The VLDB Journal (2012) 21:823–842
DOI 10.1007/s00778-012-0272-z

REGULAR PAPER

Query language-based inverses of schema mappings: semantics, computation, and closure properties

Marcelo Arenas · Jorge Pérez · Juan Reutter · Cristian Riveros

Received: 23 August 2011 / Revised: 13 January 2012 / Accepted: 5 March 2012 / Published online: 6 April 2012
© Springer-Verlag 2012

Abstract The inversion of schema mappings has been identified as one of the fundamental operators for the development of a general framework for metadata management. During the last few years, three alternative notions of inversion for schema mappings have been proposed (Fagin-inverse (Fagin, TODS 32(4), 25:1–25:53, 2007), quasi-inverse (Fagin et al., TODS 33(2), 11:1–11:52, 2008), and maximum recovery (Arenas et al., TODS 34(4), 22:1–22:48, 2009)). However, these notions lack some fundamental properties that limit their practical applicability: most of them are expressed in languages including features that are difficult to use in practice, some of these inverses are not guaranteed to exist for mappings specified with source-to-target tuple-generating dependencies (st-tgds), and it has been futile to search for a *meaningful* mapping language that is *closed* under any of these notions of inverse. In this paper, we develop a framework for the inversion of schema mappings that fulfills all of the above requirements. It is based on the notion of $\mathcal{C}$-maximum recovery, for a query language $\mathcal{C}$, a notion designed to generate inverse mappings that recover back only the information that can be retrieved with queries in $\mathcal{C}$. By focusing on the language of conjunctive queries (CQ), we are able to find a mapping language that contains the class of st-tgds, is closed under CQ-maximum recovery, and for which the chase procedure can be used to exchange data efficiently. Furthermore, we show that our choices of inverse notion and mapping language are optimal, in the sense that choosing a more expressive inverse operator or mapping language causes the loss of these properties.

Keywords Inverses · Schema mappings · Metadata management · Closure properties · Conjunctive queries · Data exchange · Data integration

A preliminary version of this article appeared in PVLDB [3].

Electronic supplementary material The online version of this article (doi:10.1007/s00778-012-0272-z) contains supplementary material, which is available to authorized users.

M. Arenas
Department of Computer Science, PUC Chile, Santiago, Chile
e-mail: marenas@ing.puc.cl

J. Pérez (✉)
Department of Computer Science, Universidad de Chile, Santiago, Chile
e-mail: jperez@dcc.uchile.cl

J. Reutter
School of Informatics, University of Edinburgh, Edinburgh, UK
e-mail: juan.reutter@ed.ac.uk

C. Riveros
Department of Computer Science, University of Oxford, Oxford, UK
e-mail: cristian.riveros@cs.ox.ac.uk

1 Introduction

A schema mapping is a specification that describes how data from a source schema are to be mapped to a target schema. Schema mappings have proved to be essential for several data interoperability tasks such as data exchange [12], data integration [23], and peer data management [18,20]. The research on this area has mainly focused on performing these tasks and has produced several applications that work with declarative specifications of schema mappings. However, as Bernstein pointed out in [8], many information system problems involve not only the design and integration of complex application artifacts, but also their subsequent manipulation. Driven by this consideration, Bernstein proposed in [8] a general framework for managing schema mappings. In this framework, schema mappings are usually specified in some logical language, and high-level algebraic

operators like *match*, *merge*, and *compose* are used to manipulate them [8,9,26,28].

One of the operators in Bernstein's framework is the *inverse* of a schema mapping that has recently received considerable attention [2,4,9,11,15]. Consider a mapping $\mathcal{M}$ from a schema **A** to a schema **B**. Intuitively, an *inverse* of $\mathcal{M}$ is a new mapping that describes the *reverse* relationship from **B** to **A** and is semantically consistent with the relationship established by $\mathcal{M}$.

In practical scenarios, the inverse of a mapping can have several applications. In a data exchange context [7,12], if a mapping $\mathcal{M}$ is used to exchange data from a source to a target schema, an inverse of $\mathcal{M}$ can be used to exchange the data back to the source, thus *reverting* the application of $\mathcal{M}$. As a second application, consider a peer data management system (PDMS) [19,20]. In a PDMS, a peer can act as a data source, a mediator, or both, and the system relates peers by establishing mappings between the peer schemas. Mappings between peers are usually *directional* and are used to reformulate queries. For example, if there is a mapping $\mathcal{M}$ from peer P_1 to peer P_2 and a query over P_2, a PDMS can use $\mathcal{M}$ to reformulate the query by using P_1 as a source. Hence, an inverse of $\mathcal{M}$ would allow the PDMS to reformulate a query over P_1 in terms of P_2, thus considering this time P_2 as a source. Another application is schema evolution, where the inverse together with the composition operator play a crucial role [9]. Consider a mapping $\mathcal{M}$ between schemas **A** and **B**, and assume that schema **A** evolves into a schema $\mathbf{A}'$. This evolution can be expressed as a mapping $\mathcal{M}'$ between **A** and $\mathbf{A}'$. Thus, the relationship between the new schema $\mathbf{A}'$ and schema **B** can be obtained by inverting mapping $\mathcal{M}'$ and then composing the result with mapping $\mathcal{M}$.

All the previous work on inverting schema mappings has been motivated by foundational issues [4,11,15,16], one of the most delicate being the definition of a good semantics for inversion of mappings. Yet, up to now, little attention has been paid to the study of practical issues regarding inverting schema mappings, and systems implementing these notions are scarce, mostly built for specific management tasks [27]. One possible reason for this is that all the notions of inverse proposed so far fail to meet some fundamental requirements that guarantee their practical applicability. Below, we identify three such crucial conditions.

Existence for the most common mappings: A first crucial requirement for any notion of inverse for schema mappings is that it has to be defined for the mappings used in practice. In particular, given that the language of *source-to-target tuple-generating dependencies* (st-tgds), or GLAV constraints, is arguably the most popular language to specify schema mappings, an inverse notion to be used in practice has to be applicable for each schema mapping specified in terms of these dependencies.

Efficiency of exchanging data: In the data exchange scenario, the standard procedure used to exchange data efficiently with a mapping is based on the classical *chase* procedure [12]. More precisely, given a mapping $\mathcal{M}$ and a source database I, a *canonical* translation of I according to $\mathcal{M}$ is computed by *chasing* I with the set of dependencies defining $\mathcal{M}$ [12]. Thus, when computing an inverse of $\mathcal{M}$, it would be desirable from a practical point of view to obtain a mapping $\mathcal{M}'$ where the chase procedure can also be used to exchange data in an efficient way.

Specification of inverses: Closely related with the previous issue, there is a *representation* issue. In the framework proposed by Bernstein [8], schema mappings are first-class citizens, and high-level algebraic operators are used to manipulate and reuse them. In this algebraic context, a natural question is whether a logical language for specifying mappings is *closed under inversion*: given a schema mapping specified in some language $\mathcal{L}$, can its inverse be also specified in $\mathcal{L}$? For example, if a mapping is specified by a set of st-tgds, can its inverse be specified in the same language? In practice, this is a desirable condition as it means that inverse mappings can be specified in a language with good properties for metadata management.

The search for an inverse notion satisfying these requirements has several practical implications, as these properties are crucial to guarantee the possibility of *combining* the inverse operator with other operators from Bernstein's framework [8]. Up to this point, no meaningful mapping language is known to be *closed* under any of the inverse notions proposed in the literature [4,11,15,16], and a few of them where the chase can be used to exchange data efficiently are not guaranteed to exist for a mapping specified by st-tgds, which obviously undermines their practical applicability (see e.g., [11]). Thus, the main goal of this paper is to find a notion of inverse that satisfies the three requirements previously mentioned. This goal amounts to (1) first choose a *natural* semantics for the inverse operator and (2) provide a *useful* mapping language $\mathcal{L}$ that contains the language of st-tgds, is closed under this notion of inverse, and where the classical chase procedure can be used to exchange data efficiently.

The issue of finding closed mapping languages for schema mapping operators has been raised as a "prominent issue" in metadata management [22]. Unfortunately, until now there have been very little positive results about this issue for an inverse operator [4,11,15,16]. This suggests that one should look for a weaker notion of inverse in order to obtain the desired closure results. But how can an inverse notion be weakened, and how much? To answer these questions, we follow earlier work on schema mapping composition [24] and schema mapping optimization [13] and propose a new inverse notion designed to generate inverse mappings that recover

back only the information that could have been retrieved by a given query language.

More precisely, our framework is based on the following idea. Let $\mathcal{C}$ be a class of queries. Then, assuming that we are only interested in answering queries that belong to $\mathcal{C}$, our inverse should only care to *recover* the initial data that could be retrieved by posing queries in $\mathcal{C}$. Moreover, there is a soundness requirement; we would like to recover only *sound information*, that is, information that was already present before the exchange. But, what does it mean to recover sound information? By answering this simple yet fundamental question, we uncover a rich theory. In fact, this gives rise to the notion of $\mathcal{C}$-recovery of a mapping: a reverse mapping that recovers sound information for a given mapping under a query language $\mathcal{C}$. We further introduce an order relation on $\mathcal{C}$ recoveries that lead to the notion of $\mathcal{C}$-maximum recovery, which is a mapping that recovers the maximum amount of information for a given mapping according to $\mathcal{C}$.

By choosing different alternatives for the query language $\mathcal{C}$ in the definition of the notion of $\mathcal{C}$-maximum recovery, we are able to fine-tune the power of the resulting inverses. In particular, this allows us to characterize the notions of inverse proposed in the literature; as an example, it is shown in the paper that the notion of Fagin-inverse proposed in [11] corresponds to the notion of $\mathcal{C}$-maximum recovery if $\mathcal{C}$ is the class of unions of conjunctive queries with inequalities. But more importantly, the parameterization of our notion of inverse by a query language allows us to find a natural notion of inverse for schema mappings that satisfies the requirements mentioned before.

Let CQ be the class of conjunctive queries. The main result of the second part of the paper states that the notion of CQ-maximum recovery satisfies our proposed desiderata. That is, there exists a well-behaved language that contains the class of st-tgds, that is closed under CQ-maximum recovery, and where the classical chase procedure can be used to exchange data efficiently. More specifically, this language is defined by the class of $\text{CQ}^{\mathbf{C},\neq}$- TO- CQ dependencies, which are, essentially, st-tgds extended in the premises with inequalities and a built-in predicate $\mathbf{C}(\cdot)$ to differentiate constants from null values.

We prove several results that justify our choices of CQ-maximum recovery as the notion of inverse for schema mappings and of $\text{CQ}^{\mathbf{C},\neq}$- TO- CQ dependencies as the language for specifying schema mappings. In particular, we show in the paper that given that the new features of $\text{CQ}^{\mathbf{C},\neq}$- TO- CQ dependencies are only allowed in the premises, this language is good as the language of st-tgds for data exchange purposes. Moreover, we provide in the paper an algorithm that, given a mapping specified by a set of $\text{CQ}^{\mathbf{C},\neq}$- TO- CQ dependencies, returns a CQ-maximum recovery of this mapping that is also specified by a set of $\text{CQ}^{\mathbf{C},\neq}$- TO- CQ dependencies. Finally, we show that our choices are *optimal* to obtain the desired aforementioned properties, in the sense that choosing a more expressive inverse operator or mapping language causes the loss of these properties. For instance, we prove that a closure result cannot be obtained if instead of using CQ-maximum recoveries as the notion for inverting mappings, we consider either UCQ-maximum recoveries or $\text{CQ}^{\neq}$-maximum recoveries.

Organization of the paper. In Sect. 2, we introduce the notation used in this paper. In Sect. 3, we present the notion of $\mathcal{C}$-maximum recovery. In Sect. 4, we compare our proposal with the previous notions of inverses proposed in the literature. In Sect. 5, we present an algorithm to compute CQ-maximum recoveries and show a closure result for the language of mappings specified by $\text{CQ}^{\mathbf{C},\neq}$- TO- CQ dependencies. In Sect. 6, we show the *optimality* of the closure result in Sect. 5. Finally, we present some concluding remarks in Sect. 7. For the sake of space, the complete proofs of the results in this paper are included as electronic supplementary material (electronic Appendix A).

1.1 New material in this paper

Preliminary versions of some of the results in this paper appeared in [3]. Nevertheless, this paper contains substantial new material. We include a new result on the separation of $\mathcal{C}$-maximum recoveries for the most common extensions of CQ (Proposition 1). The notion of *fully recoverable query* introduced in Sect. 4 is new, as well as the characterizations of Fagin-inverses and quasi-inverses that use this notion (Propositions 2 and 3). We also include a result on the relationship between maximum recoveries and $\mathcal{C}$-maximum recoveries (Theorem 3), which is not given in [3]. The closure result presented in Sect. 5 (Theorem 4) is also new as well as the results in Sect. 6, which show the optimality of this closure property (Propositions 5, 6 and 7). Beside the aforementioned new results, in this paper, we include new examples and also many proof sketches in the body of the paper plus full proofs in the electronic appendix (which are not included in [3]).

1.2 Related work and limitations of our approach

As most of the research on schema mappings [12,13,22] and in particular on inverting mappings [2–4,11,15,17], we make the assumption that source instances contain only constant values, while target instances contain constant and null values, the latter to represent missing (incomplete) information (see Sect. 2 for a formalization of our setting). We discuss here the limitations of this assumption when studying inverses, as well as the related work on relaxing this assumption [5,16].

There have been at least two investigations that relax the assumption mentioned above in order to study the inversion

of schema mappings. In [16], the authors study a setting in which both source and target instances contain null values, and make the case that inverses should be studied in this *symmetric* setting. Similarly, in [5], the authors study the problem of exchanging incomplete information in a more general setting not only including nulls in the source data, but also considering general representation systems to exchange data. In both papers, the focus is on the semantics of inversion, and less attention is paid in more practical concerns. In particular, although in [16] the authors introduce a meaningful notion of inverse in the symmetric setting, the results from an algorithmic point of view are not very encouraging; the authors provide an algorithm to compute inverses of st-tgds that uses second-order quantification to express inverses, and left open whether the full power of first-order logic is enough to express inverses even for the simple fragment of mappings specified by full st-tgds (which are st-tgds without existential quantification). Similarly, although the authors in [5] show that by adding expressiveness to source and target instances one can obtain good properties for the existence of inverses, they do not provide an algorithm to compute them.

Instead of studying a more expressive setting allowing mappings to contain incomplete information in source and target instances, we try to substantially improve our understanding of the classical data exchange setting in which source instances are considered to have only constant values. This assumption not only allows us to simplify our study but also allows us to provide several positive algorithmic and practical results. Nevertheless, as observed by Fagin et al. [16], our setting comes with a limitation on the uniform treatment of data exchange in both directions (from source to target and from target to source). It has been noted in [16] that after exchanging data with a mapping $\mathcal{M}$, the usual instance that one would like to materialize in the target, which is the result of the chase procedure, contains null values. Thus, if one uses the inverse of $\mathcal{M}$ to exchange data back, this time from the target to the source schema, nulls would naturally arise in the source. This is the reason why Fagin et al. [16] consider that the restriction of having only constant in the source is a *semantic mismatch* in the study of inverses.

From the point of view advocated in [16], the proposal for inverting mappings introduced in this paper also suffers from the aforementioned semantic mismatch. Nevertheless, it should be noticed that there is not yet consensus on the community on what is the ultimate setting to study inverses and more generally, schema mapping operators. Moreover, most of the investigation has been carried out on the classical (asymmetrical) data exchange setting, and this setting had allowed the development of several positive results on the subject [4,11,14,15,17]. Our proposal in this paper can be considered as continuing the *top-down* approach to the subject of inverting mappings, trying to close the gap between theory and practice based on the huge amount of work on the the classical setting of data exchange. This approach is complementary to the more integral approaches proposed in [5,16] of adding expressiveness to database instances and mappings. We expect that our results on the classical scenario of data exchange could be useful in the future research on mappings for databases with incomplete information, on extensions of the notions of inversion in a symmetric scenario, and on the search for the right notion of inversion together with the right language for expressing inverses that can be used in the next generation of schema mapping tools.

