Consistent Query Answering via ASP from Different Perspectives: Theory and Practice

MARCO MANNA, FRANCESCO RICCA

GIORGIO TERRACINA

Department of Mathematics, University of Calabria, Italy
(*e-mail:* {manna,ricca,terracina}@mat.unical.it)

submitted 23 November 2010; revised 1 January 2003; accepted 24 January 2011

Abstract

A data integration system provides transparent access to different data sources by suitably combining their data, and providing the user with a unified view of them, called *global schema*. However, source data are generally not under the control of the data integration process, thus integrated data may violate global integrity constraints even in presence of locally-consistent data sources. In this scenario, it may be anyway interesting to retrieve as much consistent information as possible. The process of answering user queries under global constraint violations is called *consistent query answering* (CQA). Several notions of CQA have been proposed, e.g., depending on whether integrated information is assumed to be *sound*, *complete*, *exact* or a variant of them. This paper provides a contribution in this setting: it uniforms solutions coming from different perspectives under a common ASP-based core, and provides query-driven optimizations designed for isolating and eliminating inefficiencies of the general approach for computing consistent answers. Moreover, the paper introduces some new theoretical results enriching existing knowledge on decidability and complexity of the considered problems. The effectiveness of the approach is evidenced by experimental results.
To appear in Theory and Practice of Logic Programming (TPLP).

KEYWORDS: Answer Set Programming, Data Integration, Consistent Query Answering

1 Introduction

The enormous amount of information dispersed over many data sources, often stored in different heterogeneous databases, has recently boosted the interest for data integration systems (Lenzerini 2002). Roughly speaking, a data integration system provides transparent access to different data sources by suitably combining their data, and providing the user with a unified view of them, called *global schema*. In many cases, the application domain imposes some consistency requirements on integrated data. For instance, it may be at least desirable to impose some integrity constraints (ICs), like primary/foreign keys, on the global relations. It may be the case that data stored at the sources may violate global ICs when integrated, since in general data sources are not under the control of the data integration process. The standard approach to this problem basically consists of explicitly modifying the data in order to eliminate IC violations (data cleaning). However, the explicit repair of data is not always convenient or possible. Therefore, when answering a user query, the system should be able to "virtually repair" relevant data (in the line of Arenas et al. 2003; Bertossi et al. 2005; Chomicki and Marcinkowski 2005), in order to provide consistent answers; this task is also called Consistent Query Answering (CQA).

The database community has spent considerable efforts in this area, relevant research results have been obtained to clarify semantics, decidability, and complexity of data-integration under constraints and, specifically, for CQA. In particular, several notions of CQA have been proposed (see Bertossi et al. 2005 for a survey), e.g. depending on whether the information in the database is assumed to be *sound*, *complete* or *exact*. However, while efficient systems are already available for simple data integration scenarios, solutions being both scalable and comprehensive have not been implemented yet for CQA, mainly due to the fact that handling inconsistencies arising from constraints violation is inherently hard. Moreover, mixing different kinds of constraints (e.g. denial constraints, and inclusion dependencies) on the same global database makes, often, the query answering process undecidable (Abiteboul et al. 1995; Calì et al. 2003a).

This paper provides some contributions in this setting. Specifically, it first starts from different state-of-the-art semantic perspectives (Arenas et al. 2003; Calì et al. 2003a; Chomicki and Marcinkowski 2005) and revisits them in order to provide a uniform, common core based on Answer Set Programming (ASP) (Gelfond and Lifschitz 1988; Gelfond and Lifschitz 1991). Thus, it provides query driven optimizations, in the light of the experience we gained in the INFOMIX (Leone et al. 2005) project in order to overcome the limitations observed in real-world scenarios. The main contributions of this paper can be summarized in:

- A theoretical analysis of considered semantics which extends previous results.
- The definition of a unified framework for CQA based on a purely declarative, logic based approach which supports the most relevant semantics assumptions on source data. Specifically, the problem of consistent query answering is reduced to cautious reasoning on (disjunctive) ASP programs with aggregates (Faber et al. 2010) automatically built from both the query and involved constraints.
- The definition of an optimization approach designed to *(1)* "localize" and limit the inefficient part of the computation of consistent answers to small fragments of the input, *(2)* cast down the computational complexity of the repair process if possible.
- The implementation of the entire framework in a full fledged prototype system.
- The capability of handling large amounts of data, typical of real-world data integration scenarios, using as internal query evaluator the DLV^{DB} (Terracina et al. 2008) system; indeed, DLV^{DB} allows for mass-memory database evaluations and distributed data management features.

In order to assess the effectiveness of the proposed approach, we carried out experimental activities both on a real world scenario and on synthetic data, comparing its behavior on different semantics and constraints.

The plan of the paper is as follows. Section 2 formally introduces the notion of CQA under different semantics and some new theoretical results on decidability and complexity for this problem. Section 3 first introduces a unified (general) solution to handle CQA via ASP, and then presents some optimizations. Section 4 describes the benchmark framework we adopted in the tests and discusses on obtained results. Finally, Section 5 compares related work and draws some conclusive considerations.

2 Data Integration Framework

In this paper we exploit the data integration setting to point out motivations and challenges underlying CQA. However, as it will be clarified in the following, techniques and results provided in the paper hold also for a single database setting. We next formally describe the adopted data integration framework.

The following notation will be used throughout the paper. We always denote by Γ a countably infinite domain of totally ordered values; by t a tuple of values from Γ; by X a variable; by $\bar{\mathbf{x}}$ a sequence $X_1, \ldots, X_n$ of (not necessarily distinct) variables, and by $|\bar{\mathbf{x}}| = n$ its length. Let $\bar{\mathbf{x}}, \bar{\mathbf{x}}'$ be two sequences of variables, we denote by $\bar{\mathbf{x}} - \bar{\mathbf{x}}'$ the sequence obtained from $\bar{\mathbf{x}}$ by discarding a variable if it appears in $\bar{\mathbf{x}}'$. Whenever all the variables of sequence $\bar{\mathbf{x}}$ appear in another sequence $\bar{\mathbf{x}}'$, we simply write $\bar{\mathbf{x}} \leq \bar{\mathbf{x}}'$. Given a sequence $\bar{\mathbf{x}}$ and a set $\pi \subseteq \{1, \ldots, |\bar{\mathbf{x}}|\}$, we denote by $\bar{\mathbf{x}}^{\pi}$ the sequence obtained from $\bar{\mathbf{x}}$ by discarding a variable if its position is not in π. (Similarly, given a tuple t and a set $\pi \subseteq \{1, \ldots, |t|\}$, we denote by t^{π} the tuple obtained from t by discarding a value if its position is not in π.) Moreover, we denote, by $\sigma(\bar{\mathbf{x}})$ a conjunction of comparison atoms of the form $X \odot X'$, where $\odot \in \{\leq, \geq, <, >, \neq\}$, and by $\ominus$, the symmetric difference operator between two sets.

A relational *database schema* is a pair $\mathcal{R} = \langle names(\mathcal{R}), constr(\mathcal{R}) \rangle$ where $names(\mathcal{R})$ and $constr(\mathcal{R})$ are the relation names and the integrity constraints (ICs) of $\mathcal{R}$, respectively. The arity of a given relation $r \in names(\mathcal{R})$ is denoted by $arity(r)$. A *database* (instance) for $\mathcal{R}$ is any set of facts (Abiteboul et al. 1995) of the form:

$$\mathcal{F} = \{r(t) : r \in names(\mathcal{R}) \ \wedge \ t \text{ is a tuple from } \Gamma \ \wedge \ |t| = arity(r)\}$$

In the following, we adopt the *unique name assumption*, and $\mathsf{dom}(\mathcal{F})$ denotes the subset of Γ containing all the values appearing in the facts of $\mathcal{F}$.

Let $r_1, \ldots, r_m \in names(\mathcal{R})$, the set $constr(\mathcal{R})$ contains ICs of the form:

1. $\forall \bar{\mathbf{x}}_1, \ldots, \bar{\mathbf{x}}_m \ \neg[\ r_1(\bar{\mathbf{x}}_1) \wedge \ldots \wedge r_m(\bar{\mathbf{x}}_m) \wedge \sigma(\bar{\mathbf{x}}_1, \ldots, \bar{\mathbf{x}}_m)\ ]$ (*denial constraints* – DCs)
2. $\forall \bar{\mathbf{x}}_{\forall}\ [\ r_1(\bar{\mathbf{x}}_1) \rightarrow \exists \bar{\mathbf{x}}_{2\exists}\ r_2(\bar{\mathbf{x}}_2)\ ]$ (*inclusion dependencies* – INDs);

where $arity(r_i) = |\bar{\mathbf{x}}_i|$, for each i in $[1..m]$. In particular, for INDs we require that all the variables within an $\bar{\mathbf{x}}_i$ $(1 \leq i \leq 2)$ are distinct, $\bar{\mathbf{x}}_{\forall} \leq \bar{\mathbf{x}}_1$, $\bar{\mathbf{x}}_{\forall} \leq \bar{\mathbf{x}}_2$, and $\bar{\mathbf{x}}_{2\exists} = \bar{\mathbf{x}}_2 - \bar{\mathbf{x}}_{\forall}$. Note that, if $|\bar{\mathbf{x}}_{2\exists}| = 0$, then $\bar{\mathbf{x}}_{\forall} = \bar{\mathbf{x}}_2 \leq \bar{\mathbf{x}}_1$. In the case we are only interested in emphasizing the relation names involved in an IND, we simply write $r_1(\bar{\mathbf{x}}_1) \rightarrow r_2(\bar{\mathbf{x}}_2)$ or $r_1 \rightarrow r_2$. A database $\mathcal{F}$ is said to be *consistent* w.r.t. $\mathcal{R}$ if all ICs are satisfied. A *conjunctive query* $cq(\bar{\mathbf{x}})$ over $\mathcal{R}$ is a formula of the form

$$\exists \bar{\mathbf{x}}_{1\exists}, \ldots, \bar{\mathbf{x}}_{m\exists}\ \ r_1(\bar{\mathbf{x}}_1) \wedge \ldots \wedge r_m(\bar{\mathbf{x}}_m) \wedge \sigma(\bar{\mathbf{x}}_1 \ldots, \bar{\mathbf{x}}_m)$$

where $\bar{\mathbf{x}}_{i\exists} \leq \bar{\mathbf{x}}_i$ for each i in $[1..m]$, $\bar{\mathbf{w}} = \bar{\mathbf{x}}_1 - \bar{\mathbf{x}}_{1\exists}, \ldots, \bar{\mathbf{x}}_m - \bar{\mathbf{x}}_{m\exists}$ are the *free variables* of q, and $\bar{\mathbf{x}}$ contains only and all the variables of $\bar{\mathbf{w}}$ (with no duplicates, and possibly in different order). A *union of conjunctive queries* $q(\bar{\mathbf{x}})$ is a formula of the form $cq_1(\bar{\mathbf{x}}) \vee \ldots \vee cq_n(\bar{\mathbf{x}})$. In the following, for simplicity, the term query refers to a union of conjunctive queries, if not differently specified. Given a database $\mathcal{F}$ for $\mathcal{R}$, and a query $q(\bar{\mathbf{x}})$, the *answer* to q is the set of n-tuples of values $ans(q, \mathcal{F}) = \{t : \mathcal{F} \models q(t)\}$.

2.1 The Data Integration Model

A *data integration system* is formalized (Lenzerini 2002) as a triple $\mathcal{I} = \langle \mathcal{G}, \mathcal{S}, \mathcal{M} \rangle$ where

- $\mathcal{G}$ is the *global schema*. A *global database* for $\mathcal{I}$ is any database for $\mathcal{G}$;
- $\mathcal{S}$ is the *source schema*. A *source database* for $\mathcal{I}$ is any database consistent w.r.t. $\mathcal{S}$;
- $\mathcal{M}$ is the *global-as-view* (GAV) *mapping*, that associates each element g in $names(\mathcal{G})$ with a union of conjunctive queries over $\mathcal{S}$.

Let $\mathcal{F}$ be a source database for $\mathcal{I}$. The *retrieved global database* is

$$ret(\mathcal{I}, \mathcal{F}) = \{g(t) : g \in names(\mathcal{G}) \ \wedge \ t \in ans(q, \mathcal{F}) \ \wedge \ q \in \mathcal{M}(g)\}$$

for $\mathcal{G}$ satisfying the mapping. Note that, when source data are combined in a unified schema with its own ICs, the retrieved global database might be inconsistent.

In the following, when it is clear from the context, we use simply the symbol $\mathcal{D}$ to denote the retrieved global database $ret(\mathcal{I}, \mathcal{F})$. In fact, all results provided in the paper hold for any database $\mathcal{D}$ complying with some schema $\mathcal{G}$ but possibly inconsistent w.r.t. the constraints of $\mathcal{G}$.

Example 1
Consider a bank association that desires to unify the databases of two branches. The first (source) database models managers by using a relation $man(code, name)$ and employees by a relation $emp(code, name)$, where $code$ is a primary key for both tables. The second database stores the same data in a relation $employee(code, name, role)$. Suppose that the data have to be integrated under a global schema with two relations $m(code)$ and $e(code, name)$, where the global ICs are:

- $\forall X_1, X_2, X_3 \ \neg[e(X_1, X_2) \wedge e(X_1, X_3) \wedge X_2 \neq X_3]$ namely, $code$ is the key of e;
- $\forall X_1[m(X_1) \rightarrow \exists X_2 \ e(X_1, X_2)]$ i.e., an IND imposing that each manager code must be an employee code as well.

The mapping is defined by the following Datalog rules (as usual, see Abiteboul et al. 1995):

$$e(X_c, X_n) \text{ :- } emp(X_c, X_n). \qquad m(X_c) \text{ :- } man(X_c, \_).$$
$$e(X_c, X_n) \text{ :- } employee(X_c, X_n, \_). \qquad m(X_c) \text{ :- } employee(X_c, \_, \text{‘}manager\text{'}).$$

Assume that, emp stores tuples *(‘e1’,‘john’)*, *(‘e2’,‘mary’)*, *(‘e3’,‘willy’)*, man stores *(‘e1’,‘john’)*, and $employee$ stores *(‘e1’,‘ann’,‘manager’)*, *(‘e2’,‘mary’,‘manager’)*, *(‘e3’, ‘rose’,‘emp’)*. It is easy to verify that, although the source databases are consistent w.r.t. local constraints, the global database, obtained by evaluating the mapping, violates the key constraint on e as both *john* and *ann* have the same code *e1*, and both *willy* and *rose* have the same code *e3* in table e. □

2.2 Consistent Query Answering under different semantics

In case a database $\mathcal{D}$ violates ICs, one can still be interested in querying the “consistent” information originating from $\mathcal{F}$. One possibility is to “repair” $\mathcal{D}$ (by inserting or deleting tuples) in such a way that all the ICs are satisfied. But there are several ways to “repair” $\mathcal{D}$. As an example, in order to satisfy an IND of the form $r_1 \rightarrow r_2$ one might either remove violating tuples from r_1 or insert new tuples in r_2. Moreover, the repairing strategy depends on the particular semantic assumption made on the data integration system. Semantic assumptions may range from (strict) soundness to (strict) completeness. Roughly speaking, completeness complies with the *closed world assumption* where missing facts are assumed to be false; on the contrary, soundness complies with the *open world assumption* where $\mathcal{D}$ may be incomplete. We next define consistent query answering under some relevant semantics, namely *loosely-exact, loosely-sound, CM-complete* (Arenas et al. 2003; Calì et al. 2003a; Chomicki and Marcinkowski 2005). More formally, let Σ denote a *semantics*, and $\mathcal{D}$ a possibly inconsistent database for $\mathcal{G}$, a database $\mathcal{B}$ is said to be a Σ-*repair* for $\mathcal{D}$ if it is consistent w.r.t. $\mathcal{G}$ and one of the following conditions holds:

1. $\Sigma =$ *CM-complete*, $\mathcal{B} \subseteq \mathcal{D}$, and $\nexists\, \mathcal{B}' \subseteq \mathcal{D}$ such that $\mathcal{B}'$ is consistent and $\mathcal{B}' \supset \mathcal{B}$;
2. $\Sigma =$ *loosely-sound* and $\nexists\, \mathcal{B}'$ such that $\mathcal{B}'$ is consistent and $\mathcal{B}' \cap \mathcal{D} \supset \mathcal{B} \cap \mathcal{D}$;
3. $\Sigma =$ *loosely-exact*, and $\nexists\, \mathcal{B}'$ such that $\mathcal{B}'$ is consistent and $\mathcal{B}' \ominus \mathcal{D} \subset \mathcal{B} \ominus \mathcal{D}$.

