Proceedings of the 22nd International Conference on Principles of Knowledge Representation and Reasoning
Main Track

Abstractions of Queries in Ontology-Based Data Access

Michel Leclère , Marie-Laure Mugnier , Guillaume Pérution-Kihli
LIRMM, Inria, University of Montpellier, CNRS, Montpellier, France
{michel.leclere, marie-laure.mugnier, guillaume.perution-kihli}@lirmm.fr

Abstract

In ontology-based data access (OBDA), multiple data sources are integrated via mappings to an ontology. We consider an OBDA setting based on existential rules and the certain answer semantics. We address the recent issue of query abstraction, which consists of abstracting data queries by translating them to the ontology layer. Since a perfect abstraction may not exist, the notions of minimally complete and maximally sound abstractions have been introduced. We study abstractions within an extension of UCQs with a limited form of inequality and a special predicate marking database constants. While this extension does not lead to an increased complexity of the problems of interest, it is able to express minimally complete abstractions, hence perfect abstractions when they exist. We also characterize maximally sound abstractions by making a new connection with the notion of maximum recovery stemming from data exchange.

1 Introduction

In ontology-based data access (OBDA), multiple data sources are integrated via mappings to a shared ontology (Poggi et al. 2008). This approach generalizes classical data integration by replacing the global schema with an ontology, enabling users to query data at a high level of abstraction while benefiting from reasoning over domain knowledge.

A central problem in OBDA is ontological query answering. Given an OBDA specification $\Sigma = (S, O, \mathcal{M}, \mathcal{O})$, where S is a data schema, O an ontology schema, $\mathcal{M}$ a mapping from S to O, and $\mathcal{O}$ an ontology over O, and a database D over S, the goal is to compute the answers to an ontological query Q_O over the knowledge base (KB) defined by D, $\mathcal{M}$, and $\mathcal{O}$. Such KB may remain virtual: then, Q_O is rewritten into a data-level query Q_S that is a *perfect* translation of Q_O, which means that, for any database D over S, the answers to Q_S on D coincide with the answers to Q_O on the virtual KB. While Q_S is simply evaluated on D, the answers to Q_O are logically entailed by the KB—corresponding to the standard semantics of *certain answers*, i.e., answers that hold in every model of the KB.

More recently, research in OBDA has also investigated query translation in the opposite direction, i.e., from data queries to ontological queries—a task called *query abstraction* (Cima, Poggi, and Lenzerini 2023). This issue arises in a range of relevant scenarios. First, during the (often incremental) design of an OBDA system, query abstraction can be used to verify whether the mapping provides adequate coverage of important data queries (Lutz, Marti, and Sabellek 2018). Second, query abstraction is a means to automatically characterize the semantics of data services implemented at the data-source level—which can be seen as a form of reverse engineering (Cima, Lenzerini, and Poggi 2019). This capability opens the door to promising applications, such as providing open datasets supplied by organizations with high-level semantics, or enhancing the FAIRness of data services (Cima, Poggi, and Lenzerini 2023).

In this paper, we investigate query abstraction within an OBDA setting based on *existential rules* (Baget et al. 2009; Calí, Gottlob, and Lukasiewicz 2009), aka TGDs in database theory (Abiteboul, Hull, and Vianu 1995).We use existential rules for both the mapping (we obtain Global-Local-As-View (GLAV) mappings) and the ontology. So doing, we generalize classical OBDA frameworks based on Horn description logics (DLs), as these can be seen as specific existential rule classes. Moreover, this uniform setting allows us to rely on the same fundamental tools to handle mappings and ontologies, namely the *chase* and *query rewriting*. It also helps to make connections with database theory.

Most work in OBDA has focused on ontological queries expressed as unions of conjunctive queries (UCQs), the core relational database queries. FO-rewritable ontologies—such as those expressed with the main dialects of the DL-Lite family (Calvanese et al. 2007) or certain fragments of existential rules (Baget et al. 2009; Calì, Gottlob, and Pieris 2010)—guarantee that every ontological UCQ admits a perfect rewriting as a UCQ. In the other direction, query translation appears to be much more challenging. To start with, a perfect abstraction of a data (U)CQ may not exist at all, even with an empty ontology. Apart from the fact that mappings may not transfer all the answers, they may also make source relations indistinguishable, as illustrated next.

Example 1. *Let $S = \{s_1, s_2\}$, $O = \{r\}$ and $\mathcal{M} = \{m_1, m_2\}$, with m_1: $s_1(x) \rightarrow r(x)$ and m_2: $s_2(x) \rightarrow r(x)$. The query $Q_S(u) = s_1(u)$ has no perfect abstraction through $\mathcal{M}$. In particular, the ontological query $Q_O(u) = r(u)$ captures all answers to Q_S, i.e., it is a complete abstraction of Q_S, but it is not a sound abstraction of Q_S, as it also retrieves values coming from the source relation s_2. In fact, Q_O would be a perfect abstraction of $s_1(u) \vee s_2(u)$.*

The topic began to be investigated only recently (Lutz,

Setting	Verification	Existence
(U)CQ, GAV, bounded arity	Π_2^{P}-C (*)	Π_2^{P}-C (*)
UCQ$^{\neq,\mathbf{C}}$, GLAV bounded frontier	Π_2^{P}-C	Π_2^{P}-C
UCQ$^{\neq,\mathbf{C}}$, GLAV	Π_2^{P}-C	in Co-NExpTime

(*) from (Lutz, Marti, and Sabellek 2018)

Table 1: Complexity of problems related to perfect $\mathcal{M}$-abstractions

Marti, and Sabellek 2018; Cima, Lenzerini, and Poggi 2019; Cima et al. 2021). However, OBDA can build on a rich body of previous work in database theory. As pointed out by Lutz et al. (Lutz, Marti, and Sabellek 2018), when the ontology is ignored, deciding if a source query has a perfect abstraction as a UCQ is closely related to the long-studied problem of query expressibility with views (Nash, Segoufin, and Vianu 2010). We will establish a new link with database theory by considering specific inverse mappings from data exchange (Fagin et al. 2008; Arenas, Pérez, and Riveros 2009).

Since a perfect abstraction of a UCQ Q_S may not exist, Cima et al. introduced *minimally complete* and *maximally sound* abstractions, which respectively provide a minimal superset and a maximal subset of answers to Q_S (Cima, Lenzerini, and Poggi 2019). Here, minimality and maximality are with respect to the set of queries definable in some target query language (e.g., UCQ).

With the aim of capturing perfect abstractions of UCQs, we consider an extension of UCQ, denoted by UCQ$^{\neq,\mathbf{C}}$, with a limited form of inequality (variables that occur in inequalities must be mapped to constants) and a special unary predicate marking database constants, denoted by **C**—which was introduced in (Fagin et al. 2008) to define inverse mappings. As a data query language, UCQ$^{\neq}$ is of great interest in practice, especially as inequalities are in fact not limited when the queried databases are ground. However, the true benefit of UCQ$^{\neq,\mathbf{C}}$ is as an ontological query language. Indeed, while a perfect abstraction of a UCQ, when it exists, may not be expressible as a UCQ, we will show that it is always expressible as a UCQ$^{\neq,\mathbf{C}}$.

We will now detail our main contributions. To distinguish between the settings of database integration, i.e., without ontology, and OBDA, we will use the terms $\mathcal{M}$-abstraction and Σ-abstraction, respectively.

(1) Complexity of the problems of interest within the UCQ$^{\neq,\mathbf{C}}$ class. In Section 3, we study the computational impact of extending UCQ to UCQ$^{\neq,\mathbf{C}}$. In particular, we show that this does not lead to increased complexity of verifying whether a candidate $\mathcal{M}$-abstraction is perfect, which is Π_2^{P}-complete (see Table 1, first column). We also exhibit an FO-rewritable rule fragment for which this problem with a Σ-abstraction remains in Π_2^{P} (this subsumes earlier results on DL-Lite). Other complexity results are established later.

(2) Capturing minimally-complete and perfect abstractions of UCQ$^{\neq,\mathbf{C}}$s. In Section 4, we first point out that a non-Boolean query Q_S may not have a *complete* abstraction and characterize the conditions under which it has one (which only depends on the interactions between Q_S and the mapping). For a UCQ Q_S that has a complete abstraction, it is known that, by applying $\mathcal{M}$ to Q_S (i.e., "chasing" Q_S with $\mathcal{M}$), one produces a UCQ Q_O that is *minimally complete* w.r.t. all $\mathcal{M}$-abstractions definable as UCQs (Lutz, Marti, and Sabellek 2018; Cima, Lenzerini, and Poggi 2019). However, we point out that Q_S may have a better $\mathcal{M}$-abstraction (i.e., a complete $\mathcal{M}$-abstraction with fewer unwanted answers) in the UCQ$^{\neq,\mathbf{C}}$ class, which can be computed by a modified chase. The important result here is that UCQ$^{\neq,\mathbf{C}}$ is in fact able to express a minimally complete $\mathcal{M}$-abstraction (and Σ-abstraction as well) of any source UCQ$^{\neq,\mathbf{C}}$, where minimality is w.r.t. all possible ontological queries (i.e., queries on schema O expressed in any language, provided that they are answered with the semantics of certain answers). We take special care in exhibiting the properties behind this result. It follows that, when a perfect Σ-abstraction of a UCQ$^{\neq,\mathbf{C}}$ exists, it can be expressed in this class. Moreover, UCQ$^{\neq,\mathbf{C}}$ is a minimal language with this property, even when the source query is a plain UCQ. Finally, we use these results to show that the complexity of deciding whether a perfect $\mathcal{M}$-abstraction exists for a given UCQ$^{\neq,\mathbf{C}}$ is Π_2^{P}-complete when the mapping rules have a frontier of bounded size (the frontier being the set of variables shared between the body and head) and in Co-NExpTime otherwise (see Table 1, second column).