2 Basic notation

Our study assumes that data are represented in the relational model. A *relational schema*, or just *schema*, is a finite set $\{R_1, \ldots, R_n\}$ of relation symbols, with each R_i having a fixed arity n_i. As is customary in the data exchange and schema mapping literature, we consider two types of data values when constructing database instances: constants and nulls [11,12,15]. More precisely, let $\mathbf{C}$ and $\mathbf{N}$ be infinite and disjoint sets of constants and nulls, respectively. A database instance (or just instance) I of schema $\mathbf{R}$ assigns to each $n$ary relation symbol R of $\mathbf{R}$ a finite $n$ary relation $R^I \subseteq (\mathbf{C} \cup \mathbf{N})^n$. If a tuple $\bar{a}$ belongs to R^I, we say that $R(\bar{a})$ is a *fact* in I. We sometimes describe an instance as a set of facts. If we refer to a schema $\mathbf{S}$ as a *source* schema, then we restrict all instances of $\mathbf{S}$ to consist only of constant values. On the other hand, we allow null values in the instances of any *target* schema $\mathbf{T}$.

Schema mappings and solutions. Schema mappings are used to define a semantic relationship between two schemas. In this paper, we use a general representation of mappings; given two schemas $\mathbf{R}_1$ and $\mathbf{R}_2$, a mapping $\mathcal{M}$ from $\mathbf{R}_1$ to $\mathbf{R}_2$ is a set of pairs (I, J), where I is an instance of $\mathbf{R}_1$, and J is an instance of $\mathbf{R}_2$. Further, we say that J is a *solution for I under $\mathcal{M}$*, if (I, J) is in $\mathcal{M}$. We denote by $\mathrm{Sol}_{\mathcal{M}}(I)$ the set of all solutions for I under $\mathcal{M}$.

Some of our results are focused on a special class of mappings that we call *source-to-target* mappings (st-mappings). If we refer to a mapping $\mathcal{M}$ from $\mathbf{R}_1$ to $\mathbf{R}_2$ as an *st-mapping*, then we assume that $\mathbf{R}_1$ is a source schema (that is, instances of $\mathbf{R}_1$ are constructed using only elements from $\mathbf{C}$), and $\mathbf{R}_2$ is a target schema (that is, instances of $\mathbf{R}_2$ are constructed by using elements from $\mathbf{C}$ and $\mathbf{N}$).

Composition of mappings. The notion of composition has shown to be of fundamental importance in the study of the inverse of a schema mapping [4,11,15]. Let $\mathcal{M}_{12}$ be a mapping from a schema $\mathbf{R}_1$ to a schema $\mathbf{R}_2$, and $\mathcal{M}_{23}$ a mapping from $\mathbf{R}_2$ to a schema $\mathbf{R}_3$. Then, the composition of $\mathcal{M}_{12}$ and $\mathcal{M}_{23}$, denoted by $\mathcal{M}_{12} \circ \mathcal{M}_{23}$, is defined as the standard composition of binary relations, that is, as the set of all pairs

of instances (I, J) such that I is an instance of $\mathbf{R}_1$, J is an instance of $\mathbf{R}_3$, and there exists an instance K of $\mathbf{R}_2$ such that (I, K) belongs to $\mathcal{M}_{12}$, and (K, J) belongs to $\mathcal{M}_{23}$ [14,26].

Query answering. A $k$ary query Q over a schema $\mathbf{R}$, with $k \geq 0$, is a function that maps every instance I of $\mathbf{R}$ into a k-relation $Q(I) \subseteq \text{dom}(I)^k$, where $\text{dom}(I)$ is the set of elements mentioned in I. Notice that if $k = 0$ (Q is a Boolean query), then the answer to Q is either the set with one 0-ary tuple or the empty set, denoted by true and false, respectively.

We use CQ to denote the class of conjunctive queries and UCQ to denote the class of unions of conjunctive queries. We note that we consider that CQ and UCQ are not equipped with equalities. If we extend these classes by allowing equalities or inequalities, then we use superscripts $=$ and $\neq$, respectively. Thus, for example, $\text{CQ}^{=}$ is the class of conjunctive queries with equalities, and $\text{UCQ}^{\neq}$ is the class of union of conjunctive queries with inequalities. FO is the class of all first-order queries with equality. Slightly abusing notation, we use $\mathbf{C}(\cdot)$ to denote a built-in unary predicate such that $\mathbf{C}(a)$ holds if and only if a is a constant, that is, $a \in \mathbf{C}$. If $\mathcal{L}$ is any of the previous query languages, then $\mathcal{L}^{\mathbf{C}}$ is the extension of $\mathcal{L}$ allowing predicate $\mathbf{C}(\cdot)$. For example, $\text{CQ}^{\mathbf{C},\neq}$ is the class of conjunctive queries with inequalities and predicate $\mathbf{C}(\cdot)$.

As usual, the semantics of queries in the presence of schema mappings is defined in terms of the notion of *certain answer*. Assume that $\mathcal{M}$ is a mapping from a schema $\mathbf{R}_1$ to a schema $\mathbf{R}_2$. Then, given an instance I of $\mathbf{R}_1$ and a query Q over $\mathbf{R}_2$, the *certain answers of Q for I under $\mathcal{M}$*, denoted by $\text{certain}_{\mathcal{M}}(Q, I)$, is the set of tuples that belong to the evaluation of Q over every possible solution for I under $\mathcal{M}$, that is,

$$\text{certain}_{\mathcal{M}}(Q, I) = \bigcap \{Q(J) \mid J \in \text{Sol}_{\mathcal{M}}(I)\}.$$

Dependencies and definability of mappings. Let $\mathcal{L}_1$, $\mathcal{L}_2$ be query languages and $\mathbf{R}_1$, $\mathbf{R}_2$ be schemas with no relation symbols in common. An $\mathcal{L}_1$- TO- $\mathcal{L}_2$ *dependency from* $\mathbf{R}_1$ *to* $\mathbf{R}_2$ is a formula Φ of the form

$$\forall \bar{x} \, (\varphi(\bar{x}) \rightarrow \psi(\bar{x})), \tag{1}$$

where (a) $\bar{x}$ is the tuple of free variables in both $\varphi(\bar{x})$ and $\psi(\bar{x})$; (b) $\varphi(\bar{x})$ is an $\mathcal{L}_1$-formula over $\mathbf{R}_1 \cup \{\mathbf{C}(\cdot)\}$ if $\mathbf{C}(\cdot)$ is allowed in $\mathcal{L}_1$, and over $\mathbf{R}_1$ otherwise, and (c) $\psi(\bar{x})$ is an $\mathcal{L}_2$-formula over $\mathbf{R}_2 \cup \{\mathbf{C}(\cdot)\}$ if $\mathbf{C}(\cdot)$ is allowed in $\mathcal{L}_2$, and over $\mathbf{R}_2$ otherwise. We call $\varphi(\bar{x})$ the *premise* of Φ, and $\psi(\bar{x})$ the *conclusion* of Φ. We usually omit the outermost universal quantifier in $\mathcal{L}_1$- TO- $\mathcal{L}_2$ dependencies, and thus, we write dependency (1) just as $\varphi(\bar{x}) \rightarrow \psi(\bar{x})$. Moreover, for the sake of readability, we usually write $\alpha(\bar{x}, \bar{y}) \rightarrow \psi(\bar{x})$ instead of $(\exists \bar{y} \, \alpha(\bar{x}, \bar{y})) \rightarrow \psi(\bar{x})$ in the examples.

A fundamental class of dependencies in the schema mapping literature is the class of source-to-target tuple-generating dependencies (st-tgds), which corresponds to the class of CQ- TO- CQ dependencies. Another class that we extensively use in this paper is the class of $\text{CQ}^{\neq,\mathbf{C}}$- TO- CQ dependencies, which are essentially st-tgds in which the premise is extended with inequalities and predicate $\mathbf{C}(\cdot)$.

Let $\mathbf{R}_1$ and $\mathbf{R}_2$ be schemas with no relation symbols in common and Σ a set of $\mathcal{L}_1$- TO- $\mathcal{L}_2$-dependencies from $\mathbf{R}_1$ to $\mathbf{R}_2$. We say that a mapping $\mathcal{M}$ from $\mathbf{R}_1$ to $\mathbf{R}_2$ is *specified* by Σ, denoted by $\mathcal{M} = (\mathbf{R}_1, \mathbf{R}_2, \Sigma)$, if for every instance I of $\mathbf{R}_1$ and instance J of $\mathbf{R}_2$, it holds that $(I, J) \in \mathcal{M}$ if and only if (I, J) satisfies the dependencies in Σ.

Homomorphisms and the chase. Given instances J_1 and J_2 of the same schema, a *homomorphism* h from J_1 to J_2 is a function that is the identity over constants ($h(a) = a$ for every $a \in \mathbf{C}$), maps null values to null or constant values, and for every fact $R(a_1, \ldots, a_n)$ in J_1, it holds that $R(h(a_1), \ldots, h(a_k))$ is a fact in J_2. If there exists homomorphisms from J_1 to J_2 and from J_2 to J_1, then we say that J_1 and J_2 are homomorphically equivalent.

Another notion that will be used in this document is the notion of *chase* [25]. Assume that $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \Sigma)$ is an st-mapping, where Σ is a set of FO- TO- CQ dependencies. Let I be an instance of $\mathbf{S}$, and let J_I be an instance of $\mathbf{T}$ constructed as follows. For every $\sigma \in \Sigma$ of the form $\varphi(\bar{x}) \rightarrow \exists \bar{y} \, \psi(\bar{x}, \bar{y})$, where $\bar{x} = (x_1, \ldots, x_m)$ and $\bar{y} = (y_1, \ldots, y_\ell)$ are tuples of pairwise distinct variables with no variables in common, and for every m-tuple $\bar{a}$ of elements mentioned in I such that $I \models \varphi(\bar{a})$, do the following. Choose an ℓ-tuple $\bar{n}$ of pairwise distinct fresh values from $\mathbf{N}$ and include all the conjuncts of $\psi(\bar{a}, \bar{n})$ as facts in J_I. We call instance J_I the result of *chasing* I with Σ and write $J_I = \text{chase}_{\Sigma}(I)$.

The instance $\text{chase}_{\Sigma}(I)$ has several desirable properties [1,12]. In particular, if $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \Sigma)$ is an st-mapping with Σ a set of FO- TO- CQ dependencies, then for every instance I of $\mathbf{S}$, it holds that $\text{chase}_{\Sigma}(I) \in \text{Sol}_{\mathcal{M}}(I)$. Moreover, for every solution $J \in \text{Sol}_{\mathcal{M}}(I)$, it holds that there exists a homomorphism from $\text{chase}_{\Sigma}(I)$ to J [12], reason for which $\text{chase}_{\Sigma}(I)$ is called a *canonical universal solution* for I under $\mathcal{M}$ [1,12]. This property of the chase implies that it can be used to compute certain answers of unions of conjunctive queries [12]. Formally, for every instance I of $\mathbf{S}$ and query Q in UCQ over $\mathbf{T}$, it holds that $\text{certain}_{\mathcal{M}}(Q, I) = Q(\text{chase}_{\Sigma}(I))_{\downarrow}$, where $Q(\text{chase}_{\Sigma}(I))_{\downarrow}$ denotes the set of tuples obtained from $Q(\text{chase}_{\Sigma}(I))$ by eliminating all the tuples that mention null values.

3 Query language-based inverses of schema mappings

Intuitively, an inverse of a schema mapping $\mathcal{M}$ is a *reverse* mapping that *undoes* the application of $\mathcal{M}$. Any natural notion of inverse should capture the intuition that if $\mathcal{M}$

describes how to exchange data from a source to a target schema, then the inverse of $\mathcal{M}$ must describe how to *recover* the initial data back in the source (or, at least, part of it). Moreover, one should impose a soundness requirement; one would like to recover only *sound information*, that is, information that was already present before the exchange.

A natural question at this point is how one can formally describe the idea of recovering sound information. In this paper, we give a formal definition of what it means to recover sound information with respect to a query language, and use this notion to define our query language-based notion of inverse of a schema mapping. It is important to notice that a first notion of *recovering sound information* was proposed in [4], and it was called a *recovery* of a mapping $\mathcal{M}$. A recovery is a reverse mapping that recovers sound information with respect to a mapping $\mathcal{M}$. Here, we define a relaxation of this notion that is parameterized by a query language.

Before formalizing what does it mean to recover sound information with respect to a query language, let us present the intuition of this notion with one example.

Example 1 Let **S** be a source schema consisting of binary relations $R(\cdot, \cdot)$ and $S(\cdot, \cdot)$, **T** a target schema consisting of a binary relation $T(\cdot, \cdot)$, and assume that schemas **S** and **T** are related by a mapping $\mathcal{M}$ specified by st-tgd:

$$R(x, y) \wedge S(y, z) \rightarrow T(x, z). \tag{2}$$

Thus, target relation T stores the join of source relations R and S. Consider now the *reverse* mapping $\mathcal{M}'$ relating **T** and **S** through the dependency:

$$T(x, y) \rightarrow \exists u\, R(x, u). \tag{3}$$

This dependency states that whenever an element is in the first component of relation T in the target database, it must also be in the first component of relation R in the source database. Thus, given the definition of mapping $\mathcal{M}$, one can intuitively conclude that mapping $\mathcal{M}'$ recovers sound information with respect to $\mathcal{M}$. The previous intuition can be formalized by considering the composition of $\mathcal{M}$ with $\mathcal{M}'$, which represents the idea of using $\mathcal{M}'$ to bring back to the source the information that was exchanged from the source by using mapping $\mathcal{M}$. It is important to notice that $\mathcal{M} \circ \mathcal{M}'$ is a *round-trip* mapping from **S** to **S** and, therefore, one can use queries over **S** to measure the amount of recovered information. In particular, one can claim in this example that $\mathcal{M}'$ recovers sound information with respect to $\mathcal{M}$ since for every source instance I and query Q over **S**, if a tuple $\bar{t}$ belongs to the certain answers of Q for I under $\mathcal{M} \circ \mathcal{M}'$, then $\bar{t}$ also belongs to the evaluation of Q over I.

Let us give an example of the previous discussion with a concrete scenario. Assume that I is a source database $\{R(a, b), R(c, d), S(b, e)\}$, and $Q(x)$ is the source conjunctive query $\exists y\, R(x, y)$. If we directly evaluate Q over I, we obtain the set of answers $Q(I) = \{a, c\}$. On the other hand, if we evaluate Q over the set of instances obtained by exchanging data from I through $\mathcal{M} \circ \mathcal{M}'$, then we obtain the set of answers $\{a\}$. To see why $\{a\}$ is the set of certain answers in this case, notice that every solution for I under $\mathcal{M}$ contains the tuple $T(a, e)$ and, hence, every solution for I under $\mathcal{M} \circ \mathcal{M}'$ contains the tuple $R(a, u)$, for some element u. Thus, the value a belongs to the set of certain answers of query Q over I under mapping $\mathcal{M} \circ \mathcal{M}'$. Besides, no other element belongs to this set of certain answers as $\{R(a, b)\}$ is a solution for I under $\mathcal{M} \circ \mathcal{M}'$.

We conclude this example by pointing out that given the definitions of mappings $\mathcal{M}$ and $\mathcal{M}'$, when computing the certain answers of a query Q over a source instance I under $\mathcal{M} \circ \mathcal{M}'$, we obtain a subset of the direct evaluation of the query over I, that is:

$$\text{certain}_{\mathcal{M} \circ \mathcal{M}'}(Q, I) \subseteq Q(I). \tag{4}$$

In general, we say that a mapping $\mathcal{M}'$ is a Q-recovery of a mapping $\mathcal{M}$ whenever Eq. (4) holds for every source database I. For example, for the mappings $\mathcal{M}$ and $\mathcal{M}'$ defined by dependencies (2) and (3), respectively, we have that $\mathcal{M}'$ is a Q-recovery of $\mathcal{M}$ for the query $Q(x) = \exists y\, R(x, y)$. Furthermore, it can also be shown that $\mathcal{M}'$ is a Q-recovery of $\mathcal{M}$ for every query Q. In fact, this is the reason why we claim that $\mathcal{M}'$ recovers sound information with respect to $\mathcal{M}$ in this case. This intuition gives rise to the following notion of $\mathcal{C}$-recovery, where $\mathcal{C}$ is a class of queries.

