal{S}}$ as follows: (*i*) compute $\mathcal{V}_{\mathcal{J}}$, and (*ii*) compute and return the UCQ corresponding to the perfect $\mathcal{V}_{\mathcal{J}}$-rewriting of $q_{\mathcal{S}}$.

Corollary 2. *If $\mathcal{J}$ is a data integration system with SPGAV mapping and $q_{\mathcal{S}}$ is a UCQ $\mathcal{S}$-query, then the UCQ-maximally sound $\mathcal{J}$-abstraction of $q_{\mathcal{S}}$ exists and is computable.*

Things get more complicated when we consider a data integration system $\mathcal{J}$ with PGAV mappings, which are clearly more expressive than SPGAV, for which $\mathcal{V}_{\mathcal{J}}$ is a set of UCQ views, rather than CQ views. Indeed, for UCQ views $\mathcal{V}$, UCQ-maximal $\mathcal{V}$-rewritings of CQs $q_{\mathcal{S}}$ are not guaranteed to exist [20], [21], and thus, in general, perfect $\mathcal{V}$-rewritings of CQs $q_{\mathcal{S}}$ are not expressible as UCQs. However, the perfect $\mathcal{V}$-rewritings of UCQs (actually, even of *Datalog* queries) $q_{\mathcal{S}}$ can always be expressed in $\mathsf{DD}^{\neq}$, and can always be computed using the technique presented in [21]. Thus, Corollary 1 implies that, given a data integration system $\mathcal{J} = \langle \mathcal{G}, \mathcal{S}, \mathcal{M} \rangle$ with $\mathcal{M} \in \mathsf{PGAV}$ and a UCQ $\mathcal{S}$-query $q_{\mathcal{S}}$, we can compute the $\mathsf{DD}^{\neq}$-maximally sound $\mathcal{J}$-abstraction of $q_{\mathcal{S}}$ as follows: (*i*) compute $\mathcal{V}_{\mathcal{J}}$, and (*ii*) compute and return the $\mathsf{DD}^{\neq}$ query corresponding to the perfect $\mathcal{V}_{\mathcal{J}}$-rewriting of $q_{\mathcal{S}}$.

Corollary 3. *If $\mathcal{J}$ is a data integration system with PGAV mapping and $q_{\mathcal{S}}$ is a UCQ $\mathcal{S}$-query, then the $\mathsf{DD}^{\neq}$-maximally sound $\mathcal{J}$-abstraction of $q_{\mathcal{S}}$ exists and is computable.*

[3] This is not the case when view definitions are expressed as *regular path queries* rather than UCQs [22].

B. Novel results on view-based query processing

As already observed, [20], [21] show that for a given set $\mathcal{V}$ of UCQ views, UCQ-maximal $\mathcal{V}$-rewritings of CQs may not exist. Combined with an observation made above, this means that perfect $\mathcal{V}$-rewritings of CQs are in general not expressible as UCQs. We point out that the CQ $q_{\mathcal{S}}$ used to prove such results contain more than one join existential variable. As a consequence, in the case of UCQ views $\mathcal{V}$, it is still open whether (*i*) the result holds even for $q_{\mathcal{S}}$ with just one join existential variable (*ii*) perfect $\mathcal{V}$-rewritings of UCQJFEs are expressible as UCQs. By combining Corollary 1 with results of [10] (that we will discuss in Section V), we can actually answer positively to both questions.

Corollary 4. *For a set $\mathcal{V}$ of UCQ views, the UCQ-maximal $\mathcal{V}$-rewritings of $q_{\mathcal{S}}$ may not exist, even if $q_{\mathcal{S}}$ is a CQ with one join existential variable.*

On the other hand, in Section V, we will show that for a data integration systems $\mathcal{J}$ with PGAV mapping, UCQ-maximally sound $\mathcal{J}$-abstractions of UCQJFEs are guaranteed to exist, and we will provide an algorithm to compute them (Theorem 5). Thus, given a set of UCQ views $\mathcal{V}$ over a schema $\mathcal{S}$ and a UCQJFE $\mathcal{S}$-query $q_{\mathcal{S}}$, we can compute the perfect $\mathcal{V}$-rewriting of $q_{\mathcal{S}}$ as follows: (*i*) compute $\mathcal{J}_{\mathcal{V}}$, and (*ii*) compute and return the UCQ-maximally sound $\mathcal{J}_{\mathcal{V}}$-abstraction of $q_{\mathcal{S}}$. This leads to the following positive result for $\mathcal{V}$-rewritings of UCQJFEs.

Corollary 5. *If $\mathcal{V}$ is a set of UCQ views and $q_{\mathcal{S}}$ is a UCQJFE $\mathcal{S}$-query, then the perfect $\mathcal{V}$-rewriting of $q_{\mathcal{S}}$ is computable and can be expressed as a UCQ.*

V. UCQ Abstractions

In this section we investigate the problem of checking the existence of abstractions in the class UCQ, and of their computation. We first study the case of UCQ-minimally complete $\mathcal{J}$-abstractions, then we switch to UCQ-maximally sound $\mathcal{J}$-abstractions, and finally we tackle perfect $\mathcal{J}$-abstractions in the class UCQ. We observe that all the results presented in this section appear in [10].