The *CM-complete* semantics allows a minimal number of deletions in each repair to avoid empty repairs, if possible, but does not allow insertions. The *loosely-sound* semantics allows insertions and a minimal amount of deletions. Finally, the *loosely-exact* semantics allows both insertions and deletions by minimization of the symmetric difference between $\mathcal{D}$ and the repairs.

Definition 1

Let $\mathcal{D}$ be a database for a schema $\mathcal{G}$, and Σ be a semantics. The *consistent answer* to a query q w.r.t. $\mathcal{D}$, is the set $ans_\Sigma(q, \mathcal{G}, \mathcal{D}) = \{t : t \in ans(q, \mathcal{B})$ *for each* Σ-*repair* $\mathcal{B}$ *for* $\mathcal{D}\}$ *Consistent Query Answering* (CQA) is the problem of computing $ans_\Sigma(q, \mathcal{G}, \mathcal{D})$. □

Observe that other semantics have been considered in the literature, like *sound*, *complete*, *exact*, *loosely-complete*, etc. (Calì et al. 2003a); however, some of them are trivial for CQA; as an example, in the *exact* semantics CQA makes sense only if the retrieved database is already consistent with the global constraints, whereas in the *complete* and *loosely-complete* semantics CQA will always return a void answer. Note that, the semantics considered in this paper address a wide significant range of ways to repair the retrieved database which are also relevant for CQA.

Example 2

By following Example 1, the retrieved global database admits exactly the following repairs under the *CM-complete* semantics:

$$\mathcal{B}_1 = \{e(\text{'e2','mary'}),\ e(\text{'e1','john'}),\ e(\text{'e3','willy'}),\ m(\text{'e1'}),\ m(\text{'e2'})\}$$
$$\mathcal{B}_2 = \{e(\text{'e2','mary'}),\ e(\text{'e1','john'}),\ e(\text{'e3','rose'}),\ m(\text{'e1'}),\ m(\text{'e2'})\}$$
$$\mathcal{B}_3 = \{e(\text{'e2','mary'}),\ e(\text{'e1','ann'}),\ e(\text{'e3','willy'}),\ m(\text{'e1'}),\ m(\text{'e2'})\}$$
$$\mathcal{B}_4 = \{e(\text{'e2','mary'}),\ e(\text{'e1','ann'}),\ e(\text{'e3','rose'}),\ m(\text{'e1'}),\ m(\text{'e2'})\}$$

Query $m(X)$ asking for the list of manager codes has then both *e1* and *e2* as consistent answers, whereas the query $e(X, Y)$ asking for the list of employees has only e(*'e2','mary'*) as consistent answer (e is the only tuple in each *CM-complete* repair). □

2.3 Restricted Classes of Integrity Constraints

The problem of computing CQA, under general combinations of ICs, is undecidable (Abiteboul et al. 1995). However, restrictions on ICs to retain decidability and identify tractable cases can be imposed.

Definition 2
Let r be a relation name of arity n, and π be a set of $m \leq n$ indices from $I = \{1, \ldots, n\}$. A *key dependency* (KD) for r consists of a set of $n - m$ DCs, exactly one for each index $i \in I - \pi$, of the form $\forall \bar{\mathbf{x}}_1, \bar{\mathbf{x}}_2 \;\; \neg(r(\bar{\mathbf{x}}_1) \wedge r(\bar{\mathbf{x}}_2) \wedge \; \bar{\mathbf{x}}_1^i \neq \bar{\mathbf{x}}_2^i)$ where no variable occurs twice in each $\bar{\mathbf{x}}_i$ ($1 \leq i \leq 2$), $|\bar{\mathbf{x}}_1| = |\bar{\mathbf{x}}_2| = n$, the sequence $\bar{\mathbf{x}}_1^\pi$ exactly coincides with $\bar{\mathbf{x}}_2^\pi$, and $\bar{\mathbf{x}}_1^j$ is distinct from $\bar{\mathbf{x}}_2^j$ for each $j \in I - \pi$. The set π is called the *primary-key* of r and is denoted by $key(r)$. We assume that at most one KD is specified for each relation (Calì et al. 2003a). Finally, for each relation name r' such that no DC is explicitly specified for, we say, without loss of generality, that $key(r') = \{1, \ldots, arity(r')\}$. □

Definition 3
Given an inclusion dependency d of the form $\forall \bar{\mathbf{x}}_\forall \; [\; r_1(\bar{\mathbf{x}}_1) \rightarrow \exists \bar{\mathbf{x}}_{2\exists} \; r_2(\bar{\mathbf{x}}_2) \;]$, we denote by $\pi_L^d \subseteq \{1, \ldots, arity(r_1)\}$ and $\pi_R^d \subseteq \{1, \ldots, arity(r_2)\}$ the two sets of indices induced by the positions of the variables $\bar{\mathbf{x}}_\forall$ in $\bar{\mathbf{x}}_1$ and $\bar{\mathbf{x}}_2$, respectively. More formally, $\pi_L^d = \{i : \bar{\mathbf{x}}_1^i$ *is universally quantified in* $d\}$ and $\pi_R^d = \{i : \bar{\mathbf{x}}_2^i$ *is universally quantified in* $d\}$. □

For example, let d denote the IND $\forall X_1, X_2 \; [\; r_1(X_1, X_3, X_2) \rightarrow \exists X_4 \; r_2(X_4, X_2, X_1) \;]$. We have that $\pi_L^d = \{1, 3\}$ and $\pi_R^d = \{2, 3\}$.

Definition 4
An IND d is said to be

- a *foreign key* (FK) if $\pi_R^d = key(r_2)$ (Abiteboul et al. 1995);
- a *foreign superkey* (FSK) if $\pi_R^d \supseteq key(r_2)$ (Levene and Vincent 2000);
- *non-key-conflicting* (NKC) if $\pi_R^d \not\supset key(r_2)$ (Calì et al. 2003a). □

Definition 5
An FSK d of the form $r_1 \rightarrow r_2$ is said to be *safe* (SFSK) if $\pi_L^d \subseteq key(r_1)$. In particular, if d is a *safe* FK we call it an SFK. □

For example, let d denote the FSK $\forall X_1, X_2 \; [\; r_1(X_1, X_3, X_2) \rightarrow \exists X_4 \; r_2(X_4, X_2, X_1) \;]$ where $key(r_2) = \{3\}$. Thus, if $key(r_1) = \{1, 3\}$, d is SFSK, whereas if $key(r_1) = \{1, 2\}$, d is not SFSK.

Table 1 summarizes known and new results about computability and complexity of CQA under relevant classes of ICs and the three semantic assumptions considered in this paper. In particular, given a query q (without comparison atoms if $\Sigma \in \{loosely\text{-}sound, loosely\text{-}exact\}$), we refer to the decision problem of establishing whether a tuple from $\mathsf{dom}(\mathcal{D})$ belongs to $ans_\Sigma(q, \mathcal{G}, \mathcal{D})$ or not. Note that, Chomicki and Marcinkowski (2005) have proved computability and complexity of CQA for the *CM-complete* semantics in case of conjunctive queries with comparison predicates. However, since in such a setting there is a finite number of repairs each of finite size, then their results straightforwardly hold for union of conjunctive queries as well. New decidability and complexity results for CQA under KDs and SFSKs only, with $\Sigma \in \{$*loosely-sound*, *loosely-exact*$\}$ are proved in Section 2.4.

Table 1. Data Complexity of CQA (distinguishing between cyclic/acyclic INDs)

DCs	INDs	*loosely-sound*	*loosely-exact*	*CM-complete*
no	*any*	in **PTIME** [1]	in **PTIME** [1]	in **PTIME** [2]
KD	*no*	**coNP**-c [1]	**coNP**-c [1]	**coNP**-c [2]
KD	*NKC*	**coNP**-c [1]	Π_2^p-c [1]	in Π_2^p [2] / in **coNP** [2]
KD	*SFSK*	in Π_2^p [3]	in Π_2^p [3]	in Π_2^p [2] / in **coNP** [2]
KD	*any*	undec. [1]	undec. [1]	in Π_2^p [2] / in **coNP** [2]
any	*any*	undec. [4]	undec. [4]	Π_2^p-c [2] / **coNP**-c [2]

[1] Calì et al. 2003a; [2] Chomicki and Marcinkowski 2005; [3] Section 2.4; [4] Abiteboul et al. 1995;

2.4 Loosely-exact and Loosely-sound semantics under KD and SFSK

In this section we provide new decidability and complexity results for CQA under both the loosely-exact and the loosely-sound semantics with KDs and SFSKs. In the rest of the section we always denote by:

- $\mathcal{G}$, a schema containing KDs and SFSKs only;
- $\mathcal{D}$, a possibly inconsistent database for $\mathcal{G}$;
- q, a union of conjunctive queries without comparison atoms.
- $\Sigma \in \{$*loosely-exact*, *loosely-sound*$\}$.

We first show that, in the aforementioned hypothesis, the size of each repair is finite.

Definition 6
Let $\mathcal{B}$ be a Σ-repair for $\mathcal{D}$ and $i \geq 0$ be a natural number. We inductively define the sets $\mathcal{B}^i$ as follows:

1. If $i = 0$, then $\mathcal{B}^0 = \mathcal{B} \cap \mathcal{D}$.
2. If $i > 0$, then $\mathcal{B}^i \subseteq \mathcal{B} - (\mathcal{B}^0 \cup \ldots \cup \mathcal{B}^{i-1})$ is arbitrarily chosen in such a way that its facts are necessary and sufficient for satisfying all the INDs in $constr(\mathcal{G})$ that are violated in $\mathcal{B}^0 \cup \ldots \cup \mathcal{B}^{i-1}$.

Observe that $\mathcal{B} = \bigcup_{i\geq 0} \mathcal{B}^i$ and that $\mathcal{B}^i \cap \mathcal{B}^j = \emptyset$ for each $j \neq i$. □

Lemma 1
Let $\mathcal{B}$ be a Σ-repair for $\mathcal{D}$, then

1. The key of each fact in $\mathcal{B}$ only contains values from $\mathsf{dom}(\mathcal{D})$.
2. $|\mathcal{B}|$ is finite.

Proof

(1) Let $i > 0$ be a natural number. Let $r_i(t_i)$ be a fact in $\mathcal{B}^i$ such that there is an index $j \in key(r_i)$ for which $t_i^j \notin \mathsf{dom}(\mathcal{B}^0)$. Let $r_{i-1}(t_{i-1})$ be one of the facts in $\mathcal{B}^{i-1}$ that forces the presence of $r_i(t_i)$ in $\mathcal{B}^i$ for satisfying some IND, say d. (Note that, by Definition 6, there must be at least one of such a fact because $\mathcal{B}^i$ would otherwise violate condition 2, since $r_i(t_i)$ would be unnecessary.) Moreover, since d is a safe FSK, then there must exist an index $k \in key(r_{i-1})$ such that $t_i^j = t_{i-1}^k$. Thus, $r_{i-1}(t_{i-1})$ contains a value being not in $\mathsf{dom}(\mathcal{B}^0)$ inside its key as well as $r_i(t_i)$. Since i has been chosen arbitrarily, then value t_i^j has to be part of a fact of $\mathcal{B}^0$, which is clearly a contradiction.

(2) Since, the key of each fact in $\mathcal{B}$ can only contain values from $\mathsf{dom}(\mathcal{B}^0)$, and $|\mathsf{dom}(\mathcal{B}^0)| \leq |\mathcal{B}^0|\cdot\alpha$ where $\alpha = \mathsf{max}\{arity(g) : g \in names(\mathcal{G})\}$, then $|\mathcal{B}| \leq |names(\mathcal{G})|\cdot|\mathsf{dom}(\mathcal{B}^0)|^\alpha \leq |names(\mathcal{G})| \cdot (\alpha \cdot |\mathcal{B}^0|)^\alpha \leq |names(\mathcal{G})| \cdot (\alpha \cdot |\mathcal{D}|)^\alpha$. □

We next characterize representative databases for Σ-repairs.

Definition 7

Let $\mathcal{B}$ be a Σ-repair for $\mathcal{D}$. We denote by $\mathsf{homo}(\mathcal{B})$ the (possibly infinite) set of databases defined in such a way that $\mathcal{B}' \in \mathsf{homo}(\mathcal{B})$ if and only if:

- $\mathcal{B}'$ can be obtained from $\mathcal{B}$ by replacing each value (if any) that is not in $\mathsf{dom}(\mathcal{D})$ with a value from $\Gamma - \mathsf{dom}(\mathcal{D})$; and
- none of the values in $\Gamma - \mathsf{dom}(\mathcal{D})$ occurs twice in $\mathcal{B}'$.

Finally, we denote by $h_{\mathcal{B},\mathcal{B}'} : \mathsf{dom}(\mathcal{B}') \rightarrow \mathsf{dom}(\mathcal{B})$ the function (homomorphism) associating values in $\mathsf{dom}(\mathcal{B}')$ with values in $\mathsf{dom}(\mathcal{B})$, where $h_{\mathcal{B},\mathcal{B}'}(\alpha) = \alpha$, for each $\alpha \in \mathsf{dom}(\mathcal{D}) \cap \mathsf{dom}(\mathcal{B}')$. □

Note that, since (by Lemma 1) the key of each fact in $\mathcal{B}$ only contains values from $\mathsf{dom}(\mathcal{D})$, then $|\mathcal{B}'| = |\mathcal{B}|$ holds.

For example, if $\mathcal{B} = \{p(1, \varepsilon_1, \varepsilon_2), q(2, \varepsilon_2, \varepsilon_1)\}$ with $\mathsf{dom}(\mathcal{D}) = \{1, 2\}$ and $key(p) = key(q) = \{1\}$, then all of the following databases are in $\mathsf{homo}(\mathcal{B})$: $\{p(1, \varepsilon_1, \varepsilon_3), q(2, \varepsilon_2, \varepsilon_4)\}$, $\{p(1, \varepsilon_4, \varepsilon_2), q(2, \varepsilon_3, \varepsilon_1)\}$ and $\{p(1, \varepsilon_5, \varepsilon_6), q(2, \varepsilon_7, \varepsilon_8)\}$.