Definition 1 Let $\mathcal{C}$ be a class of queries, $\mathcal{M}$ a mapping from schema **S** to schema **T**, and $\mathcal{M}'$ a mapping from **T** to **S**. Then, $\mathcal{M}'$ is a *$\mathcal{C}$-recovery* of $\mathcal{M}$ if for every query $Q \in \mathcal{C}$ over **S** and every instance I of **S**, it holds that:

$$\text{certain}_{\mathcal{M} \circ \mathcal{M}'}(Q, I) \subseteq Q(I).$$

Notice that in the previous definition we consider a class $\mathcal{C}$ of queries, as in some scenarios one is interested in retrieving sound information not for every possible query but for a class of queries of interest (for instance, for the class of conjunctive queries).

Being a $\mathcal{C}$-recovery is a sound but mild requirement. Thus, it is natural to ask whether one can compare mappings according to their ability to recover sound information, and then whether there is a natural way to define a notion of *best* possible recovery according to a given query language. It turns out that there is simple and natural way to do this, as we show in the following example.

Example 2 Let $\mathcal{M}$ and $\mathcal{M}'$ be the mappings given by dependencies (2) and (3), respectively, and $\mathcal{M}''$ a mapping specified by dependency:

$$T(x, y) \rightarrow \exists u\, (R(x, u) \wedge S(u, y)).$$

As mentioned in Example 1, $\mathcal{M}'$ recovers sound information with respect to $\mathcal{M}$ as it is a Q-recovery of $\mathcal{M}$ for every query Q. Furthermore, it can be shown that this property also holds for mapping $\mathcal{M}''$. In order to compare these mappings, we observe that for every query Q and source instance I, mappings $\mathcal{M}'$ and $\mathcal{M}''$ satisfy the following property:

$$\text{certain}_{\mathcal{M}\circ\mathcal{M}'}(Q, I) \subseteq \text{certain}_{\mathcal{M}\circ\mathcal{M}''}(Q, I) \subseteq Q(I).$$

For instance, if $I = \{R(a, b), R(c, d), S(b, e)\}$ and $Q(x, y)$ is conjunctive query $\exists z\,(R(x, z) \wedge S(z, y))$, then we have that $Q(I) = \{(a, e)\}$ and $\text{certain}_{\mathcal{M}\circ\mathcal{M}''}(Q, I) = \{(a, e)\}$, while $\text{certain}_{\mathcal{M}\circ\mathcal{M}'}(Q, I) = \emptyset$. Thus, every tuple that is retrieved by posing query Q against the space of solutions for I under $\mathcal{M} \circ \mathcal{M}'$ is also retrieved by posing this query over the space of solutions for I under $\mathcal{M} \circ \mathcal{M}''$. Hence, we can claim that $\mathcal{M}''$ is *better* than $\mathcal{M}'$ recovering sound information with respect to $\mathcal{M}$. □

The above discussion gives rise to a simple way to compare two Q-recoveries $\mathcal{M}'$ and $\mathcal{M}''$ of a mapping $\mathcal{M}$; a mapping $\mathcal{M}''$ recovers as much information as $\mathcal{M}'$ does for $\mathcal{M}$ under Q if for every source instance I, it holds that $\text{certain}_{\mathcal{M}\circ\mathcal{M}'}(Q, I) \subseteq \text{certain}_{\mathcal{M}\circ\mathcal{M}''}(Q, I)$. With this way of comparing inverse mappings, it is straightforward to define a notion of *best* possible recovery according to a query language.

Definition 2 Let $\mathcal{C}$ be a class of queries, $\mathcal{M}$ a mapping from a schema $\mathbf{S}$ to a schema $\mathbf{T}$, and $\mathcal{M}_1$ a $\mathcal{C}$-recovery of $\mathcal{M}$. Then, $\mathcal{M}_1$ is a *$\mathcal{C}$-maximum recovery* of $\mathcal{M}$ if for every $\mathcal{C}$-recovery $\mathcal{M}_2$ of $\mathcal{M}$, it holds that:

$$\text{certain}_{\mathcal{M}\circ\mathcal{M}_2}(Q, I) \subseteq \text{certain}_{\mathcal{M}\circ\mathcal{M}_1}(Q, I),$$

for every query Q in $\mathcal{C}$ over $\mathbf{S}$ and every instance I of $\mathbf{S}$.

That is, $\mathcal{M}_1$ is a $\mathcal{C}$-maximum recovery of a mapping $\mathcal{M}$ if by exchanging data from I through $\mathcal{M}\circ\mathcal{M}_1$, one can retrieve by using queries from $\mathcal{C}$ as much information as by exchanging data from I through $\mathcal{M} \circ \mathcal{M}_2$, for any other $\mathcal{C}$-recovery $\mathcal{M}_2$ of $\mathcal{M}$. For instance, for the mappings $\mathcal{M}$ and $\mathcal{M}''$ mentioned in Example 2, it holds that $\mathcal{M}''$ is an ALL-maximum recovery of $\mathcal{M}$, where ALL is the class of all queries.

3.1 On the choice of a query language

At this point, a natural question is what is the influence of the parameter $\mathcal{C}$ on the notion of $\mathcal{C}$-maximum recovery. In the following sections, we show that this parameter is essential to obtain a good mapping language for inversion, but first we need to show how different selections of the language $\mathcal{C}$ give rise to essentially different notions of inverse. Let us start with an example relating the classes UCQ and CQ.

Example 3 Let $\mathcal{M}$ be a mapping specified by st-tgds:

$$A(x, y) \rightarrow R(x, y)$$
$$B(x) \rightarrow R(x, x)$$

It can be shown that mapping $\mathcal{M}_1$ specified by dependency:

$$R(x, y) \rightarrow A(x, y) \vee \big(B(x) \wedge x = y\big) \tag{5}$$

is a UCQ-maximum recovery of $\mathcal{M}$.

Notice that to specify $\mathcal{M}_1$, we use a disjunction in the conclusion of dependency (5), which can be shown to be unavoidable as UCQ is used to retrieve information. Unfortunately, tgds with disjunctions on the conclusion make the processes of exchanging data and computing certain answers much more complicated. On the other hand, if we focus on CQ to retrieve information, then, intuitively, there is no need for disjunctions in the right-hand side of the rules as conjunctive queries cannot extract disjunctive information. In fact, it can be shown that a CQ-maximum recovery of $\mathcal{M}$ is specified by dependency:

$$R(x, y) \wedge x \neq y \rightarrow A(x, y).$$

Intuitively, the only *conjunctive information* that one can obtain from a target instance is that if $R(a, b)$ is a fact in the target with $a \neq b$, then the fact $A(a, b)$ was a fact in the initial source instance. Notice that if $R(a, a)$ is in the target, we cannot assume that this information came either from relation $A(\cdot, \cdot)$ or $B(\cdot)$. Although in this case one can safely assume that the initial source instance contained either the fact $A(a, a)$ or the fact $B(a)$, this information is useless if we only consider conjunctive queries to extract data.

The above example suggests that the notion of CQ-maximum recovery is a strict generalization of the notion of UCQ-maximum recovery. The following proposition provides a complete picture of the relationship of the notions of $\mathcal{C}$-maximum recovery, when one focuses on mappings specified by st-tgds and the most common extensions of CQ.

Proposition 1

(1) *There exist st-tgd mappings $\mathcal{M}$ and $\mathcal{M}'$ such that $\mathcal{M}'$ is a* UCQ*-maximum recovery of $\mathcal{M}$ but not a* $\text{CQ}^{\neq}$*-maximum recovery of $\mathcal{M}$.*

(2) *There exist st-tgd mappings $\mathcal{M}$ and $\mathcal{M}'$ such that $\mathcal{M}'$ is a* $\text{CQ}^{\neq}$*-maximum recovery of $\mathcal{M}$ but not a* UCQ*-maximum recovery of $\mathcal{M}$.*

Proposition 1 tells us that the notions of UCQ- and $\text{CQ}^{\neq}$-maximum recovery are incomparable, even in the case of st-tgds. Moreover, we can also conclude from this proposition that if $\mathcal{C}_1$ and $\mathcal{C}_2$ are any of the classes of queries CQ, UCQ, $\text{CQ}^{\neq}$ or $\text{UCQ}^{\neq}$, and $\mathcal{C}_1 \subsetneq \mathcal{C}_2$, then there exist mappings $\mathcal{M}$ and $\mathcal{M}'$ such that $\mathcal{M}'$ is a $\mathcal{C}_1$-maximum recovery of $\mathcal{M}$ but not a $\mathcal{C}_2$-maximum recovery of $\mathcal{M}$.

We now move to the proof of Proposition 1.

Proof of Proposition 1 (1) Consider a source schema $\mathbf{S} = \{P(\cdot), R(\cdot)\}$, a target schema $\mathbf{T} = \{T(\cdot,\cdot), S(\cdot)\}$ and a mapping $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \Sigma)$, where Σ consists of the following st-tgds:

$$P(x) \rightarrow \exists y\, T(x, y)$$
$$R(x) \rightarrow S(x)$$

Consider also a mapping $\mathcal{M}' = (\mathbf{T}, \mathbf{S}, \Sigma')$, where Σ' consists of the following dependencies:

$$T(x, x) \wedge S(y) \rightarrow P(y) \quad (6)$$
$$T(x, y) \rightarrow P(x) \wedge P(y) \quad (7)$$
$$S(x) \rightarrow R(x)$$

We first show that $\mathcal{M}'$ is not a CQ$^{\neq}$-recovery of $\mathcal{M}$, from which we conclude that $\mathcal{M}'$ is not a CQ$^{\neq}$-maximum recovery of $\mathcal{M}$. Consider instance $I = \{P(a), R(b)\}$ with $a \neq b$. Notice that every solution for I under mapping $\mathcal{M}$ contains facts $S(b)$ and $T(a, u)$ for some value u. Thus, by definition of $\mathcal{M}'$, we have that every solution for I under $\mathcal{M} \circ \mathcal{M}'$ contains facts $P(a)$, $P(u)$ (dependency (7)) and, if $u = a$, then it also contains fact $P(b)$ (dependency (6)). Assume that Q is the following boolean query in CQ$^{\neq}$:

$$\exists x \exists y\, (P(x) \wedge P(y) \wedge x \neq y).$$

Then, we have that $\text{certain}_{\mathcal{M}\circ\mathcal{M}'}(Q, I) = \underline{\text{true}}$ but $Q(I) = \underline{\text{false}}$, from which we conclude that $\mathcal{M}'$ is not a CQ$^{\neq}$-recovery of $\mathcal{M}$.

We now prove that $\mathcal{M}'$ is a UCQ-maximum recovery of $\mathcal{M}$. Let Q be a $m$ary query in UCQ over $\mathbf{S}$ ($m \geq 0$) and I an instance of $\mathbf{S}$. Next we show that $Q(I) = \text{certain}_{\mathcal{M}\circ\mathcal{M}'}(Q, I)$, from which we conclude that $\mathcal{M}'$ is a UCQ-maximum recovery of $\mathcal{M}$. Consider the instance

$$I = \{P(a_1), \ldots, P(a_k), R(b_1), \ldots, R(b_\ell)\},$$

where $\{a_1, \ldots, a_k\}$ and $\{b_1, \ldots, b_\ell\}$ are not necessarily disjoint sets, and let $J = \text{chase}_{\Sigma}(I)$. It is easy to see that

$$J = \{T(a_1, n_1), \ldots, T(a_k, n_k), S(b_1), \ldots, S(b_\ell)\},$$

where $n_1, \ldots, n_k$ is a sequence of pairwise distinct null values. Now let $K = \text{chase}_{\Sigma'}(J)$. It is straightforward to prove that K is the instance given by the set of facts

$$\{P(a_1), P(n_1), \ldots, P(a_k), P(n_k), R(b_1), \ldots, R(b_\ell)\}.$$

Thus, given that Q is a monotone query, we have that $Q(I) \subseteq Q(K)$, and given that Q is a query in UCQ and there exists a homomorphism h from K to I that is the identity on the constants, we have that $Q(K) \cap \{a_1, \ldots, a_k, b_1, \ldots, b_\ell\}^m \subseteq Q(I)$. Therefore, we conclude that $Q(I) = (Q(K) \cap \{a_1, \ldots, a_k, b_1, \ldots, b_\ell\}^m)$, from which we obtain that $Q(I) = \text{certain}_{\mathcal{M}\circ\mathcal{M}'}(Q, I)$, given that the certain answers to Q over I under $\mathcal{M} \circ \mathcal{M}'$ can be obtained by evaluating Q over K and then removing the tuples containing null values [12,14].

(2) Consider a source schema $\mathbf{S} = \{D(\cdot), E(\cdot), F(\cdot)\}$, a target schema $\mathbf{T} = \{P(\cdot), R(\cdot)\}$, and a mapping $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \Sigma)$, where Σ consists of the following st-tgds:

$$D(x) \rightarrow P(x)$$
$$E(x) \rightarrow P(x)$$
$$F(x) \rightarrow R(x)$$

Consider also a mapping $\mathcal{M}' = (\mathbf{T}, \mathbf{S}, \Sigma')$, where Σ' consists of dependency $R(x) \rightarrow F(x)$. We show next that $\mathcal{M}'$ is not a UCQ-maximum recovery of $\mathcal{M}$, and we show in electronic Appendix A.1 that $\mathcal{M}'$ is a CQ$^{\neq}$-maximum recovery of $\mathcal{M}$, from which we conclude that (2) holds.

Let Σ'' consists of dependency $P(x) \rightarrow (D(x) \vee E(x))$ and $\mathcal{M}'' = (\mathbf{T}, \mathbf{S}, \Sigma'')$. From their definitions, one can easily see that $\mathcal{M}'$ and $\mathcal{M}''$ are UCQ-recoveries of $\mathcal{M}$. We show next that $\mathcal{M}'$ is not as informative as $\mathcal{M}''$ w.r.t. queries in UCQ, which implies that $\mathcal{M}'$ is not a UCQ-maximum recovery of $\mathcal{M}$. Consider the instance $I = \{D(a)\}$ of $\mathbf{S}$, and let Q be the following boolean query in UCQ over $\mathbf{S}$:

$$\exists x D(x) \vee \exists y E(y)$$

Let $I_\emptyset$ be the empty instance of schema $\mathbf{S}$. It is clear by the definitions of $\mathcal{M}$ and $\mathcal{M}'$ that $I_\emptyset$ is a possible solution for I under the composition $\mathcal{M} \circ \mathcal{M}'$. Thus, we have that $\text{certain}_{\mathcal{M}\circ\mathcal{M}'}(Q, I) = \underline{\text{false}}$ since $Q(I_\emptyset) = \underline{\text{false}}$. On the other hand, every solution for I under $\mathcal{M} \circ \mathcal{M}''$ contains fact $D(a)$ or fact $E(a)$ (as every solution for I under $\mathcal{M}$ contains fact $P(a)$), from which we conclude that $\text{certain}_{\mathcal{M}\circ\mathcal{M}''}(Q, I) = \underline{\text{true}}$. Therefore, we have that $\text{certain}_{\mathcal{M}\circ\mathcal{M}''}(Q, I) \not\subseteq \text{certain}_{\mathcal{M}\circ\mathcal{M}'}(Q, I)$, which implies that $\mathcal{M}'$ is not a UCQ-maximum recovery of $\mathcal{M}$. □

We have learned in this section, specifically in Example 1, that by focusing on some particular query languages in the notion of $\mathcal{C}$-maximum recovery, one can avoid employing in mapping languages features that are difficult to use in practice. This observation motivates the search for a class of queries that gives rise to an inverse notion meeting the requirements mentioned in the introduction, namely that every mapping specified by a set of st-tgds admits an inverse under this new notion, that this mapping can be expressed in a language that has the same good properties for data exchange as st-tgds, and that the language is closed under this notion of inverse. A natural starting point in this search is the class of conjunctive queries, as this class is widely used in practice and, in particular, has been extensively studied in the context of data exchange [7,12]. In fact, from the results in [4], it is straightforward to prove that every mapping specified by a set of st-tgds has a CQ-maximum recovery. Hence, it remains to show that the notion of CQ-maximum recovery admits a closed mapping language with good properties for

data exchange. In Sect. 5, we show that this is indeed the case. But before going into the details of this result, we show in Sect. 4 that the general notion of $\mathcal{C}$-maximum recovery is of independent interest, as it can be used to characterize the previous notions of inverse for schema mappings proposed in the literature [4,11,15]. Moreover, in Sect. 4 we also provide some theoretical tools that are fundamental to proving that the notion of CQ-maximum recovery admits a closed mapping language.