On the positive side, we show that UCQ-minimally complete abstractions always exist, by providing an algorithm to compute them. In a nutshell, given a data integration system $\mathcal{J} = \langle \mathcal{G}, \mathcal{S}, \mathcal{M} \rangle$ and a UCQ $q_{\mathcal{S}} = q^1_{\mathcal{S}} \cup \ldots \cup q^n_{\mathcal{S}}$, an algorithm to compute the UCQ minimally-complete $\mathcal{J}$-abstraction of $q_{\mathcal{S}}$ returns the union of CQs of the form $\{\bar{x}_i \mid \exists \bar{y}_i . \mathcal{M}(q^i_{\mathcal{S}}) \wedge \top(\bar{x}_i)\}$ obtained by simply "applying" the mapping $\mathcal{M}$ to each CQ $q^i_{\mathcal{S}}$ in $q_{\mathcal{S}}$, using $\top$ to bind the distinguished variables that are not involved in the application of $\mathcal{M}$ to $q^i_{\mathcal{S}}$. Formally, applying the GLAV mapping $\mathcal{M}$ to a CQ q means to chase [26] the atoms in q by using the tuple generating dependencies corresponding to the assertions in $\mathcal{M}$.

Theorem 3. *The UCQ-minimally complete $\mathcal{J}$-abstraction of $q_{\mathcal{S}}$ always exists and is computable.*

On the negative side, the following shows that UCQ-maximally sound abstractions may not exist.

Theorem 4. *The UCQ-maximally sound $\mathcal{J}$-abstractions of $q_{\mathcal{S}}$ may not exist if at least one of the following is true:*
(a) *$q_{\mathcal{S}}$ contains a join existential variable;*
(b) *$\mathcal{M}$ contains a* LAV *mapping assertion;*
(c) *$\mathcal{M}$ contains a non-*PGAV *mapping assertion.*

Interestingly, in order to illustrate the case (a) of the above theorem we can refer to a slight modification of the data integration system $\mathcal{J}$ introduced in Example 1. In particular, let $\mathcal{J}_1 = \langle \mathcal{G}, \mathcal{S}, \mathcal{M}_1 \rangle$ be obtained from $\mathcal{J}$ by removing from $\mathcal{M}$ the mapping m_1, and consider the query $q^4_{\mathcal{S}}$ of Example 1. Note that $\mathcal{M}_1 \in$ PGAV and $q^4_{\mathcal{S}}$ contains a join existential variable, x. Clearly, removing m_1 has no impact on the abstraction of $q^4_{\mathcal{S}}$. Thus, as already discussed in Example 1, there exists no UCQ-maximally sound $\mathcal{J}_1$-abstraction of $q^4_{\mathcal{S}}$.

Motivated by Theorem 4, we next introduce a specific scenario, that we call *restricted*, obtained from the general one by limiting the mapping language to PGAV, and $q_{\mathcal{S}}$ to be UCQJFEs. It can be shown that for such a restricted scenario, UCQ-maximally sound abstractions always exist. Intuitively, the latter can be derived by showing that for any UCQJFE $q_{\mathcal{S}}$ and data integration system $\mathcal{J} = \langle \mathcal{G}, \mathcal{S}, \mathcal{M} \rangle$ with $\mathcal{M} \in$ PGAV, a CQ-maximally sound $\mathcal{J}$-abstraction of $q_{\mathcal{S}}$ may comprise at most $k^{\mathcal{M}}_{q_{\mathcal{S}}}$ atoms, where $k^{\mathcal{M}}_{q_{\mathcal{S}}}$ is an integer that depends on the number of atoms occurring in $q_{\mathcal{S}}$ and the number of mapping assertions occurring in $\mathcal{M}$. Hence, given a data integration system $\mathcal{J}$ with PGAV mapping and an UCQJFE $q_{\mathcal{S}}$, an algorithm to compute the UCQ-maximally sound $\mathcal{J}$-abstraction of $q_{\mathcal{S}}$ simply returns the union of all CQs $q_{\mathcal{G}}$ comprising at most $k^{\mathcal{M}}_{q_{\mathcal{S}}}$ atoms, that are sound $\mathcal{J}$-abstractions of $q_{\mathcal{S}}$. The crucial observation here is that in order to check whether $q_{\mathcal{G}}$ is a sound $\mathcal{J}$-abstraction of $q_{\mathcal{S}}$, it is sufficient to check whether $unf_{\mathcal{M}}(q_{\mathcal{G}}) \sqsubseteq q_{\mathcal{S}}$, which is decidable, since both $q_{\mathcal{S}}$ and $unf_{\mathcal{M}}(q_{\mathcal{G}})$ are UCQs [27].