(3) Characterizing maximally sound abstractions of UCQ$^{\neq,\mathbf{C}}$s. There is no known algorithm that builds a *maximally sound* $\mathcal{M}$-abstraction of a UCQ as a UCQ (or UCQ$^{\neq,\mathbf{C}}$), when such abstraction exists, except in a very specific case (Cima, Lenzerini, and Poggi 2019). Whether the associated existence problem is decidable is an open question. In Section 5, we make a step towards better understanding by drawing a connection with the notion of a *maximum recovery* investigated in a quite different context, namely data exchange. In data exchange, mappings are used to specify how to transfer data from a source schema to a target schema; then, a maximum recovery of a mapping $\mathcal{M}$ is an inverse mapping from target to source that allows one to recover the most answers to source queries when it is composed with $\mathcal{M}$ (Arenas, Pérez, and Riveros 2009). To express such inverse mapping, a strictly more expressive language than GLAV is required, as disjunction in rule heads is needed. We show that a maximally sound $\mathcal{M}$-abstraction of a UCQ$^{\neq,\mathbf{C}}$ Q_S can be equivalently defined as the *rewriting* of Q_S with a maximum recovery of $\mathcal{M}$. To define such rewriting, we rely on a rewriting operator for UCQs and disjunctive existential rules introduced in (Leclère, Mugnier, and Pérution-Kihli 2023). In passing, we correct a wrong claim from the literature, that a maximum recovery for CQs could always be expressed by a conjunctive mapping (i.e., without disjunction) (Arenas et al. 2009). We then extend the notion of a maximum recovery to an OBDA specification (i.e., we add an ontology) and show that a maximum recovery can still be expressed by the same form of disjunctive mapping when the ontology is FO-rewritable; hence, in this case, a maximally sound Σ-abstraction of a UCQ$^{\neq,\mathbf{C}}$ can also be characterized as a rewriting with a maximum recovery. When this rewriting is finite, it is a UCQ$^{\neq,\mathbf{C}}$.

2 Preliminaries

We assume the reader has basic knowledge in database theory and ontological query answering.

Database theory We consider a denumerable set of constants $\mathcal{C}$. A *term* is a constant from $\mathcal{C}$ or a variable. A *schema* is a finite set of predicates. An atom with predicate p is called a *p-atom*. Given a schema P, a *P-atom* is any p-atom with $p \in P$. A *P-instance* is a possibly infinite set of P-atoms, also seen as an *interpretation* of $(P, \mathcal{C})$, in which constants are interpreted by themselves. A finite instance is also seen as an existentially-closed positive conjunctive FO-formula.[1] A *database* is a ground finite instance. Given an instance I, we denote by $\texttt{vars}(I)$, $\texttt{consts}(I)$ and $\texttt{terms}(I)$ its sets of variables, constants and terms, respectively. A tuple $\vec{x}$ of pairwise distinct variables is sometimes seen as a set. Given instances I_1 and I_2, a *homomorphism* h from I_1 to I_2 is a substitution of $\texttt{vars}(I_1)$ by $\texttt{terms}(I_2)$ such that $h(I_1) \subseteq I_2$. We also denote it by $h : I_1 \to I_2$ and we say that I_1 *maps* to I_2 (by h). When convenient, we extend the domain of a homomorphism to constants.

We define the notion of a query in an abstract way, i.e., independently from any syntactic form: an n-ary *query* Q on schema P is any function that maps each P-instance I to a set of n-ary tuples on $\texttt{consts}(I)$. We denote by $Q(I)$ the set of answers to Q on I. An n-ary *FO-query* on P has the form $Q(\vec{x})$, where Q is an FO-formula on P and $\vec{x}$ are n free variables from Q (the answer variables). A *Boolean* query is 0-ary. Given an instance I, a tuple $\vec{c} \subseteq \texttt{consts}(I)$ with $|\vec{c}| = |\vec{x}|$ is an *answer* to an FO-query $Q(\vec{x})$ on I if I is a model of $Q(\vec{c})$, where $Q(\vec{c})$ is obtained by substituting each i-th variable in $\vec{x}$ by the i-th constant in $\vec{c}$. A *conjunctive query* (CQ) has the form $q(\vec{x}) = \exists \vec{y}.\ \phi[\vec{x}, \vec{y}]$, where ϕ is a finite conjunction of atoms and $\vec{x} \cup \vec{y} = \texttt{vars}(\phi)$; to denote a CQ, we write q rather than Q. A CQ is *with join-free existential variables* (CQJFE) if each existential variable occurs only once. A *union of conjunctive queries* (UCQ) is a finite disjunction of CQs with the same tuple of answer variables. A UCQ may contain equalities, but to simplify technical developments we will silently assume that, before any processing, equalities are removed by substituting variables.

Given queries Q_1 and Q_2, Q_1 is *contained* in Q_2, denoted by $Q_1 \sqsubseteq Q_2$, if $Q_1(I) \subseteq Q_2(I)$ for any instance I. Given two CQs $q_1(\vec{x}_1)$ and $q_2(\vec{x}_2)$, a *query homomorphism* from q_1 to q_2 is a homomorphism h from q_1 to q_2 such that $h(\vec{x}_1) = \vec{x}_2$. It is well known that, given CQs q_1 and q_2, $q_1 \sqsubseteq q_2$ iff there is a query homomorphism from q_2 to q_1. Note that for UCQs Q_1 and Q_2, we have $Q_1 \sqsubseteq Q_2$ iff for all $q_1 \in Q_1$, there is $q_2 \in Q_2$ such that $q_1 \sqsubseteq q_2$.

When information is incomplete, a set of instances is often considered instead of a single instance; then, the set of *certain answers* to a query Q on a set of instances $\mathcal{I}$ is $\texttt{certain}(Q, \mathcal{I}) = \bigcap_{I \in \mathcal{I}} Q(I)$.

Rules and mappings An *existential rule* R (or simply *rule* hereafter) is a closed formula of the form $\forall \vec{x}.\ (\exists \vec{y}.\ B[\vec{x}, \vec{y}]) \to \exists \vec{z}.\ H[\vec{x}, \vec{z}]$ where B and $H \neq \emptyset$ are finite conjunctions, respectively called the *body* and the *head* of R, also noted $\texttt{body}(R)$ and $\texttt{head}(R)$, and $\vec{x}, \vec{y}$ and $\vec{z}$ are pairwise disjoint tuples of variables and $\texttt{consts}(H) \subseteq \texttt{consts}(B)$. The *frontier* of R is $\texttt{fr}(R) = \vec{x}$. Note that $\texttt{body}(R)$ and $\texttt{head}(R)$ can be seen as CQs with answer variables $\texttt{fr}(R)$. R is *Datalog* if $\vec{z} = \emptyset$. Given schemas $\mathcal{S}$ and $\mathcal{T}$, called source and target respectively, with $\mathcal{S} \cap \mathcal{T} = \emptyset$, an *$\mathcal{S}$-to-$\mathcal{T}$ rule* has a body made of $\mathcal{S}$-atoms and a head made of $\mathcal{T}$-atoms. A GLAV *mapping* from $\mathcal{S}$ to $\mathcal{T}$ is a finite set of $\mathcal{S}$-to-$\mathcal{T}$ rules. A GLAV mapping is *GAV* if it is a set of Datalog rules.

OBDA An *OBDA specification* is a quadruplet $\Sigma = (S, O, \mathcal{M}, \mathcal{R})$ where S is the source schema, O the ontology schema, $\mathcal{M}$ a (GLAV) mapping from S to O and $\mathcal{R}$ a finite set of rules over O. An *OBDA system* is a pair (D, Σ) with Σ an OBDA specification and D an S-database. A *source query*, usually denoted by Q_S, is defined on S and an *ontological query*, usually denoted by Q_O, is defined on O. The answers to an ontological query Q_O on an OBDA system (D, Σ), denoted by $Q_O^{\texttt{cert}}(D, \Sigma)$, are its certain answers on the *models of the OBDA system*, denoted by $\texttt{Mod}_\Sigma(D)$:

$$\texttt{Mod}_\Sigma(D) = \{O\text{-instance } I \mid D \cup I \models \mathcal{M} \text{ and } I \models \mathcal{R}\}$$

$$Q_O^{\texttt{cert}}(D, \Sigma) = \texttt{certain}(Q_O, \texttt{Mod}_\Sigma(D))$$

When we want to ignore the set of rules $\mathcal{R}$ of an OBDA system, we use notations with $\mathcal{M}$ instead of Σ. Query containment is extended to ontological queries: $Q_1 \sqsubseteq_\Sigma Q_2$ if $Q_1^{\texttt{cert}}(D, \Sigma) \subseteq Q_2^{\texttt{cert}}(D, \Sigma)$ for every S-database D.

Query abstraction For an OBDA specification Σ, a query Q_O on O is a *complete* (resp. *sound*) Σ-abstraction of a query Q_S on S if $Q_S(D) \subseteq Q_O^{\texttt{cert}}(D, \Sigma)$ (resp. $Q_O^{\texttt{cert}}(D, \Sigma) \subseteq Q_S(D)$) for all D; Q_O is a *perfect* Σ-abstraction if it is both sound and complete. We also consider "best" approximations of perfect abstractions within a target query class: Given a query class $\mathbb{Q}$, an ontological query $Q_O \in \mathbb{Q}$ is a *$\mathbb{Q}$-minimally complete* Σ-abstraction of Q_S if Q_O is a complete Σ-abstraction of Q_S and $Q_O \sqsubseteq_\Sigma Q'_O$ for any $Q'_O \in \mathbb{Q}$ that is a complete Σ-abstraction of Q_S; similarly, Q_O is a *$\mathbb{Q}$-maximally sound* Σ-abstraction of Q_S if Q_O is a sound Σ-abstraction of Q_S and $Q'_O \sqsubseteq_\Sigma Q_O$ for any $Q'_O \in \mathbb{Q}$ that is a sound Σ-abstraction of Q_S. *When we do not specify $\mathbb{Q}$, we consider* any *query, as abstractly defined above.* Finally, when $\mathcal{R}$ is ignored, we speak of *$\mathcal{M}$-abstraction* instead of Σ-abstraction.

Let $\mathcal{X} \in \{\mathcal{M}, \Sigma\}$ and $\mathbb{Q}$ be a class of queries. We study three kinds of problems:

- *Verifying* if an abstraction is perfect: the $\mathbb{Q}$ $\mathcal{X}$-perfectness *verification* problem takes as input $\mathcal{X}$, $\mathbb{Q}$-queries Q_S and Q_O, and asks if Q_O is a perfect $\mathcal{X}$-abstraction of Q_S. This problem is decomposed into verifying $\mathcal{X}$-soundness and $\mathcal{X}$-completeness.
- *Deciding* the existence of a perfect abstraction: the $\mathbb{Q}$ $\mathcal{X}$-*expressibility* problem takes as input $\mathcal{X}$ and $Q_S \in \mathbb{Q}$, and asks if a perfect $\mathcal{X}$-abstraction of Q_S is expressible in $\mathbb{Q}$.
- *Computing* abstractions satisfying property $P \in$ {perfectness, maximal soundness, minimal completeness}; Given $\mathcal{X}$ and a $\mathbb{Q}$-query Q_S, the task is to compute a $\mathbb{Q}$-query Q_O that is an $\mathcal{X}$-abstraction of Q_S with property P, when such an abstraction exists.

