TYPE Review
PUBLISHED 23 June 2023
DOI 10.3389/frai.2023.1085754

Check for updates

OPEN ACCESS

EDITED BY
Giovanni Sileno,
University of Amsterdam, Netherlands

REVIEWED BY
Federica Mandreoli,
University of Modena and Reggio Emilia, Italy
Joao Pita Costa,
UNESCO International Research Center on Artificial Intelligence - IRCAI, Slovenia
Pablo Barcelo,
Pontifical Catholic University of Chile, Chile

*CORRESPONDENCE
Antonella Poggi
✉ poggi@diag.uniroma1.it

SPECIALTY SECTION
This article was submitted to Machine Learning and Artificial Intelligence, a section of the journal Frontiers in Artificial Intelligence

RECEIVED 31 October 2022
ACCEPTED 30 March 2023
PUBLISHED 23 June 2023

CITATION
Cima G, Console M, Lenzerini M and Poggi A (2023) A review of data abstraction. *Front. Artif. Intell.* 6:1085754. doi: 10.3389/frai.2023.1085754

COPYRIGHT
© 2023 Cima, Console, Lenzerini and Poggi. This is an open-access article distributed under the terms of the Creative Commons Attribution License (CC BY). The use, distribution or reproduction in other forums is permitted, provided the original author(s) and the copyright owner(s) are credited and that the original publication in this journal is cited, in accordance with accepted academic practice. No use, distribution or reproduction is permitted which does not comply with these terms.

A review of data abstraction

Gianluca Cima, Marco Console, Maurizio Lenzerini and Antonella Poggi*

Dipartimento di Ingegneria Informatica, Automatica e Gestionale "A. Ruberti", Sapienza University of Rome, Rome, Italy

It is well-known that Artificial Intelligence (AI), and in particular Machine Learning (ML), is not effective without good data preparation, as also pointed out by the recent wave of data-centric AI. Data preparation is the process of gathering, transforming and cleaning raw data prior to processing and analysis. Since nowadays data often reside in distributed and heterogeneous data sources, the first activity of data preparation requires collecting data from suitable data sources and data services, often distributed and heterogeneous. It is thus essential that providers describe their data services in a way to make them compliant with the FAIR guiding principles, i.e., make them automatically Findable, Accessible, Interoperable, and Reusable (FAIR). The notion of data abstraction has been introduced exactly to meet this need. Abstraction is a kind of reverse engineering task that automatically provides a semantic characterization of a data service made available by a provider. The goal of this paper is to review the results obtained so far in data abstraction, by presenting the formal framework for its definition, reporting about the decidability and complexity of the main theoretical problems concerning abstraction, and discuss open issues and interesting directions for future research.

KEYWORDS
knowledge representation, abstraction, automated reasoning, data integration, data preparation

1. Introduction

Despite the increasing centrality of data in AI, the way in which AI deals with data has remained virtually unchanged since the dawn of the discipline. This has to be contrasted with the well-known fact that Artificial Intelligence (AI), and in particular Machine Learning (ML), is not effective without good data preparation, as also pointed out by the recent wave of data-centric AI. The term "data centric" refers to an architecture where data is the primary and permanent asset. So, data preparation precedes the implementation of any given machine learning task, and can potentially support many of such tasks relying on the same domain. More specifically, data preparation is the process of gathering, transforming and cleaning raw data prior to processing and analysis. It is therefore regarded as an important step in any data engineering and data science projects, including machine learning, involving tasks such as understanding, collecting and reformatting data, aggregating, integrating, combining and enriching raw source data and making modifications and corrections in order to meet quality standards.

The first activity of data preparation requires collecting data from suitable data sources and data services, often distributed and heterogeneous. In the era of data as driving asset both for the private and public domain, the availability of services providing data, also called *data services*, is indeed growing incredibly fast. Thus, on one hand, more and more data services are available, on the other hand, more and more AI tasks and applications rely on data services. This scenario opens two crucial issues for data-centric AI. First, from a consumer point of view, how to find the "right" data, i.e., data which properly respond to an information need? Second, from a provider point of view, how to release FAIR-compliant data services, i.e., services automatically Findable, Accessible, Interoperable, and Reusable

(FAIR)? An effective answer to the former question is given by exploiting the state of the art technology for *answering queries* over *data integration* systems, which stems from more than thirty years of research. As for the second question, an answer is given by the results on a relatively new service of data integration systems, called *abstraction*. In order to elaborate more on both these answers, let us first make a step back to data integration.

Data integration is the problem of providing a unified and reconciled view of the data stored in a set of autonomous and heterogeneous sources. The theoretical works on data integration systems have advocated a three-layer architecture comprising the *data sources*, which in our setting are the output of the data services, the *global schema*, which is a unified shared conceptualization of the domain of interest, and the *mapping* between the sources and the global schema. Formally, a data integration system is 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, i.e., a set of logical assertions describing how the data at the sources relate to the elements of the global schema. Then, intuitively, given a set of data sources D, $\mathcal{J}$ represents all the (possibly incomplete) databases that are instances of $\mathcal{G}$ satisfying $\mathcal{M}$ w.r.t. D.

Once data services have been integrated by means of a data integration system specified through a triple $\mathcal{J} = \langle \mathcal{G}, \mathcal{S}, \mathcal{M} \rangle$, in order to find the "right data," a data service consumer can rely on query answering. Specifically, by unambiguously expressing an information need as a query over the shared vocabulary of $\mathcal{G}$, he can get the answers that "best correspond" to his need without even having to know the relevant data services. In particular, in most approaches, such answers have been identified as *certain answers*, i.e. answers to $q_\mathcal{G}$ that would be returned by every database represented by $\mathcal{J}$ given a set of data sources D. Also, typically, such answers are computed by first reformulating $q_\mathcal{G}$ in terms of a query $q_\mathcal{S}$ and then by evaluating $q_\mathcal{S}$ over D. Conversely, in order to make a data service FAIR-compliant, a provider can rely on abstraction over $\mathcal{J}$. Specifically, given a data service originally expressed as a query over a set of data sources, he can get a query over the shared vocabulary of $\mathcal{G}$, that unambiguously describes the data service content, thus making it both accessible, interoperable and reusable. Concretely, given a query $q_\mathcal{S}$ over the data sources, he would get a query $q_\mathcal{G}$ over the global schema whose answers "best correspond" to the data service. Obviously, also for abstraction, the meaning of "best correspond" has to be made precise. Ideally, the query $q_\mathcal{G}$ is the one whose certain answers are exactly the answers of $q_\mathcal{S}$, for every possible source database. Such a query $q_\mathcal{G}$ is called perfect J-abstraction of $q_\mathcal{S}$.