4 $\mathcal{C}$-Maximum recovery as a unifying framework for inverse operators

During the last few years, several different notions of inverse have been proposed for schema mappings, among them one can find Fagin-inverse [11], quasi-inverse [15], and maximum recovery [4]. The goal of this section is to establish the relationship between these notions and the concept of $\mathcal{C}$-maximum recovery. Specifically, we show that the notion of $\mathcal{C}$-maximum recovery provides a unified framework for the seemingly different notions of inverse previously proposed in the literature, thus improving the understanding of the problem of inverting schema mappings. We recall that in this paper we consider the typical data exchange setting in which source instances contain only constant values. Thus, we only compare with the notions of inverses proposed in this setting [4,11,15] and leave for future research the comparison with the notions proposed in the setting in which source instances can contain incomplete information [5,16].

We begin in Sects. 4.1 and 4.2 by showing that both Fagin-inverse and quasi-inverse can be characterized in terms of the notion of $\mathcal{C}$-maximum recovery, for some specific choices of the query language $\mathcal{C}$. Next, in Sect. 4.3, we formalize the intuitive idea that maximum recovery should be the best possible way to recover information given by any possible query, and show that, if $\mathcal{M}'$ is a maximum recovery of $\mathcal{M}$, then $\mathcal{M}'$ is an ALL-maximum recovery of $\mathcal{M}$, where ALL is the class of all queries. Finally, in Sect. 4.4, we show how the concept of $\mathcal{C}$-maximum recovery can be obtained from the notion of maximum recovery by taking into consideration the concept of schema mapping equivalence under a query language, which has been studied in some contexts such as schema mapping composition [24] and schema mapping optimization [13].

4.1 Fagin-inverse

We start our study by considering the notion of Fagin-inverse [11]. Roughly speaking, Fagin's definition is based on the idea that a mapping composed with its inverse should be equal to the identity schema mapping. To define this notion, Fagin first defines an *identity mapping* $\overline{\mathrm{Id}}_{\mathbf{R}}$ as $\{(I_1, I_2) \mid I_1, I_2$ instances of $\mathbf{R}$ and $I_1 \subseteq I_2\}$, which is an appropriate identity for the mappings specified by st-tgds [11]. Then given a mapping $\mathcal{M}$ from a schema $\mathbf{S}$ to a schema $\mathbf{T}$, a mapping $\mathcal{M}'$ from $\mathbf{T}$ to $\mathbf{S}$ is said to be a *Fagin-inverse* of $\mathcal{M}$ if $\mathcal{M} \circ \mathcal{M}' = \overline{\mathrm{Id}}_{\mathbf{S}}$.

In order to characterize the notion of Fagin-inverse in terms of the notion of $\mathcal{C}$-maximum recovery, we first introduce a concept that measures the ability of an inverse mapping to recover sound data according to a query language, we then characterize the notion of Fagin-inverse in terms of this concept, and we finally use this characterization to establish the relationship between Fagin-inverses and $\mathcal{C}$-maximum recoveries. More precisely, let $\mathcal{M}$ be a mapping from a schema $\mathbf{R}_1$ to a schema $\mathbf{R}_2$, $\mathcal{M}'$ a mapping from $\mathbf{R}_2$ to $\mathbf{R}_1$ and Q a query over $\mathbf{R}_1$. Then we say that $\mathcal{M}'$ *fully recovers* Q *for* $\mathcal{M}$ if for every instance I of $\mathbf{R}_1$, it holds that:

$$\mathrm{certain}_{\mathcal{M}\circ\mathcal{M}'}(Q, I) = Q(I).$$

That is, a mapping $\mathcal{M}'$ fully recovers a query Q for a mapping $\mathcal{M}$ if all data extracted by using Q can be recovered after translating the source information with $\mathcal{M}$ and then back with $\mathcal{M}'$. Moreover, given a class $\mathcal{C}$ of queries, we say that $\mathcal{M}'$ fully recovers $\mathcal{C}$ for $\mathcal{M}$ if for every query $Q \in \mathcal{C}$ over $\mathbf{R}_1$, it holds that $\mathcal{M}'$ fully recovers Q for $\mathcal{M}$.

The following proposition shows that the Fagin-inverses of a mapping $\mathcal{M}$ specified by a set of st-tgds exactly corresponds with the mappings that fully recover for $\mathcal{M}$ every union of conjunctive queries with inequalities.

Proposition 2 *Let $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \Sigma)$, where Σ is a set of st-tgds, and assume that $\mathcal{M}$ is Fagin-invertible. Then a mapping $\mathcal{M}'$ is a Fagin-inverse of $\mathcal{M}$ if and only if $\mathcal{M}'$ fully recovers* $\mathrm{UCQ}^{\neq}$ *for $\mathcal{M}$.*

Proof We start by showing the *only if* direction, that is, if $\mathcal{M}'$ is a Fagin-inverse of $\mathcal{M}$, then for every query Q in $\mathrm{UCQ}^{\neq}$ over $\mathbf{S}$, it holds that $\mathcal{M}'$ fully recovers Q for $\mathcal{M}$.

Let Q be a query in $\mathrm{UCQ}^{\neq}$ over $\mathbf{S}$ and I an instance of $\mathbf{S}$. We have to show that $\mathrm{certain}_{\mathcal{M}\circ\mathcal{M}'}(Q, I) = Q(I)$. Given that $\mathcal{M}'$ is a Fagin-inverse of $\mathcal{M}$, we have that $I \subseteq J$ for every $J \in \mathrm{Sol}_{\mathcal{M}\circ\mathcal{M}'}(I)$. Thus, given that Q is a monotone query, we have that $Q(I) \subseteq Q(J)$ for every $J \in \mathrm{Sol}_{\mathcal{M}\circ\mathcal{M}'}(I)$. It follows that $Q(I) \subseteq \mathrm{certain}_{\mathcal{M}\circ\mathcal{M}'}(Q, I)$ and, thus, $Q(I) = \mathrm{certain}_{\mathcal{M}\circ\mathcal{M}'}(Q, I)$ since $I \in \mathrm{Sol}_{\mathcal{M}\circ\mathcal{M}'}(I)$.

We show now the *if* direction, that is, if for every query Q in $\mathrm{UCQ}^{\neq}$ over $\mathbf{S}$, it holds that $\mathcal{M}'$ fully recovers Q for $\mathcal{M}$, then $\mathcal{M}'$ is a Fagin-inverse of $\mathcal{M}$. By the definition of Fagin-inverse, the latter hold if $\mathcal{M} \circ \mathcal{M}' = \overline{\mathrm{Id}}_{\mathbf{S}}$. Thus, to show that $\mathcal{M}'$ is a Fagin-inverse of $\mathcal{M}$, we prove that:

$(I, J) \in \mathcal{M} \circ \mathcal{M}'$ if and only if $I \subseteq J$.

($\Rightarrow$) Assume that $(I, J) \in \mathcal{M} \circ \mathcal{M}'$. We need to show that $I \subseteq J$. To this end, for every $R \in \mathbf{S}$ of arity k, let Q_R

be the identity query for table R, that is, $Q_R(x_1, \ldots x_k) = R(x_1, \ldots, x_k)$. Given that $\mathcal{M}'$ fully recovers each of these queries for $\mathcal{M}$, we have that $Q_R(I) = \text{certain}_{\mathcal{M}\circ\mathcal{M}'}(Q_R, I)$ for every $R \in \mathbf{S}$. Thus, we conclude that for every $K \in \text{Sol}_{\mathcal{M}\circ\mathcal{M}'}(I)$ and $R \in \mathbf{S}$, it holds that $Q_R(I) \subseteq Q_R(K)$, that is, $R^I \subseteq R^K$. Hence, we have that $I \subseteq J$ since $J \in \text{Sol}_{\mathcal{M}\circ\mathcal{M}'}(I)$.

($\Leftarrow$) Assume that $I \subseteq J$. To prove that $(I, J) \in \mathcal{M} \circ \mathcal{M}'$, we first show that $(J, J) \in \mathcal{M} \circ \mathcal{M}'$.

For the sake of contradiction, assume that $(J, J) \notin \mathcal{M} \circ \mathcal{M}'$. Then for every relation $R \in \mathbf{S}$, define a Boolean query Q_R as follows. Assuming that the arity of R is k and R^J contains n tuples, Q_R is the following query in UCQ$^{\neq}$:

$$\exists \bar{x}_1 \cdots \exists \bar{x}_n \exists \bar{x}_{n+1} \left[\left(\bigwedge_{1 \leq i \leq n+1} R(\bar{x}_i) \right) \wedge \left(\bigwedge_{1 \leq i < j \leq n+1} \bar{x}_i \neq \bar{x}_j \right) \right],$$

where $\bar{u} \neq \bar{v}$ stands for the formula $\bigvee_{\ell=1}^{k} u_i \neq v_i$, for k-tuples $\bar{u} = (u_1, \ldots, u_k)$ and $\bar{v} = (v_1, \ldots, v_k)$. Thus, Q_R says that relation R contains at least $n + 1$ tuples. Let Q be the following query in UCQ$^{\neq}$:

$$Q = \bigvee_{R \in \mathbf{S}} Q_R.$$

By the construction of Q, it is clear that $Q(J) = \underline{\text{false}}$.

By the direction ($\Rightarrow$), we know that if $(J, J') \in \mathcal{M} \circ \mathcal{M}'$, then $J \subseteq J'$ (notice that the proof of direction ($\Rightarrow$) was done for an arbitrary pair of instances in $\mathcal{M} \circ \mathcal{M}'$). Thus, we have that for every $J' \in \text{Sol}_{\mathcal{M}\circ\mathcal{M}'}(J)$, there exists $R \in \mathbf{S}$ such that $R^J \subsetneq R^{J'}$, given that $J \subseteq J'$ and $J \neq J'$ (since $(J, J) \notin \mathcal{M} \circ \mathcal{M}'$). We conclude that $Q(J') = \underline{\text{true}}$ for every $J' \in \text{Sol}_{\mathcal{M}\circ\mathcal{M}'}(J)$ and, hence, $\text{certain}_{\mathcal{M}\circ\mathcal{M}'}(Q, J) = \underline{\text{true}}$. But this contradicts the fact that $\mathcal{M}'$ fully recovers Q for $\mathcal{M}$ since $Q(J) = \underline{\text{false}}$ and, thus, $Q(J) \neq \text{certain}_{\mathcal{M}\circ\mathcal{M}'}(Q, J)$.

Given that $(J, J) \in \mathcal{M} \circ \mathcal{M}'$, we have that there exists an instance K of $\mathbf{T}$ such that $(J, K) \in \mathcal{M}$ and $(K, J) \in \mathcal{M}'$. Thus, given that $\mathcal{M}$ is specified by a set of st-tgds and $I \subseteq J$, we conclude that $(I, K) \in \mathcal{M}$. Hence, given that $(K, J) \in \mathcal{M}'$, we have that $(I, J) \in \mathcal{M} \circ \mathcal{M}'$. This concludes the proof of the *if* direction. □

Fagin et al. [16] introduce a notion similar to our notion of fully recoverable query when studying *extended inverses* in a setting in which null values are allowed to appear in source instances. In [16] the authors prove a weaker form of Proposition 2 stating that (extended) Fagin-inverses are capable of fully recovering all the class conjunctive queries (we refer the reader to Theorem 6.11 in [16] for a precise formulation). Our result is in a sense stronger as we are able to completely characterize Fagin-inverses as the mapping that fully recover the class UCQ$^{\neq}$.

We are now ready to state the main theorem of this section, that characterizes Fagin-inverses as UCQ$^{\neq}$-maximum recoveries.

Theorem 1 *Let $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \Sigma)$, where Σ is a set of st-tgds, and assume that $\mathcal{M}$ is Fagin-invertible. Then a mapping $\mathcal{M}'$ is a Fagin-inverse of $\mathcal{M}$ if and only if $\mathcal{M}'$ is a* UCQ$^{\neq}$*-maximum recovery of $\mathcal{M}$.*

Proof ($\Rightarrow$) If $\mathcal{M}'$ is a Fagin-inverse of $\mathcal{M}$, then by Proposition 2 we have that $\mathcal{M}'$ fully recovers UCQ$^{\neq}$ for $\mathcal{M}$, which immediately implies that $\mathcal{M}'$ is a UCQ$^{\neq}$-maximum recovery of $\mathcal{M}$.

($\Leftarrow$) Assume that $\mathcal{M}'$ is a UCQ$^{\neq}$-maximum recovery of $\mathcal{M}$. Given that $\mathcal{M}$ is Fagin-invertible, there exists a Fagin-inverse $\mathcal{M}^\star$ of $\mathcal{M}$. By Proposition 2, we have that $\mathcal{M}^\star$ fully recovers UCQ$^{\neq}$ for $\mathcal{M}$. Thus, we have that $\mathcal{M}^\star$ is a UCQ$^{\neq}$-recovery of $\mathcal{M}$ and, hence, we conclude, from the fact that $\mathcal{M}'$ is a UCQ$^{\neq}$-maximum recovery of $\mathcal{M}$, that for every query Q in UCQ$^{\neq}$ over $\mathbf{S}$ and every instance I of $\mathbf{S}$: $\text{certain}_{\mathcal{M}\circ\mathcal{M}^\star}(Q, I) \subseteq \text{certain}_{\mathcal{M}\circ\mathcal{M}'}(Q, I)$. Therefore, we deduce that $\mathcal{M}'$ fully recovers UCQ$^{\neq}$ for $\mathcal{M}$ from the fact that $\mathcal{M}^\star$ fully recovers UCQ$^{\neq}$ for $\mathcal{M}$. Hence, considering again Proposition 2, we deduce that $\mathcal{M}'$ is a Fagin-inverse of $\mathcal{M}$. □

4.2 Quasi-inverse

We continue our study by considering the notion of quasi-inverse [15]. The idea behind quasi-inverses is to relax the notion of Fagin-inverse by not differentiating between instances that are *data-exchange equivalent*. Two instances I_1, I_2 are data-exchange equivalent w.r.t. a mapping $\mathcal{M}$, denoted by $I_1 \sim_{\mathcal{M}} I_2$, if the space of solutions of I_1 under $\mathcal{M}$ coincides with the space of solutions of I_2 under $\mathcal{M}$, that is, if $\text{Sol}_{\mathcal{M}}(I_1) = \text{Sol}_{\mathcal{M}}(I_2)$ [15]. Given a mapping $\mathcal{M}_1$ from $\mathbf{S}$ to $\mathbf{S}$, mapping $\mathcal{M}_1[\sim_{\mathcal{M}}, \sim_{\mathcal{M}}]$ is defined as [15]:

$$\mathcal{M}_1[\sim_{\mathcal{M}}, \sim_{\mathcal{M}}] = \{(I_1, I_2) \mid \exists (I_1', I_2') \text{ such that } I_1 \sim_{\mathcal{M}} I_1', I_2 \sim_{\mathcal{M}} I_2' \text{ and } (I_1', I_2') \in \mathcal{M}_1\}$$

Then $\mathcal{M}'$ is a *quasi-inverse* of $\mathcal{M}$ if

$$(\mathcal{M} \circ \mathcal{M}')[\sim_{\mathcal{M}}, \sim_{\mathcal{M}}] = \overline{\text{Id}}_{\mathbf{S}}[\sim_{\mathcal{M}}, \sim_{\mathcal{M}}],$$

where $\overline{\text{Id}}_{\mathbf{S}}$ is the same identity mapping used in the definition of the notion of Fagin-inverse.

It turns out that the notion of quasi-inverse can be characterized in terms of the notion of $\mathcal{C}$-maximum recovery, just as for the case of Fagin-inverse. However, this characterization is slightly more technical, since now the choice of the query language $\mathcal{C}$ depends on the mapping that is being inverted. More specifically, given a mapping $\mathcal{M}$ specified by a set of st-tgds, we define next a set of unions of conjunctive queries with inequalities $\mathcal{C}_{\mathcal{M}}$ that depends on the dependencies that

specify $\mathcal{M}$, and then we show that the quasi-inverses of $\mathcal{M}$ correspond to the $\mathcal{C}_{\mathcal{M}}$-maximum recoveries of $\mathcal{M}$.