[1] Null values in instances are seen as existential variables.

Reasoning tools Let $\mathcal{R}$ be a set of rules. Given an instance I, the *chase* with $\mathcal{R}$ exhaustively applies rules from $\mathcal{R}$ to I, towards a fixpoint. [2] We denote by $\texttt{chase}(I, \mathcal{R})$ the (possibly infinite) resulting instance. Crucially, $\texttt{chase}(I, \mathcal{R})$ is a *universal* model of I and $\mathcal{R}$, i.e., it maps to any model of I and $\mathcal{R}$. Given a UCQ Q, *query rewriting* with $\mathcal{R}$ starts from Q seen as a set of CQs; it iteratively rewrites a CQ from the set with a rule from $\mathcal{R}$ and adds the resulting CQ to the set, while keeping a minimal set w.r.t. query containment, towards a fixpoint. We consider here the rewriting algorithm from (König et al. 2015), based on so-called piece-unifiers, and denote by $\texttt{rew}(Q, \mathcal{R})$ the possibly infinite set (i.e., union) of CQs it produces. Each rewriting step is based on a piece-unifier, which unifies a subset q' of a CQ q with a subset of a rule head h' while satisfying the following *piece condition*: an existential variable from h' can only be unified with variables from q', which furthermore do not occur in $q \setminus q'$. A fundamental property holds: For any instance I, set of rules $\mathcal{R}$ and Boolean UCQ Q, $I \cup \mathcal{R}$ entails Q iff $\texttt{chase}(I, \mathcal{R}) \models Q$ iff $I \models \texttt{rew}(Q, \mathcal{R})$. A pair $(Q, \mathcal{R})$ where Q is a UCQ, is *FO-rewritable* if there is a UCQ Q' such that $Q'(I) = \texttt{certain}(Q, \texttt{chase}(I, \mathcal{R}))$ for any instance I. Note that $(Q, \mathcal{R})$ is FO-rewritable iff $\texttt{rew}(Q, \mathcal{R})$ is finite. A rule set $\mathcal{R}$ is FO-rewritable, or *fus*, if $(Q, \mathcal{R})$ is FO-rewritable for any UCQ Q.

When mappings (from $\mathcal{S}$ to $\mathcal{T}$) are considered instead of rules on a single schema, the result of the chase or of query rewriting is restricted to the relevant schema ($\mathcal{T}$ or $\mathcal{S}$). Given a mapping $\mathcal{M}$ from $\mathcal{S}$ to $\mathcal{T}$ and a (finite) $\mathcal{S}$-instance I, the chase of I with $\mathcal{M}$ yields the $\mathcal{T}$-instance $\mathcal{M}(I) = \{A \in \texttt{chase}(I, \mathcal{M}) \mid A \text{ is a } \mathcal{T}\text{-atom}\}$. Given a CQ q_s, the chase of q_s with $\mathcal{M}$ is defined similarly, on the atoms of q_s seen as an instance, provided that each answer variable from q_s occurs in a $\mathcal{T}$-atom of $\mathcal{M}(q_s)$; hence, $\mathcal{M}(q_s)$ can be seen as a CQ with the same arity as q_s. The chase is further extended to a UCQ: given an $\mathcal{S}$-UCQ Q_s, $\mathcal{M}(Q_s)$ is defined only if $\mathcal{M}(q_{s_i})$ is defined for each $q_{s_i} \in Q_s$; then $\mathcal{M}(Q_s)$ is the $\mathcal{T}$-UCQ obtained by making the disjunction of the $\mathcal{M}(q_{s_i})$, for all $q_{s_i} \in Q_s$. Query rewriting with a mapping $\mathcal{M}$ takes as input a $\mathcal{T}$-UCQ Q_t and produces the $\mathcal{S}$-UCQ $\mathcal{M}^-(Q_t) = \{q \in \texttt{rew}(Q_t, \mathcal{M}) \mid q \text{ is a query on } \mathcal{S}\}$. Note that $\mathcal{M}(Q_s)$ and $\mathcal{M}^-(Q_t)$ are always finite.

It is convenient to use the same notations for mappings and general rule sets; so we also note $\mathcal{R}(I) = \texttt{chase}(I, \mathcal{R})$ and $\mathcal{R}^-(Q) = \texttt{rew}(Q, \mathcal{R})$. For an OBDA specification $\Sigma = (S, O, \mathcal{M}, \mathcal{R})$, we note $\Sigma(I) = \mathcal{R}(\mathcal{M}(I))$ and $\Sigma^-(Q) = \mathcal{M}^-(\mathcal{R}^-(Q))$. When $\mathcal{R}^-(Q)$ is not finite, it is seen as an infinitary query, as in (Lutz, Marti, and Sabellek 2018).

Properties of OBDA systems We finally list some fundamental properties of OBDA systems, which are explicit or implicit in previous work, except that we consider existential rules instead of specific Horn description logics.

Proposition 2. *For any OBDA specification $\Sigma = (S, O, \mathcal{M}, \mathcal{R})$, S-database D, S-UCQ Q_S and O-UCQ Q_O the following holds:*

1. $Q_O^{\texttt{cert}}(D, \mathcal{M}) \subseteq Q_O^{\texttt{cert}}(D, \Sigma)$
2. $Q_O^{\texttt{cert}}(D, \Sigma) = Q_O(\mathcal{R}(\mathcal{M}(D))) = (\mathcal{M}^-(\mathcal{R}^-(Q_O)))(D)$
3. *If* $Q_S \sqsubseteq Q'_S$ *then* $\mathcal{M}(Q_S) \sqsubseteq_{\mathcal{M}} \mathcal{M}(Q'_S)$
4. $Q_O \sqsubseteq_{\mathcal{M}} Q'_O$ *iff* $\mathcal{M}^-(Q_O) \sqsubseteq \mathcal{M}^-(Q'_O)$
5. $Q_O \sqsubseteq_{\Sigma} Q'_O$ *iff* $\mathcal{R}^-(Q_O) \sqsubseteq_{\mathcal{M}} \mathcal{R}^-(Q'_O)$

When a perfect $\mathcal{M}$-abstraction of a UCQ Q_S is expressible as a UCQ, $\mathcal{M}(Q_S)$ is such an abstraction (this follows, e.g., from (Nash, Segoufin, and Vianu 2010)). Moreover, when Q_S has a complete $\mathcal{M}$-abstraction, $\mathcal{M}(Q_S)$ is such an abstraction, which is even UCQ-minimally complete (Cima, Lenzerini, and Poggi 2019). Hence, a UCQ Q_S has a perfect $\mathcal{M}$-abstraction expressible as a UCQ iff $\mathcal{M}(Q_S)$ is sound, which can be checked by verifying if $\mathcal{M}^-(\mathcal{M}(Q_S)) \sqsubseteq Q_S$. This result has been extended to a Σ-abstraction in some specific OBDA settings.[3]

Next, we use the following notations: Σ is an OBDA specification defined by $(S, O, \mathcal{M}, \mathcal{R})$, D is an S-database, and Q_S, Q_O are queries over S and O respectively.

3 From UCQ to UCQ$^{\neq,\mathbf{C}}$

The class UCQ$^{\neq,\mathbf{C}}$ extends UCQ with two special predicates: a restricted form of inequality ($\neq$) and a unary predicate $\mathbf{C}$ stating that its argument is a constant.[4] In the context of query abstraction, $\mathbf{C}$ is used to mark variables in ontological queries that must be mapped to values coming from the database, whereas $\neq$ allows one to distinguish between different ways of matching query variables.

Definition 3 (UCQ$^{\neq,\mathbf{C}}$)**.** *A UCQ$^{\neq,\mathbf{C}}$ Q is a UCQ extended with atoms on special predicates $\neq$ (binary) and $\mathbf{C}$ (unary), such that: (1) All the variables of Q occur in standard atoms, and (2) The terms of any $\neq$-atom are constants, answer variables, or variables that occur in a $\mathbf{C}$-atom. We denote by $std(Q)$ the restriction of Q to standard atoms.*

The $\mathbf{C}$-atoms on answer variables and constants, as well as $\neq$-atoms over constants, can be made explicit or not. Also, $\mathbf{C}$ is useless in source queries since databases are ground, but for simplicity we keep the same query class at the data and the ontology levels. A UCQ$^{\neq,\mathbf{C}}$ Q is *consistent* if there exists a database D such that $Q(D) \neq \emptyset$, which can be checked in polynomial time. Next, we implicitly assume that queries are consistent.

All the technical tools for CQs are extended to CQ$^{\neq,\mathbf{C}}$ in the natural way. A *homomorphism* $h : q \to I$ is a homomorphism from $std(q)$ to I such that (i) for all $\mathbf{C}(t) \in q$, $h(t)$ is a constant, and (ii) for all $t_1 \neq t_2$, $h(t_1) \neq h(t_2)$. A *query homomorphism* $h : q_1 \to q_2$ is a homomorphism from $std(q_1)$ to $std(q_2)$ such that (i) for all $\mathbf{C}(t) \in q_1$, $h(\mathbf{C}(t)) \in q_2$ or $h(t)$ is a constant or $h(t)$ is an answer variable, and (ii) for all $t_1 \neq t_2 \in q_1$, $h(t_1 \neq t_2) \in Q_2$ or $h(t_1)$ and $h(t_2)$ are distinct constants. The chase of q with $\mathcal{M}$, i.e., $\mathcal{M}(q)$, is obtained from $\mathcal{M}(std(q))$ by (1) adding the atoms $(t_1 \neq t_2)$ from q if t_1 and t_2 are both in $\mathcal{M}(std(q))$, and (2) adding $\mathbf{C}(x)$ on the variables from

[2] This formulation corresponds to the simplest chase variant, called oblivious (Calì, Gottlob, and Kifer 2008).

[3] GAV mappings (Lutz, Marti, and Sabellek 2018) or ontology in DL-Lite$_{\mathcal{R}}$ (Cima, Lenzerini, and Poggi 2019).

[4] Formally: for any FO-interpretation $\mathcal{I} = (\Delta, .^{\mathcal{I}})$ it holds that $\mathbf{C}^{\mathcal{I}} = \texttt{consts}(\mathcal{I})$ and $\neq^{\mathcal{I}} = \{(d_1, d_2) \in \Delta^2 \mid d_1 \neq d_2\}$.