We next use an example for informally introducing and illustrating the main notions related to abstraction. In the example, we focus on queries that are conjunctions of atoms, called *conjunctive queries (CQ)*, and unions thereof, called *unions of conjunctive queries (UCQ)*, and 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):*

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

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

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

Consider now the query $q^4_\mathcal{S} = \{() \mid \exists x,y.s_5(x,y) \wedge s_3(x)\}$, 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 $q^4_\mathcal{S}$. 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 $q^4_\mathcal{S}$. Based on this observation, one can show that there exists no maximally sound $\mathcal{J}$-abstraction of $q^4_\mathcal{S}$ in the class UCQ. However, the following Datalog query (with goal Ans) is the maximally sound $\mathcal{J}$-abstraction of $q^4_\mathcal{S}$ in the whole class of monotone queries:

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

We point out that, apart from the scenario of data services providers, data abstraction is relevant in several other contexts. We mention three of them here. In the context of ontology-based data management, abstraction can be used to check whether the mapping provides the right coverage for expressing the relevant data services at the global schema level (Lutz et al., 2018). Also, 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 (Cima et al., 2017). Finally, abstraction can be the basis for a semantic-based approach to source profiling (Abedjan et al., 2017), again one of tasks of data preparation, 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 review the main notions and results about abstraction. We present the formal framework for its definition, and report about the decidability and complexity of the main theoretical problems concerning abstraction, i.e., verification, existence, and computation. The roadmap of the paper is as follows:

- Section 2 introduces some relevant background about databases, queries, and data integration.
- Section 3 illustrates the formal framework for abstraction in data integration by providing some of the key definitions used throughout the paper.
- Section 4 reports results appearing in Cima et al. (2021) on the relationship between abstraction and another well-studied problem, namely *view-based query processing* (see, e.g., Halevy, 2001). The latter is the problem of answering a query over a schema $\mathcal{S}$ in terms of a set of materialized views over $\mathcal{S}$. Interestingly, the established relationship between abstraction and view-based query processing sheds into light new results about both problems.
- Section 5 illustrates results related to the problem of computing best UCQ abstractions of UCQ source queries (Cima et al., 2019). The main results are that, while minimally complete abstractions are guaranteed to exist, this is not the case for maximally sound abstractions. Motivated by the latter result, a restricted scenario is introduced, in which the existence of maximally sound abstractions is always guaranteed.
- Section 6 surveys results on computing best *monotone* abstractions of UCQ source queries (Cima et al., 2022). The principal contributions are the definition of a novel monotone query language (in the context of data integration) and the discussion of how such a language is able to express all forms of the best monotone abstractions (perfect, or approximated).
- Section 7 presents results on computing abstractions of UCQ source queries in a specific, well-known *non-monotone* query language (Cima et al., 2020). The main results are that all forms of best abstractions are not guaranteed to exist in such a language, and, in virtue of this result, two interesting restricted scenarios are investigated.
- Finally, Section 8 concludes the paper by discussing possible future research on abstraction.

2. Preliminaries

2.1. Databases and queries

We assume a denumerable set of constant symbols C that is included in every alphabet that we shall consider. A *database schema* (or simply schema) $\mathcal{T}$ is a logical theory, i.e., a finite set of logical axioms, over an alphabet $\mathcal{A}_{\mathcal{T}}$ of predicate symbols and constants from C. A $\mathcal{T}$*-database* is simply a model of $\mathcal{T}$, i.e., an interpretation for $\mathcal{A}_{\mathcal{T}}$ that satisfies all the axioms of $\mathcal{T}$, with the additional requirements that (*i*) the domain of D is C, (*ii*) every constant is interpreted into itself, and (*iii*) the extention of every predicate is finite.[1] In what follows, we will often see a $\mathcal{T}$-database as a finite set of ground facts over $\mathcal{A}_{\mathcal{T}}$, each of which corresponding to a tuple in the extension of the associated predicate.

As customary, a *database query* over a schema $\mathcal{T}$ of arity n, or simply an n-ary $\mathcal{T}$-query, is a function associating to each $\mathcal{T}$-database a finite set of tuples of constants of arity n. Often, however, it is more convenient to specify queries using expressions from some formal language to which a semantics, i.e., an actual query function, is associated. In what follows, whenever we talk about a *query language* $\mathcal{L}$, we mean the class of all queries that can be expressed using $\mathcal{L}$ and its associated semantics.

A fundamental query language for our work is the language of *First-Order Logic (FOL) queries.* A FOL query q for a schema $\mathcal{T}$ is a $\mathcal{T}$-query defined by an expression of the form $\{\bar{x} \mid \phi(\bar{x})\}$, where $\bar{x}$ is a tuple of variables, called the *distinguished variables* of q, and $\phi(\bar{x})$ is a FOL formula over alphabet of $\mathcal{T}$ containing all the variables in $\bar{x}$. The arity of q is the arity of $\bar{x}$, and we will often use $q(\bar{x})$ to say that $\bar{x}$ are the free-variables of the FOL query q and write $\{\bar{x} \mid \exists\bar{y}.\phi(\bar{x},\bar{y})\}$ simply as $\phi(\bar{x})$. Moreover,we will use the predicate $\top$ to form atoms of any arity; such atoms will always be interpreted as true. Given a $\mathcal{T}$-database D and a FOL $\mathcal{T}$-query q of arity n, q^D is the set of all tuples $\bar{c} \in C^n$ such that $D \models \phi(\bar{c})$.

A *conjunctive query (CQ)* q over a schema $\mathcal{T}$ is a FOL query of the form $\{\bar{x} \mid \exists\bar{y}.\phi(\bar{x},\bar{y})\}$, where $\bar{y}$ is a tuple of variables, called the *existential variables* of q, and $\phi(\bar{x},\bar{y})$ is a finite conjunction of relational atom. 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.

Other classes of database queries considered in this paper are defined as customary in terms of both syntax and semantics. An *atomic query* is a FOL query where $\phi(\bar{x})$ consists of a single relational atom. A *union of conjunctive queries (UCQ)* (resp., *union of conjunctive queries with join-free existential variables (UCQJFE)*) is a query defined as a finite union of CQs (resp., CQJFEs) having the same arity, called its *disjuncts*, and its semantics is defined via the associated FOL query. For the definition of Datalog, Disjunctive Datalog, and Disjunctive Datalog with inequalities (denoted by $\mathrm{DD}^{\neq}$), we refer the reader to Eiter et al. (1997).

2.2. Querying sets of databases

In what follows, we will often need to extend the notion of database queries to sets of databases. A *generalized* $\mathcal{T}$*-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, called the *answers of* q *for* Σ and denoted q^{Σ}. As customary, for two $\mathcal{T}$-queries q_1 and q_2, we write $q_1 \sqsubseteq q_2$ if

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

$q_1^\Sigma \subseteq q_2^\Sigma$ for each set Σ of $\mathcal{T}$-databases, and we write $q_1 \equiv q_2$ if both $q_1 \sqsubseteq q_2$ and $q_2 \sqsubseteq q_1$.

A common method to define a generalized $\mathcal{T}$-query is to lift the semantics of a $\mathcal{T}$-query to sets of $\mathcal{T}$-databases using the notion of *certain answers*. Given a $\mathcal{T}$-query q and a set Σ of $\mathcal{T}$-databases, the *certain answers of* q *over* Σ are defined as $\bigcap_{D\in\Sigma} q^D$. Thus, in what follows, we consider that every generalized $\mathcal{T}$-query is such that given a set Σ of $\mathcal{T}$-databases, q^Σ is the set of the certain answers of q over Σ. This small abuse of notation and the observation that $q^D = q^{\{D\}}$ allow us to blur the distinction between queries and generalized queries. Therefore, from now on, unless otherwise specified, we will use the term $\mathcal{T}$-query for generalized $\mathcal{T}$-query.