Theorem 5. *In the restricted scenario, the UCQ-maximally sound $\mathcal{J}$-abstractions of $q_{\mathcal{S}}$ always exists and is computable.*

To conclude the section, we provide the last positive result about perfect abstractions in the class UCQ. Namely, we show that checking whether there exists a UCQ that is the perfect $\mathcal{J}$-abstraction of $q_{\mathcal{S}}$ is decidable. In particular, given a data integration system $\mathcal{J}$ with GLAV mapping and a UCQ $q_{\mathcal{S}}$, an algorithm to decide whether there exists a UCQ that is a perfect $\mathcal{J}$-abstraction of $q_{\mathcal{S}}$ proceeds as follows. First, it computes the query $q_{\mathcal{G}}$ that is the UCQ-minimally complete $\mathcal{J}$-abstraction of $q_{\mathcal{S}}$. Then, it checks whether $q_{\mathcal{G}}$ is a sound abstraction of $q_{\mathcal{S}}$ (as discussed above). If the answer is negative, then there exists no UCQ that is a perfect $\mathcal{J}$-abstraction of $q_{\mathcal{S}}$. If the answer is positive, then $q_{\mathcal{G}}$ is actually a UCQ, and is the perfect $\mathcal{J}$-abstraction of $q_{\mathcal{S}}$. Thus the algorithm also solves the computation problem for perfect abstractions in the UCQ language.

Theorem 6. *Checking whether there exists a query q in the class UCQ that is the perfect $\mathcal{J}$-abstraction of $q_{\mathcal{S}}$ is decidable. Moreover, there is an algorithm that computes q, whenever it exists.*

VI. Monotone Abstractions

We remind the reader that a $\mathcal{G}$-query q is monotone in the context of a data integration system $\mathcal{J} = \langle \mathcal{G}, \mathcal{S}, \mathcal{M} \rangle$ if for every pair D, D' of $\mathcal{S}$-databases, $mod(\mathcal{J}, D) \subseteq mod(\mathcal{J}, D')$ implies $q^{\mathcal{J},D'} \subseteq q^{\mathcal{J},D}$. Monotonicity is natural yet broad enough to characterize some of the most popular classes of queries. In particular, first-order queries and several Datalog variants are monotone, if evaluated under the certain answer semantics. In the light of these considerations, it is natural to ask whether perfect and approximated abstractions in the class of monotone queries always exist for a given class of source queries, and, in case, which algorithms can be used for computing them.

In the remainder of this section, we present recent results on monotone abstractions of UCQs. We introduce a novel language of monotone queries, called $\mathsf{DD}^{\mathbf{K}}$, with attractive computational properties (Section VI-A). For the case of data integration systems with no axioms in both the global schema and in the source schema, we show that minimally complete and maximally sound monotone abstractions for UCQ source queries always exist, and are expressible in $\mathsf{DD}^{\mathbf{K}}$. From these results, we also derive the decidability of checking whether a perfect monotone abstraction of a given source query exists (Section VI-B).

A. A language for monotone abstractions

Monotone queries form a natural yet expressive class of queries. Unsurprisingly, perfect and approximated monotone abstractions require a suitably expressive query language. We now introduce one such language and discuss some of its most compelling computational characteristics. The language, called $\mathsf{DD}^{\mathbf{K}}$, is based on disjunctive Datalog, extended with an epistemic operator. We present it in a form specifically tailored for querying data integration systems.

Assume a data integration system $\mathcal{J} = \langle \mathcal{G}, \mathcal{M}, \mathcal{S} \rangle$ and an alphabet of predicate symbols Int, called *intensional predicate symbols*, disjoint from the alphabets of $\mathcal{G}$ and $\mathcal{S}$. In this subsection we consider the case where the logical theories corresponding to both $\mathcal{G}$ and $\mathcal{S}$ may have a nonempty set of axioms.

A $\mathsf{DD}^{\mathbf{K}}$ query for $\mathcal{J}$ includes a set of rules, each one of two possible forms:

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

$$b_1 \wedge \ldots \wedge b_m \rightarrow i_1 \vee \ldots \vee i_n \quad (1)$$

  where $b_1, \ldots, b_m$ and $i_1, \ldots, i_n$ are atoms on intensional predicates, and
- a new form specified as follows

$$\mathbf{K}(\phi_1(\bar{x}) \vee \ldots \vee \phi_m(\bar{x})) \rightarrow \bigvee_{i \in \{1..n\}} \exists \bar{y}_i . \psi_i(\bar{x}, \bar{y}_i) \quad (2)$$

  where each ψ_i is a conjunction of intensional predicates, and each ϕ_i is of the form $\exists \bar{z} . \gamma(\bar{x}, \bar{z}) \wedge \xi(\bar{x})$, with $\gamma(\bar{x}, \bar{z})$ a conjunction of atoms over $\mathcal{G}$, and $\xi(\bar{x})$ a conjunction of inequalities involving variables in $\bar{x}$.