$\texttt{vars}(q) \cap \texttt{vars}(\mathcal{M}(std(q)))$. Hence, if the $\mathbf{C}$-atoms in q are made explicit, $\mathcal{M}(q)$ is defined as for a plain (U)CQ w.r.t. target predicates $\mathcal{T} \cup \{\mathbf{C}, \neq\}$. Similarly, the rewriting of q with $\mathcal{M}$, i.e., $\mathcal{M}^-(q)$, is defined w.r.t. source predicates $\mathcal{S} \cup \{\mathbf{C}, \neq\}$; inconsistent CQs can be removed from $\mathcal{M}^-(q)$ and the $\mathbf{C}$-atoms can be made implicit. Note that, by definition of a piece-unifier, an existential variable from a rule head cannot be unified with a variable that occurs in a $\mathbf{C}$- or $\neq$-atom. These notions are illustrated in Ex. 4. They are extended to UCQ$^{\neq,\mathbf{C}}$ as expected. Note that the fundamental properties from Prop. 2 still hold.

Example 4. *Let $\mathcal{M}$ be the following mapping:*

$m_1 : s_1(x, y) \to p(x, y)$ $\quad m_3 : s_2(x) \to p(x, x)$
$m_2 : s_1(x, x) \to r(x)$ $\quad m_4 : s_3(x) \to \exists y.p(x, y)$

Let the source CQ$^{\neq}$ $q(u) = \exists v.\, s_1(u, v) \wedge u \neq v$. Then, $\mathcal{M}(q(u)) = p(u, v) \wedge u \neq v \wedge \boldsymbol{C}(v)$ (with $\boldsymbol{C}(u)$ left implicit as u is an answer variable) and $\mathcal{M}^-(\mathcal{M}(q(u))) \equiv q(u)$. Let us detail the rewriting of $\mathcal{M}(q(u))$: $p(u, v)$ can be unified with $\texttt{head}(m_1)$, which yields $q(u)$, as well as with $\texttt{head}(m_3)$, which yields the inconsistent CQ $s_2(u) \wedge u \neq u$ (discarded); $p(u, v)$ cannot be unified with $\texttt{head}(m_4)$ because v would be unified with the existential variable y, while it also occurs in $\boldsymbol{C}(v)$ and in $u \neq v$.

The next example shows that a UCQ may have no perfect abstraction in UCQ but one in UCQ$^{\neq,\mathbf{C}}$.

Example 5 (Perfect abstraction within UCQ$^{\neq,\mathbf{C}}$)**.** *Consider again $\mathcal{M}$ from Ex. 4. The CQ $q_S(u) = \exists v.s_1(u, v)$ has no perfect $\mathcal{M}$-abstraction in UCQ. Indeed, $\mathcal{M}(q_S) = \exists v.p(u, v)$, and $Q'_S = \mathcal{M}^-(\mathcal{M}(q_S)) = \exists v.s_1(u, v) \vee s_2(u) \vee s_3(u)$ strictly contains q_S. Hence, $\mathcal{M}(q_S)$ is not a sound abstraction. However, the following UCQ$^{\neq,\boldsymbol{C}}$ is a perfect abstraction of q_S: $Q_O(u) = q^1_O(u) \vee q^2_O(u)$, with $q^1_O(u) = \exists v.\, p(u, v) \wedge \boldsymbol{C}(v) \wedge u \neq v$ and $q^2_O(u) = r(u) \wedge p(u, u)$. Such query Q_O is the output of the algorithm given in Section 4. Intuitively, Q_O is a sound abstraction because it does not retrieve the p-atoms produced by m_3 and m_4. Indeed, q^1_O only retrieves p-atoms produced by m_1, thanks to $u \neq v$ and $\boldsymbol{C}(v)$. However, q^1_O is not a complete abstraction as it avoids atoms of the form $p(a, a)$ that can be produced by m_1. Subquery q^2_O compensates for this elimination. Note that the atom $p(u, u)$ in q^2_O is actually not needed: indeed, when an atom $r(a)$ is produced (by m_2), the atom $p(a, a)$ is necessarily produced (by m_1). More formally, let us check that $\mathcal{M}^-(Q_O) \equiv q_S$. The rewriting of q^1_O yields the CQ $\exists v.\, s_1(u, v) \wedge u \neq v$, see Ex. 4. The rewriting of q^2_O yields two CQs: 1. $s_1(u, u)$, obtained by unifying $r(u)$ with $\texttt{head}(m_2)$ and $p(u, u)$ with $\texttt{head}(m_1)$; and 2. $(s_1(u, u) \wedge s_2(u))$, obtained by unifying $r(u)$ with $\texttt{head}(m_2)$ and $p(u, u)$ with $\texttt{head}(m_3)$. The latter query is contained in the former, hence can be ignored. We obtain $\mathcal{M}^-(Q_O) = (\exists v.\, s_1(u, v) \wedge u \neq v) \vee s_1(u, u) \equiv q_S$.*

A natural question is whether this extension increases the complexities of the problems we are interested in. First note that a homomorphism from a CQ$^{\neq,\mathbf{C}}$ q to an instance I necessarily maps terms from a $\neq$-atom to terms known to be constants. It follows that, for q Boolean, if I entails q then q maps to I (and reciprocally); hence, query answering can still rely on homomorphism. This is different for query containment: given CQs$^{\neq,\mathbf{C}}$ q_1 and q_2, the existence of a query homomorphism from q_2 to q_1 is no longer a necessary condition for $q_1 \sqsubseteq q_2$, even when considering only databases. Let *UCQ$^{\neq,\boldsymbol{C}}$ containment* be the problem that takes as input two queries Q_1 and Q_2 in UCQ$^{\neq,\mathbf{C}}$, and asks if $Q_1 \sqsubseteq Q_2$. This problem is known to be Π^{P}_2-complete, already when both queries are in CQ$^{\neq,\mathbf{C}}$ (van der Meyden 1997; Kolaitis, Martin, and Thakur 1998). As a complementary result, we prove that Π^{P}_2-hardness already holds when Q_1 is a very simple kind of CQ.

Theorem 6 (Complexity of UCQ$^{\neq,\mathbf{C}}$ containment)**.** *The UCQ$^{\neq,\boldsymbol{C}}$ containment problem is Π^{P}_2-hard when Q_1 is a Boolean CQJFE and Q_2 is a Boolean CQ$^{\neq,\boldsymbol{C}}$.*

Proof sketch. By a reduction from $\forall\exists$3CNF adapted from (Abiteboul, Kanellakis, and Grahne 1991). □

We now study the complexity of $\mathcal{M}$-perfectness verification, by decomposing that problem into $\mathcal{M}$-completeness and $\mathcal{M}$-soundness verifications. The $\mathcal{M}$-completeness (resp. $\mathcal{M}$-soundness) verification problem can be recast as verifying if $Q_S \sqsubseteq \mathcal{M}^-(Q_O)$ (resp. $\mathcal{M}^-(Q_O) \sqsubseteq Q_S$). There is an immediate reduction from UCQ$^{\neq,\mathbf{C}}$ containment to verification, taking a trivial mapping $\mathcal{M}$ that bijectively translates n-ary predicates in S into n-ary predicates in O.

Theorem 7 (Complexity of $\mathcal{M}$-completeness)**.** *The UCQ$^{\neq,\boldsymbol{C}}$ $\mathcal{M}$-completeness verification problem is Π^{P}_2-complete, even if Q_S is a CQJFE and Q_O is a CQ$^{\neq}$.*

Proof. To verify that $Q_S \sqsubseteq \mathcal{M}^-(Q_O)$, we can check if, for every ground instantiation D of a CQ$^{\neq}$ from Q_S, there is a CQ$^{\neq}$ $q_i \in \mathcal{M}^-(Q_O)$ that maps to D (with answer variables mapped correctly). We can universally choose a D in polynomial time as it is given by a substitution of the variables of a $q_s \in Q_S$ by fresh constants and we can guess q_i and a homomorphism from q_i to D in polynomial time. Indeed, to obtain a q_i, we guess a CQ$^{\neq}$ $q_j \in Q_O$, a subset of $\mathcal{M}$ with at most $|q_j|$ rules and associated piece-unifiers. Hence, $\mathcal{M}$-completeness is in Π^{P}_2. Hardness follows from Th. 6. □

Theorem 8 (Complexity of $\mathcal{M}$-soundness)**.** *The UCQ$^{\neq,\boldsymbol{C}}$ $\mathcal{M}$-soundness verification problem is Π^{P}_2-complete, even if Q_S is a Boolean CQ$^{\neq}$ and Q_O is a is a Boolean CQJFE.*

Proof. Similar to that of Th. 7. □

Since $Q_1 \sqsubseteq Q_2$ iff $Q_1 \equiv Q_1 \wedge Q_2$, query equivalence is as hard as query containment, hence:

Corollary 9. *The UCQ$^{\neq,\boldsymbol{C}}$ $\mathcal{M}$-perfectness verification problem is Π^{P}_2-complete.*

It is known that $\mathcal{M}$-perfectness verification is Π^{P}_2-hard already for Q_S and Q_O CQs and $\mathcal{M}$ a GAV mapping in a DL setting, i.e., with mapping heads restricted to unary and binary predicates (Lutz, Marti, and Sabellek 2018; Cima, Lenzerini, and Poggi 2019). Hence, considering (U)CQ$^{\neq,\mathbf{C}}$ (and GLAV mappings) does not lead to increased complexity of verification.

When it comes to taking an ontology into account, most previous works have considered lightweight DLs that are

fus.[5] The key point is that, for any *fus* rule set $\mathcal{R}$ and Q_O in UCQ$^{\neq,\mathbf{C}}$, $\mathcal{R}^-(Q_O)$ is also in UCQ$^{\neq,\mathbf{C}}$; hence the techniques designed for $\mathcal{M}$-abstractions can be extended, however at the risk of increased complexity. Next, we show that perfectness verification remains in Π_2^{P} when $\mathcal{R}$ is a set of *linear* rules—i.e., existential rules whose body has a single atom—over predicates with bounded arity. This rule class generalizes several dialects of the DL-Lite family, in particular DL-Lite$_\mathcal{R}$ (Calí, Gottlob, and Lukasiewicz 2009).

Theorem 10. *The UCQ$^{\neq,\mathbf{C}}$ Σ-perfectness verification problem is in Π_2^{P} when $\mathcal{R}$ is linear with bounded-arity predicates.*

Proof sketch. W.l.o.g. assume Q_O is a CQ$^{\neq,\mathbf{C}}$. We show we can guess a CQ$^{\neq,\mathbf{C}}$ q' from $\mathcal{R}^-(Q_O)$ in polynomial time. Since $\mathcal{R}$ is linear, any such q' has at most $|Q_O|$ atoms, which only share terms from Q_O. Hence, the length of a rewriting sequence to q' can be bounded by $|Q_O| \times A$, where A is an upper-bound on the number of "non-isomorphic" atoms—with isomorphism being the identity on $\mathtt{terms}(Q_O)$— i.e., $A = |P| \times (|\mathtt{terms}(Q_O)| + a)^a$, where P is the set of predicates and a is the maximal arity of a predicate in P. □

This result subsumes previous results establishing Π_2^{P}-membership of Σ-perfectness verification with UCQs and DL-Lite$_\mathcal{R}$ (Lutz, Marti, and Sabellek 2018; Cima, Lenzerini, and Poggi 2019). [6] The theorem actually applies to any FO-rewritable pair $(Q_O, \mathcal{R})$ such that all the CQs in $\mathcal{R}^-(Q_O)$ can be generated in a polynomial number of rewriting steps.