Lemma 2

If $\mathcal{B}$ is a Σ-repair for $\mathcal{D}$, then each $\mathcal{B}' \in \mathsf{homo}(\mathcal{B})$ also is.

Proof

Let $\mathcal{B}' \in \mathsf{homo}(\mathcal{B})$. First of all, we prove that $\mathcal{B}'$ is consistent w.r.t. $\mathcal{G}$. In particular, since the key of each fact in $\mathcal{B}$ only contains values from $\mathsf{dom}(\mathcal{D})$ (by Lemma 1), then $\mathcal{B}'$ cannot violate any KD (by Definition 7); Moreover, since each IND has to be satisfied through values of a key (by definition of safe FSKs), and since the key of each fact in $\mathcal{B}$ only contains values from $\mathsf{dom}(\mathcal{D})$ (by Lemma 1), then $\mathcal{B}'$ cannot violate any IND (by Definition 7);

We now prove that $\mathcal{B}'$ is a repair, first for the loosely-sound semantics and then for the loosely-exact semantics.

[loosely-sound] If $\Sigma =$ *loosely-sound*, then observe that $\mathcal{B}' \cap \mathcal{D} = \mathcal{B} \cap \mathcal{D}$, by definition of $\mathsf{homo}(\mathcal{B})$. Thus, if $\mathcal{B}'$ was consistent but not a loosely-sound repair there would exist a loosely-sound repair $\mathcal{B}''$ such that $\mathcal{B}'' \cap \mathcal{D} \supset \mathcal{B}' \cap \mathcal{D} = \mathcal{B} \cap \mathcal{D}$. Contradiction.

[loosely-exact] If $\Sigma =$ *loosely-exact*, then assume that $\mathcal{B}$ is a loosely-exact repair but $\mathcal{B}'$ (although consistent w.r.t. $\mathcal{G}$) is not. By definition, there must be a loosely-exact repair $\mathcal{B}''$ such that $\mathcal{B}'' \ominus \mathcal{D} \subset \mathcal{B}' \ominus \mathcal{D}$. In particular, we distinguish three cases:

(1) $\mathcal{B}'' - \mathcal{D} = \mathcal{B}' - \mathcal{D}$ and $\mathcal{D} - \mathcal{B}'' \subset \mathcal{D} - \mathcal{B}'$
(2) $\mathcal{B}'' - \mathcal{D} \subset \mathcal{B}' - \mathcal{D}$ and $\mathcal{D} - \mathcal{B}'' = \mathcal{D} - \mathcal{B}'$
(3) $\mathcal{B}'' - \mathcal{D} \subset \mathcal{B}' - \mathcal{D}$ and $\mathcal{D} - \mathcal{B}'' \subset \mathcal{D} - \mathcal{B}'$

CASE 1: Since, by Definition 7, for each fact in $\mathcal{B}$ there is a fact in $\mathcal{B}'$ with the same key, if we could add the facts in $\mathcal{B}'' - \mathcal{B}'$ to $\mathcal{B}'$ without violating any KD, then such facts could also be added to $\mathcal{B}$ without violating any KD. Moreover, if we could add to $\mathcal{B}'$ the facts in $\mathcal{B}'' - \mathcal{B}'$ without violating any IND, then such facts could be also added to $\mathcal{B}$ preserving consistency. This follows by the definition of safe FSKs (because each IND has to be satisfied through values of a key), by Lemma 1 (because the key of each fact in a loosely-exact repair only contains values from $\mathsf{dom}(\mathcal{D})$) and, by Definition 7 (because for each fact in $\mathcal{B}'$ there is a fact in $\mathcal{B}$ with the same key and with the same values from $\mathsf{dom}(\mathcal{D})$). Consequently, we could add all the facts in $\mathcal{B}'' - \mathcal{B}'$ to $\mathcal{B}$ preserving consistency. But this is not possible since $\mathcal{B}$ is a loosely-exact repair.

CASE 2: Since in $\mathcal{B}'$ we have unnecessary facts (those in $\mathcal{B}' - \mathcal{B}''$) or equivalently the facts in $\mathcal{B}''$ do not violate any IND, then the corresponding facts in $\mathcal{B}$ do not violate any IND by Lemma 1 and by Definition 7. Consequently, if each fact $f \in \mathcal{B}$, such that there is a fact $f' \in \mathcal{B}' - \mathcal{B}''$ that is homomorphic to f, was removed from $\mathcal{B}$, then we would obtain a database preserving consistency and with a smaller symmetric difference than $\mathcal{B}$. But this is not possible since $\mathcal{B}$ is a loosely-exact repair.

CASE 3: Analogous considerations can be done by combining case 1 and case 2. □

We next define the finite database $\mathcal{D}^*$ having among its subsets a number of Σ-repairs sufficient for solving CQA.

Definition 8
Let c be a value in $\Gamma - \mathsf{dom}(\mathcal{D})$. Consider the largest (possibly inconsistent) database, say C, constructible on the domain $\mathsf{dom}(\mathcal{D}) \cup \{c\}$ such that $f \in C$ iff the value c does not appear in the key of f. Let $\mathcal{N}$ be a fixed set of values arbitrarily chosen from $\Gamma - \mathsf{dom}(\mathcal{D})$ whose cardinality is equal to the number of occurrences of c in C. We denote by $\mathcal{D}^*$ one possible database for $\mathcal{G}$ obtained from C by replacing each occurrence of c with a value from $\mathcal{N}$ in such a way that each value in $\mathcal{N}$ occurs exactly once in $\mathcal{D}^*$. ($|C| = |\mathcal{D}^*|$.) □

For example, if $\mathsf{dom}(\mathcal{D}) = \{1, 2\}$ and $\mathcal{G} = \{p\}$ with $arity(p) = 2$ and $key(p) = \{1\}$, then $C = \{p(1,1), p(1,2), p(1,c), p(2,1), p(2,2), p(2,c)\}$. Let us fix $\mathcal{N} = \{\varepsilon_1, \varepsilon_2\}$. Thus, $\mathcal{D}^*$ has the following form: $\{p(1,1), p(1,2), p(1,\varepsilon_1), p(2,1), p(2,2), p(2,\varepsilon_2)\}$.

Proposition 1
The following hold:

- $|\mathcal{N}| = \sum_{g \in \mathcal{G}} (arity(g) - |key(g)|) \cdot |\mathsf{dom}(\mathcal{D})|^{|key(g)|} \cdot (|\mathsf{dom}(\mathcal{D})| + 1)^{arity(g) - |key(g)| - 1}$
- $|\mathcal{D}^*| \leq \sum_{g \in \mathcal{G}} (|\mathsf{dom}(\mathcal{D})| + 1)^{arity(g)} \leq \sum_{g \in \mathcal{G}} (arity(g) \cdot |\mathcal{D}| + 1)^{arity(g)}$

Lemma 3

If $\mathcal{B}$ is a Σ-repair for $\mathcal{D}$, then there exists $\mathcal{B}' \in \mathsf{homo}(\mathcal{B})$ such that $\mathcal{B}' \subseteq \mathcal{D}^*$.

Proof

$\mathcal{B}'$ can be obtained from $\mathcal{B}$ by replacing each fact $r(t_1) \in \mathcal{B}$ with the unique fact $r(t_2) \in \mathcal{D}^*$ such that for each $i \in arity(r)$ either $t_2^i = t_1^i$, if $t_1^i \in \mathsf{dom}(\mathcal{D})$, or $t_2^i \in \mathcal{N}$, if $t_1^i \notin \mathsf{dom}(\mathcal{D})$. Moreover, note that, since $\mathcal{B}$ cannot contain two facts with the same key and since keys only have values from $\mathsf{dom}(\mathcal{D})$, then each fact in $\mathcal{D}^*$ can replace at most one fact in $\mathcal{B}$. Finally, $\mathcal{B}' \in \mathsf{homo}(\mathcal{B})$ by Definition 7. $\square$

Lemma 4

Let $\mathcal{B}$ be a Σ-repair for $\mathcal{D}$, $\mathcal{B}' \in \mathsf{homo}(\mathcal{B})$, q be a query, and t be a tuple of values from $\mathsf{dom}(\mathcal{D})$. If $t \in ans(q, \mathcal{B}')$, then $t \in ans(q, \mathcal{B})$.

Proof

Let q_i be one of the conjunctions in q, if $t \in ans(q_i, \mathcal{B}')$, then there is a substitution μ' from the variables of q_i to values in Γ such that $\mathcal{B}' \models q_i(t)$. But since, by Definition 7, each fact in $\mathcal{B}'$ is univocally associated with a unique fact in $\mathcal{B}$ by preserving the values in $\mathsf{dom}(\mathcal{D})$, and since all the extra values in $\mathcal{B}'$ are distinct, then there must also be a substitution μ such that $\mathcal{B} \models q_i(t)$. In particular, let x be a variable in q_i, we can define μ in such a way that $\mu(x) = h_{\mathcal{B},\mathcal{B}'}(\mu'(x))$, where h is the homomorphism from $\mathcal{B}'$ to $\mathcal{B}$ (see Definition 7). Clearly, if $t \in ans(q_i, \mathcal{B}')$ for at least one q_i in q then $t \in ans(q, \mathcal{B}')$ too and, consequently, $t \in ans(q, \mathcal{B})$ $\square$

The next theorem states the decidability of CQA under both the loosely-exact and the loosely-sound semantics with KDs and SFSKs only.

Theorem 1

Let $\mathcal{B}$ be a Σ-repair for $\mathcal{D}$, q a query, and t a tuple from $\mathsf{dom}(\mathcal{D})$. Let $\mathbb{B} \subseteq 2^{\mathcal{D}^*}$ denote the set of all Σ-repairs contained in $\mathcal{D}^*$. Then, $t \in ans_\Sigma(q, \mathcal{G}, \mathcal{D})$ **iff** $t \in ans(q, \mathcal{B})\ \forall \mathcal{B} \in \mathbb{B}$.

Proof

($\Rightarrow$) We have to prove that, *if* $t \in ans_\Sigma(q, \mathcal{G}, \mathcal{D})$, *then* $t \in ans(q, \mathcal{B})$ *for each* $\mathcal{B} \in \mathbb{B}$, or equivalently *if* $t \notin ans(q, \mathcal{B})$ *for some* $\mathcal{B} \in \mathbb{B}$, *then* $t \notin ans_\Sigma(q, \mathcal{G}, \mathcal{D})$. This follows, by the definition of $ans_\Sigma(q, \mathcal{G}, \mathcal{D})$ and from the fact that $\mathbb{B}$ only contains Σ-repairs.

($\Leftarrow$) We have to prove that, *if* $t \in ans(q, \mathcal{B})$ *for each* $\mathcal{B} \in \mathbb{B}$, *then* $t \in ans_\Sigma(q, \mathcal{G}, \mathcal{D})$. Assume that $t \in ans(q, \mathcal{B})$ for each $\mathcal{B} \in \mathbb{B}$ but $t \notin ans_\Sigma(q, \mathcal{G}, \mathcal{D})$. This would entail that there is a repair $\mathcal{B}_0$ such that $t \notin ans(q, \mathcal{B}_0)$. But, since $t \notin ans(q, \mathcal{B}')$ for each $\mathcal{B}' \in \mathsf{homo}(\mathcal{B}_0)$ (by Lemma 4), and since $\mathbb{B} \cap \mathsf{homo}(\mathcal{B}_0)$ always contains a repair, say $\mathcal{B}''$ (by Lemma 3), then we have a contradiction since $t \notin ans(q, \mathcal{B}'')$ has to hold whereas we have assumed that $t \in ans(q, \mathcal{B})$ for each $\mathcal{B} \in \mathbb{B}$. $\square$

Decidability and complexity results, under KDs and SFSKs only, follow from Theorem 1.

Corollary 1
Let $\mathcal{G}$ be a global schema containing KDs and SFSKs only, $\mathcal{D}$ be a possibly inconsistent database for $\mathcal{G}$, q be a query, $\Sigma \in \{$*loosely-exact*, *loosely-sound*$\}$, and t be a tuple of values from $\mathsf{dom}(\mathcal{D})$. The problem of establishing whether $t \in ans_{\Sigma}(q, \mathcal{G}, \mathcal{D})$ is in Π_2^p in data complexity.

Proof
It suffices to prove that the problem of establishing whether $t \notin ans_{\Sigma}(q, \mathcal{G}, \mathcal{D})$ is in Σ_2^p. This can be done by (i) building $\mathcal{D}^*$, and (ii) guessing $\mathcal{B} \in 2^{\mathcal{D}^*}$ such that $\mathcal{B}$ is a Σ-repair and $t \notin ans(q, \mathcal{B})$. Since, by Proposition 1, $|\mathcal{D}^*| \in \mathcal{O}(|\mathcal{D}|^{\alpha})$ where $\alpha = \mathsf{max}\{arity(g) : g \in names(\mathcal{G})\}$, then step (i) (enumerate the facts of $\mathcal{D}^*$) can be done in polynomial time. Since checking that $t \notin ans(q, \mathcal{B})$ can be done in **PTIME**. It remains to show that checking whether $\mathcal{B}$ is a Σ-repair can be done in **coNP**.
[loosely-exact] If $\Sigma =$ *loosely-exact*, this task corresponds to checking that there is no consistent $\mathcal{B}' \subseteq \mathcal{D} \cup \mathcal{B}$ such that $\mathcal{B}' \ominus \mathcal{D} \subset \mathcal{B} \ominus \mathcal{D}$, where this last task is doable in **PTIME**.
[loosely-sound] If $\Sigma =$ *loosely-sound*, this task corresponds to checking that there is no consistent $\mathcal{B}' \subseteq \mathcal{D}^*$ such that $\mathcal{B}' \cap \mathcal{D} \supset \mathcal{B} \cap \mathcal{D}$, where this last task is doable in **PTIME**.
Then the thesis follows. □

2.5 *Equivalence of CQA under loosely-exact and CM-complete semantics*

In this section we define some relevant cases in which CQA under loosely-exact and CM-complete semantics coincide.

Lemma 5
Given a database $\mathcal{D}$ for a schema $\mathcal{G}$, if $\mathcal{B}$ is a CM-complete repair for $\mathcal{D}$, then it is a loosely-exact repair for $\mathcal{D}$.

Proof
Suppose that $\mathcal{B}$ is a *CM-complete* repair for $\mathcal{D}$ (so, it is consistent w.r.t. $\mathcal{G}$), but it is not a *loosely-exact* one. This means that its symmetric difference with $\mathcal{D}$ can be still reduced. But, by definition of *CM-complete* semantics, $\mathcal{B}$ does not contain anything else but tuples in $\mathcal{D}$, namely $\mathcal{B} - \mathcal{D} = \emptyset$. So, the only way for "improving" it is to extend it with tuples from $\mathcal{D}$. But, this is not possible because $\mathcal{B}$ is already maximal due to the *CM-complete* semantics, namely the addition of any other tuple would violate at least one IC. □

Corollary 2
$ans_{loosely-exact}(q, \mathcal{G}, \mathcal{D}) \subseteq ans_{CM-complete}(q, \mathcal{G}, \mathcal{D})$

Proof
This directly follows by Lemma 5 in light of Definition 1. □

Theorem 2
There are cases where $ans_{loosely\text{-}exact}(q, \mathcal{G}, \mathcal{D}) \subset ans_{CM\text{-}complete}(q, \mathcal{G}, \mathcal{D})$

Proof
By Chomicki and Marcinkowski (2005), stating that the two semantics are different, and by Corollary 2. □

Proposition 2
Let $\mathcal{B}$ be a database consistent w.r.t. a set of ICs C.