An n-ary $\mathsf{DD}^{\mathbf{K}}$ query q for $\mathcal{J}$ is a pair $q = \langle Ans, \mathcal{R} \rangle$ where $\mathcal{R}$ is a finite set of $\mathsf{DD}^{\mathbf{K}}$ rules, called the definition of q, and Ans is an n-ary intensional predicate in Int, called the answer predicate of q.

Answers for $\mathsf{DD}^{\mathbf{K}}$ queries are defined based on the notions presented in [28]. An interpretation for q is a pair $I = (E, f)$, where E is a set of interpretations for $\mathcal{J}$, and f is an interpretation for Int with domain C. An interpretation $I = (E, f)$ satisfies a $\mathsf{DD}^{\mathbf{K}}$ rule ρ of q (written $I \models \rho$) if the following conditions hold:

- If ρ is a formula of the form (1), then $I \models \rho$ if $f \models \rho$, i.e., f satisfies the implication in (1).
- If ρ is a formula of the form (2), then $I \models \rho$ if for all tuples $\bar{c}$ of values in C, if $\bar{c}$ is an answer to the query $\{\bar{x} \mid \phi_1(\bar{x})\} \cup \ldots \cup \{\bar{x} \mid \phi_m(\bar{x})\}$ in all the interpretations in E, then there is a j such that $\exists \bar{y}_j . \psi_j(\bar{c}_j, \bar{y}_j)$ is true in f.

An interpretation I for q is called a *model* of q if all the rules in the definition of q are satisfied by I. It should be clear that, under this definition of semantics, $\mathbf{K}$ represents the "knowledge" operator of the modal logic system S5. In other words, the formula $\mathbf{K}\alpha$ should be read as "α is known (i.e., logically implied) by the system".

We are ready to define what is the answer $q^{\mathcal{J},D}$ of a $\mathsf{DD}^{\mathbf{K}}$ query $q = \langle Ans, \mathcal{R} \rangle$ with respect to $\mathcal{J}$ and the $\mathcal{S}$-database D. Specifically, $q^{\mathcal{J},D} = \bigcap \{ \bar{c} \in Ans^f \mid (mod(\mathcal{J}, D), f)$ is a model of $q \}$.

While a thorough analysis of $\mathsf{DD}^{\mathbf{K}}$ is outside the scope of the present work, we mention some of its most appealing characteristics. Firstly, we observe that $\mathsf{DD}^{\mathbf{K}}$ generalizes UCQs. In particular, every UCQ q of m disjuncts is equivalent to a $\mathsf{DD}^{\mathbf{K}}$ query with one rule of the form (2) where the disjuncts of q are in the scope of $\mathbf{K}$. Secondly, every $\mathsf{DD}^{\mathbf{K}}$ query q over $\mathcal{J}$ inherits the monotonicity property of disjunctive Datalog. Intuitively, monotonicity follows from the fact that the $\mathbf{K}$ operator controls the interaction between q and $\mathcal{J}$, based on a form of stratification separating the certain answers to UCQs (rules of the form (2)) and the computation according to the rules (1). This simple form of stratification guarantees that answering q over $\mathcal{J}$ boils down to the following: (*i*) computing certain answers for the UCQs in the scope of $\mathbf{K}$ in the left-hand side of rules of the form (1) in q, and (*ii*) computing the answers for the remaining rules (form (2)) over the result of the previous step. Monotonicity follows from the monotonicity of certain answers to UCQs, and from the fact that the rules of the form (2) define a monotone query. These considerations indicate a third appealing characteristic of $\mathsf{DD}^{\mathbf{K}}$. Specifically, the decidability of answering a $\mathsf{DD}^{\mathbf{K}}$ query q w.r.t. $\mathcal{J}$ and D depends exclusively on the decidability of answering UCQs over $\mathcal{J}$, as the following proposition shows.

Proposition 3. *Answering $\mathsf{DD}^{\mathbf{K}}$ queries w.r.t. $\mathcal{J}$ and D is decidable if and only if computing the certain answers of UCQs w.r.t. $\mathcal{J}$ and D is decidable.*

These results sharply contrast with similar results obtained for plain (non-disjunctive) Datalog. In particular, the undecid-

ability of the latter can be proved even in the case of global schema axioms expressed in very simple Description Logics of the *DL-Lite* family (see, e.g., [29], [30]).

From the algorithm sketched above, we can even derive a tighter upper bound for the complexity of answering $\mathsf{DD}^{\mathbf{K}}$ queries. In particular, we can show that answering a $\mathsf{DD}^{\mathbf{K}}$ query q w.r.t. $\mathcal{J}$ and D is in coNP in data complexity (i.e., the complexity w.r.t. the size of the $\mathcal{S}$-database) whenever computing certain answers for UCQs w.r.t. $\mathcal{J}$ and D is in coNP in data complexity. The following result follows from the observations above and well-known lower bounds for answering disjunctive Datalog queries.