4 Computing Minimally Complete and Perfect Abstractions

Let us first point out that a complete abstraction of a non-Boolean query Q_S may not exist, simply because $\mathcal{M}$ may not transfer all the constants that occur in the answers to Q_S. This is independent from any target query language. E.g., let $\mathcal{M} = \{s(x, y) \to r(y)\}$ and the CQ $q_S(u) = \exists v.s(u, v)$: q_S has no complete Σ-abstraction, for any Σ with $\mathcal{M}$. Let us characterize when a UCQ$^{\neq,\mathbf{C}}$ has a complete Σ-abstraction:

Proposition 11 (Existence of a complete abstraction)**.** *A CQ$^{\neq,\mathbf{C}}$ $q_S(\vec{x})$ has a complete Σ-abstraction iff for all $x \in \vec{x}$ there are $m \in \mathcal{M}$ and a homomorphism $h : \mathtt{body}(m) \to q_S(\vec{x})$ s.t. $x \in h(\mathtt{fr}(m))$. A UCQ$^{\neq,\mathbf{C}}$ $Q_S(\vec{x})$ has a complete Σ-abstraction iff each $q_i(\vec{x}) \in Q_S$ has one.*

Hence, deciding if a non-Boolean UCQ$^{\neq,\mathbf{C}}$ has a complete Σ-abstraction is NP-complete, while it is trivial for a Boolean UCQ$^{\neq,\mathbf{C}}$.

UCQ$^{\neq,\mathbf{C}}$ captures perfect Σ-abstractions of UCQ$^{\neq(,\mathbf{C})}$ source queries. As already mentioned, chasing a (relevant) UCQ with $\mathcal{M}$ yields a UCQ that is minimally complete within this class. However, as illustrated by Ex. 5, the class UCQ$^{\neq,\mathbf{C}}$ may provide a more faithful translation: the UCQ $\mathcal{M}(Q_S)$ is minimally complete within UCQs but not sound, while the UCQ$^{\neq,\mathbf{C}}$ Q_O is a perfect abstraction. We now state the main result of this section: the class UCQ$^{\neq,\mathbf{C}}$ captures minimally complete abstractions of source UCQs$^{\neq,\mathbf{C}}$, where minimality is w.r.t. *any* ontological query class (still with certain answer semantics)[7].

Theorem 12 (Minimal completeness)**.** *For any mapping $\mathcal{M}$ and any UCQ$^{\neq,\mathbf{C}}$ Q_S that has a complete $\mathcal{M}$-abstraction, there is a UCQ$^{\neq,\mathbf{C}}$ Q_O such that, for any Σ with mapping $\mathcal{M}$, Q_O is a minimally complete Σ-abstraction of Q_S.*

Note that Q_O is also a minimally complete $\mathcal{M}$-abstraction (we take Σ with $\mathcal{R} = \emptyset$). However, a minimally complete $\mathcal{M}$-abstraction is generally not a minimally complete Σ-abstraction, and vice-versa; to obtain Th. 12, we will rely on the specific abstraction computed by the $\mathcal{M}$-chase. Moreover, if there is a perfect Σ-abstraction of Q_S, any minimally complete Σ-abstraction of Q_S is perfect, hence UCQ$^{\neq,\mathbf{C}}$ also captures perfect abstractions:

Corollary 13 (Perfectness)**.** *For any Σ and any UCQ$^{\neq,\mathbf{C}}$ Q_S, if there is a perfect Σ-abstraction of Q_S, then it can be expressed as a UCQ$^{\neq,\mathbf{C}}$.*

Furthermore, it is easy to find examples in which **C** or the limited $\neq$ is required to express a perfect abstraction of a UCQ, hence one can argue that UCQ$^{\neq,\mathbf{C}}$ is a *minimal* class to express perfect abstractions of UCQs and UCQ$^{\neq,\mathbf{C}}$.

To prove Th. 12, we first state a fundamental semantic property of OBDA systems.

Proposition 14. *For any databases D and D' on S, if $\Sigma(D) \models \mathcal{M}(D')$, then:* [8]

1. $\mathtt{Mod}_\Sigma(D) \subseteq \mathtt{Mod}_\mathcal{M}(D')$.
2. *Hence: for any Q_O on O, $Q_O^{\mathtt{cert}}(D', \mathcal{M}) \subseteq Q_O^{\mathtt{cert}}(D, \Sigma)$.*

We now explain how to build the desired abstraction. Only $\mathcal{M}$ is required (not $\mathcal{R}$), the resulting query being minimally complete for any Σ with mapping $\mathcal{M}$. In a nutshell, before chasing Q_S, we first split each $q_i \in Q_S$ into an equivalent UCQ$^{\neq,\mathbf{C}}$, whose CQs encode all the ways of mapping q_i to a database: terms substituted identically are merged, remaining terms are declared distinct ($\neq$) and marked by **C**. We will show that chasing the output with $\mathcal{M}$ yields the desired minimally complete Σ-abstraction.

Note that a similar split operation is presented in (Cima et al. 2022) to compute minimally complete abstractions expressed in a more complex target language. Such operation is also commonly used to build inverse mappings, see e.g. Ex. 19. For the sake of self-containedness, and to include the processing of constants, we detail our split operation next. Given a CQ$^{\neq,\mathbf{C}}$ q, a partition of $\mathtt{terms}(q)$ is said *admissible* if none of its classes contains two constants nor both terms of a $\neq$-atom from q. Informally, each class of the partition gathers the terms of q mapped to the same database constant. To each admissible partition P_σ can be

[5] An exception is (Lutz, Marti, and Sabellek 2018) considering also non-fus DLs from the $\mathcal{EL}$ family.

[6] We can ignore the disjointness axioms from DL-Lite$_\mathcal{R}$, as they have no impact on the complexity results.

[7] See the discussion at the end of this section.

[8] As regards the formulation of the proposition, note that $\Sigma(D) \models \mathcal{M}(D')$ is stronger than $\Sigma(D) \models \Sigma(D')$: indeed, $\Sigma(D) \models \mathcal{M}(D')$ implies $\mathcal{R}(\Sigma(D)) \models \mathcal{R}(\mathcal{M}(D'))$, with $\mathcal{R}(\Sigma(D)) \equiv \Sigma(D)$ and $\mathcal{R}(\mathcal{M}(D')) \equiv \Sigma(D')$

assigned a substitution σ, which is obtained by (1) selecting one term t_i in each class $C_i \in P_\sigma$, with priority given to constants, then to answer variables if any, and (2) setting $\sigma(t_j) = t_i$ for each $t_j \in C_i$ s.t. $t_j \neq t_i$. E.g., to $P_\sigma = \{\{x, u\}, \{v, w, a\}\}$, with x an answer variable and a a constant, is assigned $\sigma = \{u \mapsto x, v \mapsto a, w \mapsto a\}$. Given a UCQ$^{\neq,\mathbf{C}}$ $Q_S(\vec{x})$, *split*(Q_S) is a UCQ$^{\neq,\mathbf{C}}$ built as follows:

Let *split*$(Q_S) = \emptyset$
For each $q_i \in Q_S$
  For each admissible P_σ on $\mathtt{terms}(q_i) \cup \mathtt{consts}(\mathcal{M})$
    Let $q' = \sigma(q_i)$ // q' *is consistent*
    For any $v \in \mathtt{vars}(q')$
      If $v \notin \vec{x}$ then add $\mathbf{C}(v)$ to q'
      For any $t \in \mathtt{terms}(q') \cup \mathtt{consts}(\mathcal{M})$ with $v \neq t$
        Add $v \neq t$ to q'
    Add q' to *split*(Q_S)

Note that q' may include atoms of the form $v \neq c$, where c is a constant from $\mathcal{M}$ that does not occur in *std*(q'): this is necessary to ensure the desired behavior of *split*(Q_S) (see Lemma 16). It is easy to check that for any database D, $Q_S(D) =$ *split*$(Q_S)(D)$.

Example 15. *(Minimally complete abstraction) Consider again Example 5 with* $q_s(u) = \exists v.s_1(u, v)$.

$$\textit{split}(q_s(u)) = (\exists v.s_1(u, v) \wedge \boldsymbol{C}(v) \wedge u \neq v) \vee s_1(u, u)$$

$$\mathcal{M}(\textit{split}(q_s(u))) = Q_O(u)$$

The following lemma states the crucial property of *split*(Q_S). We call *grounding* of a CQ$^{\neq,\mathbf{C}}$ q_S a substitution σ of each variable in q_S by a constant s.t. $\sigma(q_S)$ is consistent. The key point is that any grounding of a $q^i_s \in$ *split*(Q_S) is *injective*. It follows that any rule body maps "in the same way" to q^i_s and to any of its groundings:

Lemma 16. *Let* σ^i_s *be a grounding of* $q^i_s \in$ *split*(Q_S). *Then:* $\mathcal{M}(\sigma^i_s(q^i_s)) \equiv \sigma^i_s(\mathcal{M}(q^i_s))$.