2.3. Data integration

A *data integration system* (Lenzerini, 2002) $\mathcal{J}$ 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 the $\mathcal{G}$-databases that satisfy $\mathcal{M}$ with respect to D. A $\mathcal{G}$-database B satisfies $\mathcal{M}$ with respect to D, denoted by $(D, B) \models \mathcal{M}$, if it satisfies all the assertions in $\mathcal{M}$, i.e., $q_\mathcal{S}^D \subseteq q_\mathcal{G}^B$ for each $(q_\mathcal{S} \rightarrow q_\mathcal{G}) \in \mathcal{M}$.

Formally, the semantics of $\mathcal{J}$ relative to D, denoted as $mod(\mathcal{J}, D)$, is defined as $\{B | B \text{ is a } \mathcal{G}\text{-database such that } (D,B) \models \mathcal{M}\}$. 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 $q_\mathcal{S}(\vec{x}) \rightarrow q_\mathcal{G}(\vec{x})$, where both $q_\mathcal{S}$ and $q_\mathcal{G}$ are conjunctive queries over $\mathcal{S}$ and $\mathcal{G}$ respectively, with distinguished variables $\vec{x}$.

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 $\mathcal{S}$ and (*ii*) $A(\vec{x})$ is an atomic $\mathcal{T}$-query. 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 the right-hand side.

A LAV mapping is a special case of GLAV, constituted by a set of assertions $A(\vec{x}) \rightarrow q_\mathcal{G}(\vec{x})$, where (*i*) $A(\vec{x})$ is an atomic $\mathcal{T}$-query and (*ii*) $q_\mathcal{G}$ is a conjunctive query over $\mathcal{G}$ with distinguished variables $\vec{x}$.

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).

2.4. The EQL-Lite(UCQ) language

EQL-Lite(UCQ) is a powerful query language in the context of data integrations systems introduced and studied in Calvanese et al. (2007a). An *EQL-Lite*(UCQ) $\mathcal{T}$-query q is an expression of the form $q = \{\bar{x} \mid \varphi(\bar{x})\}$ where $\varphi(\bar{x})$ is an *EQL* formula built according to the following syntax:

$$\varphi(\bar{x}) ::= \mathbf{K}\varrho \mid \exists y.\varphi \mid \varphi_1 \wedge \varphi_2 \mid \varphi_1 \vee \varphi_2 \mid \neg\varphi$$

with ϱ being a disjunction of conjunction of relational atoms over $\mathcal{T}$ possibly involving existentially quantified variables. The semantics is based on the notion of satisfaction of *EQL* sentences w.r.t. *epistemic interpretations*, which are pairs $\langle E, \mathcal{I} \rangle$ with E being a set of interpretations and $\mathcal{I} \in E$. We now inductively define when an epistemic interpretation $\langle E, \mathcal{I} \rangle$ satisfies an *EQL* sentence φ, written $\langle E, \mathcal{I} \rangle \models \varphi$:

$$\begin{array}{lll}
\langle E, \mathcal{I} \rangle \models P(\vec{c}) & \text{if} & \mathcal{I} \models P(\vec{c}) \\
\langle E, \mathcal{I} \rangle \models \varphi_1 \wedge \varphi_2 & \text{if} & \langle E, \mathcal{I} \rangle \models \varphi_1 \text{ and } \langle E, \mathcal{I} \rangle \models \varphi_2 \\
\langle E, \mathcal{I} \rangle \models \neg\varphi & \text{if} & \langle E, \mathcal{I} \rangle \not\models \varphi \\
\langle E, \mathcal{I} \rangle \models \exists x.\varphi & \text{if} & \langle E, \mathcal{I} \rangle \models \varphi_c^x \text{ for some constant } c \\
\langle E, \mathcal{I} \rangle \models \mathbf{K}\varphi & \text{if} & \langle E, \mathcal{I}' \rangle \models \varphi \text{ for every } \mathcal{I}' \in E,
\end{array}$$

Then, the answers $q^{\mathcal{J},D}$ of an *EQL-Lite*(UCQ) query $q = \{\bar{x} \mid \varphi(\bar{x})\}$ w.r.t. a data integration system $\mathcal{J} = \langle \mathcal{G}, \mathcal{S}, \mathcal{M} \rangle$ and an $\mathcal{S}$-database D are those tuples $\bar{c}$ of constants such that $\langle mod(\mathcal{J}, D), B \rangle \models \varphi(\bar{c})$ for every $B \in mod(\mathcal{J}, D)$.

Example 2. *Consider Example 1 and suppose we are interested in asking for all x such that there exists y such that we know (x, y) belongs to g_1. This can be expressed in EQL-Lite(UCQ) as follows:*

$$q_\mathcal{G}^6 = \{x \mid \exists y.\boldsymbol{K}(g_1(x, y))\}$$

Note that the query $q_\mathcal{G}^6$ is different from the query asking for all x such that we know there exists y such that (x, y) belongs to g_1, which is expressed as follows:

$$q_\mathcal{G}^7 = \{x \mid \boldsymbol{K}(\exists y.g_1(x, y))\}$$

Indeed, while it can be verified that the answers to $q_\mathcal{G}^6$ over $\mathcal{J}$ coincide with the answers to the query $q_\mathcal{S}^2 = \{x \mid \exists y.s_2(x, y)\}$, the answers to $q_\mathcal{G}^7$ over $\mathcal{J}$ coincide with the answers to the query $q_\mathcal{S}^5 = q_\mathcal{S}^2 \cup \{x \mid s_1(x)\}$. $\triangle$

3. Framework

We proceed to introduce the notion of query abstraction following Cima et al. (2019) for the basic definitions. 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, i.e., if q' is a perfect $\mathcal{J}$-abstraction of $q_\mathcal{S}$ then $q' =_\mathcal{J} q_\mathcal{G}$. Therefore in the following we will talk about *the* perfect $\mathcal{J}$-abstraction of $q_\mathcal{S}$.

Example 3. *Consider Example 1. It is easy to verify that $q^1_{\mathcal{G}}$ is the perfect $\mathcal{J}$-abstraction of $q^1_{\mathcal{S}}$.* △

The following theorem presents a preliminary characterization of the existence of perfect $\mathcal{J}$-abstractions.

Theorem 1. [(Cima et al., 2021, 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 $q^D_{\mathcal{S}} = q^{D'}_{\mathcal{S}}$.*

As the condition of being a perfect $\mathcal{J}$-abstraction of source query is rather strong one, it might be very well the case that such a global schema query may not exist.