In order to define the class of queries $\mathcal{C}_{\mathcal{M}}$ for a given mapping $\mathcal{M}$, we need to recall some concepts regarding query rewriting. Assume that $\mathcal{M}$ is a mapping from a schema $\mathbf{S}$ to a schema $\mathbf{T}$, and that Q is a query over $\mathbf{T}$. We say that a query Q' over $\mathbf{S}$ is a *source rewriting of Q w.r.t. $\mathcal{M}$* if for every instance I of $\mathbf{S}$, the set $Q'(I)$ is exactly the set of certain answers of Q over I with respect to $\mathcal{M}$, that is, $Q'(I) = \text{certain}_{\mathcal{M}}(Q, I)$. It is known that if $\mathcal{M}$ is specified by a set of st-tgds and Q is a conjunctive query over the target schema, a source rewriting of Q always exists [1,29]. Moreover, it can be shown that in this case a source rewriting of Q can be expressed as a union of conjunctive queries with equality predicates (UCQ$^{=}$). As an example, consider a mapping given by the following st-tgds:

$$A(x, y) \rightarrow P(x, y),$$
$$B(x) \rightarrow P(x, x),$$

and let Q be the target query $P(x, y)$. Then a source rewriting of Q is given by the query:

$$A(x, y) \vee (B(x) \wedge x = y),$$

which is a query in UCQ$^{=}$. Notice that in this rewriting, we do need disjunction and the equality $x = y$.

Let $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \Sigma)$, where Σ is a set of st-tgds, and consider the following set of queries over $\mathbf{S}$:

$$\mathcal{P}_{\mathcal{M}} = \{\chi(\bar{x}) \mid \text{there exists } \varphi(\bar{x}) \rightarrow \psi(\bar{x}) \in \Sigma$$
such that $\chi(\bar{x})$ is a source rewriting of $\psi(\bar{x})$ w.r.t. $\mathcal{M}\}$.

Notice that $\mathcal{P}_{\mathcal{M}}$ is a set of queries in UCQ$^{=}$ since Σ is a set of st-tgds. Then define $\mathcal{C}_{\mathcal{M}}$ as the set of union of conjunctive queries with inequalities obtained by closing $\mathcal{P}_{\mathcal{M}}$ under conjunction, disjunction, existential quantification, variable substitution, and the use of inequalities between free variables.

Example 4 Consider again the mapping $\mathcal{M}$ given by st-tgds:

$$A(x, y) \rightarrow P(x, y),$$
$$B(x) \rightarrow P(x, x).$$

We have that

$$\mathcal{P}_{\mathcal{M}} = \{A(x, y) \vee (B(x) \wedge x = y),\ A(x, x) \vee B(x)\}$$

since $A(x, y) \vee (B(x) \wedge x = y)$ and $A(x, x) \vee B(x)$ are the source rewritings of $P(x, y)$ and $P(x, x)$ w.r.t. $\mathcal{M}$, respectively. Thus, the following are some of the queries in the set $\mathcal{C}_{\mathcal{M}}$:

$$A(x, y) \vee (B(x) \wedge x = y), \quad \exists y A(x, y) \vee B(x),$$
$$A(x, y) \wedge x \neq y, \quad \exists x \exists y\, (A(x, y) \wedge x \neq y),$$
$$\exists x A(x, x) \vee \exists y B(y)$$

As our first result regarding the notion of quasi-inverse, we show that the quasi-inverses of a mapping $\mathcal{M}$ specified by a set of st-tgds exactly corresponds with the mappings that fully recover $\mathcal{C}_{\mathcal{M}}$ for $\mathcal{M}$. Notice that this result is the analogue of Proposition 2.

Proposition 3 *Let $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \Sigma)$, where Σ is a set of st-tgds, and assume that $\mathcal{M}$ is quasi-invertible. Then a mapping $\mathcal{M}'$ is a quasi-inverse of $\mathcal{M}$ if and only if $\mathcal{M}'$ fully recovers $\mathcal{C}_{\mathcal{M}}$ for $\mathcal{M}$.*

The proof of Proposition 3 is given in electronic Appendix A.2. With this result, we are ready to state the main theorem of this section, that characterizes the quasi-inverses of a mapping $\mathcal{M}$ as $\mathcal{C}_{\mathcal{M}}$-maximum recoveries.

Theorem 2 *Let $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \Sigma)$, where Σ is a set of st-tgds, and assume that $\mathcal{M}$ is quasi-invertible. Then a mapping $\mathcal{M}'$ is a quasi-inverse of $\mathcal{M}$ if and only if $\mathcal{M}'$ is a $\mathcal{C}_{\mathcal{M}}$-maximum recovery of $\mathcal{M}$.*

The proof of Theorem 2 is given in electronic Appendix A.3.

4.3 Maximum recovery

We consider now the notion of maximum recovery proposed in [4]. In that paper, the authors follow a different approach to define a notion of inversion. In fact, the main goal of [4] is not to define a notion of inverse mapping, but instead to give a formal definition for what it means for a mapping $\mathcal{M}'$ to recover *sound information* with respect to a mapping $\mathcal{M}$. Such a mapping $\mathcal{M}'$ is called a recovery of $\mathcal{M}$ in [4]. In general, there may exist many possible recoveries for a given mapping $\mathcal{M}$. Given that, an order relation on recoveries is introduced in [4], and then it is shown that this naturally gives rise to the notion of maximum recovery, which is a mapping that brings back the maximum amount of sound information.

Let $\mathcal{M}$ be a mapping from a schema $\mathbf{R}_1$ to a schema $\mathbf{R}_2$, and $\text{Id}_{\mathbf{R}_1}$ the identity schema mapping over $\mathbf{R}_1$, that is, $\text{Id}_{\mathbf{R}_1} = \{(I, I) \mid I \text{ is an instance of } \mathbf{R}_1\}$. When trying to invert $\mathcal{M}$, the ideal would be to find a mapping $\mathcal{M}'$ from $\mathbf{R}_2$ to $\mathbf{R}_1$ such that $\mathcal{M} \circ \mathcal{M}' = \text{Id}_{\mathbf{R}_1}$. Unfortunately, in most cases this ideal is impossible to reach (for example, for the case of mappings specified by st-tgds [11]). If for a mapping $\mathcal{M}$, there is no mapping $\mathcal{M}_1$ such that $\mathcal{M} \circ \mathcal{M}_1 = \text{Id}_{\mathbf{R}_1}$, at least one would like to find a schema mapping $\mathcal{M}_2$ that does not forbid the possibility of recovering the initial source data. This gives rise to the notion of recovery proposed in [4]. Formally, given a mapping $\mathcal{M}$ from a schema $\mathbf{R}_1$ to a schema $\mathbf{R}_2$, a mapping $\mathcal{M}'$ from $\mathbf{R}_2$ to $\mathbf{R}_1$ is a *recovery* of $\mathcal{M}$ if $(I, I) \in \mathcal{M} \circ \mathcal{M}'$ for every instance I of $\mathbf{R}_1$ [4]. In general, if $\mathcal{M}'$ is a recovery of $\mathcal{M}$, then the smaller the space of solutions generated by $\mathcal{M} \circ \mathcal{M}'$, the more informative $\mathcal{M}'$ is about the initial source

instances. This naturally gives rise to the notion of maximum recovery; given a mapping $\mathcal{M}$ and a recovery $\mathcal{M}'$ of it, $\mathcal{M}'$ is said to be a *maximum recovery* of $\mathcal{M}$ if for every recovery $\mathcal{M}''$ of $\mathcal{M}$, it holds that $\mathcal{M} \circ \mathcal{M}' \subseteq \mathcal{M} \circ \mathcal{M}''$ [4].

The following theorem presents our main result for this section regarding the notion of maximum recovery. Part (1) of Theorem 3 shows that for every class of queries $\mathcal{C}$, if a mapping $\mathcal{M}'$ is a maximum recovery of $\mathcal{M}$, then $\mathcal{M}'$ is also a $\mathcal{C}$-maximum recovery of $\mathcal{M}$. Thus, a maximum recovery is the best possible alternative to retrieve sound information. In particular, one obtains as a corollary (part (2) of Theorem 3) that if $\mathcal{M}'$ is a maximum recovery of $\mathcal{M}$ and a query Q can be fully recovered for $\mathcal{M}$, then $\mathcal{M}'$ also fully recovers Q for $\mathcal{M}$.

Theorem 3 *Let $\mathcal{M}$ be a mapping from a schema $\mathbf{R}_1$ to a schema $\mathbf{R}_2$, $\mathcal{M}'$ a maximum recovery of $\mathcal{M}$ and Q an arbitrary query over $\mathbf{R}_1$.*

(1) *If $\mathcal{M}''$ recovers sound information for $\mathcal{M}$ under Q, then for every instance I of $\mathbf{R}_1$:*

$$\text{certain}_{\mathcal{M}\circ\mathcal{M}''}(Q, I) \subseteq \text{certain}_{\mathcal{M}\circ\mathcal{M}'}(Q, I) \subseteq Q(I).$$

(2) *If some mapping fully recovers Q for $\mathcal{M}$, then $\mathcal{M}'$ fully recovers Q for $\mathcal{M}$.*

The proof of this theorem can be found in electronic Appendix A.4. It is important to notice that from the previous theorem it is possible to conclude that if $\mathcal{M}'$ is a maximum recovery of $\mathcal{M}$, then $\mathcal{M}'$ is an ALL-maximum recovery of $\mathcal{M}$. The converse of this last statement does not hold, as it can be proved that there exist mappings $\mathcal{M}$ and $\mathcal{M}'$ such that $\mathcal{M}'$ is an ALL-maximum recovery of $\mathcal{M}$ but $\mathcal{M}'$ is not a maximum recovery of $\mathcal{M}$. The example that prove this fact is very involved and for the sake of the space we skip this construction here.

4.4 $\mathcal{C}$-maximum recovery, maximum recovery, and $\mathcal{C}$-equivalence

The idea of parameterizing a notion by a class of queries is not new in schema mappings management. In fact, this idea was developed by Madhavan and Halevy [24] to study the composition operator, and was also used by Fagin et al. [13] to develop a theory of schema mapping optimization. In particular, the authors of these papers considered the notion of *certain-answers equivalence* of mappings [13,24]. Let $\mathcal{C}$ be a class of queries. Then two mappings $\mathcal{M}$ and $\mathcal{M}'$ from a schema $\mathbf{R}_1$ to a schema $\mathbf{R}_2$ are $\mathcal{C}$-equivalent, denoted by $\mathcal{M} \equiv_{\mathcal{C}} \mathcal{M}'$, if for every query $Q \in \mathcal{C}$ over $\mathbf{R}_2$ and every instance I of $\mathbf{R}_1$: $\text{certain}_{\mathcal{M}}(Q, I) = \text{certain}_{\mathcal{M}'}(Q, I)$.

If $\mathcal{M}_1$ and $\mathcal{M}_2$ are $\mathcal{C}$-equivalent, then we know that they behave in the same way with respect to the queries in $\mathcal{C}$. Thus, if one is going to retrieve information by using only queries from $\mathcal{C}$, a mapping $\mathcal{M}$ can be replaced by any other $\mathcal{C}$-equivalent mapping. In particular, it can be replaced by a mapping that can be handled more efficiently, thus optimizing the initial schema mapping [13]. In the notion of $\mathcal{C}$-maximum recovery, this idea of only considering a particular query language to retrieve information is also present. The following result shows that the notions of maximum recovery and $\mathcal{C}$-maximum recovery can be related through the notion of $\mathcal{C}$-equivalence [13]. This result will play a central role when we present our algorithm to compute CQ-maximum recoveries in Sect. 5.

Proposition 4 *Let $\mathcal{M}$ be a mapping from a schema $\mathbf{R}_1$ to a schema $\mathbf{R}_2$, $\mathcal{M}'$, $\mathcal{M}''$ be mappings from $\mathbf{R}_2$ to $\mathbf{R}_1$, and assume that $\mathcal{M}'$ is a maximum recovery of $\mathcal{M}$.*

(1) *$\mathcal{M}''$ is a $\mathcal{C}$-maximum recovery of $\mathcal{M}$ iff $(\mathcal{M} \circ \mathcal{M}'') \equiv_{\mathcal{C}} (\mathcal{M} \circ \mathcal{M}')$.*
(2) *If $\mathcal{M}'' \equiv_{\mathcal{C}} \mathcal{M}'$, then $\mathcal{M}''$ is a $\mathcal{C}$-maximum recovery of $\mathcal{M}$.*

The proof of this proposition can be found in electronic Appendix A.5.

5 A schema mapping language closed under inversion

The main result of this section is that, when we consider the notion of CQ-maximum recovery as our semantics for inverting st-mappings, there exists a language that is closed under inversion, contains the class of st-tgds, and for which the standard *chase* procedure can be used to exchange data. The language corresponds to $\text{CQ}^{\mathbf{C},\neq}$- TO- CQ dependencies an extension of the st-tgds by allowing formulas in the premises to contain inequalities and a built-in predicate $\mathbf{C}(\cdot)$ to differentiate constants from null values. The fact that $\text{CQ}^{\mathbf{C},\neq}$- TO- CQ dependencies contain the class of st-tgds is obvious, and it is an easy observation that the standard *chase* procedure can be used to construct a *canonical universal solution* in the same way as it is done for st-tgds (see e.g., [12]). Thus, we devote the rest of the section in showing that our choice of CQ-maximum recoveries and $\text{CQ}^{\mathbf{C},\neq}$- TO- CQ dependencies satisfy the sought-after *closure* result. More specifically, we prove that every st-mapping specified by a set of $\text{CQ}^{\mathbf{C},\neq}$- TO- CQ dependencies has a CQ-maximum recovery also specified by a set of $\text{CQ}^{\mathbf{C},\neq}$- TO- CQ dependencies (Theorem 4). Although this language has appeared before in the literature about inverses of schema mappings [15,17], it has not been used to study closure properties such as the ones considered in this paper.

It should be noticed that our closure result (Theorem 4) is specific to the case of st-mappings, that is, mappings that consider only constant values in source instances. In this scenario inverses are ts-mappings which are mappings that transform instances with constant and null values into source instances that only contain constant values. This has been a common assumption on the literature on inverting mappings [4,11,15,17]. As we have mentioned, Fagin et al. [16] have recently raised the issue of the asymmetry in the study of the inverse operator, and have proposed to study the inverse operator in a symmetric scenario in which both source and target schemas have constant and null values. We leave the study of closure properties in this symmetric scenario for future work.

The following Theorem formalizes the closure result.

Theorem 4 *Every st-mapping specified by a set of* $\text{CQ}^{\mathbf{C},\neq}$- TO- CQ *dependencies, has a* CQ*-maximum recovery specified by a set of* $\text{CQ}^{\mathbf{C},\neq}$- TO- CQ *dependencies.*

To prove the theorem, we provide in this section an algorithm for computing a CQ-maximum recovery of an st-mapping specified by a set of $\text{CQ}^{\mathbf{C},\neq}$- TO- CQ dependencies (see Theorem 5), and whose output is a mapping specified in the same language. In order to simplify the exposition of the procedure, for the rest of this section fix a set Σ of $\text{CQ}^{\mathbf{C},\neq}$- TO- CQ dependencies from a source schema $\mathbf{S}$ to a target schema $\mathbf{T}$, and let $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \Sigma)$ be the st-mapping specified by Σ that is the input for our algorithm.

5.1 An overview of the algorithm

Our algorithm works as follows. We start by computing a maximum recovery $\mathcal{M}'$ of $\mathcal{M}$ by using the algorithm presented in [4]. Given that $\mathcal{M}'$ is a maximum recovery of $\mathcal{M}$, we have by Theorem 3 that $\mathcal{M}'$ is also a CQ-maximum recovery of $\mathcal{M}$. However, by the results in [4], we know that $\mathcal{M}'$ is specified by a set of $\text{CQ}^{\mathbf{C},\neq}$- TO- $\text{UCQ}^{=,\neq}$ dependencies, which have features beyond $\text{CQ}^{\mathbf{C},\neq}$- TO- CQ dependencies such as disjunctions, equalities, and inequalities in the conclusions of a rule. Thus, our algorithm performs a series of transformations on $\mathcal{M}'$ to eliminate these features while preserving conjunctive-query equivalence. To be more precise, we first use a procedure ELIMINATEEQINEQ that eliminates some equalities and inequalities from $\mathcal{M}'$ to produce a mapping $\mathcal{M}''$ that is also a maximum recovery of $\mathcal{M}$. Then, we use some graph-theoretical techniques to devise a procedure ELIMINATEDISJUNCTIONS that produces a mapping $\mathcal{M}^\star$ from $\mathcal{M}''$ such that $\mathcal{M}^\star \equiv_{\text{CQ}} \mathcal{M}''$ and the dependencies specifying $\mathcal{M}^\star$ do not include disjunctions in the conclusions. That is, the output of this last procedure is a set of $\text{CQ}^{\mathbf{C},\neq}$- TO- CQ dependencies. By using Proposition 4, we conclude that the mapping $\mathcal{M}^\star$ is also a CQ-maximum recovery of $\mathcal{M}$, thus obtaining our desired result.