Proof of Th. 12 (Sketch). Let $Q_O(\vec{x}) = \mathcal{M}(\textit{split}(Q_S))$. We show that Q_O is a minimally complete Σ-abstraction of Q_S, for any Σ with mapping $\mathcal{M}$. Completeness follows from the properties of the $\mathcal{M}$-chase. To prove that Q_O is minimally complete, we consider any D and $\vec{c} \in Q_O(\Sigma(D))$ and show that $\vec{c}$ is a certain answer to any complete Σ-abstraction of Q_S. Let $q^i_O \in Q_O$ that maps by h_i to $\Sigma(D)$ with $h_i(\vec{x}) = \vec{c}$. Let $q^i_S \in$ *split*(Q_S) such that $q^i_O = \mathcal{M}(q^i_S)$. Let D_i be obtained by a grounding of $h_i(q^i_S)$. Since $h_i(\vec{x}) = \vec{c}$, $\vec{c} \in q^i_S(D_i)$. Since q^i_O is a complete Σ-abstraction of q^i_S, $\vec{c} \in q^i_O(\Sigma(D_i))$. We have $\mathcal{M}(h_i(q^i_S)) \equiv \mathcal{M}(D_i)$, hence, from Lemma 16, $\mathcal{M}(D_i)$ maps to $h_i(q^i_O)$. Since $h_i(q^i_O) \subseteq \Sigma(D)$, $\mathcal{M}(D_i)$ maps to $\Sigma(D)$. So, by Prop. 14, for all ontological query Q, $Q^{\mathtt{cert}}(D_i, \Sigma) \subseteq Q^{\mathtt{cert}}(D, \Sigma)$, hence if Q is Σ-complete then $\vec{c} \in Q^{\mathtt{cert}}(D, \Sigma)$. □

Complexity of expressibility With these results in hand, we can now study the complexity of determining whether a UCQ$^{\neq,\mathbf{C}}$ Q_S has a perfect $\mathcal{M}$-abstraction. Let us say that $\mathcal{M}$ is *frontier-bounded* if the frontier of all its rules is bounded by a constant. From (Lutz, Marti, and Sabellek 2018) we know that UCQ $\mathcal{M}$-expressibility is Π^{P}_2-complete in a GAV setting with bounded predicates (in rule heads). We observe that Π^{P}_2-membership can be extended to GLAV mappings with unbounded predicate arity provided that $\mathcal{M}$ is frontier-bounded. Indeed, for a CQ $q_{s_i} \in Q_S$, $\mathcal{M}(q_{s_i})$ is built from rule heads whose frontier is substituted by $\mathtt{terms}(q_{s_i})$; hence, for each $m \in \mathcal{M}$, the number of substitutions that need to be considered is bounded by $\mathtt{terms}(q_{s_i})^{|\mathtt{fr}(m)|}$. Therefore, when $\mathcal{M}$ is frontier-bounded, we can build each $\mathcal{M}(q_{s_i})$ by making a polynomial number of calls to an NP oracle, asking for each $m \in \mathcal{M}$ with $\mathtt{fr}(m) = (x_1, \ldots, x_k)$ (according to an arbitrary total ordering of the frontier variables) and tuple $t = (y_1, \ldots, y_k) \in \mathtt{terms}(q_{s_i})^k$ if there is a homomorphism $h : \mathtt{body}(m) \to q_{s_i}$ such that $h(\mathtt{fr}(m)) = t$. When $\mathcal{M}$ is not frontier-bounded, $\mathcal{M}(Q_S)$ can be computed in ExpTime, which yields a Co-NExpTime upper bound. These arguments can be generalized to (unrestricted) $\mathcal{M}$-expressibility of UCQ$^{\neq,\mathbf{C}}$ queries, as shown next.

Theorem 17. *(Complexity of* $\mathcal{M}$*-expressibility) UCQ*$^{\neq,\mathbf{C}}$ $\mathcal{M}$*-expressiblity is* Π^{P}_2*-complete when* $\mathcal{M}$ *is frontier-bounded, otherwise it is in Co-NExpTime.*

Proof sketch. To check that Q_s is *not* $\mathcal{M}$-expressible, one can guess a CQ$^{\neq}$ from Q_s and a partition on its terms, which yields a CQ$^{\neq}$ q_{s_i} from split(Q_s), compute $q_{o_i} = \mathcal{M}(q_{s_i})$ and guess a rewriting q'_{s_i} of q_{o_i} (of polynomial size in q_{o_i}) such that $q'_{s_i} \not\sqsubseteq Q_s$ (test in Σ^P_2). If $\mathcal{M}$ is frontier-bounded, $q_{o_i} = \mathcal{M}(q_{s_i})$ can be built by making a polynomial number of calls to an NP oracle, otherwise, it can be computed in ExpTime. Hence, the (co-)problem is in Σ^P_2 if $\mathcal{M}$ is frontier-bounded, otherwise in NExpTime. Π^{P}_2-hardness follows from (Lutz, Marti, and Sabellek 2018). □

Discussion on related frameworks We will now discuss our framework further in relationship with previous work.

As shown above, one can decide if a UCQ$^{\neq,\mathbf{C}}$ Q_S has a perfect $\mathcal{M}$-abstraction by simply checking if $Q_O = \mathcal{M}(\textit{split}(Q_S))$ is a sound $\mathcal{M}$-abstraction, i.e., $\mathcal{M}^-(Q_O) \sqsubseteq Q_S$. This may seem contradictory with other results from the literature. In particular, it is shown in (Cima et al. 2021) that determining if a CQ has a perfect $\mathcal{M}$-abstraction is undecidable. In fact, the crucial point is the semantics of ontological queries. We consider the widely adopted semantics of certain answers. As a consequence, ontological queries are necessarily monotone, in the following sense: Q_O is *monotone* if for all D_1, D_2 on S, if $\mathtt{Mod}_\Sigma(D_2) \subseteq \mathtt{Mod}_\Sigma(D_1)$ then any answer to Q_O on (D_1, Σ) is an answer to Q_O on (D_2, Σ). This is a corollary of our Prop. 14. A more general notion of ontological query is investigated in the above-mentionned paper, which allows for non-monotone queries. Note that a source query may have no perfect monotone abstraction but a perfect abstraction in this more general setting, which is studied in (Cima, Lenzerini, and Poggi 2020).

The $\mathbf{C}$ predicate was introduced in (Fagin et al. 2008), under the name *is-constant*, to define specific kinds of inverses of GLAV mappings, which are disjunctive$^{\neq,\mathbf{C}}$ mappings (Definition 18). It has been commonly used since then in the data exchange litterature, not only in inverse mappings but also in queries, see e.g., (Arenas, Pérez, and Riveros

2008). Similar notions have also been studied in KR, as closed-world variables (Amendola et al. 2018), or nominal variables in description logics (Krötzsch and Rudolph 2014). We think that **C** yields a very simple and effective way of dealing with unknown values introduced by mappings (and ontologies). First, it is easy to understand for a user and its introduction has no impact on computational complexity. Second, it can be handled using off-the-shelf tools. Indeed, one can slighly modify the mapping by adding a head atom $\mathbf{C}(t)$ for each frontier variable or constant t: then, **C**-atoms in ontological queries can be processed just like the standard atoms. A more general way of introducing some closed world reasoning would have been to extend queries with the epistemic operator **K** ("is known"), as in (Cima, Lenzerini, and Poggi 2020; Cima et al. 2022). However, this operator does not have the simplicity of **C**, which seems a better choice to us in the context of abstractions under standard certain answer semantics. On the other hand, **K** comes into its own in the context of *non-monotone* ontological queries, which inherently requires a more general semantics than certain answers.

5 Computing Maximally Sound Abstractions

A (U)CQ$^{(\neq)}$ always has a sound abstraction (the empty UCQ, which has no answer) but may not have a *maximally sound* abstraction expressible as a UCQ$^{(\neq,\mathbf{C})}$. Finding a suitable language for such abstractions remains open. In this section, we make progress by providing a characterization of maximally sound $\mathcal{M}$-abstractions of UCQ$^{(\neq)}$s, which we further extend to Σ-abstractions with *fus* rules. For that, we rely on specific inverse mappings from O to S, which, as explained later, correspond to so-called *maximum recoveries*. Such mappings have disjunctive heads, as defined next.

Definition 18 (Disjunctive$^{\neq,\mathbf{C}}$ mapping)**.** *A* disjunctive *(resp.* disjunctive$^{\neq,\mathbf{C}}$*)* mapping *from a source schema $\mathcal{S}$ to a target schema $\mathcal{T}$ is a set of $\mathcal{S}$-to-$\mathcal{T}$ disjunctive existential rules of the form* $\forall\vec{x}.(\exists\vec{y}.B[\vec{x},\vec{y}]) \rightarrow \bigvee_{i=1}^{n} \exists\vec{z_i}.H_i[\vec{x},\vec{z_i}]$*, where B is a CQ (resp. a CQ$^{\neq,\mathbf{C}}$) and each H_i is a CQ, all with answer variables $\vec{x}$.*

Given a (GLAV) mapping $\mathcal{M}$ from S to O, we will consider a disjunctive$^{\neq,\mathbf{C}}$ mapping $\mathcal{M}_\vee$ from O to S, which has the property of being a maximum recovery of $\mathcal{M}$ (Arenas, Pérez, and Riveros 2009; Arenas et al. 2009). Before entering into the formal framework of maximum recoveries, we first explain how $\mathcal{M}_\vee$ is built. Briefly, each rule of $\mathcal{M}_\vee$ is obtained by rewriting a rule head from $\mathcal{M}$ against $\mathcal{M}$. Precisely, for each $m \in \mathcal{M}$ with head $\exists\vec{y}.H[\vec{x},\vec{y}]$ (seen as a CQ), $\mathcal{M}_\vee$ has the following disjunctive rule:

$$\forall\vec{x}.(\exists\vec{y}.H[\vec{x},\vec{y}] \wedge \mathbf{C}[\vec{x}]) \rightarrow \mathcal{M}^{-}(\exists\vec{y}.H[\vec{x},\vec{y}])$$

This is illustrated in Ex. 19. As shown in the example, the head of the obtained rule may contain equalities; these equalities can be turned into inequalities in the rule body, by a split operation similar in spirit to that described in Sect. 4, which yields a rule complying with Def. 18; see (Arenas et al. 2009) for details.

Example 19. *Let* $\mathcal{M} =$
$\begin{cases} m_1 = s_1(x) \rightarrow \exists y.p(x,y) & m_3 = s_3(x,y) \rightarrow r(x,y) \\ m_2 = s_2(x,y) \rightarrow p(x,y) & m_4 = s_4(x) \rightarrow r(x,x) \end{cases}$
By rewriting the CQs $\mathtt{head}(m_i)(\mathtt{fr}(m_i))$*, one gets* $\mathcal{M}'=$
$\begin{cases} m'_1 = p(x,y) \wedge \mathbf{C}(x) & \rightarrow s_1(x) \vee \exists z.s_2(x,z) \\ m'_2 = p(x,y) \wedge \mathbf{C}(x) \wedge \mathbf{C}(y) & \rightarrow s_2(x,y) \\ m'_3 = r(x,y) \wedge \mathbf{C}(x) \wedge \mathbf{C}(y) & \rightarrow s_3(x,y) \vee (s_4(x) \wedge x = y) \\ m'_4 = r(x,x) \wedge \mathbf{C}(x) & \rightarrow s_3(x,x) \vee s_4(x) \end{cases}$
Moreover, Rule m'_3 with equality can be replaced by two rules obtained by considering that, in $\mathtt{body}(m'_3)$*, either $x = y$ (which yields m'_4, already present) or $x \neq y$, which yields: $m''_3 = r(x,y) \wedge \mathbf{C}(x) \wedge \mathbf{C}(y) \wedge x \neq y \rightarrow s_3(x,y)$*
Finally, $\mathcal{M}_\vee = \{m'_1, m'_2, m''_3, m'_4\}$.