Example 4. *Consider again Example 1. Using Theorem 1, we can show that there exists no perfect $\mathcal{J}$-abstraction for $q^4_{\mathcal{S}}$. In fact, for the databases $D = \{s_5(a,b)\}$ and $D' = \{s_5(a,b), s_3(a)\}$, we have $mod(\mathcal{J}, D) = mod(\mathcal{J}, D')$ but $D \not\models q^5_{\mathcal{S}}$ while $D' \models q^5_{\mathcal{S}}$.* △

In these cases, it is reasonable to consider weaker notions, such as sound or complete approximations of perfectness. We say that $q_{\mathcal{G}}$ is a *complete $\mathcal{J}$-abstraction of $q_{\mathcal{S}}$* if $q^D_{\mathcal{S}} \subseteq q^{\mathcal{J},D}_{\mathcal{G}}$, for each $\mathcal{S}$-database D consistent with $\mathcal{J}$. Similarly, we say that $q_{\mathcal{G}}$ is a *sound $\mathcal{J}$-abstraction of $q_{\mathcal{S}}$* if $q^{\mathcal{J},D}_{\mathcal{G}} \subseteq q^D_{\mathcal{S}}$, for each $\mathcal{S}$-database D consistent with $\mathcal{J}$. Obviously, one is 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 $\mathcal{J}$-abstraction of $q_{\mathcal{S}}$* if $q_{\mathcal{G}}$ is a complete $\mathcal{J}$-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 $\mathcal{J}$-abstraction of $q_{\mathcal{S}}$ and $q'_{\mathcal{G}} \sqsubset_{\mathcal{J}} q_{\mathcal{G}}$. Similarly, we say that a global schema query $q_{\mathcal{G}} \in \mathcal{L}_{\mathcal{G}}$ is an *$\mathcal{L}_{\mathcal{G}}$-maximally sound $\mathcal{J}$-abstraction of $q_{\mathcal{S}}$* if $q_{\mathcal{G}}$ is a 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 sound $\mathcal{J}$-abstraction of $q_{\mathcal{S}}$ and resp., $q_{\mathcal{G}} \sqsubset_{\mathcal{J}} q'_{\mathcal{G}}$.

Example 5. *Consider again Example 1. Queries $q^2_{\mathcal{G}}$ and $q^3_{\mathcal{G}}$ are, respectively, the UCQ-minimally complete and UCQ-maximally sound $\mathcal{J}$-abstraction of $q^3_{\mathcal{S}}$.* △

Depending on the chosen language $\mathcal{L}_{\mathcal{G}}$, it may be the case that no $\mathcal{L}_{\mathcal{G}}$-minimally complete or $\mathcal{L}_{\mathcal{G}}$-maximally sound $\mathcal{J}$-abstraction exists (see again Example 1 for some concrete cases). Moreover, even if one such abstraction exists, it may not be unique. For some classes $\mathcal{Q}$ of queries, however, one can show that a $\mathcal{Q}$-maximally sound (resp., $\mathcal{Q}$-minimally complete) $\mathcal{J}$-abstraction of $q_{\mathcal{S}}$ exists, then it is unique up to $\mathcal{J}$-equivalence. This is the case, for example, of the class of UCQs for which, if a UCQ-maximally sound (resp., UCQ-minimally complete) $\mathcal{J}$-abstraction of exists, then it is unique up to $\mathcal{J}$-equivalence. Thus, in the following, we simply talk about *the* UCQ-maximally sound and *the* UCQ-minimally complete $\mathcal{J}$-abstraction of a source query $q_{\mathcal{S}}$. Other classes of queries with this properties will be introduced in the subsequent sections.

In the next sections, we will study $\mathcal{J}$-abstraction for 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.

4. View-based query processing and query abstraction

It is well-known that there is a relationship between data integration and view-based query processing, grounded on the idea that the sources of a LAV data integration systems can be considered as views defined over the global schema, in particular sound views (Lenzerini, 2002). 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 as views defined over the source schema.

This section is organized as follows. We first recall the basic notions about view-based query processing. Then, in Section 4.1 we make clear the relationship between GAV data integration systems and views, while in Section 4.2 we establish the connection between abstractions and rewriting queries using views. Finally, in Sections 4.3 and 4.4 we use the above connection to introduce results for abstraction and view-based query processing, respectively. All the results presented in this section appear in Cima et al. (2021).

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 $\{V(\bar{c}) \mid V \in \mathcal{V}$ and $\bar{c} \in V^D_{\mathcal{T}}\}$. 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* (Calvanese et al., 2000, 2007b).

In the former notion, originated in Levy et al. (1995), 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 $q^{\mathcal{V}(D)}_{\mathcal{V}} \subseteq q^D_{\mathcal{T}}$ (*ii*) an *exact $\mathcal{V}$-rewriting of $q_{\mathcal{T}}$* if for every $\mathcal{T}$-database D it holds that $q^{\mathcal{V}(D)}_{\mathcal{V}} = q^D_{\mathcal{T}}$. 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*) $q^{\mathcal{V}(D)}_{\mathcal{V}} \subseteq q^{\mathcal{V}(D)}_1$ for each $\mathcal{T}$-database D, and (*c*) there is a $\mathcal{T}$-database D for which $q^{\mathcal{V}(D)}_{\mathcal{V}} \subsetneq q^{\mathcal{V}(D)}_1$.

As argued in Nash et al. (2010), 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* Calvanese et al., 2007b) is equivalent to the problem, called *view determinacy* (Nash et al., 2010), 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 hence an exact $\mathcal{V}$-rewriting of $q_{\mathcal{S}}$ cannot exist.

In the view-based query answering, originated in Duschka and Genesereth (1997), 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 $\bar{c} \in q_{\mathcal{T}}^{D}$ 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* $q_{\mathcal{T}}$.

4.1. View-based query processing and data integration

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 6. *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 7. *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. [(Cima et al., 2021, 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* $q_{\mathcal{S}}$.

4.2. Abstractions and rewritings of DD$^{\neq}$

We now turn our attention to a concrete class of queries, namely DD$^{\neq}$. From now on, when we use $\mathcal{L}$, we refer to a sublanguage of DD$^{\neq}$. By exploiting well-known results, we provide connections between the notion of $\mathcal{J}$-abstractions and $\mathcal{V}$-rewritings in the context of DD$^{\neq}$ and its sublanguages. 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 (Lenzerini, 2002), 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 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 (Levy et al., 1995), 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

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 Afrati and Chirkova (2019) for the complications that can arise when this assumption is removed.

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. [(Cima et al., 2021, 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 $q_{\mathcal{S}}$.*

Actually, as shown in Duschka and Genesereth (1998, 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. [(Cima et al., 2021, 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 query processing literature can be directly translated into results and techniques in the context of abstraction, and viceversa.

4.3. From view-based query processing to abstraction

By combining Proposition 1 with a well-known undecidability result about view determinacy, we can derive a 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. [(Cima et al., 2021, Theorem 2)] *Given a data integration system $\mathcal{J} = \langle \mathcal{G}, \mathcal{S}, \mathcal{M} \rangle$ with $\mathcal{M} \in$ 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.*

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 Levy et al. (1995), 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 (Levy et al., 1996) or the *MiniCon* algorithm (Pottinger and Halevy, 2001)]. Thus Corollary 1 implies that, given a data integration system $\mathcal{J} = \langle \mathcal{G}, \mathcal{S}, \mathcal{M} \rangle$ with $\mathcal{M} \in$ 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. [(Cima et al., 2021, 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 (Duschka and Genesereth, 1998; Afrati and Chirkova, 2019), 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 $\mathrm{DD}^{\neq}$, and can always be computed using the technique presented in Duschka and Genesereth (1998). Thus, Corollary 1 implies that, given a data integration system $\mathcal{J} = \langle \mathcal{G}, \mathcal{S}, \mathcal{M} \rangle$ with $\mathcal{M} \in$ PGAV and a UCQ $\mathcal{S}$-query $q_{\mathcal{S}}$, we can compute the $\mathrm{DD}^{\neq}$-maximally sound $\mathcal{J}$-abstraction of $q_{\mathcal{S}}$ as follows: (*i*) compute $\mathcal{V}_{\mathcal{J}}$, and (*ii*) compute and return the $\mathrm{DD}^{\neq}$ query corresponding to the perfect $\mathcal{V}_{\mathcal{J}}$-rewriting of $q_{\mathcal{S}}$.