1. If C are DCs only, then each $\mathcal{B}' \subset \mathcal{B}$ is consistent w.r.t. C, as well.
2. If C are INDs only, then $\mathcal{B} \cup \mathcal{B}'$ is consistent w.r.t. C for each $\mathcal{B}'$ consistent w.r.t. C.

Proof
(1) Deletion of tuples can not introduce new DCs violations.

(2) Let $r(t)$ be a fact in $\mathcal{B}'$. Let d_1 be an IND of the form $r_1 \rightarrow r$ ($r \neq r_1$). Clearly, $r(t)$ cannot violate d_1 in any database because r is in the righthand side of d_1. In particular, $r(t)$ cannot violate d_1 in $\mathcal{B} \cup \mathcal{B}'$. Let d_2 be an IND of the form $r \rightarrow r_2$ (possibly, $r = r_2$). Since $r(t)$ does not violate d_2 in $\mathcal{B}'$, then it cannot violate d_2 in $\mathcal{B} \cup \mathcal{B}'$. □

Theorem 3
Given a database $\mathcal{D}$ for a schema $\mathcal{G}$, let $\mathcal{B}$ be a loosely-exact repair for $\mathcal{D}$, and $\overline{\mathcal{B}} = \mathcal{B} \cap \mathcal{D}$. There is a CM-complete repair $\mathcal{B}' \subseteq \overline{\mathcal{B}}$ for $\mathcal{D}$ if at least one of the following restrictions holds:

I $\mathcal{G}$ contains DCs only (no INDs);
II $\mathcal{G}$ contains INDs only (no DCs);
III $\mathcal{G}$ contains KDs and FKs only, and $\mathcal{D}$ is consistent w.r.t. KDs;
IV $\mathcal{G}$ contains KDs and SFKs only;

Proof
Case I: By Proposition 2, since $\mathcal{B}$ is consistent w.r.t. DCs, then $\overline{\mathcal{B}} \subseteq \mathcal{B}$ is consistent as well. Now, if $\mathcal{B} - \mathcal{D} \neq \emptyset$, then we would have a contradiction because $\overline{\mathcal{B}} \ominus \mathcal{D} \subset \mathcal{B} \ominus \mathcal{D}$ would hold. Thus, $\mathcal{B} - \mathcal{D} = \emptyset$ and so, $\mathcal{B} = \overline{\mathcal{B}}$ is already a *CM-complete* repair itself.

Case II: Since there is no DC, there exists only one *CM-complete* repair, say $\mathcal{B}'$, obtained from $\mathcal{D}$ after removing all the facts violating INDs. Now, if $\mathcal{B}'$ was not contained in $\mathcal{B}$, then, by Proposition 2, $\mathcal{B}' \cup \mathcal{B}$ would still be consistent, that is a larger *CM-complete* repair. Contradiction. Finally $\overline{\mathcal{B}} = \mathcal{B}'$.

Case III: Since $\mathcal{D}$ is consistent w.r.t. DCs, we have only one *CM-complete* repair, say $\mathcal{B}'$, obtained from $\mathcal{D}$ after removing all the facts violating INDs. But, as in case II, if the set $\mathcal{B}' - \overline{\mathcal{B}}$ was nonempty, then we could add all these facts into $\mathcal{B}$ without violating any IND. Anyway, one of these facts, say f, could violate a DC due to a fact f' in $\mathcal{B} - \mathcal{D}$. Now, note that f' is in $\mathcal{B}$ only for fixing an IND violation. But in this case, as we are only considering FKs, there would be no reason to have f' in $\mathcal{B}$ instead of f. So, we could (safely) replace f with f' in $\mathcal{B}$ and no KD would be violated as well as no FK. But this leads to a contradiction. So, there is no fact in $\mathcal{B}'$ which is not in $\overline{\mathcal{B}}$.

Case IV: First of all, we observe that if $\mathcal{B} - \mathcal{D} = \emptyset$, then either $\overline{\mathcal{B}}$ is a *CM-complete* repair or $\mathcal{B}$ is not a *loosely-exact* repair. So the statement holds. Now assume that $\mathcal{B} - \mathcal{D} \neq \emptyset$. We distinguish three different cases:

(1) $\overline{\mathcal{B}}$ is both consistent and maximal (it is a *CM-complete* repair);
(2) $\overline{\mathcal{B}}$ is consistent but not maximal (it is not a *CM-complete* repair);
(3) $\overline{\mathcal{B}}$ is inconsistent (it is not a *CM-complete* repair).

In case (1), we have a contradiction because $\mathcal{B}$ is assumed to be a *loosely-exact* repair, but it does not minimize the symmetric difference with $\mathcal{D}$ since $\overline{\mathcal{B}} \ominus \mathcal{D} \subset \mathcal{B} \ominus \mathcal{D}$.

In case (2), we have again a contradiction because $\mathcal{B}$ is assumed to be a *loosely-exact* repair but it does not minimize the symmetric difference with $\mathcal{D}$ since there is a *CM-complete* repair $\widetilde{\mathcal{B}} \supset \overline{\mathcal{B}}$ such that $\widetilde{\mathcal{B}} \ominus \mathcal{D} \subset \mathcal{B} \ominus \mathcal{D}$.

In case (3), we observe that since, by hypothesis, $\mathcal{B}$ is consistent, then the inconsistency of $\overline{\mathcal{B}}$ arises, by Proposition 2, only due to INDs. Now, assume that (i) $\overline{\mathcal{B}}$ contains a fact $r_1(t_1)$; (ii) there is an IND d of the form $\forall \bar{\mathbf{x}}_\forall\ [\ r_1(\bar{\mathbf{x}}_1) \rightarrow \exists \bar{\mathbf{x}}_{2\exists}\ r_2(\bar{\mathbf{x}}_2)\ ]$; (iii) there is no fact for r_2 in $\overline{\mathcal{B}}$ satisfying d. This means that a fact of the form $r_2(t_2)$ must be in $\mathcal{B} - \mathcal{D}$, where $t_1^{\pi_L^d} = t_2^{\pi_R^d}$.

Now, we claim that there is no fact of the form $r_2(t_3)$ in $\mathcal{D} - \overline{\mathcal{B}}$, where $t_1^{\pi_L^d} = t_3^{\pi_R^d}$. Suppose that $\mathcal{D} - \overline{\mathcal{B}}$ contained such a fact $r_2(t_3)$. Consider the new database $(\mathcal{B} \cup \{r_2(t_3)\}) - \{r_2(t_2)\}$. This would necessarily be consistent because the addition of $r_2(t_3)$ (after removing $r_2(t_2)$ as well) cannot violate any KD since d is an FK (remember that $key(r_2) = \pi_R^d$), and cannot violate any IND since each IND d' of the form $r_2 \rightarrow r_3$ is an SFK (remember that $key(r_2) \supseteq \pi_L^{d'}$). But this is not possible because $\mathcal{B}$ is assumed to be a *loosely-exact* repair, and $(\mathcal{B} \cup \{r_2(t_3)\}) - \{r_2(t_2)\}$ would improve the symmetric difference. This means, that each *CM-complete* repair cannot contain the tuple $r_1(t_1)$ (this goes in the direction of the statement).

Let us call $\overline{\mathcal{B}}'$ the consistent (w.r.t. both KDs and SFKs) database obtained from $\overline{\mathcal{B}}$ after removing all the facts violating some IND. It remains to show that there is no other fact in $\mathcal{D} - \overline{\mathcal{B}}$ such that $\overline{\mathcal{B}}' \cup \{r_1(t_1)\}$ does not violate any constraint. Assume that such a fact $r_1(t_1)$ exists, then:

- $\overline{\mathcal{B}}' \cup \{r_1(t_1)\}$ would not violate any IND;
- $\mathcal{B} \cup (\overline{\mathcal{B}}' \cup \{r_1(t_1)\}) = \mathcal{B} \cup \{r_1(t_1)\}$ would not violate any IND, by Proposition 2;
- $\mathcal{B} \cup \{r_1(t_1)\}$ would violate some KD, since $\mathcal{B}$ is a *loosely-exact* repair.

Thus, there would necessarily be a fact in $\mathcal{B}$, say $r_1(t_2)$, being not in $\overline{\mathcal{B}}'$, with the same key of $r_1(t_1)$. Since such a fact cannot stay in $\overline{\mathcal{B}} - \overline{\mathcal{B}}'$ because it does not violate any IND, then it must be in $\mathcal{B} - \mathcal{D}$. But this is not possible because we could replace $r_1(t_2)$ by $r_1(t_1)$ in $\mathcal{B}$ without violating any KD and also without violating any IND, since we are only considering SFKs. But since $\mathcal{B}$ is already a repair, this is clearly a contradiction. Finally, $\overline{\mathcal{B}}'$ is a *CM-complete* repair. □

Corollary 3
$ans_{loosely\text{-}exact}(q, \mathcal{G}, \mathcal{D}) = ans_{CM\text{-}complete}(q, \mathcal{G}, \mathcal{D})$ in the following cases:

- $\mathcal{G}$ contains DCs only (no INDs);
- $\mathcal{G}$ contains INDs only (no DCs);
- $\mathcal{G}$ contains KDs and FKs only, and $\mathcal{D}$ is consistent w.r.t. KDs;
- $\mathcal{G}$ contains KDs and SFKs only;

Proof
This directly follows by both Theorem 3 and Lemma 5, in light of Definition 1. □

Proposition 3
In general, Theorem 3 does not hold in case $\mathcal{G}$ contains SFSKs and KDs only.

Proof
Consider a database containing two relations of arity 2, namely: r and s. Moreover, the schema contains the following ICs: $key(r) = \{1, 2\}$, and $key(s) = \{1\}$ and $r(X, Y) \rightarrow s(X, Y)$. Note that, the last is a safe FSK. Suppose also that a DB $\mathcal{D}$ for this schema contains the following facts: $r(a, b)$, $s(a, c)$. The *loosely-exact* repairs are $\mathcal{B}_1 = \{s(a, c)\}$ and $\mathcal{B}_2 = \{r(a, b), s(a, b)\}$, but only the first one is also a *CM-Complete* repair. However, $\overline{\mathcal{B}} = \mathcal{B}_2 \cap \mathcal{D} = \{r(a, b)\}$ is not a CM-complete repair (it is inconsistent). The only consistent database contained in $\overline{\mathcal{B}}$ is the empty set that is not a *CM-Complete* repair (deletions are not minimized). □

3 Computation of CQA via ASP

In this section, we show how to exploit *Answer Set Programming* (ASP) (Gelfond and Lifschitz 1988; Gelfond and Lifschitz 1991) for efficiently computing consistent answers to user queries under different semantic assumptions. ASP is a powerful logic programming paradigm allowing (in its general form) for disjunction in rule heads (Minker 1982) and nonmonotonic negation in rule bodies. In the following, we assume that the reader is familiar with ASP with aggregates, and in particular we adopt the DLV syntax (Faber et al. 2010; Leone et al. 2006).

The suitability of ASP for implementing CQA has been already recognized in the literature (Lenzerini 2002; Arenas et al. 2003; Bertossi et al. 2005; Chomicki and Marcinkowski 2005). The general approaches are based on the following idea: produce an ASP program P whose answer sets represent possible repairs, so that the problem of computing CQA corresponds to cautious reasoning on P. One of the hardest challenges in this context is the automatic identification of a program P considering a minimal number of repairs actually relevant to answering user queries.

In order to face these challenges, we first introduce a general encoding which unifies in a common core the solutions for CQA under the semantics considered in this paper. Then, based on this unified framework, we define optimization strategies precisely aiming at reducing the computational cost of CQA. This is done in several ways: *(i)* by casting down the original program to complexity-wise easier programs; *(ii)* by identifying portions of the database not requiring repairs at all, according to the query requirements; *(iii)* exploiting equivalence classes between some semantics in such a way to adopt optimized solutions.

We next present the general encoding first and, then, the optimizations.

3.1 General Encoding

The general approach generates a program Π_{cqa} and a new query q_{cqa} obtained by rewriting both the constraints and the query q in such a way that CQA reduces to cautious reasoning on Π_{cqa} and q_{cqa}. Recall that a union of conjunctive queries in ASP is expressed as a set of rules having the same head predicate with the same arity.

In what follows, we first present how to generate Π_{cqa} and q_{cqa} and then formally prove under which hypothesis cautious reasoning on such Π_{cqa} and q_{cqa} corresponds to CQA.

Given a database $\mathcal{D}$ for a schema $\mathcal{G}$ and a query q on $\mathcal{G}$, the ASP program Π_{cqa} is created by rewriting each IC belonging to $constr(\mathcal{G})$ and q as follows:

Denial Constraints. Let $\Sigma \in \{$*CM-complete, loosely-sound, loosely-exact*$\}$. For each DC of the form $\forall \bar{\mathbf{x}}_1, \ldots, \bar{\mathbf{x}}_m \ \neg[g_1(\bar{\mathbf{x}}_1) \wedge \ldots \wedge g_m(\bar{\mathbf{x}}_m) \wedge \sigma(\bar{\mathbf{x}}_1, \ldots, \bar{\mathbf{x}}_m)]$ in $constr(\mathcal{G})$, insert the following rule into Π_{cqa}:

- $g_1^c(\bar{\mathbf{x}}_1) \vee \cdots \vee g_m^c(\bar{\mathbf{x}}_m) \coloneq g_1(\bar{\mathbf{x}}_1), \cdots, g_m(\bar{\mathbf{x}}_m), \sigma(\bar{\mathbf{x}}_1, \ldots, \bar{\mathbf{x}}_m)\cdot$

This rule states that in presence of a violated denial constraint it must be guessed the tuple(s) to be removed in order to repair the database.

Inclusion dependencies. Let $\Sigma = \{$*CM-complete, loosely-exact*$\}$. For each IND d in $constr(\mathcal{G})$ of the form $\forall \bar{\mathbf{x}}_\forall\ [\ g_1(\bar{\mathbf{x}}_1) \rightarrow \exists \bar{\mathbf{x}}_{2\exists}\ g_2(\bar{\mathbf{x}}_2)\ ]$, add the following rules into Π_{cqa}:

- $g_1^c(\bar{\mathbf{x}}_1) \coloneq g_1(\bar{\mathbf{x}}_1),\ \#\mathsf{count}\{\bar{\mathbf{x}}_{2\exists} : g_2^c(\bar{\mathbf{x}}_2)\} = \#\mathsf{count}\{\bar{\mathbf{x}}_{2\exists} : g_2(\bar{\mathbf{x}}_2)\}\cdot$ if $|\bar{\mathbf{x}}_{2\exists}| > 0$
- $g_1^c(\bar{\mathbf{x}}_1) \coloneq g_1(\bar{\mathbf{x}}_1),\ g_2^c(\bar{\mathbf{x}}_2)\cdot$
  $g_1^c(\bar{\mathbf{x}}_1) \coloneq g_1(\bar{\mathbf{x}}_1),\ not\ g_2(\bar{\mathbf{x}}_2)\cdot$ if $|\bar{\mathbf{x}}_{2\exists}| = 0$

The first rule states that a tuple of g_1 must be deleted iff either all the tuples in g_2 previously referred to by g_1 via d have been deleted due to the repairing process, or there is no tuple in g_2 referred to by g_1 via d. (This is done by comparing the total count of tuples in g_2 and g_2^c). Observe that if there is a cyclic set of INDs, the set of rules generated by this rewriting would contain recursive aggregates. Their semantics is described in (Faber et al. 2010). The latter two rules replace the first one in the special case of $|\bar{\mathbf{x}}_{2\exists}| = 0$.