We furthermore consider the rewriting operator for UCQs against *disjunctive* mappings introduced in (Leclère, Mugnier, and Pérution-Kihli 2023). This operator is sound and complete and yields a possibly infinite disjunction of CQs. Its extension to UCQ$^{\neq,\mathbf{C}}$s and disjunctive$^{\neq,\mathbf{C}}$ mappings is straightforward. We can now outline our characterization of maximal sound $\mathcal{M}$-abstractions: For any mapping $\mathcal{M}$ (from S to O) and UCQ$^{\neq,\mathbf{C}}$ Q_S on S, let $\mathcal{M}_\vee$ be the disjunctive$^{\neq,\mathbf{C}}$ mapping from O to S built as above; then, $\mathcal{M}_\vee^{-}(Q_S)$, i.e., the rewriting of Q_S against $\mathcal{M}_\vee$, is a maximally sound abstraction of Q_S. This result relies on the fact that $\mathcal{M}_\vee$ is a maximum recovery of $\mathcal{M}$ and is proven in Th. 21.

Maximum recoveries The following definitions and results come from (Arenas, Pérez, and Riveros 2009; Arenas et al. 2009). The notion of a maximum recovery is defined on *abstract mappings*, which may then be *specified* by *concrete* mappings, i.e., provided with a specific syntax (e.g., GLAV). An *abstract mapping* $\mathcal{M}_\mathcal{A}$ from a schema $\mathcal{S}$ to a schema $\mathcal{T}$ is any relation from the $\mathcal{S}$-instances to the $\mathcal{T}$-instances.[9] Let $Q_\mathcal{T}$ be a query on $\mathcal{T}$. Given an $\mathcal{S}$-instance I, we denote by $\mathtt{certain}_{\mathcal{M}_\mathcal{A}}(Q_\mathcal{T}, I) = \bigcap_{(I,J)\in\mathcal{M}_\mathcal{A}} Q_\mathcal{T}(J)$ the *certain answers* to $Q_\mathcal{T}$ through I and $\mathcal{M}_\mathcal{A}$. A query $Q_\mathcal{S}$ on $\mathcal{S}$ such that $Q_\mathcal{S}(I) = \mathtt{certain}_{\mathcal{M}_\mathcal{A}}(Q_\mathcal{T}, I)$ for all instance I on $\mathcal{S}$, is called a *perfect rewriting* of $Q_\mathcal{T}$ through $\mathcal{M}_\mathcal{A}$ (such $Q_\mathcal{S}$ may not exist). The composition of two abstract mappings $\mathcal{M}_\mathcal{A}$ and $\mathcal{M}'_\mathcal{A}$ is denoted by $\mathcal{M}_\mathcal{A} \bullet \mathcal{M}'_\mathcal{A}$.[10] Given an abstract mapping $\mathcal{M}_\mathcal{A}$ from $\mathcal{S}$ to $\mathcal{T}$, the abstract mapping $\mathcal{M}'_\mathcal{A}$ from $\mathcal{T}$ to $\mathcal{S}$ is a *recovery* of $\mathcal{M}_\mathcal{A}$ if for any query $Q_\mathcal{S}$ on $\mathcal{S}$ and instance I on $\mathcal{S}$, $\mathtt{certain}_{\mathcal{M}_\mathcal{A}\bullet\mathcal{M}'_\mathcal{A}}(Q_\mathcal{S}, I) \subseteq Q_\mathcal{S}(I)$; and $\mathcal{M}'_\mathcal{A}$ is a *maximum* recovery if, moreover, for any recovery $\mathcal{M}''_\mathcal{A}$ of $\mathcal{M}_\mathcal{A}$, $\mathtt{certain}_{\mathcal{M}_\mathcal{A}\bullet\mathcal{M}''_\mathcal{A}}(Q_\mathcal{S}, I) \subseteq \mathtt{certain}_{\mathcal{M}_\mathcal{A}\bullet\mathcal{M}'_\mathcal{A}}(Q_\mathcal{S}, I)$.

An abstract mapping $\mathcal{M}_\mathcal{A}$ from $\mathcal{S}$ to $\mathcal{T}$ is *specified* by a (GLAV) mapping $\mathcal{M}$ from $\mathcal{S}$ to $\mathcal{T}$ if: for every pair of $(\mathcal{S},\mathcal{T})$-instances (I,J), $(I,J) \in \mathcal{M}_\mathcal{A}$ iff $I \cup J \models \mathcal{M}$. In this case, $\mathtt{certain}_{\mathcal{M}_\mathcal{A}}(Q_\mathcal{T}, I) = Q_\mathcal{T}^{\mathtt{cert}}(I, \mathcal{M})$ holds for any query $Q_\mathcal{T}$ and, when $Q_\mathcal{T}$ is a UCQ$^{\neq,C}$, $\mathcal{M}^{-}(Q_\mathcal{T})$ is a perfect rewriting of $Q_\mathcal{T}$ through $\mathcal{M}_\mathcal{A}$. Not all abstract map-

[9] In the cited work, instances are finite, but the definitions work in the infinite case.

[10] We use $\bullet$ to avoid confusion with the classical $\circ$: $\mathcal{M}_\mathcal{A} \bullet \mathcal{M}'_\mathcal{A}$ can be read $\mathcal{M}'_\mathcal{A} \circ \mathcal{M}_\mathcal{A}$.

pings have a maximum recovery. However, when the source instances are ground, a GLAV mapping always has one, taking the form of a disjunctive$^{\neq,\mathbf{C}}$ mapping (Def. 18).[11]

Maximally sound $\mathcal{M}$-abstractions Let $\mathcal{M}_\mathcal{A}$ be an abstract mapping specified by a GLAV mapping $\mathcal{M}$. The following lemma shows that a perfect rewriting of a source query $Q_\mathcal{S}$ through a maximum recovery of $\mathcal{M}_\mathcal{A}$ behaves similarly to a maximally sound $\mathcal{S}$-to-$\mathcal{T}$ translation of $Q_\mathcal{S}$ through $\mathcal{M}$ (i.e., an $\mathcal{M}$-abstraction of $Q_\mathcal{S}$ when $\mathcal{S} = S$ and $\mathcal{T} = O$). Indeed, Point (1) corresponds to the soundness of an $\mathcal{M}$-abstraction and Point (2) to its maximality.

Lemma 20. *Let $Q_\mathcal{S}$ be a query on $\mathcal{S}$, $\mathcal{M}_\mathcal{A}$ be an abstract mapping from $\mathcal{S}$ to $\mathcal{T}$ that has a maximum recovery $\mathcal{M}'_\mathcal{A}$. Let $Q_\mathcal{T}$ be a perfect rewriting of $Q_\mathcal{S}$ through $\mathcal{M}'_\mathcal{A}$. Then, for any $\mathcal{S}$-instance I: (1) $\texttt{certain}_{\mathcal{M}_\mathcal{A}}(Q_\mathcal{T}, I) \subseteq Q_\mathcal{S}(I)$, and (2) $\texttt{certain}_{\mathcal{M}_\mathcal{A}}(Q'_\mathcal{T}, I) \subseteq \texttt{certain}_{\mathcal{M}_\mathcal{A}}(Q_\mathcal{T}, I)$ for any query $Q'_\mathcal{T}$ such that $\texttt{certain}_{\mathcal{M}_\mathcal{A}}(Q'_\mathcal{T}, I) \subseteq Q_\mathcal{S}(I)$.*

Finally, Th. 21 directly relies on Lemma 20:

Theorem 21. *Let $\mathcal{M}$ be a (GLAV) mapping from S to O, $\mathcal{M}_\vee$ be a disjunctive$^{\neq,\mathbf{C}}$ mapping that is a (concrete) maximum recovery of $\mathcal{M}$, and Q_S be a UCQ$^{\neq,\mathbf{C}}$ on S. Then ${\mathcal{M}_\vee}^-(Q_S)$ is a maximally sound $\mathcal{M}$-abstraction of Q_S.*

In general, ${\mathcal{M}_\vee}^-(Q_S)$ is a possibly infinite disjunction of UCQ$^{\neq,\mathbf{C}}$. Yet, the next proposition gives cases where it is a UCQ$^{\neq,\mathbf{C}}$. In such cases, the rewriting algorithm from (Leclère, Mugnier, and Pérution-Kihli 2023) can be used to effectively output a maximally sound $\mathcal{M}$-abstraction.

Proposition 22. *The maximally sound $\mathcal{M}$-abstraction of a UCQ$^{\neq,\mathbf{C}}$ Q is a UCQ$^{\neq,\mathbf{C}}$ when:*

1. *$\mathcal{M}^-(\texttt{head}(m))$ is a CQ$^{\neq,\mathbf{C}}$ for all $m \in \mathcal{M}$;*
2. *Q contains only full CQs$^{\neq,\mathbf{C}}$ (i.e., without existential variables) and $\mathcal{M}$ is GAV;*
3. *Q contains only atomic CQs$^{\neq,\mathbf{C}}$ (i.e., with at most one standard atom).*

Proof. Let $\mathcal{M}_\vee$ be a maximum recovery of $\mathcal{M}$. (1) $\mathcal{M}_\vee$ is a conjunctive mapping. (2) All rules in $\mathcal{M}_\vee$ are lossless (all body variables are frontier) which guarantees to get a UCQ$^{\neq,\mathbf{C}}$-rewriting from any full CQ$^{\neq,\mathbf{C}}$. (3) Disjunctive source-to-target rule sets guarantee to get a UCQ$^{\neq,\mathbf{C}}$-rewriting from any atomic CQ$^{\neq,\mathbf{C}}$. Points (2) and (3) follow from the rewriting algorithm in (Leclère, Mugnier, and Pérution-Kihli 2023). □

Note. We remark that Th. 21 contradicts a result from (Arenas et al. 2009) (see Th. 4.4). This result states that a recovery that maximally recovers answers to CQs (not UCQs), called a CQ-maximum recovery, can be specified by a conjunctive mapping (i.e., without disjunctive heads). But then, the rewriting of a CQ through a CQ-maximum recovery would always be finite (this is a property of conjunctive mappings), hence a CQ would always have a maximally sound $\mathcal{M}$-abstraction as a UCQ$^{\neq,\mathbf{C}}$, which is false.