Corollary 3. [(Cima et al., 2021, 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 $\mathrm{DD}^{\neq}$-maximally sound $\mathcal{J}$-abstraction of $q_{\mathcal{S}}$ exists and is computable.*

4.4. From abstraction to view-based query processing

As already observed, Duschka and Genesereth (1998) and Afrati and Chirkova (2019) 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 Cima et al. (2019) (that we will discuss in Section 5), we can actually answer positively to both questions.

Corollary 4. [(Cima et al., 2021, 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 5, 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. [(Cima et al., 2021, 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.*

3 This is not the case when view definitions are expressed as *regular path queries* rather than UCQs (Calvanese et al., 2002).

5. 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 Cima et al. (2019).

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 (Fagin et al., 2005) the atoms in q by using the tuple generating dependencies corresponding to the assertions in $\mathcal{M}$.

Theorem 3. [(Cima et al., 2019, Theorem 13)] *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. [(Cima et al., 2019, Theorem 16)] *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 (Sagiv and Yannakakis, 1980).

Theorem 5. [(Cima et al., 2019, Theorem 21)] *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. [Cima et al. (2019)] *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.*

6. Monotone abstractions

The notion of monotonicity defines a very natural class of queries that is popular in the field of databases and knowledge representation alike. The intuition behind monotone queries is simple: a query q is monotone if, whenever the data we posses increases, the answers for q do not decrease. In the literature, however, this notion has been formalized in two distinct ways. In the context of databases, a $\mathcal{T}$-query q is monotone if, for every pair of $\mathcal{T}$-databases D, D' such that $D \subseteq D'$, we have $q^D \subseteq q^{D'}$. Even very simple FOL queries can be shown not to be monotone under this notion. On the other hand, in the context of mathematical logic, the notion of monotonicity comes in a different flavor: a $\mathcal{T}$-query q is monotone, if, for every every set of interpretations Σ, Σ' for $\mathcal{T}$ such that $\Sigma \subseteq \Sigma'$, we have $q^\Sigma \subseteq q^{\Sigma'}$. We observe here that, under the semantics of certain answers, FOL queries are monotone in this sense.

To define the notion of monotone queries in the context of a data integration system, we use the notion of monotonicty from logic. 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 $q^{\mathcal{J},D'} \subseteq q^{\mathcal{J},D}$. In the following, we use $\mathfrak{M}^\mathcal{J}$ 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 $\mathfrak{M}$.

This notion of monotonicity is natural yet broad enough to characterize some of the most popular classes of queries. For example, it is trivial to see that queries evaluated under certain answer semantics are monotone. In the light of this consideration, 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 whether they can be computed. Moreover, one can show that, whenever an $\mathfrak{M}$-maximally sound (resp., $\mathfrak{M}$-minimally complete) $\mathcal{J}$-abstraction exists, then it is unique. Therefore, from now on, given a source query $q_{\mathcal{S}}$, we will talk about the $\mathfrak{M}$-maximally sound (resp., the $\mathfrak{M}$-minimally complete) $\mathcal{J}$-abstraction of $q_{\mathcal{S}}$.

In the remainder of this section, we survey recent results on monotone abstractions of UCQs presented in Cima et al. (2022). We introduce a language of monotone queries, called DD$^{\mathbf{K}}$, with attractive computational properties (Section 6.1). 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 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 6.2).

6.1. 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 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 $\mathcal{S}$. We now consider the case where the logical theories corresponding to both $\mathcal{G}$ and $\mathcal{S}$ may have a nonempty set of axioms.

A 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.,

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

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

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

  where each ψ_i is a conjunction of atoms over *Int*, 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 over variables in $\bar{x}$ only.

An n-ary 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 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 DD$^{\mathbf{K}}$ queries are defined based on the notions presented in Calvanese et al. (2007a). 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 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 I satisfies the epistemic formula $\mathbf{K}(\bigwedge_i \phi_i(\bar{c}))\}$, then there is 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 DD$^{\mathbf{K}}$ query $q = \langle Ans, \mathcal{R} \rangle$ with respect to $\mathcal{J}$ and the $\mathcal{S}$-database D. Specifically, $q^{\mathcal{J},D} = \bigcap \{\bar{c} \in Ans^f \,|\, (mod(\mathcal{J}, D), f) \text{ is a model of } q\}$.

While a thorough analysis of DD$^{\mathbf{K}}$ is outside the scope of the present work, we mention some of its most appealing characteristics. Firstly, we observe that DD$^{\mathbf{K}}$ generalizes UCQs. In particular, every UCQ q of m disjuncts is equivalent to a 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 DD$^{\mathbf{K}}$ query q over $\mathcal{J}$ is monotone in the context of $\mathcal{J}$. Intuitively, monotonicity follows from a simple form of stratification where certain answers to UCQs (rules of the form (2)) and recursive computations (rules (1)) never mix. In turn, 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 DD$^{\mathbf{K}}$. Specifically, the decidability of answering a 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. [(Cima et al., 2022, Proposition 2)] *Answering* 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 undecidability 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., Levy and Rousset, 1998; Calvanese and Rosati, 2003).

6.2. Monotone abstractions via **DD**$^{\mathbf{K}}$

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

Proposition 4. [(Cima et al., 2022, Proposition 3)] *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 $i = 1, \ldots, n$.

6.2.1. $\mathfrak{M}$-maximally sound abstractions

In Cima et al. (2022), 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 8. *Given the following mapping in $\mathcal{J}$:*

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

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

$$\begin{aligned}
&\boldsymbol{K}(g_1(x, x)) \rightarrow (\exists y.s_1(x) \land s_2(x, y)) \lor (s_1(x) \land s_3(x, x))\\
&\boldsymbol{K}(g_1(x, y) \land x \neq y) \rightarrow s_1(x) \land s_3(x, y)\\
&\boldsymbol{K}(\exists y.g_1(x, y)) \rightarrow s_4(x) \lor (\exists y.s_1(x) \land s_3(x, y)) \lor\\
&\qquad\qquad\qquad\qquad (\exists y.s_1(x) \land s_2(x, y))
\end{aligned}$$

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 $\{x|\exists y.s_1(x) \land s_2(x, y)\}$ or from the answers to the source query $\{x|s_1(x) \land s_3(x, x)\}$. 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 $\{x, y|s_1(x) \land s_3(x, y)\}$. 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 5. [(Cima et al., 2022, Theorem 2)] *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 5 and from the existence of an algorithm to compute $\mathcal{R}_{\mathcal{J}} \cup \mathcal{A}$, we obtain the following.

Theorem 7. [(Cima et al., 2022, Theorem 2)] *The $\mathfrak{M}$-maximally sound $\mathcal{J}$-abstraction of $q_{\mathcal{S}}$ always exists, is computable, and can be expressed in* $\mathsf{DD}^{\mathbf{K}}$.

6.2.2. $\mathfrak{M}$-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 $q_h = \{\bar{x} \mid \exists \bar{y}.\phi(\bar{x}, \bar{y}) \land \chi(\bar{x}, \bar{y})\}$, 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 $\mathbf{K}(\mathcal{M}(q_h) \land \top(\bar{x}) \land \chi(\bar{x}, \bar{y})) \rightarrow Ans(\bar{x})$. 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 6. [(Cima et al., 2022, Theorem 1)] *q_c is the $\mathfrak{M}$-minimally complete $\mathcal{J}$-abstraction of $q_{\mathcal{S}}$.*

The following statement is a straightforward consequence of Proposition 6.