Repaired Relations. Let $\Sigma \in \{$*CM-complete, loosely-sound, loosely-exact*$\}$. For each relation name $g \in names(\mathcal{G})$, insert the following rule into Π_{cqa}:

- $g^r(\bar{\mathbf{x}}) \coloneq g(\bar{\mathbf{x}}),\ not\ \ g^c(\bar{\mathbf{x}})\cdot$

Query rewriting. Build $q_{cqa}(\bar{\mathbf{x}})$ from $q(\bar{\mathbf{x}})$ as follows:

1. If $\Sigma =$ *loosely-sound*, then apply onto q the perfect rewriting algorithm that deals with INDs described in (Calì et al. 2003b)[1].
2. For each atom $g(\bar{\mathbf{y}})$ in q, replace $g(\bar{\mathbf{y}})$ by $g^r(\bar{\mathbf{y}})$

The perfect rewriting introduced in (Calì et al. 2003b) is intuitively described next. Given a query $q(\bar{\mathbf{x}})$ and a set of INDs, the algorithm iteratively computes a new query Q as follows. Q is first initialized with q; then, at each iteration it carries out the following two steps: *(1)* For each conjunction cq' in Q, and for each pair of atoms g_1, g_2 in cq' that unify (i.e., for which there exists a substitution transforming g_1 into g_2), g_1 and g_2 are substituted

[1] Observe that, when $\Sigma =$ *loosely-sound*, INDs are not encoded into logic rules.

by one single unifying atom. *(2)* For each conjunction cq' in Q, and for each *applicable* IND d of the form $g_1 \rightarrow g$ such that g is in cq', it adds to Q a new conjunction cq'' obtained from cq' by interpreting d as a rewriting rule on g, applied from right to left. The algorithm stops when no further modifications are possible on Q with the two steps above.

The following theorems show how and when cautious reasoning on Π_{cqa} and q_{cqa} correspond to CQA. First we consider the CM-complete semantics.

Theorem 4
Let $\Sigma =$ *CM-complete*, let $\mathcal{D}$ be a database for a schema $\mathcal{G}$ with arbitrary DCs and (possibly cyclic) INDs, and let q be a union of conjunctive queries. $t \in ans_\Sigma(q, \mathcal{G}, \mathcal{D})$ iff $q_{cqa}(t)$ is a cautious consequence of the ASP program $\mathcal{D} \cup \Pi_{cqa}$.

Proof
We claim that Π_{cqa} allows to consider only and all the repairs, exactly one per model. Let $\mathcal{B}^r$ be a repair. In the following, we describe how to obtain a model containing for each relation, say g, exactly only and all the tuples of g that do not appear in $\mathcal{B}^r$. We collect such tuples in the new relation g^c, while we collect in g^r only and all the tuples of g appearing in $\mathcal{B}^r$. For each relation, say g:

(a) By the disjunctive rules (if any) involving g, of the form

$$\cdots \vee g^c(\bar{\mathbf{x}}) \vee \cdots \ \ :\!-\ \ \cdots,\ g(\bar{\mathbf{x}}),\ \cdots,\ \sigma(\cdots, \bar{\mathbf{x}}, \cdots)\cdot$$

we guess a set of tuples of g, collected in g^c, that must not appear in $\mathcal{B}^r$.

(b) Next, for each IND of the form $g(\bar{\mathbf{x}}_1) \rightarrow g_1(\bar{\mathbf{x}}_2)$ (involving g in the left-hand side), we use the rule

$$g^c(\bar{\mathbf{x}}_1) :\!-\ g(\bar{\mathbf{x}}_1),\ \#\mathsf{count}\{\bar{\mathbf{x}}_{2\exists} : g_1^c(\bar{\mathbf{x}}_2)\} = \#\mathsf{count}\{\bar{\mathbf{x}}_{2\exists} : g_1(\bar{\mathbf{x}}_2)\}\cdot$$

for deciding which tuples of g cannot appear in $\mathcal{B}^r$ due to an IND violation. Note that in case $|\bar{\mathbf{x}}_{2\exists}| = 0$, the rule is rewritten without the #count aggregate.

(c) Finally, by the rule $g^r(\bar{\mathbf{x}}) :\!-\ g(\bar{\mathbf{x}}),\ not\ \ g^c(\bar{\mathbf{x}})$ we obtain the repaired relations.

Importantly, for computing the extension of each g^c we only exploit the minimality of answer sets semantics; later, the extension of each g^r is computed. Observe that, by the splitting theorem (Lifschitz and Turner 1994) Π_{cqa} can be divided (split) into two parts . It is clear that, by construction, Π_{cqa} has exactly one answer set per repair. Finally, the query is reorganized to exploit the repaired relations, and cautious reasoning does the rest. □

Example 3
Consider again Example 2, the program (and the query built from $q(X) :\!-\ m(X)$) under the *CM-complete* semantics obtained for it, is:

- $e^c(X_c, X_n) \vee e^c(X_c, X_n') :\!-\ e(X_c, X_n),\ e(X_c, X_n'),\ X_n \neq X_n'\cdot$
- $m^c(X_c) :\!-\ m(X_c),\ \#\mathsf{count}\{X_n' : e^c(X_c, X_n')\} = \#\mathsf{count}\{X_n : e(X_c, X_n)\}\cdot$
- $e^r(X_c, X_n) :\!-\ e(X_c, X_n),\ not\ \ e^c(X_c, X_n)\cdot$
- $m^r(X_c) :\!-\ m(X_c),\ not\ \ m^c(X_c)\cdot$
- $q_{cqa}(X_c) :\!-\ m^r(X_c)\cdot$

When this program is evaluated on the database we obtain four answer sets. It can be verified that, all the answer sets contain $m^r($*'e1'*$)$ and $m^r($*'e2'*$)$, (i.e., they are cautious consequences of Π_{cqa}) and, thus, *'e1'* and *'e2'* are the consistent answers to the query. □

Theorem 5
Let $\Sigma = $ *loosely-sound*, let $\mathcal{D}$ be a database for a schema $\mathcal{G}$ with KDs (and exactly one key for each relation) and (possibly cyclic) NKC INDs, and let q be a union of conjunctive queries without comparison atoms[2]. $t \in ans_{\Sigma}(q, \mathcal{G}, \mathcal{D})$ iff $q_{cqa}(t)$ is a cautious consequence of the ASP program $\mathcal{D} \cup \Pi_{cqa}$.

Proof
Considerations analogous to the *CM-complete* case can be drawn. Disjunctive rules guess a minimal set of tuples to be removed, whereas the perfect rewriting algorithm allows to deal with NKC INDs. Observe that, the separation theorem introduced in (Calì et al. 2003b) shows that INDs can be taken into account as if the KDs where not expressed on $\mathcal{G}$; in particular, it states that it is sufficient to compute the perfect rewriting q' of q and evaluate q' on the maximal subsets of $\mathcal{D}$ consistent with KDs. In our case, these are computed by the part of Π_{cqa} dealing with KDs, whereas the separation is carried out by renaming each g in q' by g^r. □

The general encoding for the *loosely-exact* semantics is inherently more complex than the ones for *loosely-sound* and *CM-complete*, since both tuple deletions and tuple insertions are subject to minimization. As a consequence, we tackled the *loosely-exact* encoding by considering that there are common cases in which CQA under the *loosely-exact* semantics and the *CM-complete* semantics actually coincide (see Corollary 3). These cases can be easily checked and, thus, it is possible to handle the *loosely-exact* semantics with the encoding defined for the *CM-complete* case.

Theorem 6
Let $\Sigma = $ *loosely-exact*, $\mathcal{D}$ be a database for a schema $\mathcal{G}$ such that one of the following holds:

- $\mathcal{G}$ contains DCs only (no INDs);
- $\mathcal{G}$ contains INDs only (no DCs);
- $\mathcal{G}$ contains KDs and FKs only, and $\mathcal{D}$ is consistent w.r.t. KDs;
- $\mathcal{G}$ contains KDs and SFKs only;

Let q be a union of conjunctive queries. $t \in ans_{\Sigma}(q, \mathcal{G}, \mathcal{D})$ iff $q_{cqa}(t)$ is a cautious consequence of the ASP program $\mathcal{D} \cup \Pi_{cqa}$.

Proof
Follows from Corollary 3 and Theorem 5. □

3.2 Optimized Solution

The strategy reported in the previous section is a general solution for solving the CQA problem but, in several cases, more efficient ASP programs can be produced. First of all, note that the general algorithm blindly considers all the ICs on the global schema, including those that have no effect on the specific query. Consequently, useless logic rules might

[2] Recall that equalities are expressed in terms of variables having the same name.

be produced which may slow down program evaluation. Then, a very simple optimization may consist of considering relevant ICs only. However, there are several cases in which the complexity of CQA stays in **PTIME**; but disjunctive programs, for which cautious reasoning becomes a hard task (Eiter et al. 1997), are generated even in presence of denial constraints only. This means that the evaluation of the produced logic programs might be much more expensive than required in those "easy" cases. In the following, we provide semantic-specific optimizations aiming to overcome such problems for the settings pointed out in Theorem 4, Theorem 5, and Theorem 6.

Given a query q and an atom g in q, we define the set of *relevant indices* of g in q, say $relevant(q, g)$ in such a way that an index i in $[1..arity(g)]$ belongs to $relevant(q, g)$ if at least one of the following holds for an occurrence $g(X_1, \ldots, X_n)$ of g in q:

- X_i is not existentially quantified (it is a free variable, it is an output variable of q);
- X_i is involved in some comparison atom (even if it is existentially quantified);
- X_i appears more than once in the same conjunction;
- X_i is a constant value;

If g does not appear in q, we say that $relevant(q, g) = \emptyset$;

In the following, we denote by π a set of indices. Moreover, given a sequence of variables $\bar{\mathbf{x}}$ and a set $\pi \subseteq \{1, \ldots, |\bar{\mathbf{x}}|\}$, we denote by $\bar{\mathbf{x}}^\pi$ the sequence obtained from $\bar{\mathbf{x}}$ by discarding a variable if its position is not in π. Finally, given a relation name g, a set of indices π and a label ℓ we denote by $g^{\ell\text{-}\pi}(\bar{\mathbf{x}}^\pi)$ an auxiliary atom derived from g, marked by ℓ, and using only variables in $\bar{\mathbf{x}}^\pi$.

$\Sigma =$ ***loosely-sound***. The objective of this optimization is to single out, for each relation involved by the query, the set of attributes actually relevant to answer it and apply the necessary repairs only on them. As we show next, this may allow both to reduce (even to zero) the number of disjunctive rules needed to repair key violations and to reduce the cardinality of relations involved in such disjunctions.

Given a schema $\mathcal{G}$ and a query q, perform the following steps for building the program Π_{cqa} and the query Q_{cqa}.

1. Apply the the perfect rewriting algorithm that deals with INDs described in (Calì et al. 2003b).
2. Let Q be the union of conjunctive queries obtained from q after Step 1. For each $g \in names(\mathcal{G})$, build the sets

$$\pi_R^g = relevant(Q, g) \qquad \pi_S^g = \pi_R^g \cup key(g)$$

These two sets capture the fact that a key attribute is relevant for the repairing process, but it may not be strictly relevant for answering the query.

Observe that the perfect rewriting dealing with INDs must be applied *before* singling out relevant attributes. In fact, q may depend, through INDs, also on attributes of relations not explicitly mentioned in it. However, in the last step of this algorithm the rewriting of the query is completed by substituting each relation in the query with its repaired (and possibly reduced) version.

3. For each $g \in names(\mathcal{G})$ such that $\pi_R^g \neq \emptyset$ and $key(g) \not\supseteq \pi_R^g$, add the following rules into Π_{cqa}:

- $g^{sr\text{-}\pi_S^g}(\bar{\mathbf{x}}^{\pi_S^g})$:– $g(\bar{\mathbf{x}})$.
- $g^{c\text{-}\pi_S^g}(\bar{\mathbf{x}}_1^{\pi_S^g}) \vee g^{c\text{-}\pi_S^g}(\bar{\mathbf{x}}_2^{\pi_S^g})$:– $g^{sr\text{-}\pi_S^g}(\bar{\mathbf{x}}_1^{\pi_S^g}),\ g^{sr\text{-}\pi_S^g}(\bar{\mathbf{x}}_2^{\pi_S^g}),\ \bar{\mathbf{x}}_1^i \neq \bar{\mathbf{x}}_2^i$. $\forall i \in \pi_S^g - key(g)$
- $g^{r\text{-}\pi_R^g}(\bar{\mathbf{x}}^{\pi_R^g})$:– $g^{sr\text{-}\pi_S^g}(\bar{\mathbf{x}}^{\pi_S^g}),\ not\ \ g^{c\text{-}\pi_S^g}(\bar{\mathbf{x}}^{\pi_S^g})$.

Observe that if there exists at least one relevant non-key attribute for g, the repairing process can not be avoided; however, violations caused by irrelevant attributes only (i.e, not in π_S^g) can be ignored, since the projection of g on π_S^g is still safe and sufficient for query answering purposes.

4. For each $g \in names(\mathcal{G})$ such that $\pi_R^g \neq \emptyset$ and $key(g) \supseteq \pi_R^g$, add the following rule into Π_{cqa}:
   - $g^{r\text{-}\pi_R^g}(\bar{\mathbf{x}}^{\pi_R^g})$:– $g(\bar{\mathbf{x}})$.

Observe that, if the relevant attributes of g are a subset of its key, the repair process of g for key violations through disjunction can be avoided at all. In fact, the projection of g on π_R^g is still safe and sufficient for query answering purposes. Moreover, for the same reason, it is not needed to take all the key of g into account.

5. For each atom of the form $g(\bar{\mathbf{x}})$ in Q, replace $g(\bar{\mathbf{x}})$ by $g^{r\text{-}\pi_R^g}(\bar{\mathbf{x}}^{\pi_R^g})$.

Σ = ***CM-complete***. For the optimization of the CM-complete semantics, we exploit a graph which is used to navigate the query and the database in order to single out those relations and projections actually relevant for answering the query. Moreover, it allows to identify possible cycles generated by ICs which must be suitably handled; in fact, acyclic ICs induce a partial order among them and this information can be effectively exploited for the optimization. On the contrary cyclic ICs must be handled in a more standard way.