Proposition 4. *If computing the certain answers of UCQs w.r.t. $\mathcal{J}$ and D is in coNP in data complexity, then computing the answers to $\mathsf{DD}^{\mathbf{K}}$ queries w.r.t. $\mathcal{J}$ and D is coNP-complete in data complexity.*

B. Monotone abstractions via $\mathsf{DD}^{\mathbf{K}}$

We now turn our attention to discussing the use of $\mathsf{DD}^{\mathbf{K}}$ in expressing monotone abstractions. We start by observing that, in terms of computational complexity, $\mathsf{DD}^{\mathbf{K}}$ perfectly fits the problem of computing approximated abstractions, as the following proposition shows.

Proposition 5. *There exists a data integration system $\mathcal{J}$ with* PGAV *mapping and a UCQ $q_{\mathcal{S}}$ such that answering the $\mathfrak{M}$-maximally sound $\mathcal{J}$-abstraction of $q_{\mathcal{S}}$ is coNP-hard in data complexity.*

In the remainder of this section, we show that $\mathsf{DD}^{\mathbf{K}}$ is well-suited to express monotone abstractions, both perfect and approximated. In discussing this issue, we go back to our assumption of dealing with data integration systems with no axioms in both the global and the source schema. So, in what follows, we implicitly deal with a data integration system $\mathcal{J} = \langle \mathcal{G}, \mathcal{M}, \mathcal{S} \rangle$, where $\mathcal{G}$ and $\mathcal{S}$ have no axioms, and a UCQ $\mathcal{S}$-query $q_{\mathcal{S}} = q_1 \cup \ldots \cup q_n$, where $q_i = \{\bar{x} \mid \exists \bar{y}_i.\phi(\bar{x}, \bar{y})\}$, for $i = 1, \ldots, n$.

$\mathfrak{M}$-Maximally Sound Abstractions. In [31], it is shown that $\mathsf{DD}^{\mathbf{K}}$ can always express $\mathfrak{M}$-maximally sound $\mathcal{J}$-abstractions of UCQs, by illustrating a technique that, given query $q_{\mathcal{S}}$, builds a set $\mathcal{R}_{\mathcal{J}}$ of $\mathsf{DD}^{\mathbf{K}}$ rules whose intensional predicates are the predicates in $\mathcal{S}$, and then uses such rules to construct the $\mathfrak{M}$-maximally sound $\mathcal{J}$-abstractions of $q_{\mathcal{S}}$ as a $\mathsf{DD}^{\mathbf{K}}$ query. We do not describe the technique in detail here. Rather, we use an example to give an intuition of the construction.

Example 4. *Given the following mapping in $\mathcal{J}$:*

$m_1 : \exists y.s_1(x) \wedge s_2(x, y) \rightarrow g_1(x, x)$
$m_2 : s_1(x) \wedge s_3(x, y) \rightarrow g_1(x, y)$
$m_3 : s_4(x) \rightarrow \exists y.g_1(x, y)$

$\mathcal{R}_{\mathcal{J}}$ is the following set of $\mathsf{DD}^{\mathbf{K}}$ rules:

$$\mathbf{K}(g_1(x, x)) \rightarrow (\exists y.s_1(x) \wedge s_2(x, y)) \vee (s_1(x) \wedge s_3(x, x))$$
$$\mathbf{K}(g_1(x, y) \wedge x \neq y) \rightarrow s_1(x) \wedge s_3(x, y)$$
$$\mathbf{K}(\exists y.g_1(x, y)) \rightarrow s_4(x) \vee (\exists y.s_1(x) \wedge s_3(x, y)) \vee (\exists y.s_1(x) \wedge s_2(x, y))$$

Intuitively, the rules of $\mathcal{R}_{\mathcal{J}}$ specify, for the various facts over $\mathcal{G}$ that are certain, i.e., that are known *to hold, the queries over the sources that generate them. For example, the first rule of $\mathcal{R}_{\mathcal{J}}$ specifies that, if a constant is known to satisfy $g_1(x, x)$, then this knowledge derives either from the answers to the source query $\{x \mid \exists y.s_1(x) \wedge s_2(x, y)\}$ or from the answers to the source query $\{x \mid s_1(x) \wedge s_3(x, x)\}$. As another example, the second rule of $\mathcal{R}_{\mathcal{J}}$ specifies that the pairs of distinct constants x, y known to satisfy $g_1(x, y)$ derive from the query $\{x, y \mid s_1(x) \wedge s_3(x, y)\}$. It can be shown that this is crucial for ensuring that the abstraction of queries involving the join of s_1 and s_3, which is based on the certain answers of g_1, do not include data deriving from source queries whose abstraction is based on the certain answers of the projection of g_1. Finally, the third rule of $\mathcal{R}_{\mathcal{J}}$ takes care of those constants x known to satisfy $g_1(x, y)$, for some, not necessarily known, y. Such constants may derive from each of source queries above.* △

Using the notion of $\mathcal{R}_{\mathcal{J}}$, we can immediately obtain the $\mathfrak{M}$-maximally sound $\mathcal{J}$-abstraction of $q_{\mathcal{S}}$, by adding to $\mathcal{R}_{\mathcal{J}}$ the set $\mathcal{A}$ constituted by one rule of the form $\phi_i(\bar{x}, \bar{y}) \rightarrow Ans(\bar{x})$ for each disjunct $q_i = \{\bar{x} \mid \exists \bar{y}_i.\phi(\bar{x}, \bar{y})\}$ in $q_{\mathcal{S}}$.