Theorem 8. [(Cima et al., 2022, Theorem 1)] *The $\mathfrak{M}$-minimally complete $\mathcal{J}$-abstraction of $q_{\mathcal{S}}$ always exists, is computable, and can be expressed in* DD^{K}.

6.2.3. 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 6 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. [(Cima et al., 2022, Theorem 3)] *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.*

7. Non-monotone abstractions

So far, we have limited our analysis of the abstraction reasoning task by focusing on monotone query languages in the context of data integration systems. There exist, however, very simple scenarios in which the perfect abstraction can only be expressed by means of a non-monotone query.

Example 9. *Let $\mathcal{J} = \langle \mathcal{G}, \mathcal{S}, \mathcal{M} \rangle$ be such that the global schema $\mathcal{G}$ has the predicates $\{A/1, B/1, C/1\}$, the source schema $\mathcal{S}$ has the predicates $\{s_1/1, s_2/1\}$, and $\mathcal{M} = \{m_1, m_2, m_3, m_4\}$, where:*

$m_1 : s_1(x) \rightarrow A(x)$

$m_2 : s_2(x) \rightarrow A(x)$

$m_3 : s_2(x) \rightarrow B(x)$

$m_4 : s_1(x) \wedge s_2(x) \rightarrow C(x)$

Consider the query $q_{\mathcal{S}} = \{x \mid s_1(x)\}$. One can verify that the perfect $\mathcal{J}$-abstraction of $q_{\mathcal{S}}$ is the non-monotone query $q_{\mathcal{G}}$ such that, given an $\mathcal{S}$-database D, returns those x for which either $(A(x) \wedge \neg B(x))$ or $C(x)$ is known to be true, i.e. holds in every $\mathcal{G}$-database B such that $B \in mod(\mathcal{J}, D)$. $\triangle$

Motivated by the above example, in this section we summarize the most salient aspects of the results in Cima et al. (2020), which investigates the problem of finding perfect (resp. minimally complete, maximally sound) abstractions expressed in the query language *EQL-Lite*(UCQ).[4] For instance, refer to Example 9. The perfect $\mathcal{J}$-abstraction of $q_{\mathcal{S}}$ written there in natural language can be formulated through the *EQL-Lite*(UCQ) query $q_{\mathcal{G}} = \{(x) \mid \mathbf{K}(A(x) \wedge \neg B(x)) \vee \mathbf{K}(C(x))\}$. As in the case of the UCQ and the $\mathfrak{M}$ classes, it can be shown that if an *EQL-Lite*(UCQ)-maximally sound (resp., *EQL-Lite*(UCQ)-minimally complete) $\mathcal{J}$-abstraction of $q_{\mathcal{S}}$ exists, then it is unique up to $\mathcal{J}$-equivalence. Thus, in what follows, we will simply talk about *the EQL-Lite*(UCQ)-maximally sound (resp., *EQL-Lite*(UCQ)-minimally complete) $\mathcal{J}$-abstraction of $q_{\mathcal{S}}$.

A natural question that arises is whether "best" abstractions in the *EQL-Lite*(UCQ) query language always exist. Unfortunately, the following theorem shows that this is not the case for both *EQL-Lite*(UCQ)-minimally complete abstractions and *EQL-Lite*(UCQ)-maximally sound abstractions.

Theorem 10. [(Cima et al., 2020, Theorems 1 and 2)] *Both the EQL-Lite(UCQ)-minimally complete $\mathcal{J}$-abstractions of $q_{\mathcal{S}}$ and the EQL-Lite(UCQ)-maximally sound $\mathcal{J}$-abstractions of $q_{\mathcal{S}}$ may not exist.*

Due to the above negative result, which holds already for CQJFE queries $q_{\mathcal{S}}$ and data integrations systems with PGAV mappings, we now explore two alternative restricted scenarios. The former weakens the target query language for expressing abstractions by considering a fragment of *EQL-Lite*(UCQ), whereas the latter weakens the mapping language by considering a special case of GLAV. In both the restricted scenarios, we assume that source queries are CQs rather than UCQs.

4 Actually, we consider the slightly restricted version of EQL-Lite(UCQ) which does not allow the use of (in)equalities.

7.1. A restricted non-monotone query language

We now consider the problem of finding abstractions expressed in *EQL-Lite*$^{-}$(UCQ), which corresponds to the fragment of *EQL-Lite*(UCQ) where both *nested* negation and union operators are disallowed. More formally, an *EQL-Lite*$^{-}$(UCQ) query q is an expression of the form $q = \{\bar{x} \mid \varphi(\vec{x})\}$ where $\varphi(\vec{x})$ is an *EQL* formula built according to the following syntax:

$$\begin{aligned} \varphi &::= \mathbf{K}\varrho \mid \exists y.\varphi \mid \varphi_1 \wedge \varphi_2 \mid \neg\delta \\ \delta &::= \mathbf{K}\varrho \mid \exists y.\delta \end{aligned}$$

with ϱ being a disjunction of conjunction of atoms over $\mathcal{G}$ possibly involving existentially quantified variables. For instance, the *EQL-Lite*(UCQ) query $q_{\mathcal{G}}$ illustrated above, which corresponds to the perfect $\mathcal{J}$-abstraction of $q_{\mathcal{S}}$ in Example 9, is not an *EQL-Lite*$^{-}$(UCQ) query.

On the negative side, even in this scenario, maximally sound abstractions are not guaranteed to exist, and this holds already for CQJFE queries $q_{\mathcal{S}}$ and data integrations systems with PGAV mappings.

Theorem 11. [(Cima et al., 2020, Theorem 2)] *The EQL-Lite^{-}(UCQ)-maximally sound $\mathcal{J}$-abstractions of $q_{\mathcal{S}}$ may not exist.*

On the positive side, we now provide an algorithm for computing *EQL-Lite*$^{-}$(UCQ)-minimally complete $\mathcal{J}$-abstractions of CQs $q_{\mathcal{S}}$. The algorithm is similar to the one for the UCQ case (cf. Section 5), expect that all the atoms obtained when applying the mapping to the given CQ occur inside the scope of the epistemic operator **K**, binding also the existential variables coming from the input query. More precisely, given a data integration system $\mathcal{J} = \langle \mathcal{G}, \mathcal{S}, \mathcal{M} \rangle$ and a CQ $q_{\mathcal{S}} = \{\bar{x} \mid \exists \bar{y}.\phi(\bar{x}, \bar{y})\}$, the algorithm returns the *EQL-Lite*$^{-}$(UCQ) query $q_{\mathcal{G}} = \{\bar{x} \mid \exists \bar{y}.\mathbf{K}(\exists \bar{z}.\mathcal{M}(q_{\mathcal{S}}) \wedge \top(\bar{x}))\}$, where $\bar{y} \subseteq \bar{y}$ are the existential variables of $q_{\mathcal{S}}$ occurring in $\mathcal{M}(q_{\mathcal{S}})$, while $\bar{z}$ are the fresh existential variables introduced when applying $\mathcal{M}$ to $q_{\mathcal{S}}$. To see the difference with the UCQ case, recall Example 1 in the introduction and the CQ $q^2_{\mathcal{S}}$ therein. While $q^2_{\mathcal{G}}$ is the UCQ-minimally complete $\mathcal{J}$-abstraction of $q^2_{\mathcal{S}}$, the *EQL-Lite*$^{-}$(UCQ) query $\{x \mid \exists y.\mathbf{K}(g_1(x, y))\}$ returned by the above algorithm is a better complete approximation than $q^2_{\mathcal{G}}$, and is in fact the perfect $\mathcal{J}$-abstraction of $q^2_{\mathcal{S}}$.