[11] More precisely: For every GLAV mapping $\mathcal{M}$, which specifies an abstract mapping $\mathcal{M}_\mathcal{A}$, there is a concrete mapping $\mathcal{M}_\vee$ that specifies a maximum recovery of $\mathcal{M}_\mathcal{A}$ and can be expressed as a disjunctive mapping$^{\neq,\mathbf{C}}$. For the sake of simplicity, we say that $\mathcal{M}_\vee$ is a (concrete) maximum recovery of $\mathcal{M}$.

Maximally sound Σ-abstractions We now extend previous results to an OBDA specification with a *fus* ontology $\mathcal{R}$. A suitable disjunctive$^{\neq,\mathbf{C}}$ mapping from O to S, say $\mathcal{M}_\vee^\Sigma$, is obtained by rewriting each rule head of $\mathcal{M} \cup \mathcal{R}$ against $\mathcal{M} \cup \mathcal{R}$. For a rule head $\exists\vec{y}.H[\vec{x}, \vec{y}]$, this yields the disjunctive rule $\forall\vec{x}.(\exists\vec{y}.H[\vec{x}, \vec{y}] \wedge \mathbf{C}[\vec{x}]) \rightarrow \Sigma^-(\exists\vec{y}.H[\vec{x}, \vec{y}])$. To bring OBDA specifications into the maximum recovery framework, we say that an abstract mapping $\mathcal{M}_\mathcal{A}$ from S to O is *specified* by $\Sigma = (S, O, \mathcal{M}, \mathcal{R})$ if, for all S-database D and O-instance J, $(D, J) \in \mathcal{M}_\mathcal{A}$ iff $D \cup J \models \mathcal{M}$ and $J \models \mathcal{R}$ both hold.

Theorem 23. *Let Σ be an OBDA specification with FO-rewritable $\mathcal{R}$. Then:*

1. *$\mathcal{M}_\vee^\Sigma$ is a (concrete) maximum recovery of Σ.*
2. *For any UCQ$^{\neq,\mathbf{C}}$ Q_S on S, ${\mathcal{M}_\vee^\Sigma}^-(Q_S)$ is a maximally sound Σ-abstraction of Q_S.*

Proof (sketch). (1) Since $\mathcal{M} \cup \mathcal{R}$ is fus, $\mathcal{M}_\vee^\Sigma$ is well defined. We first prove that $\mathcal{M}_\vee^\Sigma$ specifies a recovery $\Sigma'_\mathcal{A}$ of the abstract mapping $\Sigma_\mathcal{A}$ specified by Σ. To do that, we prove that for all S-databases D, there is an O-instance J s.t. $(D, J) \in \Sigma_\mathcal{A}$ and $(J, D) \in \Sigma'_\mathcal{A}$. Such J always exists, f.i. $J = \Sigma(D)$. Then, we prove that $\Sigma'_\mathcal{A}$ is a maximum recovery of $\Sigma_\mathcal{A}$, using Prop. 3.8 from (Arenas, Pérez, and Riveros 2009), which follows that $\Sigma'_\mathcal{A}$ is a maximum recovery of $\Sigma_\mathcal{A}$ iff $\Sigma'_\mathcal{A}$ is a recovery and for every $(D_1, D_2) \in \Sigma_\mathcal{A} \bullet \Sigma'_\mathcal{A}$, it is the case that $\emptyset \neq \texttt{Mod}_{\Sigma_\mathcal{A}}(D_2) \subseteq \texttt{Mod}_{\Sigma_\mathcal{A}}(D_1)$. $\Sigma'_\mathcal{A}$ has this property by construction of $\mathcal{M}_\vee^\Sigma$. (2) The proof is similar to the proof of Th. 21, using Point (1) and Lemma 20. □

Prop. 22 can be extended to Σ-abstractions as follows: (1) taking $\mathcal{M} \cup \mathcal{R}$ instead of $\mathcal{M}$; (2) and (3): taking rule classes ensuring that $\mathcal{R}^-(Q)$ has the desired property, in particular lossless rules for (2) and linear rules for (3).

6 Conclusion

We have investigated the properties of the query class UCQ$^{\neq,\mathbf{C}}$ for capturing abstractions in an OBDA setting under certain answer semantics. We found that this class enjoys nice computational behavior in this context. We proved that it is able to express any minimally complete—and therefore any perfect—abstraction of a source UCQ$^{\neq,\mathbf{C}}$, when such an abstraction exists. Although a maximally sound abstraction of a UCQ always exists, it may not be expressible in UCQ$^{\neq,\mathbf{C}}$. However, we identified an interesting connection with the notion of maximum recovery from data exchange, and showed that a maximally sound $\mathcal{M}$-abstraction of a source UCQ$^{\neq,\mathbf{C}}$ is precisely its rewriting with a maximum recovery of $\mathcal{M}$. While the ontology plays no role in minimal completeness, it does in maximal soundness. Accordingly, we extended the preceding result to OBDA specifications with *fus* ontologies.

Among the open questions, it remains unknown whether the problem of determining if a (U)CQ admits a maximally sound abstraction in UCQ$^{\neq,\mathbf{C}}$ is decidable. Moreover, no known algorithm is guaranteed to terminate whenever such a finite abstraction exists.

Acknowledgements

We thank the reviewers for their helpful comments.

References

Abiteboul, S.; Hull, R.; and Vianu, V. 1995. *Foundations of Databases*. Addison-Wesley.

Abiteboul, S.; Kanellakis, P. C.; and Grahne, G. 1991. On the representation and querying of sets of possible worlds. *Theor. Comput. Sci.* 78(1):158–187.

Amendola, G.; Leone, N.; Manna, M.; and Veltri, P. 2018. Enhancing existential rules by closed-world variables. In Lang, J., ed., *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI 2018, July 13-19, 2018, Stockholm, Sweden*, 1676–1682. ijcai.org.

Arenas, M.; Pérez, J.; Reutter, J. L.; and Riveros, C. 2009. Inverting schema mappings: Bridging the gap between theory and practice. *Proc. VLDB Endow.* 2:1018–1029.

Arenas, M.; Pérez, J.; and Riveros, C. 2008. The recovery of a schema mapping: bringing exchanged data back. In *ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems*.

Arenas, M.; Pérez, J.; and Riveros, C. 2009. The recovery of a schema mapping: Bringing exchanged data back. *ACM Trans. Database Syst.* 34:22:1–22:48.

Baget, J.-F.; Leclère, M.; Mugnier, M.-L.; and Salvat, E. 2009. Extending Decidable Cases for Rules with Existential Variables. In *Proceedings of the 21st International Joint Conference on Artificial Intelligence, IJCAI 2009*, 677–682.

Calì, A.; Gottlob, G.; and Kifer, M. 2008. Taming the infinite chase: Query answering under expressive relational constraints. In Brewka, G., and Lang, J., eds., *Principles of Knowledge Representation and Reasoning: Proceedings of the Eleventh International Conference, KR 2008, Sydney, Australia, September 16-19, 2008*, 70–80. AAAI Press.

Calí, A.; Gottlob, G.; and Lukasiewicz, T. 2009. A general datalog-based framework for tractable query answering over ontologies. In *ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems*.

Calì, A.; Gottlob, G.; and Pieris, A. 2010. Advanced processing for ontological queries. *Proc. VLDB Endow.* 3(1):554–565.

Calvanese, D.; Giacomo, G. D.; Lembo, D.; Lenzerini, M.; and Rosati, R. 2007. Tractable reasoning and efficient query answering in description logics: The *DL-Lite* family. *J. Autom. Reason.* 39(3):385–429.

Cima, G.; Console, M.; Lenzerini, M.; and Poggi, A. 2021. Abstraction in data integration. In *36th Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2021, Rome, Italy, June 29 - July 2, 2021*, 1–11. IEEE.

Cima, G.; Console, M.; Lenzerini, M.; and Poggi, A. 2022. Monotone Abstractions in Ontology-Based Data Management. In *AAAI Conference on Artificial Intelligence*.

Cima, G.; Lenzerini, M.; and Poggi, A. 2019. Semantic characterization of data services through ontologies. In *International Joint Conference on Artificial Intelligence*.

Cima, G.; Lenzerini, M.; and Poggi, A. 2020. Non-monotonic ontology-based abstractions of data services. In *International Conference on Principles of Knowledge Representation and Reasoning*.

Cima, G.; Poggi, A.; and Lenzerini, M. 2023. The notion of abstraction in ontology-based data management. *Artif. Intell.* 323:103976.

Fagin, R.; Kolaitis, P. G.; Popa, L.; and Tan, W.-C. 2008. Quasi-inverses of schema mappings. *ACM Transactions on Database Systems (TODS)* 33(2):1–52.

Kolaitis, P. G.; Martin, D. L.; and Thakur, M. N. 1998. On the complexity of the containment problem for conjunctive queries with built-in predicates. In Mendelzon, A. O., and Paredaens, J., eds., *Proceedings of the Seventeenth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems, June 1-3, 1998, Seattle, Washington, USA*, 197–204. ACM Press.

König, M.; Leclère, M.; Mugnier, M.; and Thomazo, M. 2015. Sound, complete and minimal ucq-rewriting for existential rules. *Semantic Web* 6(5):451–475.

Krötzsch, M., and Rudolph, S. 2014. Nominal schemas in description logics: Complexities clarified. In Baral, C.; Giacomo, G. D.; and Eiter, T., eds., *Principles of Knowledge Representation and Reasoning: Proceedings of the Fourteenth International Conference, KR 2014, Vienna, Austria, July 20-24, 2014*. AAAI Press.

Leclère, M.; Mugnier, M.; and Pérution-Kihli, G. 2023. Query rewriting with disjunctive existential rules and mappings. In *Proceedings of the 20th International Conference on Principles of Knowledge Representation and Reasoning, KR 2023, Rhodes, Greece, September 2-8, 2023*, 429–439.

Lutz, C.; Marti, J.; and Sabellek, L. 2018. Query expressibility and verification in ontology-based data access. In *International Conference on Principles of Knowledge Representation and Reasoning*. Erratum at https://www.informatik.uni-leipzig.de/kr/research/papers.html.

Nash, A.; Segoufin, L.; and Vianu, V. 2010. Views and queries: Determinacy and rewriting. *ACM Trans. Database Syst.* 35(3):21:1–21:41.

Poggi, A.; Lembo, D.; Calvanese, D.; Giacomo, G. D.; Lenzerini, M.; and Rosati, R. 2008. Linking data to ontologies. *J. Data Semant.* 10:133–173.

van der Meyden, R. 1997. The complexity of querying indefinite data about linearly ordered domains. *J. Comput. Syst. Sci.* 54(1):113–135.