Proposition 6. *The $\mathsf{DD}^{\mathbf{K}}$ query $\langle Ans, \mathcal{R}_{\mathcal{J}} \cup \mathcal{A} \rangle$ is the $\mathfrak{M}$-maximally sound $\mathcal{J}$-abstraction of $q_{\mathcal{S}}$.*

In the light of Proposition 6 and from the existence of an algorithm to compute $\mathcal{R}_{\mathcal{J}} \cup \mathcal{A}$, we obtain the following.

Theorem 7. *The $\mathfrak{M}$-maximally sound $\mathcal{J}$-abstraction of $q_{\mathcal{S}}$ always exists, is computable, and can be expressed in $\mathsf{DD}^{\mathbf{K}}$.*

$\mathfrak{M}$-Minimally Complete Abstractions. We show that $\mathsf{DD}^{\mathbf{K}}$ can always express $\mathfrak{M}$-minimally complete $\mathcal{J}$-abstractions of UCQs.

Let us first introduce a useful notion. Given a CQ $q = \{\bar{x} \mid \exists \bar{y}.\phi(\bar{x}, \bar{y})\}$, $\mathsf{Saturate}(q)$ denotes the UCQ with inequalities obtained as follows. For each possible unifier μ on the variables in $\bar{x} \cup \bar{y}$ such that $\mu(x) \in \bar{x}$ for each $x \in \bar{x}$, $\mathsf{Saturate}(q)$ contains a query obtained from $\mu(q)$ by adding an inequality atom $(t_1 \neq t_2)$ for each pair of distinct variables t_1, t_2 occurring in $\mu(q)$. For a UCQ Q, we denote by $\mathsf{Saturate}(Q)$ the UCQ with inequalities consisting of the union of $\mathsf{Saturate}(q)$, for each disjunct q of Q. It is easy to see that $\mathsf{Saturate}(Q)$ is equivalent to Q, for every UCQ Q.

Consider a disjunct q_h in in $\mathsf{Saturate}(q_{\mathcal{S}})$. Clearly, q_h is a CQ with inequalities of the form $q_h = \{\bar{x} \mid \exists \bar{y}.\phi(\bar{x}, \bar{y}) \wedge \chi(\bar{x}, \bar{y})\}$, where $\chi(\bar{x}, \bar{y})$ are inequality atoms. Let $\mathcal{M}(q_h)$ denote the result of chasing the set of relational atoms occurring in q_h with $\mathcal{M}$. Let ρ_{q_h} denote the $\mathsf{DD}^{\mathbf{K}}$ rule $\mathbf{K}(\mathcal{M}(q_h) \wedge \top(\bar{x}) \wedge \chi(\bar{x}, \bar{y})) \rightarrow Ans(\bar{x})$. Finally, let q_c denote the $\mathsf{DD}^{\mathbf{K}}$ query consisting of all the rules ρ_{q_h} for the various q_h in $\mathsf{Saturate}(q_{\mathcal{S}})$ and with answer predicate Ans. We can now prove the following.

Proposition 7. *q_c is the $\mathfrak{M}$-minimally complete $\mathcal{J}$-abstraction of $q_{\mathcal{S}}$.*

The following statement is a straightforward consequence of Proposition 7.

Theorem 8. *The $\mathfrak{M}$-minimally complete $\mathcal{J}$-abstraction of $q_{\mathcal{S}}$ always exists, is computable, and can be expressed in* $\mathsf{DD}^{\mathbf{K}}$.

Perfect Monotone Abstractions. From the results presented above, we can derive an algorithm for checking whether there exists a query in $\mathfrak{M}$ that is the perfect $\mathcal{J}$-abstraction of $q_{\mathcal{S}}$. In particular, observe that if the perfect $\mathcal{J}$-abstraction of $q_{\mathcal{S}}$ can be expressed as a query in $\mathfrak{M}$, then it is $\mathcal{J}$-equivalent to the $\mathfrak{M}$-minimally complete $\mathcal{J}$-abstraction of $q_{\mathcal{S}}$. Then, from Proposition 7 we know that, in order to check whether there exists a query in $\mathfrak{M}$ that is the perfect $\mathcal{J}$-abstraction of $q_{\mathcal{S}}$, we have to check whether $q_{\mathcal{S}}$ is equivalent to q_c modulo $\mathcal{J}$.

To this end, we observe the following. There exists a UCQ with inequalities $\mathcal{S}$-query q_{min} such that $q_{min}^D = q_c^{\mathcal{J},D}$, for every $\mathcal{S}$-database D. Moreover, q_{min} is computable. These two properties result from $\mathcal{J}$ being a GLAV data integration system with no source and global schema axioms, and from the specific form of q_c. Therefore, in order to check whether there exists a query in $\mathfrak{M}$ that is the perfect $\mathcal{J}$-abstraction of $q_{\mathcal{S}}$, we just need to check whether $q_{min} \sqsubseteq q_{\mathcal{S}}$. The next claim follows from these considerations.