Theorem 12. [(Cima et al., 2020, Theorem 5)] *The EQL-Lite^{-}(UCQ)-minimally complete $\mathcal{J}$-abstraction of a CQ $q_{\mathcal{S}}$ always exists and is computable.*

We further notice that the above algorithm returns queries that are monotone and that are expressible in DD$^{\mathbf{K}}$, thus proving that, without disjunction, the limited form of negation allowed in *EQL-Lite*$^{-}$(UCQ) does not give more expressive power in finding minimally complete (and therefore also perfect) abstractions of CQs. On the contrary, it can be shown that inequalities give more expressive power in finding abstractions. In particular, there exist $\mathfrak{M}$-minimally complete $\mathcal{J}$-abstractions of CQs that cannot be expressed in *EQL-Lite*$^{-}$(UCQ), whereas, as shown in the previous section, they can be expressed in DD$^{\mathbf{K}}$.

Given a query $q_{\mathcal{G}}$ as returned by the above algorithm, it is always possible to compute a UCQ q_u such that $q_u^D = q_{\mathcal{G}}^{\mathcal{J},D}$ for every $\mathcal{S}$-database D. Thus, following the same line of reasoning as the one at the end of the previous section, in this scenario we can solve the computation problem for perfect abstractions of CQs.

Theorem 13. [Cima et al. (2020)] *Checking whether there exists a query q in EQL-Lite*$^-$*(UCQ) that is the perfect* $\mathcal{J}$*-abstraction of a CQ* $q_{\mathcal{S}}$ *is decidable. Moreover, if it exists, then q is a monotone query and there is an algorithm that computes it.*

7.2. One-to-one mapping

We now examine the problem of finding abstractions in the presence of data integration systems $\mathcal{J} = \langle \mathcal{G}, \mathcal{S}, \mathcal{M} \rangle$ such that $\mathcal{M}$ is a *one-to-one mapping*. A one-to-one mapping is a special case of GLAV, constituted by a set of assertions of the form $\exists \bar{y}.s(\bar{x}, \bar{y}) \rightarrow \exists \bar{z}.g(\bar{x}, \bar{y})$, where $s(\bar{x}, \bar{y})$ and $g(\bar{x}, \bar{y})$ are single atoms without constants or repeated variables.

The first result is that the algorithm previously presented for computing *EQL-Lite*$^-$(UCQ)-minimally complete abstractions of CQs can be also used for computing *EQL-Lite*(UCQ)-minimally complete abstractions of CQs for data integration systems $\mathcal{J}$ with PGAV mapping.

Theorem 14. [(Cima et al., 2020, Theorem 3)] *Under one-to-one mappings, the EQL-Lite(UCQ)-minimally complete* $\mathcal{J}$*-abstraction of a CQ* $q_{\mathcal{S}}$ *always exists, is computable, and is a monotone query.*

Thus, using exactly the same considerations done for the case of *EQL-Lite*$^-$(UCQ), we can solve the computation problem for perfect abstractions in *EQL-Lite*(UCQ) of CQs under one-to-one mappings.

Theorem 15. [Cima et al. (2020)] *Under one-to-one mappings, checking whether there exists a query q in EQL-Lite(UCQ) that is the perfect* $\mathcal{J}$*-abstraction of a CQ* $q_{\mathcal{S}}$ *is decidable. Moreover, if it exists, then q is a monotone query and there is an algorithm that computes it.*

We now turn to the sound case under one-to-one mappings. Specifically, in this scenario, while the existence of *EQL-Lite*(UCQ)-maximally sound $\mathcal{J}$-abstractions of CQs is still an open problem, we present an algorithm for computing *EQL-Lite*(UCQ)-maximally sound $\mathcal{J}$-abstractions of CQJFEs $q_{\mathcal{S}}$. Roughly speaking, given a data integration system $\mathcal{J} = \langle \mathcal{G}, \mathcal{S}, \mathcal{M} \rangle$ with $\mathcal{M}$ a one-to-one mapping and a CQJFE $q_{\mathcal{S}}$, as a first step the algorithm computes the *EQL-Lite*(UCQ)-minimally complete $q_{\mathcal{G}}$ of $q_{\mathcal{S}}$ and its UCQ reformulation q_u such that $q_u^D = q_{\mathcal{G}}^{\mathcal{J},D}$ for each $\mathcal{S}$-database D. Then, for each CQ q' which is a disjunct of q_u such that $q' \not\sqsubseteq q_{\mathcal{S}}$, the algorithm adds in conjunction to the body of $q_{\mathcal{G}}$ the negation of the body of the *EQL-Lite*(UCQ)-minimally complete of q'. Informally, this last step prevents $q_{\mathcal{G}}$ to return answers that are not answers of $q_{\mathcal{S}}$, guaranteeing soundness of the output query. For instance, recall Example 1, and let $\mathcal{J}' = \langle \mathcal{G}, \mathcal{S}, \mathcal{M}' \rangle$ be the data integration system with $\mathcal{M}' = \{m_1, m_2, m_3\}$ a one-to-one mapping. The query returned by the algorithm is the *EQL-Lite*$^-$(UCQ) query $\{x \mid \mathbf{K}(A(x)) \land \neg\mathbf{K}(B(x))\}$, which is the *EQL-Lite*(UCQ)-maximally sound $\mathcal{J}'$ abstraction of $q_{\mathcal{S}}$.

Theorem 16. [(Cima et al., 2020, Theorem 4)] *Under one-to-one mappings, the EQL-Lite(UCQ)-maximally sound* $\mathcal{J}$*-abstraction of a CQJFE* $q_{\mathcal{S}}$ *always exists and is computable.*

We conclude this section with the following observation. The algorithms sketched above for computing "best" abstractions always return an *EQL-Lite*$^-$(UCQ) query. This directly implies that, under one-to-one mappings, the query languages *EQL-Lite*(UCQ) and *EQL-Lite*$^-$(UCQ) have the same expressive power in finding all three kinds of abstractions (perfect, minimally complete, and maximally sound).

8. Open problems

We have provided an overview of data abstraction, 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.

8.1. Data quality

While data quality is one the main issues proposed in Data-centric AI, there is no general and well-established methodology for leveraging data quality for improving Machine Learning methods.

As pointed out in Chen et al. (2021), poor data quality has a direct impact on the performance of the machine learning system that is built on the data. It is therefore important to devise techniques for validating the quality of both training and testing datasets. Recent work in this direction shows a strong correlation between the quality of the datasets and the performance of the machine learning system, and demonstrates that a rigorous evaluation of data quality is necessary for guiding the quality improvement of machine learning. We believe that formal methods like data abstraction can provide some contributions toward this goal. For example, by helping in making the semantics of training data explicit, abstraction can provide support for recognizing biases or other problems in the data used to train a Machine Learning Model. Making concrete steps in this direction is a stimulating research challenge.