5.2 A detailed description of the algorithm

We start with a simple observation. Consider the set $\overline{\Sigma}$ obtained from Σ by dropping all the atoms of the form $\mathbf{C}(x)$ that appear in the premises of the dependencies in Σ. Notice that, since $\mathcal{M}$ is a *source-to-target* mapping, every instance in the domain of $\mathcal{M}$ is composed only by elements in $\mathbf{C}$. This implies that the st-mapping specified by $\overline{\Sigma}$ is exactly $\mathcal{M}$ (that is, if $\overline{\mathcal{M}} = (\mathbf{S}, \mathbf{T}, \overline{\Sigma})$, then we have that for every instance I of $\mathbf{S}$: $\text{Sol}_{\mathcal{M}}(I) = \text{Sol}_{\overline{\mathcal{M}}}(I)$). Thus, in what follows we work with $\overline{\Sigma}$ instead of Σ.

We split the presentation of our algorithm in several subprocedures.

Computing a maximum recovery for $\overline{\Sigma}$

We start by computing a set Σ' that specifies a maximum recovery of our initial mapping $\mathcal{M}$. For this we use the algorithm MAXIMUMRECOVERY proposed in [4], which works for mappings specified by sets of FO- TO- CQ dependencies, with $\overline{\Sigma}$ as input.

Let $\overline{\mathcal{M}}$ be the st-mapping specified by $\overline{\Sigma}$ and $\mathcal{M}' = (\mathbf{T}, \mathbf{S}, \Sigma')$ the output of the call MAXIMUMRECOVERY($\overline{\mathcal{M}}$). Given that $\overline{\mathcal{M}}$ is specified by a set of $\text{CQ}^{\neq}$- TO- CQ dependencies, we have by the definition of algorithm MAXIMUMRECOVERY that Σ' is a set of $\text{CQ}^{\mathbf{C}}$- TO- $\text{UCQ}^{=,\neq}$ dependencies such that every dependency in Σ' is of the form

$$\exists \bar{y}\ \psi(\bar{x}, \bar{y}) \wedge \mathbf{C}(\bar{x}) \rightarrow \beta_1(\bar{x}) \vee \cdots \vee \beta_k(\bar{x}),$$

where $k \geq 1$ and

1. $\exists \bar{y}\ \psi(\bar{x}, \bar{y})$ is a query in CQ which is the conclusion of some of the dependencies in $\overline{\Sigma}$,
2. $\bar{x}$ is the tuple of free variables of $\exists \bar{y}\ \psi(\bar{x}, \bar{y})$ and of $\beta_i(\bar{x})$, for every $i \in \{1, \ldots, k\}$,
3. $\mathbf{C}(\bar{x})$ is a conjunction of formulas $\mathbf{C}(x)$ for every x in $\bar{x}$, and
4. every formula $\beta_i(\bar{x})$ is a query in $\text{CQ}^{=,\neq}$ such that:
   - if the inequality $z \neq z'$ occurs in $\beta_i(\bar{x})$, then z and z' occur in some relational atom of $\beta_i(\bar{x})$,
   - if the equality $z = z'$ occurs in $\beta_i(\bar{x})$, then z or z' (but not necessarily both) occur in some relational atom of $\beta_i(\bar{x})$.

Moreover, we can assume, without loss of generality, that for every $i \in \{1, \ldots, k\}$, the equalities occurring in the formula $\beta_i(\bar{x})$ are only among free variables (equalities among existentially quantified variables, or among free variables and existentially quantified variables, can be eliminated by the corresponding variable replacements).

In what follows, we show how disjunctions, equalities, and inequalities can be eliminated from the conclusions of the dependencies defining $\mathcal{M}'$, in such a way that the obtained mapping is a CQ-maximum recovery of $\mathcal{M}$.

Eliminating equalities and inequalities among free variables from the conclusions of Σ'

In this step, we construct a set Σ'' that defines a maximum recovery of $\mathcal{M}$, and such that the dependencies in Σ'' have neither equalities among free variables nor inequalities among free variables in their conclusions. To this end, we use a notion similar to what is called *complete description* in [15]. Let $\bar{x} = (x_1, \ldots, x_n)$ be a tuple of distinct variables. Consider a partition π of the set $\{x_1, \ldots, x_n\}$, and let $[x_i]_\pi$ be the equivalence class induced by π to which x_i belongs ($1 \leq i \leq n$). Let $f_\pi : \{x_1, \ldots, x_n\} \to \{x_1, \ldots, x_n\}$ be a function such that $f_\pi(x_i) = x_j$ if j is the minimum over all the indexes of the variables in $[x_i]_\pi$. That is, f_π is a function that selects a unique representative from every equivalence class induced by π. For example, if $\bar{x} = (x_1, x_2, x_3, x_4, x_5)$ and π is the partition $\{\{x_1, x_4\}, \{x_2, x_5\}, \{x_3\}\}$, then $f_\pi(x_1) = x_1$, $f_\pi(x_2) = x_2$, $f_\pi(x_3) = x_3$, $f_\pi(x_4) = x_1$, $f_\pi(x_5) = x_2$. We also consider the formula δ_π that is constructed by taking the conjunction of the inequalities $f_\pi(x_i) \neq f_\pi(x_j)$ whenever $f_\pi(x_i)$ and $f_\pi(x_j)$ are distinct variables. In the above example we have that δ_π is the formula $x_1 \neq x_2 \wedge x_1 \neq x_3 \wedge x_2 \neq x_3$. Finally, given a conjunction α of equalities and inequalities and a conjunction β of inequalities, we say α is *consistent* with β if there is an assignment of values to the variables in α and β that satisfies all the equalities and inequalities in these formulas. For example, $x_1 = x_2$ is consistent with $x_1 \neq x_3$, while $x_1 = x_2 \wedge x_2 = x_3$ is not consistent with $x_1 \neq x_3$.

Recall that Σ' is a set of $\text{CQ}^{\mathbf{C}}$- TO- $\text{UCQ}^{=,\neq}$ dependencies that specifies the mapping $\mathcal{M}'$, which is a maximum recovery of $\mathcal{M}$. At this point, we have the necessary ingredients to describe the procedure that constructs a set Σ'' from Σ' such that every inequality in the conclusion of a dependency in Σ'' includes at least one quantified variable, and the mapping specified by Σ'' is also a maximum recovery of $\mathcal{M}$. We call this procedure EliminateEqIneq. For a tuple $\bar{x}$ of variables, in the procedure we use $\mathbf{C}(\bar{x})$ to denote a conjunction of formulas $\mathbf{C}(x)$ for every x in $\bar{x}$.

Procedure EliminateEqIneq(Σ')

1. Let Σ'' be the empty set.
2. For every dependency σ in Σ' of the form

$$\exists \bar{y}\ \psi(\bar{x}, \bar{y}) \wedge \mathbf{C}(\bar{x}) \to \beta_1(\bar{x}) \vee \cdots \vee \beta_k(\bar{x})$$

with $\bar{x} = (x_1, \ldots, x_n)$ a tuple of distinct variables, and for every partition π of $\{x_1, \ldots, x_n\}$ do the following:

- Let $\alpha(\bar{x}) = \beta_1(\bar{x}) \vee \cdots \vee \beta_k(\bar{x})$ and $f_\pi(\bar{x}) = (f_\pi(x_1), \ldots, f_\pi(x_n))$
- Construct a formula γ from $\alpha(f_\pi(\bar{x}))$ as follows. For every $i \in \{1, \ldots, k\}$:
  - If the conjunction of equalities and inequalities among free variables in $\beta_i(f_\pi(\bar{x}))$ is consistent with δ_π, then drop the equalities and inequalities among free variables in $\beta_i(f_\pi(\bar{x}))$ and add the resulting formula as a disjunct in γ.
- If γ has at least one disjunct, then add to Σ'' the dependency σ_π given by formula

$$\exists \bar{y}\ \psi(f_\pi(\bar{x}), \bar{y}) \wedge \mathbf{C}(f_\pi(\bar{x})) \wedge \delta_\pi \to \gamma.$$

3. Return Σ'' □

For example, assume $\bar{x} = (x_1, x_2, x_3)$ and consider the following formulas:

$$\exists y\ \psi(\bar{x}, y) : \exists y\ (A(x_1, x_2, y) \wedge B(x_3, y))$$
$$\beta_1(\bar{x}) : \exists u\ (P(x_1, x_2, x_3, u) \wedge u \neq x_1) \wedge x_2 = x_3$$
$$\beta_2(\bar{x}) : R(x_2, x_3) \wedge x_1 = x_2 \wedge x_1 \neq x_3$$

Moreover, assume that $\sigma \in \Sigma'$ is the following dependency:

$$\exists y\ \psi(\bar{x}, y) \wedge \mathbf{C}(x_1) \wedge \mathbf{C}(x_2) \wedge \mathbf{C}(x_3) \to \beta_1(\bar{x}) \vee \beta_2(\bar{x}).$$

Consider a partition $\pi_1 = \{\{x_1, x_2\}, \{x_3\}\}$. Then we have that δ_{π_1} is the formula $x_1 \neq x_3$. Notice that the equality $x_2 = x_3$ in $\beta_1(\bar{x})$ becomes $x_1 = x_3$ after applying f_{π_1}, which is not consistent with δ_{π_1}. Considering now $\beta_2(\bar{x})$. In this case, the formula $x_1 = x_2 \wedge x_1 \neq x_3$ becomes $x_1 = x_1 \wedge x_1 \neq x_3$ after applying f_{π_1}, which is consistent with δ_{π_1}. Then to construct the conclusion of dependency σ_{π_1}, we just consider the formula $R(x_1, x_3)$, which is obtained from $\beta_2(f_{\pi_1}(\bar{x}))$ by dropping the equalities and inequalities. The premise of σ_{π_1} is constructed as the conjunction of $\exists y\ \psi(f_{\pi_1}(\bar{x}), y)$, $\mathbf{C}(f_{\pi_1}(\bar{x}))$ and δ_{π_1}. That is, we have that σ_{π_1} is the formula

$$\exists y\ (A(x_1, x_1, y) \wedge B(x_3, y)) \wedge \mathbf{C}(x_1) \wedge \mathbf{C}(x_3) \wedge x_1 \neq x_3 \to R(x_1, x_3),$$

which is added to Σ'' by procedure EliminateEqIneq. Consider now the partition $\pi_2 = \{\{x_1\}, \{x_2, x_3\}\}$. Since $f_{\pi_2}(x_3) = x_2$, the conjunction of equalities and inequalities in $\beta_2(f_{\pi_2}(\bar{x}))$ is the unsatisfiable formula $x_1 = x_2 \wedge x_1 \neq x_2$, which is not consistent with δ_{π_2}. On the other hand, the equalities and inequalities of $\beta_1(f_{\pi_2}(\bar{x}))$ are consistent with δ_{π_2}. Thus, in this case we have that σ_{π_2} is the dependency

$$\exists y\ (A(x_1, x_2, y) \wedge B(x_2, y)) \wedge \mathbf{C}(x_1) \wedge \mathbf{C}(x_2) \wedge x_1 \neq x_2 \to \exists u\ (P(x_1, x_2, x_2, u) \wedge u \neq x_1),$$

and σ_{π_2} is added to Σ''. If we now consider partition $\pi_3 = \{\{x_1\}, \{x_2\}, \{x_3\}\}$, then no dependency is added since neither $\beta_1(f_{\pi_3}(\bar{x}))$ nor $\beta_2(f_{\pi_3}(\bar{x}))$ is consistent with δ_{π_3}.

Notice that the set Σ'' obtained after the described process is a set of $\text{CQ}^{\mathbf{C},\neq}$- TO- $\text{UCQ}^{\neq}$ dependencies, that also satisfies that for every disjunct $\beta(\bar{x})$ in the conclusion of a

dependency in Σ', and for every inequality $x \neq x'$ occurring in $\beta(\bar{x})$, it holds that x or x' is existentially quantified. But more importantly, Σ'' satisfies the following key property for our algorithm.

Lemma 1 *Let $\mathcal{M}''$ be the ts-mapping specified by the set Σ'' returned by* EliminateEqIneq(Σ'). *Then $\mathcal{M}''$ is a maximum recovery of $\mathcal{M}$.*

The proof of Lemma 1 is rather technical and can be found in electronic Appendix A.6.

We continue now with the description of the procedure to compute a CQ-maximum recovery of $\mathcal{M}$. In what follows, we assume that Σ'' is the set constructed from Σ' as described above, and that $\mathcal{M}''$ is the ts-mapping specified by Σ''.

Eliminating the inequalities in the conclusions of Σ''

In this step, we just drop all the remaining inequalities in the disjunctions of the conclusions of the dependencies of Σ''. It turns out that, although the obtained set of dependencies may no longer define a maximum recovery of $\mathcal{M}$, it does define a CQ-maximum recovery of $\mathcal{M}$. In fact, a stronger result holds, namely that the obtained set of dependencies defines a UCQ-maximum recovery of $\mathcal{M}$.

From now on, assume that Σ''' is the set obtained from Σ'' by dropping all the inequalities in the conclusions of the dependencies in Σ'', and that $\mathcal{M}'''$ is the ts-mapping specified by Σ'''. Then we have that:

Lemma 2 *$\mathcal{M}''' \equiv_{\text{UCQ}} \mathcal{M}''$ and $\mathcal{M}'''$ is a* UCQ*-maximum recovery of $\mathcal{M}$.*

The proof of this lemma can be found in electronic Appendix A.7.

Before going to the last step of our procedure, it is worth recalling how the dependencies in Σ''' look. Every element of Σ''' is a dependency of the form:

$$\varphi(\bar{x}) \wedge \mathbf{C}(\bar{x}) \wedge \delta(\bar{x}) \rightarrow \beta_1(\bar{x}) \vee \cdots \vee \beta_k(\bar{x}),$$

where

1. $\bar{x}$ is a tuple of distinct variables,
2. $\varphi(\bar{x})$ and $\beta_i(\bar{x})$ $(1 \leq i \leq k)$ are conjunctive queries with $\bar{x}$ as their tuple of free variables,
3. $\mathbf{C}(\bar{x})$ is a conjunction of formulas $\mathbf{C}(x)$ for every variable x in $\bar{x}$, and
4. $\delta(\bar{x})$ is a conjunction of inequalities $x \neq x'$ for every pair of distinct variables x, x' in $\bar{x}$.

In the last step of our algorithm, we eliminate the disjunctions from the conclusions of the dependencies of Σ''', to obtain a set of $\text{CQ}^{\mathbf{C},\neq}$- to- CQ dependencies that specifies a CQ-maximum recovery of $\mathcal{M}$.

Eliminating the disjunctions in the conclusions of Σ'''

We explain first the machinery needed in this step of our algorithm. We borrow some notions and tools from graph theory, in particular, properties about graph homomorphisms.

Given two instances J_1 and J_2 composed by constants and null values, we define the *product* of J_1 and J_2, denoted by $J_1 \times J_2$, as the instance constructed by the following procedure. Consider an injective function $f : (\mathbf{C} \cup \mathbf{N}) \times (\mathbf{C} \cup \mathbf{N}) \rightarrow (\mathbf{C} \cup \mathbf{N})$ such that (1) $f(a, a) = a$ for every constant value $a \in \mathbf{C}$, and (2) $f(a, b) = n_{(a,b)}$ is a null value, whenever a or b is a null value, or a and b are distinct constant values. Then for every $k$ary relation symbol R and every pair of facts $R(a_1, \ldots, a_k)$ in J_1 and $R(b_1, \ldots, b_k)$ in J_2, the fact $R(f(a_1, b_1), \ldots, f(a_k, b_k))$ is included in the instance $J_1 \times J_2$. For example, consider the instances

$$J_1 = \{P(a, b), R(n_1, a), R(n_1, c)\},$$
$$J_2 = \{P(a, c), R(n_2, a), R(n_2, c)\},$$

where a, b, c are distinct constant values, and n_1, n_2 are distinct null values. Then $J_1 \times J_2$ is the instance

$$\{P(a, m_1), R(m_2, a), R(m_2, c), R(m_2, m_3), R(m_2, m_4)\},$$

where m_1, m_2, m_3, m_4 are distinct null values. In this case, we have used a function f such that $f(b, c) = m_1$, $f(n_1, n_2) = m_2$, $f(a, c) = m_3$ and $f(c, a) = m_4$. Notice that the product of two instances could be the empty instance. For example, if we consider $J_1 = \{P(a, b)\}$ and $J_2 = \{R(a, b)\}$, then $J_1 \times J_2$ is the empty instance.