Theorem 9. *Checking whether there exists a query q in the class $\mathfrak{M}$ that is the perfect $\mathcal{J}$-abstraction of $q_{\mathcal{S}}$ is decidable. Moreover, there is an algorithm that computes q, whenever it exists.*

VII. Open Problems

We have provided an overview of abstraction in data integration, and we have illustrated some results obtained in recent years on computing abstractions. We conclude the paper by discussing a set of issues related to abstractions that deserve more investigation.

Languages for Abstractions. A crucial issue related to abstraction is to compute perfect and approximated abstractions within specific classes of queries. For the fundamental class UCQ, the decidability of checking whether there exists a UCQ-maximally sound abstraction of a UCQ source query is still open. More generally, there are many interesting classes of queries that can be used to express abstractions, and for which it would be interesting to compute perfect, or approximated abstractions. For example, in the case of graph databases as virtual views, relevant classes of queries for abstractions include regular path queries, or two-way conjunctive regular path queries.

Abstraction and Monotonicity. In this paper we have discussed the use of $\mathsf{DD}^{\mathbf{K}}$ to express monotone abstractions of source queries in the class UCQ. It would be interesting to investigate which is the minimal expressive power needed for capturing perfect and approximated monotone abstractions of source queries. Also, it is not difficult to see that there are queries for which the perfect abstraction is non-monotone. Although first results on non-monotone abstractions have appeared in [11], the issue of checking the existence of and computing non-monotone abstractions is largely unexplored.

Expressive Source Queries. The majority of work on abstraction so far focused on source queries in the class UCQ. It would be interesting to address the problem of computing perfect and approximated abstractions of source queries expressed in more expressive languages such as Datalog. More expressive mapping languages (e.g., UCQ with inequalities in the GLAV type of mapping) also deserve attention.

Axioms. The computation of abstractions in the presence of axioms in the global schema or in the source schema is another interesting problem to study. First results in this direction appeared in [8]–[10], but the topic requires a more thorough analysis.

Reverse Engineering. Abstraction has also interesting connections with the reverse-engineering problem [32]. When casted in data integration, given a source database D and set P of tuples, this problem aims at finding a global schema query q that captures P, i.e., such that the answers of q with respect to D captures the tuples in P. Despite the intuitive connection, a detailed analysis of the relationship between the two problems is missing.

User Requirements. Finally, crucial aspects of abstractions, such as succinctness and clarity, have not been considered in this paper. More generally, issues related to the adequacy of the formulation of abstractions with respect to user requirements deserve greater attention.

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

[1] A. Doan, A. Y. Halevy, and Z. G. Ives, *Principles of Data Integration*. Morgan Kaufmann, 2012.

[2] M. Lenzerini, "Data integration: A theoretical perspective." in *Proceedings of the Twenty-First ACM SIGACT SIGMOD SIGART Symposium on Principles of Database Systems (PODS 2002)*, 2002, pp. 233–246.

[3] M. Arenas, P. Barceló, L. Libkin, and F. Murlak, *Foundations of Data Exchange*. Cambridge University Press, 2014.

[4] A. Y. Halevy, "Answering queries using views: A survey," *Very Large Database Journal*, vol. 10, no. 4, pp. 270–294, 2001.

[5] M. Jarke, M. Lenzerini, Y. Vassiliou, and P. Vassiliadis, Eds., *Fundamentals of Data Warehouses*, 2nd ed. Springer, 2003.

[6] G. Xiao, D. Calvanese, R. Kontchakov, D. Lembo, A. Poggi, R. Rosati, and M. Zakharyaschev, "Ontology-based data access: A survey," in *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence (IJCAI 2018)*, 2018, pp. 5511–5519.

[7] C. Gutierrez and J. F. Sequeda, "Knowledge graphs," *Commun. ACM*, vol. 64, no. 3, p. 96–104, 2021.

[8] G. Cima, "Preliminary results on ontology-based open data publishing," in *Proceedings of the Thirtieth International Workshop on Description Logics (DL 2017)*, ser. CEUR Electronic Workshop Proceedings, http://ceur-ws.org/, vol. 1879, 2017.

[9] C. Lutz, J. Marti, and L. Sabellek, "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)*, 2018, pp. 389–398.

[10] G. Cima, M. Lenzerini, and A. Poggi, "Semantic characterization of data services through ontologies," in *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence (IJCAI 2019)*, 2019, pp. 1647–1653.

[11] ——, "Non-monotonic ontology-based abstractions of data services," in *Proceedings of the Seventeenth International Conference on Principles of Knowledge Representation and Reasoning (KR 2020)*, 2020, pp. 243–252.

[12] G. Cima, "Abstraction in ontology-based data management," Ph.D. dissertation, Sapienza University of Rome, 12 2020.