8.2. 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.

8.3. Abstraction and monotonicity

In this paper we have discussed the use of DD^{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 Cima et al. (2020), the issue of checking the existence of and computing non-monotone abstractions is largely unexplored.

8.4. 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.

8.5. 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 Cima (2017), Lutz et al. (2018), and Cima et al. (2019), but the topic requires a more thorough analysis.

8.6. Reverse engineering

Abstraction has also interesting connections with the reverse-engineering problem (Barceló and Romero, 2017). 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.

8.7. 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.

Author contributions

All authors listed have made a substantial, direct, and intellectual contribution to the work and approved it for publication.

Funding

This work has been partially supported by MUR under the PRIN 2017 project HOPE (prot. 2017MMJJRE), by the EU under the H2020-EU.2.1.1 project TAILOR, grant id. 952215, and by MUR under the PNRR project PE0000013-FAIR.

Conflict of interest

The authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

Publisher's note

All claims expressed in this article are solely those of the authors and do not necessarily represent those of their affiliated organizations, or those of the publisher, the editors and the reviewers. Any product that may be evaluated in this article, or claim that may be made by its manufacturer, is not guaranteed or endorsed by the publisher.

References

Abedjan, Z., Golab, L., and Naumann, F. (2017). "Data profiling: a tutorial," in *Proceedings of the 2017 ACM International Conference on Management of Data (SIGMOD 2017)* (Chicago, IL), 1747–1751. doi: 10.1145/3035918.3054772

Afrati, F. N., and Chirkova, R. (2019). *Answering Queries Using Views. Synthesis Lectures on Data Management*, 2nd ed. San Rafael, CA: Morgan and Claypool Publishers. doi: 10.1007/978-3-031-01871-8

Barceló, P., and Romero, M. (2017). "The complexity of reverse engineering problems for conjunctive queries," in *Proceedings of the Twentieth International Conference on Database Theory (ICDT 2017), Volume 68 of Leibniz International Proceedings in Informatics*, 7:1–7:17. Available online at: https://www.dagstuhl.de/en/publications/lipics (accessed June 15, 2023).

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)* (Hyderabad), 274–279.

Calvanese, D., De Giacomo, G., Lenzerini, M., and Vardi, M. Y. (2000). "What is view-based query rewriting?" in *Proceedings of the Seventh International Workshop on Knowledge Representation meets Databases (KRDB 2000), Volume 29 of CEUR Electronic Workshop Proceedings*, 17–27. Available online at: http://ceur-ws.org/ (accessed June 15, 2023).

Calvanese, D., De Giacomo, G., Lenzerini, M., and Vardi, M. Y. (2002). "Lossless regular views," in *Proceedings of the Twenty-First ACM SIGACT SIGMOD SIGART Symposium on Principles of Database Systems (PODS 2002)* (Madison, WI: ACM), 58–66. doi: 10.1145/543613.543646

Calvanese, D., De Giacomo, G., Lenzerini, M., and Vardi, M. Y. (2007b). View-based query processing: on the relationship between rewriting, answering and losslessness. *Theor. Comput. Sci.* 371, 169–182. doi: 10.1016/j.tcs.2006.11.006

Calvanese, D., and Rosati, R. (2003). "Anwering 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*. Available online at: http://ceur-ws.org/ (accessed June 15, 2023).

Chen, H., Chen, J., and Ding, J. (2021). Data evaluation and enhancement for quality improvement of machine learning. *IEEE Trans. Reliab.* 70, 831–847. doi: 10.1109/TR.2021.3070863

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*. Available online at: http://ceur-ws.org/ (accessed June 15, 2023).

Cima, G., Console, M., Lenzerini, M., and Poggi, A. (2021). "Abstraction in data integration," in *Proceedings of the Thirty Sixth IEEE Symposium on Logic in Computer Science (LICS 2021)* (Rome: IEEE), 1–11. doi: 10.1109/LICS52264.2021.9470716

Cima, G., Console, M., Lenzerini, M., and Poggi, A. (2022). "Monotone abstractions in ontology-based data management," in *Proceedings of the Thirty-Sixth AAAI Conference on Artificial Intelligence (AAAI 2022)*, 5556–5563. doi: 10.1609/aaai.v36i5.20495

Cima, G., Lenzerini, M., and Poggi, A. (2017). "Semantic technology for open data publishing," in *Proceedings of the Seventh International Conference on Web Intelligence, Mining and Semantics (WIMS 2017)* (Amantea), 1. doi: 10.1145/3102254.3102255

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)* (Macao), 1647–1653. doi: 10.24963/ijcai.2019/228

Cima, G., Lenzerini, M., and Poggi, A. (2020). "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. doi: 10.24963/kr.2020/25

Duschka, O. M., and Genesereth, M. R. (1997). "Answering recursive queries using views," in *Proceedings of the Sixteenth ACM SIGACT SIGMOD SIGART Symposium on Principles of Database Systems (PODS 1997)* (New York, NY), 109–116. doi: 10.1145/263661.263674

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* (Cambridge, MA: AAAI/The MIT).

Eiter, T., Gottlob, G., and Mannilla, H. (1997). Disjunctive datalog. *ACM Trans. Database Syst.* 22, 364–418. doi: 10.1145/261124.261126

Fagin, R., Kolaitis, P. G., Miller, R. J., and Popa, L. (2005). Data exchange: semantics and query answering. *Theor. Comput. Sci.* 336, 89–124. doi: 10.1016/j.tcs.2004.10.033

Halevy, A. Y. (2001). Answering queries using views: a survey. *Very Large Database J.* 10, 270–294. doi: 10.1007/s007780100054

Lenzerini, M. (2002). "Data integration: a theoretical perspective," in *Proceedings of the Twenty-First ACM SIGACT SIGMOD SIGART Symposium on Principles of Database Systems (PODS 2002)* (New York, NY: ACM), 233–246. doi: 10.1145/543613.543644

Levy, A. Y., Mendelzon, A. O., Sagiv, Y., and Srivastava, D. (1995). "Answering queries using views," in *Proceedings of the Fourteenth ACM SIGACT SIGMOD SIGART Symposium on Principles of Database Systems (PODS 1995)* (San Jose, CA: ACM Press), 95–104. doi: 10.1145/212433.220198

Levy, A. Y., Rajaraman, A., and Ordille, J. J. (1996). "Querying heterogenous information sources using source descriptions," in *Proceedings of the Twenty-Second International Conference on Very Large Data Bases (VLDB 1996)* (Mumbai), 251–262.

Levy, A. Y., and Rousset, M.-C. (1998). Combining Horn rules and description logics in CARIN. *Artif Intell.* 104, 165–209. doi: 10.1016/S0004-3702(98)00048-4

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)* (Tempe, AZ), 389–398.

Nash, A., Segoufin, L., and Vianu, V. (2010). Views and queries: aeterminacy and rewriting. *ACM Trans. Database Syst.* 35, 1–21. doi: 10.1145/1806907.1806913

Pottinger, R., and Halevy, A. Y. (2001). MiniCon: a scalable algorithm for answering queries using views. *Very Large Database J.* 10, 182–198. doi: 10.1007/s007780100048

Sagiv, Y., and Yannakakis, M. (1980). Equivalences among relational expressions with the union and difference operators. *J. ACM* 27, 633–655. doi: 10.1145/322217.322221