If we consider a schema with a single binary relation and two instances J_1 and J_2 composed only by null values, the product $J_1 \times J_2$ corresponds to the graph-theoretical Cartesian product [21]. As for graph products, the operation $\times$ between instances satisfies several algebraic properties. One of the most important properties is the following.

Lemma 3 (c.f. [21]) *Let J_1, J_2 be instances composed by constant and null values.*

(1) *There exists a homomorphism from $J_1 \times J_2$ to J_1, and a homomorphism from $J_1 \times J_2$ to J_2.*
(2) *If there exists a homomorphism from J to J_1 and a homomorphism from J to J_2, then there exists a homomorphism from J to $J_1 \times J_2$.*

The above lemma intuitively states that from the space of all possible instances, $J_1 \times J_2$ is the *closest instance* to both J_1 and J_2, taking homomorphisms as our proximity criterion. Since the answering process of conjunctive queries can be characterized in terms of homomorphisms [10], this property gives us the following intuition. If a tuple $\bar{t}$ is an answer

to a conjunctive query Q over J_1 and also over J_2, then $\bar{t}$ should be an answer to Q over $J_1 \times J_2$. And, conversely, if $\bar{t}$ is an answer of Q over $J_1 \times J_2$, then it should be an answer to Q over J_1 and over J_2. This is one of the key ingredients for the last step of our algorithm, which is presented in this section.

The notion of product of instances can also be applied to conjunctive queries. Let Q_1 and Q_2 be two $n$ary conjunctive queries, and assume that $\bar{x}$ is the tuple of free variables of Q_1 and Q_2. The *product* of Q_1 and Q_2, denoted by $Q_1 \times Q_2$, is defined as a $k$ary conjunctive query (with $k \leq n$) constructed as follows. Let $f(\cdot, \cdot)$ be a one-to-one mapping from pairs of variables to variables such that:

1. $f(x, x) = x$ for every variable x in $\bar{x}$, and
2. $f(y, z)$ is a fresh variable (mentioned neither in Q_1 nor in Q_2) in any other case.

Then for every atoms $R(y_1, \ldots, y_m)$ in Q_1 and $R(z_1, \ldots, z_m)$ in Q_2, the atom $R(f(y_1, z_1), \ldots, f(y_m, z_m))$ is included as a conjunct in the query $Q_1 \times Q_2$. Furthermore, the set of free variables of $Q_1 \times Q_2$ is the set of variables from $\bar{x}$ that are mentioned in $Q_1 \times Q_2$. For example, consider conjunctive queries:

$$Q_1(x_1, x_2) : P(x_1, x_2) \wedge R(x_2, x_1)$$
$$Q_2(x_1, x_2) : \exists y\,(P(x_1, y) \wedge R(y, x_2)).$$

Then we have that $Q_1 \times Q_2$ is the following conjunctive query (with a single free variable x_1):

$$(Q_1 \times Q_2)(x_1) : \exists z_1 \exists z_2\,(P(x_1, z_1) \wedge R(z_1, z_2)).$$

In this case, we used a mapping f such that $f(x_1, x_1) = x_1$, $f(x_2, y) = z_1$ and $f(x_1, x_2) = z_2$. As shown in the example, the free variables of $Q_1 \times Q_2$ do not necessarily coincide with the free variables of Q_1 and Q_2. Notice that the product of two queries may be empty. For example, if Q_1 is the query $\exists y_1 \exists y_2 P(y_1, y_2)$ and Q_2 is the query $\exists z_1 R(z_1, z_1)$, then $Q_1 \times Q_2$ is empty.

We finally have all the necessary ingredients to construct a set of dependencies $\Sigma^\star$ from Σ''', such that $\Sigma^\star$ defines a CQ-maximum recovery of $\mathcal{M}$ and the dependencies in $\Sigma^\star$ do not have disjunctions in their conclusions.

Procedure ELIMINATEDISJUNCTIONS(Σ''')

1. Let $\Sigma^\star$ be empty.
2. For every dependency in Σ''' of the form

$$\varphi(\bar{x}) \wedge \mathbf{C}(\bar{x}) \wedge \delta(\bar{x}) \rightarrow \beta_1(\bar{x}) \vee \cdots \vee \beta_k(\bar{x}),$$

do the following. If $\beta_1(\bar{x}) \times \cdots \times \beta_k(\bar{x})$ is not empty, then add to $\Sigma^\star$ the dependency

$$\varphi(\bar{x}) \wedge \mathbf{C}(\bar{x}) \wedge \delta(\bar{x}) \rightarrow \beta_1(\bar{x}) \times \cdots \times \beta_k(\bar{x}).$$

3. Return $\Sigma^\star$. □

For example, assume that Σ''' contains the dependency

$$A(x_1, x_2) \wedge \mathbf{C}(x_1) \wedge \mathbf{C}(x_2) \wedge x_1 \neq x_2 \rightarrow [R(x_1, x_2) \wedge R(x_1, x_1)] \vee [\exists y(P(x_1, y) \wedge R(x_2, x_2))]. \quad (8)$$

Then we add to $\Sigma^\star$ the dependency

$$A(x_1, x_2) \wedge \mathbf{C}(x_1) \wedge \mathbf{C}(x_2) \wedge x_1 \neq x_2 \rightarrow \exists z(R(z, x_2) \wedge R(z, z)) \quad (9)$$

since $\exists z(R(z, x_2) \wedge R(z, z))$ is the result of

$$[R(x_1, x_2) \wedge R(x_1, x_1)] \times [\exists y(P(x_1, y) \wedge R(x_2, x_2))].$$

Notice that the set $\Sigma^\star$ obtained as output of the above procedure is a set of $\text{CQ}^{\mathbf{C},\neq}$- TO- CQ dependencies. The following lemma shows the key property of $\Sigma^\star$.

Lemma 4 *Let $\mathcal{M}'''$ be the ts-mapping specified by Σ''' and $\mathcal{M}^\star$ the ts-mapping specified by $\Sigma^\star$, where $\Sigma^\star$ is the set of* $\text{CQ}^{\mathbf{C},\neq}$- TO- CQ *dependencies obtained as the result of the call* ELIMINATEDISJUNCTIONS(Σ'''). *Then $\mathcal{M}^\star$ is* CQ*-equivalent with $\mathcal{M}'''$.*

We give some intuition of why this result holds (the complete proof of the lemma can be found in electronic Appendix A.8). Consider a set Γ_1 containing dependency (8), and a set Γ_2 containing dependency (9). One of the main steps in the proof of the lemma is based on computing the chase with dependencies containing disjunctions in the conclusions (like dependency (8)). In particular, if we want to chase an instance with the set Γ_1, we need to consider the *disjunctive chase* [15]. As in the classical (non-disjunctive) chase, to apply a step of the disjunctive chase we need to select a particular dependency of the form $\alpha \rightarrow \beta_1 \vee \cdots \vee \beta_k$ in the set, but in this case we also need to select a particular disjunct β_i in the conclusion of the dependency, and then apply a standard chase step for dependency $\alpha \rightarrow \beta_i$. The result of the disjunctive chase is the set of all possible instances that are obtained by considering all possible choices of disjuncts in the conclusions (we include a detailed definition of the disjunctive chase in electronic Appendix A.6).

Consider now the instance $J = \{A(a, b)\}$, with a, b distinct constant values. If we consider the disjunctive chase of instance J with set Γ_1, we obtain a set $\mathcal{V}$ consisting of instances $K_1 = \{R(a, b), R(a, a)\}$ and $K_2 = \{P(a, n), R(b, b)\}$, where n is a null value. By the properties of the disjunctive chase [15], we know that every solution

K of J under Γ_1 is such that there exists a homomorphism from K_1 to K, or there exists a homomorphism from K_2 to K. Thus, when considering the *conjunctive information* contained in the space of solutions of J under Γ_1, we certainly know that the value b appears in the second component of relation R, and that some element appears in a single tuple in both components of R. Notice that the conjunctive information contained in $\mathcal{V}$ is captured by the instance $K_1 \times K_2 = \{R(n', b), R(n', n')\}$, where n' is a null value. If we now chase J with Γ_2, we obtain the instance $K = \{R(m, b), R(m, m)\}$ with m a null value, which is homomorphically equivalent to $K_1 \times K_2$. From this, it can be formally proved that for every conjunctive query Q, the certain answers of Q under Γ_1 coincides with the certain answers of Q under Γ_2.

The strategy in the proof of Lemma 4 is a generalization of the above argument. Let Σ''' and $\Sigma^\star$ be the sets of dependencies in the statement of the lemma and J an arbitrary source instance. Recall that Σ''' is a set of $\text{CQ}^{\mathbf{C},\neq}$- TO- UCQ dependencies, thus the result of the (disjunctive) chase of J with Σ''' is a set of instances $\mathcal{V} = \{K_1, \ldots, K_\ell\}$. Now let K be the result of chasing J with $\Sigma^\star$. The key ingredient in the proof of Lemma 4 is the fact that K and $K_1 \times \cdots \times K_\ell$ are homomorphically equivalent. From this and the properties of the product of instances, it can be shown that for every conjunctive query Q the certain answers of Q under $\Sigma^\star$ coincide with the certain answers of Q under Σ'''.

Putting it all together

The following is the complete algorithm that uses all the previous procedures to compute CQ-maximum recoveries of st-mappings specified by $\text{CQ}^{\mathbf{C},\neq}$- TO- CQ dependencies.

Algorithm CQ- MAXIMUMRECOVERY($\mathcal{M}$)
Input: An st-mapping $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \Sigma)$, where Σ is a set of $\text{CQ}^{\mathbf{C},\neq}$- TO- CQ dependencies.
Output: A ts-mapping $\mathcal{M}^\star = (\mathbf{T}, \mathbf{S}, \Sigma^\star)$, where $\Sigma^\star$ is a set of $\text{CQ}^{\mathbf{C},\neq}$- TO- CQ dependencies such that $\mathcal{M}^\star$ is a CQ-maximum recovery of $\mathcal{M}$.

1. Let $\overline{\Sigma}$ be the set of dependencies obtained from Σ by dropping all the atoms of the form $\mathbf{C}(x)$ that appear in the premises of Σ, and let $\overline{\mathcal{M}} = (\mathbf{S}, \mathbf{T}, \overline{\Sigma})$.
2. Let $\mathcal{M}' = (\mathbf{T}, \mathbf{S}, \Sigma')$ be the ts-mapping obtained as the output of algorithm MAXIMUMRECOVERY($\overline{\mathcal{M}}$) [4].
3. Let Σ'' be the set of dependencies obtained as the output of ELIMINATEEQINEQ(Σ').
4. Let Σ''' be the set obtained from Σ'' by dropping all the inequalities that appear in the conclusions of the dependencies in Σ''.
5. Let $\Sigma^\star$ be the set of dependencies obtained as the output of ELIMINATEDISJUNCTIONS(Σ''').
6. Return $\mathcal{M}^\star = (\mathbf{T}, \mathbf{S}, \Sigma^\star)$. □

Theorem 5 *Let $\mathcal{M}$ be an st-mapping specified by a set of* $\text{CQ}^{\mathbf{C},\neq}$- TO- CQ *dependencies. Then algorithm* CQ- MAXIMUMRECOVERY($\mathcal{M}$) *computes a* CQ*-maximum recovery of $\mathcal{M}$ specified by set of* $\text{CQ}^{\mathbf{C},\neq}$- TO- CQ *dependencies.*

Proof From Lemma 1, we have that the mapping $\mathcal{M}''$ specified by the set of $\text{CQ}^{\mathbf{C},\neq}$- TO- $\text{UCQ}^{\neq}$ dependencies Σ'' (computed in step 3) is a maximum recovery of $\mathcal{M}$. Moreover, we know from Lemmas 2 and 4 that the mapping $\mathcal{M}^\star$ specified by the set of $\text{CQ}^{\mathbf{C},\neq}$- TO- CQ dependencies $\Sigma^\star$ (computed in step 5) is CQ-equivalent to $\mathcal{M}''$. Thus, we conclude from Proposition 4 (2) that $\mathcal{M}^\star$ is a CQ-maximum recovery of $\mathcal{M}$, from which the theorem follows. □

We conclude this section by pointing out that the closure property stated in Theorem 4 follows directly from Theorem 5.

6 Optimality of the closure result

The closure result presented in the previous section depends on both the mapping language and the class $\mathcal{C}$ used in the notion of $\mathcal{C}$-maximum recovery. Thus, a natural question is whether this result could be strengthened by considering other alternatives for these parameters. In Sects. 6.1 and 6.2, we prove several negative results in this respect. These results show that our choice of CQ-maximum recovery as the semantics for inversion and $\text{CQ}^{\mathbf{C},\neq}$- TO- CQ as the mapping language is, in a technical sense, optimal for obtaining a mapping language closed under inversion.

6.1 $\text{CQ}^{\mathbf{C},\neq}$- TO- CQ is the right language

Most of the dependencies considered in the data exchange literature [4,11,12,15] are $\mathcal{L}_1$- TO- $\mathcal{L}_2$ dependencies, where $\mathcal{L}_1$ and $\mathcal{L}_2$ are fragments of $\text{UCQ}^{\mathbf{C},\neq}$. In this section, we show that among all of them, $\text{CQ}^{\mathbf{C},\neq}$- TO- CQ is the *right* language for the notion of CQ-maximum recovery; if one adds or removes features from this class of dependencies, then closure under CQ-maximum recovery no longer holds.

We start by showing that both inequalities and predicate $\mathbf{C}(\cdot)$ are necessary for the closure property in Theorem 4.

Theorem 6

(1) *There exists an st-mapping specified by a set of st-tgds that has no* CQ*-maximum recovery specified by a set of* $\text{CQ}^{\mathbf{C}}$- TO- CQ *dependencies.*
(2) *There exists an st-mapping specified by a set of st-tgds that has no* CQ*-maximum recovery specified by a set of* $\text{CQ}^{\neq}$- TO- CQ *dependencies.*

Proof We show here mappings that satisfy conditions (1) and (2), and give some intuition on why they satisfy these properties. The complete proof of the theorem can be found in Appendix A.9.

(1) Consider a mapping $\mathcal{M}$ specified by st-tgds:

$$A(x, y) \rightarrow P(x, y),$$
$$B(x, x) \rightarrow P(x, x).$$

Intuitively, in this case the only *conjunctive information* that one can recover are the tuples coming from A whose elements are distinct, as a fact of the form $P(a, a)$ can be generated by any of the two rules defining the mapping. To specify this we need inequalities. In fact, a CQ-maximum recovery for $\mathcal{M}$ is the mapping specified by:

$$P(x, y) \wedge x \neq y \rightarrow A(x, y).$$

In Appendix A.9, it is proved that $\mathcal{M}$ does not have a CQ-maximum recovery specified by a set of $\text{CQ}^{\mathbf{C}}$- TO- CQ dependencies.

(2) Consider a mapping $\mathcal{M}$ specified by st-tgds:

$$A(x) \rightarrow \exists y P(y),$$
$$B(x) \rightarrow P(x).$$

Intuitively, predicate $\mathbf{C}(\cdot)$ is needed in a CQ-maximum recovery of $\mathcal{M}$ to discriminate whether a value comes from relation B in the source. In fact, a CQ-maximum recovery for $\mathcal{M}$ is the mapping specified by:

$$P(x) \wedge \mathbf{C}(x) \rightarrow B(x).$$

In Appendix A.9, it is proved that $\mathcal{M}$ does not have a CQ-maximum recovery specified by a set of $\text{CQ}^{\neq}$- TO- CQ dependencies. □

We have shown that both inequalities and predicate $\mathbf{C}(\cdot)$ are necessary for the closure property of Theorem 4. A natural question at this point is whether a similar closure property can be obtained if one adds some extra features to $\text{CQ}^{\mathbf{C},\neq}$- TO- CQ. For example, it could be the case that the language of $\text{CQ}^{\mathbf{C},\neq}$- TO- $\text{CQ}^{\neq}$ dependencies is closed under CQ-maximum recovery. Unfortunately, the following proposition shows that if one includes disjunctions or inequalities in the conclusions, then mappings do not necessarily admit CQ-maximum recoveries, even if these features are added to st-tgds.