[13] G. Cima, M. Lenzerini, and A. Poggi, "Semantic technology for open data publishing," in *Proceedings of the Seventh International Conference on Web Intelligence, Mining and Semantics (WIMS 2017)*, 2017, p. 1:1.

[14] Z. Abedjan, L. Golab, and F. Naumann, "Data profiling: A tutorial," in *Proceedings of the 2017 ACM International Conference on Management of Data (SIGMOD 2017)*, 2017, pp. 1747–1751.

[15] D. Calvanese, G. De Giacomo, M. Lenzerini, and M. Y. Vardi, "What is view-based query rewriting?" in *Proceedings of the Seventh International Workshop on Knowledge Representation meets Databases (KRDB 2000)*, ser. CEUR Electronic Workshop Proceedings, http://ceur-ws.org/, vol. 29, 2000, pp. 17–27.

[16] D. Calvanese, G. De Giacomo, M. Lenzerini, and M. Y. Vardi, "View-based query processing: On the relationship between rewriting, answering and losslessness," *Theoretical Computer Science*, vol. 371, no. 3, pp. 169–182, 2007.

[17] A. Y. Levy, A. O. Mendelzon, Y. Sagiv, and D. Srivastava, "Answering queries using views," in *Proceedings of the Fourteenth ACM SIGACT SIGMOD SIGART Symposium on Principles of Database Systems (PODS 1995)*, 1995, pp. 95–104.

[18] A. Nash, L. Segoufin, and V. Vianu, "Views and queries: Determinacy and rewriting," *ACM Transactions on Database Systems*, vol. 35, no. 3, pp. 21:1–21:41, 2010.

[19] O. M. Duschka and M. R. Genesereth, "Answering recursive queries using views," in *Proceedings of the Sixteenth ACM SIGACT SIGMOD SIGART Symposium on Principles of Database Systems (PODS 1997)*, 1997, pp. 109–116.

[20] F. N. Afrati and R. Chirkova, *Answering Queries Using Views*, 2nd ed., ser. Synthesis Lectures on Data Management. Morgan & Claypool Publishers, 2019.

[21] O. M. Duschka and M. R. Genesereth, "Query planning with disjunctive sources," in *Proceedings of the AAAI-98 Workshop on AI and Information Integration*, 1998.

[22] D. Calvanese, G. De Giacomo, M. Lenzerini, and M. Y. Vardi, "Lossless regular views," in *Proceedings of the Twenty-First ACM SIGACT SIGMOD SIGART Symposium on Principles of Database Systems (PODS 2002)*, 2002, pp. 58–66.

[23] T. Gogacz and J. Marcinkowski, "Red spider meets a rainworm: Conjunctive query finite determinacy is undecidable," in *Proceedings of the Thirty-Fifth ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems (PODS 2016)*, 2016, pp. 121–134.

[24] A. Y. Levy, A. Rajaraman, and J. J. Ordille, "Querying heterogenous information sources using source descriptions," in *Proceedings of the Twenty-Second International Conference on Very Large Data Bases (VLDB 1996)*, 1996.

[25] R. Pottinger and A. Y. Halevy, "MiniCon: A scalable algorithm for answering queries using views," *Very Large Database Journal*, vol. 10, no. 2–3, pp. 182–198, 2001.

[26] R. Fagin, P. G. Kolaitis, R. J. Miller, and L. Popa, "Data exchange: Semantics and query answering," *Theoretical Computer Science*, vol. 336, no. 1, pp. 89–124, 2005.

[27] Y. Sagiv and M. Yannakakis, "Equivalences among relational expressions with the union and difference operators," *Journal of the ACM*, vol. 27, no. 4, pp. 633–655, 1980.

[28] D. Calvanese, G. De Giacomo, D. Lembo, M. Lenzerini, and R. Rosati, "EQL-Lite: Effective first-order query processing in description logics," in *Proceedings of the Twentieth International Joint Conference on Artificial Intelligence (IJCAI 2007)*, 2007, pp. 274–279.

[29] A. Y. Levy and M.-C. Rousset, "Combining Horn rules and description logics in CARIN," *Artificial Intelligence*, vol. 104, no. 1–2, pp. 165–209, 1998.

[30] D. Calvanese and R. Rosati, "Answering recursive queries under keys and foreign keys is undecidable," in *Proceedings of the Tenth International Workshop on Knowledge Representation meets Databases (KRDB 2003)*, ser. CEUR Electronic Workshop Proceedings, http://ceur-ws.org/, vol. 79, 2003.

[31] G. Cima, M. Console, M. Lenzerini, and A. Poggi, "Abstraction in ontology-based data management: The case of monotone queries," 2021, Submitted for publication.

[32] P. Barceló and M. Romero, "The complexity of reverse engineering problems for conjunctive queries," in *Proceedings of the Twentieth International Conference on Database Theory (ICDT 2017)*, ser. Leibniz International Proceedings in Informatics, https://www.dagstuhl.de/en/publications/lipics, vol. 68, 2017, pp. 7:1–7:17.