Given a schema $\mathcal{G}$ and a query q, build the directed labelled graph $G_q = \langle N, A\rangle$ as follows:

- $N = \{q\} \cup names(\mathcal{G})$;
- $(g_1, g_2, c) \in A$ iff c is a DC in $constr(\mathcal{G})$ involving both g_1 and g_2;
- $(g_1, g_2, d) \in A$ iff d is an IND in $constr(\mathcal{G})$ of the form $g_1 \rightarrow g_2$;
- $(q, g, \varepsilon) \in A$ iff g appears in a conjunction of q.

Perform the following steps for building program Π_{cqa}:

1. Visit G_q starting from node q;
2. Discard unreachable nodes and update the sets N and A;
3. Partition the set N in (N_{cf}, N_{ncf}) in such a way that a node n belongs to N_{cf} if it is not involved in any cycle (q always belongs to N_{cf}). Contrariwise, a node n belongs to N_{ncf} if it is involved in some cycle.
4. For each node $g \in N - \{q\}$ compute the sets

   $\pi_R^g = (\bigcup_{(g_L, g, d) \in A} \pi_R^d) \cup relevant(q, g)$;
   $\pi_S^g = \pi_R^g \cup key(g)$, only if g has exactly one primary key as DCs; $\pi_S^g = \emptyset$ otherwise.

   here π_R^g is the set of relevant variable indices of g, and π_S^g adds to π_R^g the key of g.

Observe that Steps 1–4 implement a pre-processing phase in which relevant relations and their relevant indices are singled out, and each relevant relation is classified as cycle free or non cycle free.

5. For each node $g \in N_{cf}$, if g has only one key as DCs, then add the following rules into Π_{cqa}:

   - $g^{\xi\text{-}\pi^g_\chi}(\bar{\mathbf{x}}^{\pi^g_\chi}) :- \ g(\bar{\mathbf{x}}),\ g_1^{r\text{-}\pi^{d_1}_R}(\bar{\mathbf{x}}_1^{\pi^{d_1}_R}),\ \ldots,\ g_k^{r\text{-}\pi^{d_k}_R}(\bar{\mathbf{x}}_k^{\pi^{d_k}_R}).$
   - $g_i^{r\text{-}\pi^{d_i}_R}(\bar{\mathbf{x}}_i^{\pi^{d_i}_R}) :- \ g_i^{r\text{-}\pi^{g_i}_R}(\bar{\mathbf{x}}_i^{\pi^{g_i}_R}).$ $\forall i \in [1..k]$ s.t. $\pi^{g_i}_R \supset \pi^{d_i}_R$

   where:

   - $k \geq 0$ is the number of arcs in G_q labelled by INDs, and outgoing from g;
   - the pair (ξ, χ) is either (r, R) or (sr, S), according to whether $key(g) \supseteq \pi^g_R$ or not, respectively. Intuitively, if $key(g) \supseteq \pi^g_R$ holds, then the repair $g^{r\text{-}\pi^g_R}$ of g can be directly computed; otherwise the computation must first go through a semi-reparation step for computing $g^{sr\text{-}\pi^g_S}$. Intuitively, this semi-reparation step collects those tuples that violate no IND of the form $g \rightarrow g_i$, but that must be anyway processed in order to fix some key violation (see Steps 6 - 10).
   - atom $g_i^{r\text{-}\pi^{d_i}_R}$ is in the body of the first rule ($1 \leq i \leq k$) only if both $(g, g_i, d_i) \in A$, and d_i is an IND of the form $g(\bar{\mathbf{x}}) \rightarrow g_i(\bar{\mathbf{x}}_i)$. This atom is just a projection of $g_i^{r\text{-}\pi^{g_i}_R}(\bar{\mathbf{x}}_i^{\pi^{g_i}_R})$.

6. For each node $g \in N_{cf}$ if g has only one primary key as DCs, and $key(g) \subset \pi^g_R$, and g has incoming arcs only from q, and all the relevant variables of g w.r.t. q are in the head of q, and each occurrence of g in q contains all of its relevant variables, then add the following rules into Π_{cqa} by considering that the key of g is defined by rules of the form $\forall \bar{\mathbf{x}}_1, \bar{\mathbf{x}}_2\ \neg[g(\bar{\mathbf{x}}_1) \wedge g(\bar{\mathbf{x}}_2) \wedge \bar{\mathbf{x}}^i_1 \neq \bar{\mathbf{x}}^i_2]$:

   - $g^{c\text{-}\pi^g_S}(\bar{\mathbf{x}}_1^{\pi^g_S}) :- \ g^{sr\text{-}\pi^g_S}(\bar{\mathbf{x}}_1^{\pi^g_S}),\ g^{sr\text{-}\pi^g_S}(\bar{\mathbf{x}}_2^{\pi^g_S}),\ \bar{\mathbf{x}}^i_1 \neq \bar{\mathbf{x}}^i_2.$ $\forall i \in \pi^g_S - key(g)$
   - $g^{r\text{-}\pi^g_R}(\bar{\mathbf{x}}_1^{\pi^g_R}) :- \ g^{sr\text{-}\pi^g_S}(\bar{\mathbf{x}}_1^{\pi^g_S}),\ not\ \ g^{c\text{-}\pi^g_S}(\bar{\mathbf{x}}_1^{\pi^g_S}).$

7. For each node $g \in N_{cf}$ if g has only one primary key as DCs, and $key(g) \not\supseteq \pi^g_R$, and case 6 does not apply, then add the following rules into Π_{cqa} by considering that the key is defined by rules of the form, $\forall \bar{\mathbf{x}}_1, \bar{\mathbf{x}}_2\ \neg[g(\bar{\mathbf{x}}_1) \wedge g(\bar{\mathbf{x}}_2) \wedge \bar{\mathbf{x}}^i_1 \neq \bar{\mathbf{x}}^i_2]$:

   - $g^{c\text{-}\pi^g_S}(\bar{\mathbf{x}}_1^{\pi^g_S}) \vee g^{c\text{-}\pi^g_S}(\bar{\mathbf{x}}_2^{\pi^g_S}) :- \ g^{sr\text{-}\pi^g_S}(\bar{\mathbf{x}}_1^{\pi^g_S}),\ g^{sr\text{-}\pi^g_S}(\bar{\mathbf{x}}_2^{\pi^g_S}),\ \bar{\mathbf{x}}^i_1 \neq \bar{\mathbf{x}}^i_2.$
     $\forall i \in \pi^g_S - key(g)$
   - $g^{r\text{-}\pi^g_R}(\bar{\mathbf{x}}_1^{\pi^g_R}) :- \ g^{sr\text{-}\pi^g_S}(\bar{\mathbf{x}}_1^{\pi^g_S}),\ not\ \ g^{c\text{-}\pi^g_S}(\bar{\mathbf{x}}_1^{\pi^g_S}).$

   Observe that, in this case, disjunctive rules are defined only on the set of relevant indices that are not in the key and that each $g^{c\text{-}\pi^g_S}$ contains only the projection of deleted tuples on the set π^g_S.

Here, Steps 5–7 handle relations for which a key is defined and are classified as cycle free. In particular, if $key(g) \supseteq \pi^g_R$ holds, key reparation can be avoided at all (and thus disjunctive rules too); otherwise a semi-reparation step is required, but Step 6 identifies further cases in which even if key reparation is needed, disjunction can be still avoided. Finally, Step 7 handles all the other cases. Importantly, through Steps 5-7 we take into account only the minimal projections of involved relations in order to reduce as much as possible computational costs (and even disjunctive rules) not considering irrelevant attributes.

8. For each node $g \in N_{ncf}$ add the following rules into Π_{cqa}:

   - $g^c(\bar{\mathbf{x}}) \text{ :- } g(\bar{\mathbf{x}}),\ not\ \ g_1^{r\text{-}\pi_R^d}(\bar{\mathbf{x}}_1^{\pi_R^d})$.
     $g_1^{r\text{-}\pi_R^d}(\bar{\mathbf{x}}_1^{\pi_R^d}) \text{ :- } g_1^{r\text{-}\pi_R^g}(\bar{\mathbf{x}}_1^{\pi_R^g})$.
     for each IND d of the form $g(\bar{\mathbf{x}}) \rightarrow g_1(\bar{\mathbf{x}}_1)$ such that there is no cycle in G_q involving both g_1 and g;
   - $g^c(\bar{\mathbf{x}}) \text{ :- } g(\bar{\mathbf{x}}),\ \#\text{count}\{\bar{\mathbf{x}}_{1\exists} : g_1^c(\bar{\mathbf{x}}_1)\} = \#\text{count}\{\bar{\mathbf{x}}_{1\exists} : g_1(\bar{\mathbf{x}}_1)\}\cdot$
     for each IND d of the form $\forall \bar{\mathbf{x}}_\forall\ [\ g(\bar{\mathbf{x}}) \rightarrow \exists \bar{\mathbf{x}}_{2\exists}\ g_1(\bar{\mathbf{x}}_1)\ ]$ such that $g_1 \in N_{ncf}$;
   - $g^c(\bar{\mathbf{x}}_1) \vee g^c(\bar{\mathbf{x}}_2) \text{ :- } g(\bar{\mathbf{x}}_1),\ g(\bar{\mathbf{x}}_2),\ \bar{\mathbf{x}}_1^i \neq \bar{\mathbf{x}}_2^i\cdot$ $\forall i \in \pi$
     where $\pi = \{1, \ldots, arity(g)\} - key(g)$ and the key of g is defined by DCs of the form $\forall \bar{\mathbf{x}}_1, \bar{\mathbf{x}}_2\ \neg[g(\bar{\mathbf{x}}_1) \wedge g(\bar{\mathbf{x}}_2) \wedge \bar{\mathbf{x}}_1^i \neq \bar{\mathbf{x}}_2^i]$;
   - $g^{r\text{-}\pi_R^g}(\bar{\mathbf{x}}^{\pi_R^g}) \text{ :- } g(\bar{\mathbf{x}}),\ not\ \ g^c(\bar{\mathbf{x}})$.
     if there is at least one node in N_{cf} with an arc to g, or g appears in q;

9. For each DC of the form $\forall \bar{\mathbf{x}}_1, \ldots, \bar{\mathbf{x}}_m\ \ \neg[g_1(\bar{\mathbf{x}}_1) \wedge \ldots \wedge g_m(\bar{\mathbf{x}}_m) \wedge \sigma(\bar{\mathbf{x}}_1, \ldots, \bar{\mathbf{x}}_m)]$ involving at least two different relation names (entailing that each $g_i \in N_{ncf}$), add the following rules into Π_{cqa}:

   - $g_1^c(\bar{\mathbf{x}}_1) \vee \cdots \vee g_m^c(\bar{\mathbf{x}}_m) \text{ :- } g_1(\bar{\mathbf{x}}_1), \cdots, g_m(\bar{\mathbf{x}}_m), \sigma(\bar{\mathbf{x}}_1, \ldots, \bar{\mathbf{x}}_m)\cdot$

Steps 8 and 9 handle non cycle free relations; the repairing process in this case mimics the standard rewriting, but projects relations on the relevant attributes whenever possible.

10. For each node $g \in N_{cf}$ if g is involved in DCs that do not form a primary key, then add the following rules into Π_{cqa}:

    - $g^{sr}(\bar{\mathbf{x}}) \text{ :- } g(\bar{\mathbf{x}}),\ g_1^{r\text{-}\pi_R^{d_1}}(\bar{\mathbf{x}}_1^{\pi_R^{d_1}}),\ \ldots,\ g_k^{r\text{-}\pi_R^{d_k}}(\bar{\mathbf{x}}_k^{\pi_R^{d_k}})$.
    - $g_i^{r\text{-}\pi_R^{d_i}}(\bar{\mathbf{x}}_i^{\pi_R^{d_i}}) \text{ :- } g_i^{r\text{-}\pi_R^{g_i}}(\bar{\mathbf{x}}_i^{\pi_R^{g_i}})$. $\forall i \in [1..k]$ s.t. $\pi_R^{g_i} \supset \pi_R^{d_i}$
    - $g^c(\bar{\mathbf{x}}_1) \vee \cdots \vee g^c(\bar{\mathbf{x}}_m) \text{ :- } g^{sr}(\bar{\mathbf{x}}_1), \cdots, g^{sr}(\bar{\mathbf{x}}_m), \sigma_d(\bar{\mathbf{x}}_1, \ldots, \bar{\mathbf{x}}_m)\cdot$ $\forall d$
    - $g^{r\text{-}\pi_R^g}(\bar{\mathbf{x}}^{\pi_R^g}) \text{ :- } g^{sr}(\bar{\mathbf{x}}),\ not\ \ g^c(\bar{\mathbf{x}})$.

    where:

    - $k \geq 0$ is the number of arcs, labelled by INDs, outgoing from g;
    - atom $g_i^{r\text{-}\pi_R^{d_i}}$ is in the body of the first rule ($1 \leq i \leq k$) iff both $(g, g_i, d_i) \in A$ and d_i is an IND of the form $g(\bar{\mathbf{x}}) \rightarrow g_i(\bar{\mathbf{x}}_i)$;
    - d is a DC of the form $\forall \bar{\mathbf{x}}_1, \ldots, \bar{\mathbf{x}}_m\ \ \neg[g(\bar{\mathbf{x}}_1) \wedge \ldots \wedge g(\bar{\mathbf{x}}_m) \wedge \sigma_d(\bar{\mathbf{x}}_1, \ldots, \bar{\mathbf{x}}_m)]$

Step 10 handles the special case in which there is no key for a relation but denial constraints are defined (only) on it.

11. For each atom of the form $g(\bar{\mathbf{x}})$ in q, replace $g(\bar{\mathbf{x}})$ by $g^{r\text{-}\pi_R^g}(\bar{\mathbf{x}}^{\pi_R^g})$.

Example 4

Consider again Example 1; suppose to extend the global schema by adding the relation $c(code, name)$ which represents the list of customers, where $code$ is the primary key of c. Moreover, suppose that we ask for the query $q(X_c, X_n) :- c(X_c, X_n), e(X_c, X_n)$ retrieving the customers that are also employees of the bank. In this case, after building the graph G_q it is easy to see that m is unreachable (so it is discarded) and that both c and e comply with the requirements described at Steps 5 and 6 of the optimized algorithm. Consequently, the optimized program under the *CM-complete* semantics is:

$$
\begin{array}{l}
e^{sr\text{-}1,2}(X_c, X_n) :- e(X_c, X_n). \quad c^{sr\text{-}1,2}(X_c, X_n) :- c(X_c, X_n).\\
e^{c\text{-}1,2}(X_c, X_n) :- e^{sr\text{-}1,2}(X_c, X_n),\ e^{sr\text{-}1,2}(X_c, X'_n),\ X_n \neq X'_n.\\
c^{c\text{-}1,2}(X_c, X_n) :- c^{sr\text{-}1,2}(X_c, X_n),\ c^{sr\text{-}1,2}(X_c, X'_n),\ X_n \neq X'_n.\\
e^{r\text{-}1,2}(X_c, X_n) :- e^{sr\text{-}1,2}(X_c, X_n),\ not\ e^{c\text{-}1,2}(X_c, X_n).\\
c^{r\text{-}1,2}(X_c, X_n) :- c^{sr\text{-}1,2}(X_c, X_n),\ not\ c^{c\text{-}1,2}(X_c, X_n).\\
q_{cqa}(X_c, X_n) :- c^{r\text{-}1,2}(X_c, X_n),\ e^{r\text{-}1,2}(X_c, X_n).
\end{array}
$$

Note that, since both e and c are not affected by IND violations, and they have no irrelevant variables, the semi-reparation step cannot actually discard tuples. However, the obtained program is non-disjunctive and stratified. Thus, it can be evaluated in polynomial time (Leone et al. 2006).