Proposition 5 *There exist st-mappings specified by sets of* (1) CQ- TO- UCQ *dependencies,* (2) CQ- TO- $\text{CQ}^{\neq}$ *dependencies, that have no* CQ*-maximum recoveries.*

Proof We just present a sketch of the proof (the complete proof can be found in electronic Appendix A.10). For the case (1), we use an st-mapping $\mathcal{M}$ specified by the following set of CQ- TO- UCQ dependencies:

$$B(x) \wedge C_1(x) \rightarrow R_1(x),$$
$$B(x) \wedge C_2(x) \rightarrow R_2(x),$$
$$A(x) \rightarrow R_1(x) \vee R_2(x).$$

This mapping has as CQ-recoveries a mapping $\mathcal{M}_1$ specified by $R_1(x) \rightarrow C_1(x) \wedge B(x)$, and a mapping $\mathcal{M}_2$ specified by $R_2(x) \rightarrow C_2(x) \wedge B(x)$. Given that neither $\mathcal{M}_1$ is better than $\mathcal{M}_2$ nor $\mathcal{M}_2$ is better than $\mathcal{M}_1$ as a CQ-recovery of $\mathcal{M}$, one could try to find a mapping $\mathcal{M}_3$ which is better than both by considering the two dependencies $R_1(x) \rightarrow C_1(x) \wedge B(x)$, $R_2(x) \rightarrow C_2(x) \wedge B(x)$ together. Although it seems that $\mathcal{M}_3$ is more informative than both $\mathcal{M}_1$ and $\mathcal{M}_2$, the problem is that $\mathcal{M}_3$ is not a CQ-recovery of $\mathcal{M}$. To see why this is the case, notice that for the source instance $I = \{A(a)\}$, we have that $B(a)$ belongs to every possible solution for I under $\mathcal{M} \circ \mathcal{M}_3$. Thus, for the conjunctive query Q given by $\exists x\, B(x)$, we have that $Q(I) = \underline{\text{false}}$ but $\text{certain}_{\mathcal{M}\circ\mathcal{M}_3}(Q, I) = \underline{\text{true}}$, which shows that $\mathcal{M}_3$ does not recover sound information according to Q. In fact, this is a general phenomenon, as it can be proved that there is no mapping which is a CQ-recovery of $\mathcal{M}$ and is better than any other mapping in terms of its ability to recover sound information for $\mathcal{M}$ according to the language of conjunctive queries. Therefore, one concludes that $\mathcal{M}$ does not have a CQ-maximum recovery.

Now for the case (2), consider an st-mappings given by the following CQ- TO- $\text{CQ}^{\neq}$ dependencies:

$$B(x) \wedge C_1(x) \rightarrow R(x, x),$$
$$B(x) \wedge C_2(x) \rightarrow \exists y(R(x, y) \wedge x \neq y),$$
$$A(x) \rightarrow \exists y R(x, y).$$

In this case one can provide a similar argument as in (1). Consider, for example, a mapping $\mathcal{M}_1$ specified by dependency $R(x, x) \rightarrow C_1(x) \wedge B(x)$, and a mapping $\mathcal{M}_2$ specified by dependency $R(x, y) \wedge x \neq y \rightarrow C_2(x) \wedge B(x)$. Both are CQ-recoveries of $\mathcal{M}$, but neither is better than the other. If one tries to improve these mappings by considering a mapping $\mathcal{M}_3$ defined by both dependencies $R(x, x) \rightarrow B(x) \wedge C_1(x)$, $R(x, y) \wedge x \neq y \rightarrow B(x) \wedge C_2(x)$ together, then one can show that the resulting mapping is not a CQ-recovery of $\mathcal{M}$. To see why this is the case, just consider the source instance $\{A(1)\}$ and the query $\exists x\, B(x)$ as in case (1). □

6.2 CQ-maximum recovery is the right notion

In this section, we consider several alternatives to CQ for the semantics of inverse operators, and show that none of them is appropriate to obtain a closure property as in Theorem 4.

We start with the notion of UCQ-maximum recovery. The following result shows that to express the UCQ-maximum recovery of a mapping given by a set of st-tgds, one needs

dependencies with disjunctions in their conclusions even if the full power of FO is allowed in the premises of the dependencies.

Proposition 6 *There exists an st-mapping $\mathcal{M}$ specified by a set of st-tgds such that:*

(a) *$\mathcal{M}$ has a* UCQ*-maximum recovery specified by a set of* CQ- TO- UCQ *dependencies.*
(b) *$\mathcal{M}$ does not have a* UCQ*-maximum recovery specified by a set of* $\text{FO}^{\mathbf{C}}$- TO- CQ *dependencies.*

Proof Consider a source schema $\mathbf{S} = \{A(\cdot), B(\cdot)\}$, a target schema $\mathbf{T} = \{T(\cdot)\}$, and the st-mapping $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \Sigma)$, where Σ consists of the following st-tgds:

$$A(x) \rightarrow T(x),$$
$$B(x) \rightarrow T(x).$$

Let $\mathcal{M}^\star = (\mathbf{T}, \mathbf{S}, \Sigma^\star)$ be a ts-mapping specified by the CQ- TO- UCQ dependency

$$T(x) \rightarrow A(x) \vee B(x). \tag{10}$$

By using the tools developed in [4], it is straightforward to show that $\mathcal{M}^\star$ is a maximum recovery of $\mathcal{M}$, which implies that $\mathcal{M}^\star$ is a UCQ-maximum recovery of $\mathcal{M}$. In order to prove that the disjunction in (10) is essential to obtain a UCQ-maximum recovery of $\mathcal{M}$, we show that every mapping specified by a set of tgds without disjunctions in the conclusions is strictly less informative than $\mathcal{M}^\star$, even if one allows the full power of FO in the premises of dependencies. The formalization of this argument can be found in Appendix A.11 □

Proposition 5 shows that there exist mappings specified by CQ- TO- UCQ dependencies that have no CQ-maximum recoveries and, hence, have no UCQ-maximum recoveries. Thus, Propositions 5 and 6 show that if we use UCQ-maximum recovery as our notion of inverse, then we are doomed to failure.

Now we consider the notion of $\text{CQ}^{\neq}$-maximum recovery. By Theorem 6, we have that inequalities in the premises of dependencies are needed to express CQ-maximum recoveries of mappings given by st-tgds. Thus, if a mapping language contains the class of st-tgds and is closed under $\text{CQ}^{\neq}$-maximum recovery, then it has to include inequalities in the premises of dependencies. Our next result shows that in order to express the $\text{CQ}^{\neq}$-maximum recovery of a mapping given by a set of $\text{CQ}^{\neq}$- TO- CQ dependencies, one needs to use inequalities in the conclusions, even if the full power of FO is allowed in the premises of the dependencies.

Proposition 7 *There exists an st-mapping $\mathcal{M}$ specified by a set of* $\text{CQ}^{\neq}$- TO- CQ *dependencies such that:*

(a) *$\mathcal{M}$ has a* $\text{CQ}^{\neq}$*-maximum recovery specified by a set of* CQ- TO- $\text{CQ}^{\neq}$ *dependencies.*
(b) *$\mathcal{M}$ does not have a* $\text{CQ}^{\neq}$*-maximum recovery specified by a set of* $\text{FO}^{\mathbf{C}}$- TO- CQ *dependencies.*

Proof Consider a source schema $\mathbf{S} = \{P(\cdot, \cdot)\}$, a target schema $\mathbf{T} = \{T(\cdot)\}$, and the st-mapping $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \Sigma)$, where Σ consists of the following $\text{CQ}^{\neq}$- TO- CQ dependency:

$$P(x, y) \wedge x \neq y \rightarrow T(x).$$

Let $\mathcal{M}^\star = (\mathbf{T}, \mathbf{S}, \Sigma^\star)$ be a ts-mapping specified by the CQ- TO- $\text{CQ}^{\neq}$ dependency:

$$T(x) \rightarrow \exists y\, (P(x, y) \wedge x \neq y). \tag{11}$$

By using the tools developed in [4], it is straightforward to prove that $\mathcal{M}^\star$ is a maximum recovery of $\mathcal{M}$, which implies that $\mathcal{M}^\star$ is a $\text{CQ}^{\neq}$-maximum recovery of $\mathcal{M}$. Moreover, it can be shown, as in the proof of Proposition 6, that the inequality in the conclusion of (11) is essential to obtain a $\text{CQ}^{\neq}$-maximum recovery of $\mathcal{M}$, as every mapping specified by a set of tgds without inequalities in the conclusions is strictly less informative than $\mathcal{M}^\star$ (even if the full power of FO is allowed in the premises). The formalization of this argument can be found in Appendix A.12. □

Proposition 5 shows that there exist mappings specified by CQ- TO- $\text{CQ}^{\neq}$ dependencies that have no CQ-maximum recoveries and, hence, have no $\text{CQ}^{\neq}$-maximum recoveries. Thus, Propositions 5 and 7 show that if we use $\text{CQ}^{\neq}$-maximum recovery as our notion of inverse, then we cannot hope for a closure result as in Theorem 4.

We conclude this section by pointing out that the negative results for UCQ- and $\text{CQ}^{\neq}$-maximum recoveries imply a negative result for the notion of $\mathcal{C}$-maximum recovery, for every class $\mathcal{C}$ of queries containing UCQ or $\text{CQ}^{\neq}$. In particular, these results, together with the results in Sect. 6.1, show that our choices of the notion CQ-maximum recovery as the semantics for inversion and of $\text{CQ}^{\mathbf{C},\neq}$- TO- CQ as the language for defining mappings are optimal to obtain a closure result for inverting schema mappings, which is a fundamental property toward the applicability of inversion in practice.

7 Concluding remarks

In this paper, we have revisited the problem of inverting schema mappings paying special attention to the practical limitations of the previous approaches. We proposed a general query language-based notion of inverse, the $\mathcal{C}$-maximum recovery with $\mathcal{C}$ a query language. By fine-tuning the language $\mathcal{C}$, we show that the notion of CQ-maximum recovery satisfies our main requirements. In particular, we proved

that every st-mapping specified by a set of $CQ^{C,\neq}$- TO- CQ dependencies has a CQ-maximum recovery that can be specified in the same language. Interestingly, the language of $CQ^{C,\neq}$- TO- CQ dependencies has several good properties, being one of the most important the fact that the chase procedure can be used to exchange data with these dependencies efficiently (in data complexity), ensuring its practical applicability in data exchange and integration. Our results show that our choices of CQ-maximum recovery as the notion of inverse, and $CQ^{C,\neq}$- TO- CQ dependencies as the mapping specification language, are promising options toward the practical implementation of inversion of schema mappings.

Fagin et al. have been argued that when studying inverses of schema mappings in the data exchange context, incomplete information naturally arise in source instances [16]. We have made the assumption that source instances contain only constant values, thus, an important line of future work is how to extend our results on closure properties and query language-base notion of inverse to the setting in which incomplete information is allowed in the source. Another interesting issue is the integration between inversion and composition of schema mappings. Closure properties for composition have been obtained in the literature [6,14], but the languages used do not coincide with our proposed language for closure of inversion. Thus, a very important and also challenging topic for future research is to find a mapping language that is closed under both operation.

Acknowledgments We thank the anonymous reviewers for their careful reading and for providing many useful comments. Arenas was supported by Fondecyt grant 1090565, Pérez by Fondecyt grant 11110404 and by VID grant U-Inicia 11/04 Universidad de Chile, and Reutter by EPSRC grant G049165 and FET-Open project FoX.

References

1. Arenas, M., Barceló, P., Fagin, R., Libkin, L.: Locally consistent transformations and query answering in data exchange. In: PODS, pp. 229–240 (2004)
2. Arenas, M., Pérez, J., Reutter, J.L., Riveros, C.: Composition and inversion of schema mappings. SIGMOD Rec. **38**(3), 17–28 (2009)
3. Arenas, M., Pérez, J., Reutter, J.L., Riveros, C.: Inverting schema mappings: bridging the gap between theory and practice. PVLDB **2**(1), 1018–1029 (2009)
4. Arenas, M., Pérez, J., Riveros, C.: The recovery of a schema mapping: bringing exchanged data back. TODS **34**(4), 22:1–22:48 (2009)
5. Arenas, M., Pérez, J., Reutter, J.L.: Data exchange beyond complete data. In: PODS, pp. 83–94 (2011)
6. Arocena, P., Fuxman, A., Miller, R.J.: Composing local-as-view mappings: closure and applications. In: ICDT, pp. 209–218 (2010)
7. Barceló, P.: Logical foundations of relational data exchange. SIGMOD Rec. **38**(1), 49–58 (2009)
8. Bernstein, P.: Applying model management to classical meta data problems. In: CIDR (2003)
9. Bernstein, P., Melnik, S.: Model management 2.0: manipulating richer mappings. In: SIGMOD, pp. 1–12 (2007)
10. Chandra, A.K., Merlin, P.M.: Optimal implementation of conjunctive queries in relational data bases. In: STOC, pp. 77–90 (1977)
11. Fagin, R.: Inverting schema mappings. TODS **32**(4), 25:1–25:53 (2007)
12. Fagin, R., Kolaitis, P.G., Miller, R.J., Popa, L.: Data exchange: semantics and query answering. TCS **336**(1), 89–124 (2005)
13. Fagin, R., Kolaitis, P.G., Nash, A., Popa, L.: Towards a theory of schema-mapping optimization. In: PODS, pp. 33–42 (2008)
14. Fagin, R., Kolaitis, P.G., Popa, L., Tan, W.C.: Composing schema mappings: second-order dependencies to the rescue. TODS **30**(4), 994–1055 (2005)
15. Fagin, R., Kolaitis, P.G., Popa, L., Tan, W.C.: Quasi-inverses of schema mappings. TODS **33**(2), 11:1–11:52 (2008)
16. Fagin, R., Kolaitis, P.G., Popa, L., Tan, W.C.: Reverse data exchange: coping with nulls. ACM Trans. Database Syst. **36**(2), 11 (2011)
17. Fagin, R., Nash, A.: The structure of inverses in schema mappings. J. ACM **57**(6), 31 (2010)
18. Fuxman, A., Kolaitis, P.G., Miller, R.J., Tan, W.C.: Peer data exchange. In: PODS, pp. 160–171 (2005)
19. de Giacomo, G., Lembo, D., Lenzerini, M., Rosati, R.: On reconciling data exchange, data integration, and peer data management. In: PODS, pp. 133–142 (2007)
20. Halevy, A.Y., Ives, Z., Madhavan, J., Mork, P., Suciu, D., Tatarinov, I.: The piazza peer data management system. IEEE Trans. Knowl. Data Eng. **16**(7), 787–798 (2004)
21. Hell, P., Nešetřil, J.: Graphs and Homomorphisms. Oxford University Press, USA (2004)
22. Kolaitis, P.G.: Schema mappings, data exchange, and metadata management. In: PODS, pp. 61–75 (2005)
23. Lenzerini, M.: Data integration: a theoretical perspective. In: PODS, pp. 233–246 (2002)
24. Madhavan, J., Halevy, A.Y.: Composing mappings among data sources. In: VLDB, pp. 572–583 (2003)
25. Maier, D., Mendelzon, A., Sagiv, Y.: Testing implications of data dependencies. TODS **4**(4), 455–469 (1979)
26. Melnik, S.: Generic Model Management: concepts and Algorithms. Lecture Notes in Computer Science, vol. 2967. Springer, Berlin (2004)
27. Melnik, S., Adya, A., Bernstein, P.A.: Compiling mappings to bridge applications and databases. ACM Trans. Database Syst. **33**(4), (2008)
28. Melnik, S., Bernstein, P.A., Halevy, A.Y., Rahm, E.: Supporting executable mappings in model management. In: SIGMOD, pp. 167–178 (2005)
29. ten Cate, B., Kolaitis, P.G.: Structural characterizations of schema-mapping languages. In: ICDT, pp. 63–72 (2009)