In this case, the only answer set of the program contains the consistent answers to the original query. □

$\Sigma =$ ***loosely-exact***. In Section 3.1 we proved that there are common cases in which CQA under the *loosely-exact* semantics and the *CM-complete* semantics actually coincide. As a consequence, in these cases, all the optimizations defined for the *CM-complete* semantics apply also to the *loosely-exact* semantics.

4 Experiments

In this section we present some of the experiments we carried out to assess the effectiveness of our approach to consistent query answering.

Testing has been performed by exploiting our complete system for data integration, which is intended to simplify both the integration system design and the querying activities by exploiting a user-friendly GUI. Indeed, this system both supports the user in designing the global schema and the mappings between global relations and source schemas, and it allows to specify user queries over the global schema via a QBE-like interface. The query evaluation engine adopted for the tests is DLV^{DB} (Terracina et al. 2008) coupled, via ODBC, with a PostgreSQL DBMS where input data were stored. DLV^{DB} is a DLP evaluator born as a database oriented extension of the well known DLV system (Leone et al. 2006). It has been recently extended for dealing with unstratified negation, disjunction and external function calls.

We first address tests on a real world scenario and then report on tests for scalability issues on synthetic data.

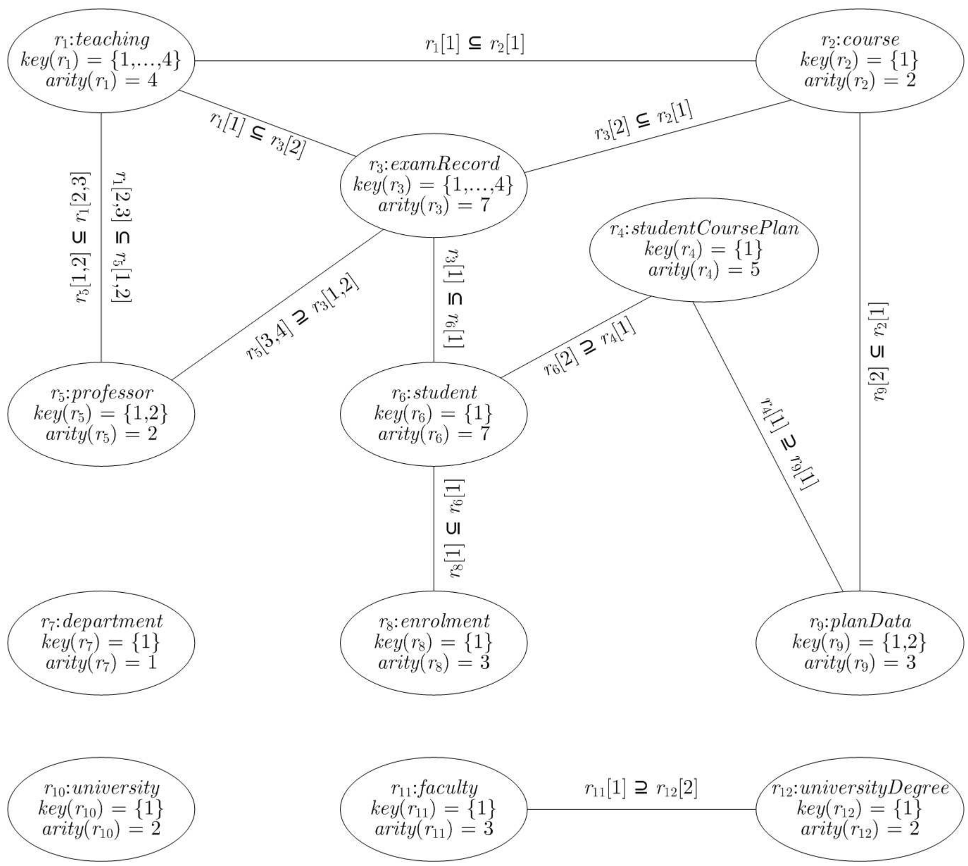


Fig. 1. INFOMIX database.

4.1 Tests on a real world scenario

Data Set. We have exploited the real-world data integration framework developed in the INFOMIX project (IST-2001-33570) (Leone et al. 2005) which integrates data from a real university context. In particular, considered data sources were available at the University of Rome "La Sapienza". These comprise information on students, professors, curricula and exams in various faculties of the university.

There are about 35 data sources in the application scenario, which are mapped into 12 global schema relations with 20 GAV mappings and 21 integrity constraints. We call this data set **Infomix** in the following. Figure 1 reproduces the main characteristics of the global database: each node corresponds to a global relation showing its arity and key. An edge between r_1 and r_2 labelled by $r_1[I] \subseteq r_2[J]$ indicates an IND of the form $\forall \bar{\mathbf{x}}_\forall \; [\; r_1(\bar{\mathbf{x}}_1) \rightarrow \exists \bar{\mathbf{x}}_{2\exists} \; r_2(\bar{\mathbf{x}}_2) \;]$ where I and J are the positions of $\bar{\mathbf{x}}_\forall$ in $\bar{\mathbf{x}}_1$ and $\bar{\mathbf{x}}_2$, respectively; the arc is labelled with the attributes of a and b involved in the IND. Observe that there are cyclic INDs involving *teaching*, *exam_record* and *professor*.

Besides the original source database instance (which takes about 16Mb on DBMS), we obtained bigger instances artificially. Specifically, we generated a number of copies of the original database; each copy is disjoint from the other ones but maintains the same data correlations between instances as the original database. This has been carried out by mapping each original attribute value to a new value having a copy-specific prefix.

Then, we considered two further datasets, namely **Infomix-x-10** and **Infomix-x-50** storing 10 copies (for a total amount of 160Mb of data) and 50 copies (800Mb) of the original database, respectively. It holds that **Infomix** $\subset$ **Infomix-x-10** $\subset$ **Infomix-x-50**.

Compared Methods and Tested Queries. In order to assess the characteristics of the proposed optimizations, we measured the execution time of different queries with *(i)* the standard encoding (identified as **STD** in the following), *(ii)* a naïve optimization obtained by only removing relations not strictly needed for answering the queries (**OPT1** in the following), and *(iii)* the fully optimized encoding presented in Section 3 (**OPT2** in the following). Each of these cases has been evaluated for the three semantics considered in this paper. In order to isolate the impact of our optimizations, we disabled other optimizations (like magic sets) embedded in the datalog evaluation engine. Clearly, such optimizations are complementary to our own and might further improve the overall performances.

Tested queries are as follows:

```
Q1(X1) :- course(X2,X1), plan_data(PL,X2,_),
   student_course_plan(PL,"09089903",_,_,_).
Q2(X1) :- university(X1,_).
Q3(X1,X2,X3) :- university_degree(X1,X2), faculty(X2,_,X3).
Q4(X1,X2,X3) :- student(S,_,X1,_,_,_,_), enrollment(S,_,_),
   exam_record(S,_,_,X2,X3,_,_), S == "09089903".
Q5(X1,X2) :- student_r(S1,_,X1,_,_,_,_), exam_record_r(S1,C,_,_,_,_,_),
   student_r(S2,_,X2,_,_,_,_), exam_record_r(S2,C,_,_,_,_,_),
   S1 == "09089470", S1<>S2.
Q6(X1,X2,X3) :- student(X1,_,_,_,_,_,_), exam_record(X1,_,_,X2,X3,_,_),
   X1 == "09089903".
```

Observe that `Q2` involves key constraints only, `Q1`, and `Q3` involve both keys and acyclic INDs; specifically, `Q3` involves a SFK while `Q1` involves NKC INDs. Finally, `Q4`, `Q5` and `Q6` involve keys and cyclic NKC INDs.

Results and discussion. All tests have been carried out on an Intel Xeon X3430, 2.4 GHz, with 4 Gb Ram, running Linux Operating System. We set a time limit of 120 minutes after which query execution has been killed. Figures 2 and 3 show obtained results for the *loosely-sound* and the *CM-complete* semantics. It is worth recalling that, as we pointed out in Section 3.2, optimizations for the *loosely-exact* semantics are inherent to the equivalence classes to the *CM-complete* semantics discovered in this paper. As a consequence, we tested this semantics only on queries Q2 and Q3 for which such equivalence holds. Then, since the execution times of the optimized encoding coincide with the *CM-complete* graphs for queries Q2 and Q3, we do not report specific figures for them.

Analyzing the figures, we observe that: the proposed optimizations do not introduce computational overhead and, in most cases, transform practically untractable queries in tractable ones; in fact, for all the tested queries the execution time of the standard rewriting exceeded the time limit. **OPT1** helps mostly on the smallest data set; in fact for **Infomix-x-10** it shows some gain in 33% of cases and only in two cases for **Infomix-x-50**.

As for the comparison among the optimized encodings, we can observe that if INDs are not involved by the query (`Q2`) the *loosely-sound* and the *CM-complete* optimizations

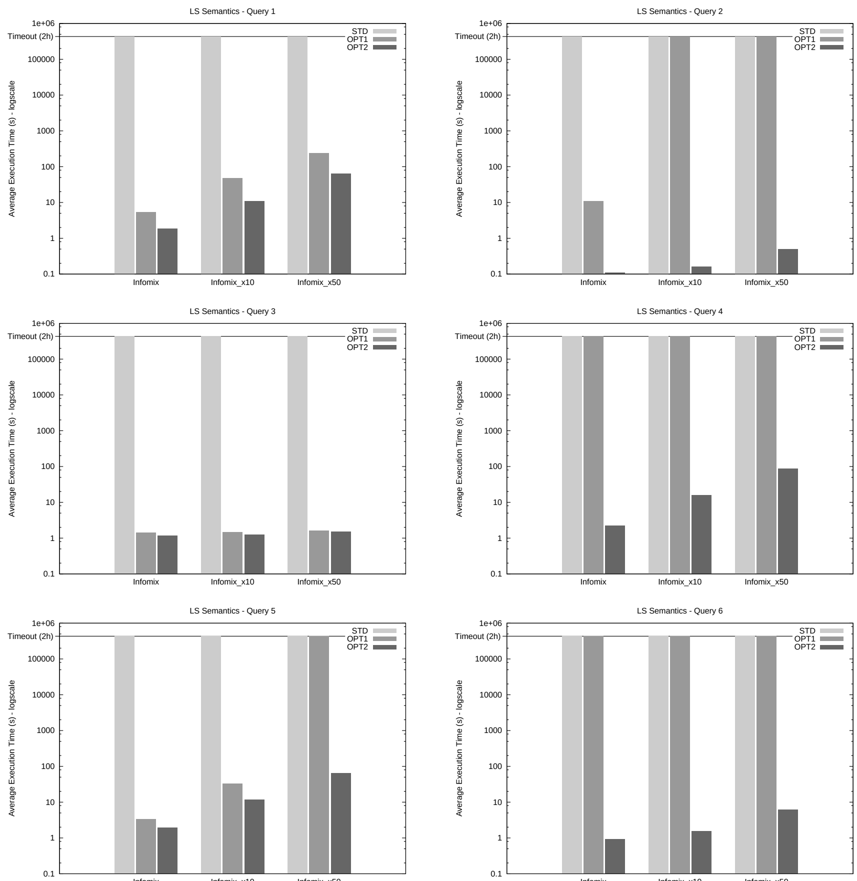


Fig. 2. Query evaluation execution times for the *loosely-sound* semantics.

have the same performances; this confirms theoretical expectations. When acyclic INDs are involved (`Q1`, `Q3`), the *loosely-sound* optimization performs slightly better because the *CM-complete* must choose the tuples to be deleted due to IND violations, whereas the *loosely-sound* semantics just works on the original data. Finally, when involved INDs are cyclic (`Q4`, `Q5`, `Q6`) the performance of the *CM-complete* optimization further degrades w.r.t. the *loosely-sound* one because recursive aggregates must be exploited to choose deletions and, this, increases the complexity of query evaluation.

4.2 Scalability analysis w.r.t. the number and kind of constraint violations

Since, in the real world scenario emerged that the *CM-complete* semantics is more affected than the *loosely sound* one from the kind of involved constraints, we carried out a scalability analysis on this semantics, whose results are reported next.

We considered a synthetic data set composed of three relations named r_1, r_2, and r_3 over which we imposed different sets of ICs in order to analyze the scalability of our methods depending on the presence of keys and/or in presence/absence of acyclic and cyclic INDs.

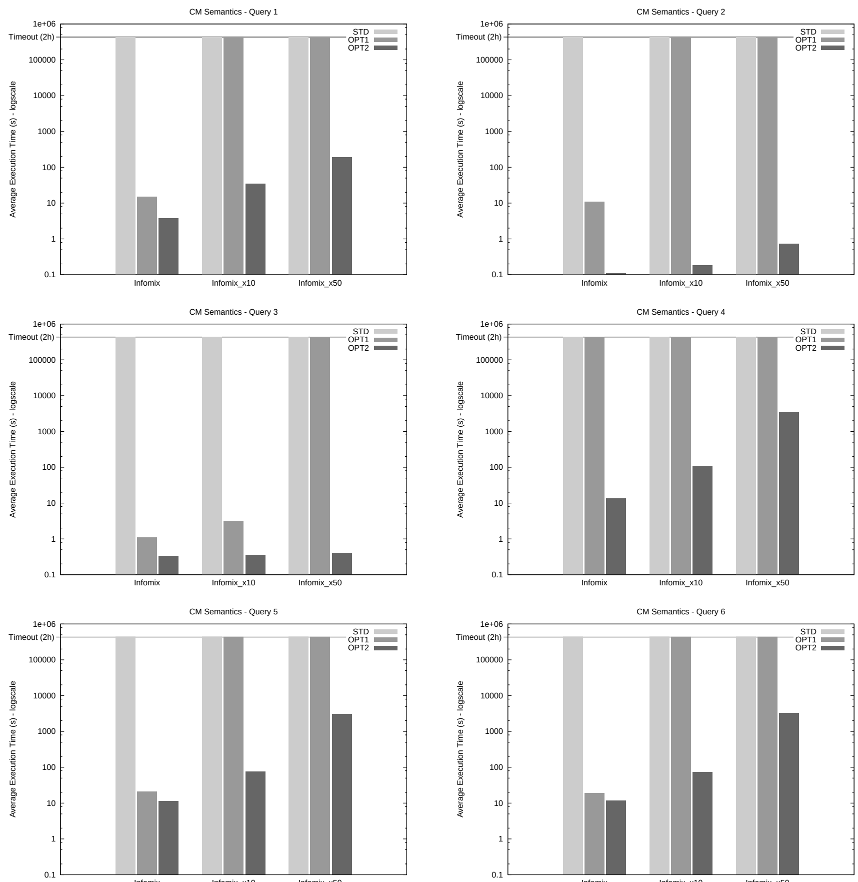
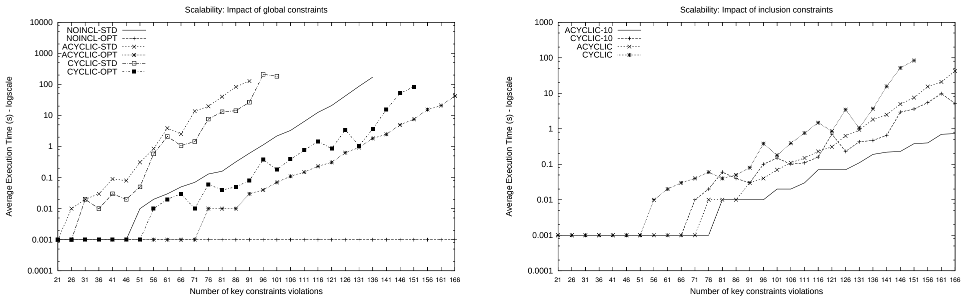


Fig. 3. Query evaluation execution times for the *CM-Complete* semantics.

In particular, we imposed the following key constraints: $key(r_2) = \{1, 2\}$, $key(r_3) = \{1\}$, and we experimented with three different sets of INDs: $NOINCL = \emptyset$, $ACYCLIC = \{r_1(X_1, X_2, X_3, X_4) \rightarrow r_2(X_2, X_5, X_3, X_6), r_1(X_1, X_2, X_3, X_4) \rightarrow r_3(X_1, X_5, X_6, X_7)\}$ and $CYCLIC = ACYCLIC \cup \{r_2(X_1, X_2, X_3, X_4) \rightarrow r_1(X_5, X_6, X_7, X_2)\}$. The employed query is: $query(X1, X3) :- r_1(X1, X2, X3, X4), r_2(X2, X3, X5, X6)$? We have randomly generated synthetic databases having a growing number of key violations on table r_2. The generation process progressively adds key violations to r_2 by generating pairs of conflicting tuples; after an instance of r_2 is obtained, tables r_1 and r_3 are generated by taking values from r_2 in such a way that INDs are satisfied. In addition, for each tuple of r_3 a key-conflicting tuple is generated. In order to assess the impact of the number of INDs violations, for each database instance DB_x, containing x key violations on table r_2, we generated a DB_x-10 instance where the 10% of tuples is (randomly) removed from tables r_1 and r_3 (causing INDs violations). We have generated six database instances per size (number of key violations on table r_2), and plotted the time (averaged over the instances of the same size) in Figure 4.

In detail, Figure 4(a) shows the results for incrementally higher KD violations with



(a) (b)

Fig. 4. Scalability Analysis

no IND violations. Both standard and optimized encodings have been tested. Figure 4(b) compares the optimized encoding only, when the percentage of IND violations is 0% or 10%. Observe that, in general, even when there is no initial IND violation, the KD repairing process may induce some of them.

The analysis of these figures shows that even if cyclic INDs are generally harder, their scaling is almost the same as the acyclic ones. On the contrary, in the absence of INDs the optimization may boost the performances (see the flat line in Figure 4(a)). Figure 4(b) points out that when the number of IND violations increases, the performance may improve. This behavior is justified by the fact that tuple deletions due to IND repairs may, in their turn, remove KD violations. This reduces the number of disjunctions to be evaluated.

5 Related work and concluding remarks

From the 90ies – when the founding notions of *CQA* (Bry 1997), *GAV mapping* (Garcia-Molina et al. 1997; Tomasic et al. 1998; Goh et al. 1999), and *database-repair* (Arenas et al. 1999) were introduced – *data integration* (Lenzerini 2002) and *inconsistent databases* (Bertossi et al. 2005) have been studied quite in depth.

Detailed characterizations of the main problems arising in a data integration system have been provided, taking into account different semantics, constraints, and query types (Calì et al. 2003a; Calì et al. 2003b; Arenas et al. 2003; Chomicki and Marcinkowski 2005; Grieco et al. 2005; Fuxman and Miller 2007; Eiter et al. 2008).

This paper provides a contribution in this scenario by extending the decidability boundaries for the *loosely-exact* semantics (as called in Calì et al. 2003a but firstly introduced by Arenas et al. 1999) and the *loosely-sound* semantics, in case of both KDs and SFSK INDs.

A first proposal of an unifying framework for CQA in a Data Integration setting is presented in (Calì et al. 2005) using first-order logic; it considers different semantics defined by interpreting the mapping assertions between the global and the local schemas of the data integration system. A common framework for computing repairs in a single database setting is proposed in (Eiter et al. 2008); it covers a wide range of semantics relying on the general notion of preorder for candidate repairs, but only universally quantified constraints are allowed. Moreover, the authors introduce an abstract logic programming framework to compute consistent answers. Finally, the authors propose an optimization strategy called factorization that, as will be clarified below, is orthogonal to our own.

This paper provides a contribution in this setting since it unifies different semantics, as in (Calì et al. 2005) and (Eiter et al. 2008), but also provides an algorithm that, given a retrieved database, a user query q, and a semantics, automatically composes an ASP program capable of computing the consistent answers to q. In particular, our ASP-rewriting offers a natural, compact, and direct way for encoding even hard cases where the CQA problem belongs to the Π_2^p complexity class.

Theoretical studies gave rise to concrete implementations most of which were conceived to operate on some specific semantics and/or constraint types. (Arenas et al. 1999; Calì et al. 2002; Greco and Zumpano 2000; Greco et al. 2001; Calì et al. 2003b; Arenas et al. 2003; Chomicki et al. 2004a; Calì et al. 2004; Chomicki et al. 2004b; Lembo 2004; Grieco et al. 2005; Leone et al. 2005; Fuxman et al. 2005; Fuxman and Miller 2007). As an example, in (Leone et al. 2005) only the *loosely-sound* semantics was supported. In this paper, we provide both a unified framework based on ASP, and a complete system supporting (i) all the three aforementioned significant semantics in case of conjunctive queries and the most commonly used database constraints (KDs and INDs), (ii) specialized optimizations, and (iii) a user-friendly GUI.

Another general contribution of our work comes from a novel optimization technique that, after analyzing the query and localizing a minimal number of relevant ICs, tries to "simplify" their structure to reduce the number of database repairs – as they could be exponentially many (Arenas et al. 2001). Such technique could be classified as "vertical" due to the fact that it reduces (whenever possible) the arity of each active relation (with the effect, e.g., of decreasing the number of key conflicts) without looking at the data. It is orthogonal to other "horizontal" approaches, such as magic-sets (Faber et al. 2007) and factorization (Eiter et al. 2008) which are based on data filtering strategies. In particular, a system exploiting ASP incorporating magic-set techniques for CQA is described in (Marileo and Bertossi 2010). Other approaches complementary to our own are based on first-order rewritings of the query (Arenas et al. 1999; Chomicki and Marcinkowski 2002; Calì et al. 2003b; Grieco et al. 2005; Fuxman and Miller 2007).

The combination of our optimizations with such approaches, and further extensions of decidability boundaries for CQA are some of our future line of research.

Acknowledgments. This work has been partially supported by the Calabrian Region under PIA (Pacchetti Integrati di Agevolazione industria, artigianato e servizi) project DLVSYSTEM approved in BURC n. 20 parte III del 15/05/2009 - DR n. 7373 del 06/05/2009.

References

ABITEBOUL, S., HULL, R., AND VIANU, V. 1995. *Foundations of Databases: The Logical Level.* Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA.

ARENAS, M., BERTOSSI, L., AND CHOMICKI, J. 1999. Consistent query answers in inconsistent databases. In *Proceedings of PODS'99.* ACM, New York, NY, USA, 68–79.

ARENAS, M., BERTOSSI, L., AND CHOMICKI, J. 2001. Scalar Aggregation in FD-Inconsistent Databases. In *Proceedings of ICDT'01.* LNCS, vol. 1973. Springer Berlin / Heidelberg, 39–53.

ARENAS, M., BERTOSSI, L., AND CHOMICKI, J. 2003. Answer sets for consistent query answering in inconsistent databases. *TPLP 3,* 4, 393–424.

BERTOSSI, L. E., HUNTER, A., AND SCHAUB, T., Eds. 2005. *Inconsistency Tolerance*. LNCS, vol. 3300. Springer, Berlin / Heidelberg.

BRY, F. 1997. Query Answering in Information Systems with Integrity Constraints. In *Proceedings of IICIS'97*. Chapman & Hall, Ltd., London, UK, UK, 113–130.

CALÌ, A., CALVANESE, D., DE GIACOMO, G., AND LENZERINI, M. 2002. On the Role of Integrity Constraints in Data Integration. *IEEE Data Eng. Bull. 25,* 3, 39–45.

CALÌ, A., CALVANESE, D., DE GIACOMO, G., AND LENZERINI, M. 2004. Data integration under integrity constraints. *Inf. Syst. 29,* 2, 147–163.

CALÌ, A., LEMBO, D., AND ROSATI, R. 2003a. On the decidability and complexity of query answering over inconsistent and incomplete databases. In *Proceedings of PODS'03*. ACM, New York, NY, USA, 260–271.

CALÌ, A., LEMBO, D., AND ROSATI, R. 2003b. Query rewriting and answering under constraints in data integration systems. In *Proceedings of IJCAI'03*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 16–21.

CALÌ, A., LEMBO, D., AND ROSATI, R. 2005. A comprehensive semantic framework for data integration systems. *Journal of Algorithms 3,* 2, 308–328.

CHOMICKI, J. AND MARCINKOWSKI, J. 2002. On the Computational Complexity of Consistent Query Answers. *CoRR cs.DB/0204010*, 1–9.

CHOMICKI, J. AND MARCINKOWSKI, J. 2005. Minimal-change integrity maintenance using tuple deletions. *Inf. Comput. 197,* 1-2, 90–121.

CHOMICKI, J., MARCINKOWSKI, J., AND STAWORKO, S. 2004a. Computing consistent query answers using conflict hypergraphs. In *Proceedings of CIKM'04*. ACM, New York, NY, USA, 417–426.

CHOMICKI, J., MARCINKOWSKI, J., AND STAWORKO, S. 2004b. Hippo: A System for Computing Consistent Answers to a Class of SQL Queries. In *Advances in Database Technology - EDBT 2004*. LNCS, vol. 2992. Springer Berlin / Heidelberg, 661–662.

EITER, T., FINK, M., GRECO, G., AND LEMBO, D. 2008. Repair localization for query answering from inconsistent databases. *ACM TODS 33,* 2, 10:1–10:51.

EITER, T., GOTTLOB, G., AND MANNILA, H. 1997. Disjunctive datalog. *ACM TODS 22,* 3, 364–418.

FABER, W., GRECO, G., AND LEONE, N. 2007. Magic Sets and their application to data integration. *JCSS 73,* 4, 584–609.

FABER, W., PFEIFER, G., AND LEONE, N. 2010. Semantics and complexity of recursive aggregates in answer set programming. *Artificial Intelligence In Press, Corrected Proof*, 1–21.

FUXMAN, A., FAZLI, E., AND MILLER, R. J. 2005. ConQuer: efficient management of inconsistent databases. In *Proceedings of SIGMOD'05*. ACM, New York, NY, USA, 155–166.

FUXMAN, A. AND MILLER, R. J. 2007. First-order query rewriting for inconsistent databases. *JCSS 73,* 4, 610–635. Special Issue: Database Theory 2005.

GARCIA-MOLINA, H., PAPAKONSTANTINOU, Y., QUASS, D., RAJARAMAN, A., SAGIV, Y., ULLMAN, J., VASSALOS, V., AND WIDOM, J. 1997. The TSIMMIS Approach to Mediation: Data Models and Languages. *JIIS 8,* 2, 117–132.

GELFOND, M. AND LIFSCHITZ, V. 1988. The Stable Model Semantics for Logic Programming. In *Proceedings of ICLP/SLP'88*. MIT Press, 1070–1080.

GELFOND, M. AND LIFSCHITZ, V. 1991. Classical Negation in Logic Programs and Disjunctive Databases. *New Gen. Comput. 9,* 3-4, 365–385.

GOH, C. H., BRESSAN, S., MADNICK, S., AND SIEGEL, M. 1999. Context interchange: new features and formalisms for the intelligent integration of information. *ACM TOIS 17,* 3, 270–293.

GRECO, G., GRECO, S., AND ZUMPANO, E. 2001. A Logic Programming Approach to the Integration, Repairing and Querying of Inconsistent Databases. In *Proceedings of ICLP'01*. Number 17 in LNCS. Springer Berlin / Heidelberg, 348–364.

GRECO, S. AND ZUMPANO, E. 2000. Querying inconsistent databases. In *Proceedings of LPAR'00.* Springer-Verlag, Berlin, Heidelberg, 308–325.

GRIECO, L., LEMBO, D., ROSATI, R., AND RUZZI, M. 2005. Consistent query answering under key and exclusion dependencies: algorithms and experiments. In *Proceedings of CIKM'05.* ACM, New York, NY, USA, 792–799.

LEMBO, D. 2004. Dealing with Inconsistency and Incompleteness in Data Integration. Ph.D. thesis, Dipartimento di Informatica e Sistemistica, Universitaà di Roma "La Sapienza".

LENZERINI, M. 2002. Data integration: a theoretical perspective. In *Proceedings of PODS'02.* ACM, New York, NY, USA, 233–246.

LEONE, N., GRECO, G., IANNI, G., LIO, V., TERRACINA, G., EITER, T., FABER, W., FINK, M., GOTTLOB, G., ROSATI, R., LEMBO, D., LENZERINI, M., RUZZI, M., KALKA, E., NOWICKI, B., AND STANISZKIS, W. 2005. The INFOMIX system for advanced integration of incomplete and inconsistent data. In *Proceedings of SIGMOD'05.* ACM, New York, NY, USA, 915–917.

LEONE, N., PFEIFER, G., FABER, W., EITER, T., GOTTLOB, G., PERRI, S., AND SCARCELLO, F. 2006. The DLV System for Knowledge Representation and Reasoning. *ACM TOCL 7,* 3, 499–562.

LEVENE, M. AND VINCENT, M. W. 2000. Justification for Inclusion Dependency Normal Form. *IEEE TKDE 12,* 2, 281–291.

LIFSCHITZ, V. AND TURNER, H. 1994. Splitting a logic program. In *Proceedings of ICLP'94.* MIT Press, Cambridge, MA, USA, 23–37.

MARILEO, M. C. AND BERTOSSI, L. E. 2010. The consistency extractor system: Answer set programs for consistent query answering in databases. *Data Knowl. Eng. 69,* 6, 545–572.

MINKER, J. 1982. On Indefinite Data Bases and the Closed World Assumption. In *Proceedings of CADE'82.* LNCS, vol. 138. Springer, Berlin / Heidelberg, 292–308.

TERRACINA, G., DE FRANCESCO, E., PANETTA, C., AND LEONE, N. 2008. Enhancing a DLP System for Advanced Database Applications. In *Proceedings of RR'08.* LNCS, vol. 5341. Springer, Berlin / Heidelberg, 119–134.

TERRACINA, G., LEONE, N., LIO, V., AND PANETTA, C. 2008. Experimenting with recursive queries in database and logic programming systems. *TPLP 8,* 2, 129–165.

TOMASIC, A., RASCHID, L., AND VALDURIEZ, P. 1998. Scaling Access to Heterogeneous Data Sources with DISCO. *IEEE TKDE 10,* 5, 808–823.