Project Number:	IST-2001-33570
Project Acronym:	INFOMIX
Project Full Name:	Infomix: Boosting Information Integration
Deliverable Number:	D5.1
Title:	**Methods and techniques for query answering**
Authors:	Domenico Lembo, Maurizio Lenzerini, Riccardo Rosati
Workpackage:	WP3
Document Type:	Deliverable
Distribution:	Infomix Consortium
Status:	final
Document file:	WP5_T1_D5.1_0200.ps
Version:	2.0
Number of pages:	40
Due Date:	October 31, 2003
Delivery Date:	November 5, 2003
Partners contributed:	UNIROME1,UNICAL,TUWIEN
Partners owning:	UNIROME1
Man Months:	8

Short Description:
In this document, we investigate techniques, methods and complexity of query answering in the INFOMIX framework. More precisely, we provide algorithms for computing answers to user queries in the presence of the different forms of integrity constraints allowed on the global schema of the Information Integration Model (IIM), under different semantics that allow to deal with incomplete and inconsistent data sources. Furthermore, we establish the computational complexity of the query answering problem in the IIM. Results presented in this document complete the study on query answering that we have already faced in deliverable D3.1, where cases in which the problem is undecidable have been addressed, and D3.3, where also complexity lower bounds have been provided.

Document Change Record		
Version	**Date**	**Reason for Change**
v.1.0	October 23, 2003	First draft
	October 31, 2003	Review from the project partners
v.2.0	November 5, 2003	Final version

Contents

1 Introduction

In this document, we study the problem of computing answers to queries expressed by the user against the global schema of the Information Integration Model (IIM). In particular, we investigate techniques, methods and complexity of query answering in the INFOMIX framework.

As described in our previous deliverables D3.1 [15] and D3.3 [16], the formal framework underlying the IIM of the INFOMIX system accounts for data integration systems consisting of a global schema $\mathcal{G}$, which specifies the global (user) elements, a source schema $\mathcal{S}$, which describes the structure of the data sources in the system, and a mapping $\mathcal{M}$, which specifies the relationship between the sources and the global schema. User queries are posed on the global schema, and the system provides the answers to such queries by exploiting the information supplied by the mapping and accessing the sources that contain relevant data. In our framework, $\mathcal{M}$ can express both the approaches for specifying the mapping commonly adopted in data integration [17]: the global-as-view (GAV) approach, in which global elements are defined as views over the sources, and the local-as-view (LAV), in which, conversely, source elements are characterized as views over the global schema. Furthermore, different semantics assumptions can be adopted on the mapping for specifying different ways of interpreting the data at the sources with respect to data satisfying the corresponding portion of the global schema. More specifically, three "strict" assumptions are considered for such interpretation: the assumption of *sound* mapping, adopted when all data that can be retrieved at the sources satisfy the corresponding portion of the global schema but may result incomplete; the assumption of *complete* mapping, adopted when no data other than those retrieved at the sources satisfy the corresponding portion of global schema, i.e., they are complete but not all sound; the assumption of *exact* mapping, when data are both sound and complete.

Since the global schema is a representation of the domain of interest of the integration application, *integrity constraints* are expressed on such a schema to enhance its expressiveness, thus improving its capability of representing the real world. In the IIM we consider classical *Key Dependencies* (KDs), *Inclusion Dependencies* (IDs), expressing that (a projection of) a relation is included in (a projection of) another relation, and *Exclusion Dependencies* (EDs), expressing that two relations (or projections on them) are disjoint. Since data sources are in general autonomous, extracted data are likely not to satisfy the constraints on the global schema. In this respect, in our framework we adopt suitable relaxations of the original mapping assumptions in order to deal with inconsistency of data at the sources with respect to the constraints on the global schema, thus giving raise to "loose" semantics.

Despite the rich expressiveness allowed, the interaction between such classes of constraints makes query answering difficult, and may lead to non-decidable cases. This problem has been already studied in [15], where we have shown undecidable cases of query answering under both strict and loose semantics, and in [16], where we have provided complexity lower bounds for the other cases. However, establishing the complexity classes for which the query answering problem is complete has been left as an open problem to be addressed in the present deliverable, due to its strict connection with the problem of defining algorithms for query answering in our framework.

In a nutshell, in this document we study query answering under the different semantics and for different combinations of IDs, KDs and EDs. More specifically,

- in order to make this document self contained, we briefly recall here the integration framework underlying the IIM, and the non-decidable cases for query answering already shown in [15] and [16];
- we provide query answering algorithms for decidable cases under both strict and loose semantics;
- we establish the computational complexity of the query answering problem by combining the hardness results shown in [16], and the membership results provided by the query answering algorithms of the present document.

We point out that the results provided here, together with the results of [16], extend our previous studies on inclusion and key dependencies, recently published in [4], to the presence of exclusion dependencies on the global schema.

This document is organized as follows. In Section 2, we present some theoretical background. In Section 3 we describe our formal framework for information integration. Section 4 provides an in-depth investigation of the query answering problem in our framework. For the sake of clarity, we initially abstract from the integration context, and address the problem in the framework of a single relational database (Section 4.1). In particular, we define query answering algorithms for decidable cases under the sound semantics in Section 4.2.1, whereas we address query answering under the loose semantics in Section 4.3.1 and Section 4.3.2. A summary of the complexity results provided in this setting is presented in Section 4.4. Algorithms for query answering in data integration that exploit the algorithms developed in the setting of a single relational database are then discussed in Section 4.5. Finally, Section 5 concludes the paper.

2 Theoretical Background

In this section we recall some theoretical background that will be useful for following discussions. In particular, we present the basic notions of the relational model, on which we will construct our formal framework for data integration, and briefly recall the complexity classes that will be mentioned in this report. This section is intended to be a brief introduction to such matters; for further background we refer the reader to [11, 1, 20].

2.1 The relational Model

We consider to have an infinite, fixed database domain $\mathcal{U}$ whose elements can be referenced by constants $c_1, \ldots, c_n$ under the *unique name assumption*, i.e., different constants denote different real-world objects.

A *relational schema* (or simply *schema*) $\mathcal{RS}$ is a pair $\langle \Psi, \Sigma \rangle$, where:

- Ψ is a set of relations, each with an associated arity that indicates the number of its attributes. The attributes of a relation $r \in \Psi$ of arity n (denoted as r/n) are represented by the integers $1, \ldots, n$;
- Σ is a set of *integrity constraints* expressed on the relations in Ψ, i.e., assertions on the relations in Ψ that are intended to be satisfied by database instances.

A *database instance* (or simply *database*) $\mathcal{DB}$ for a schema $\mathcal{RS} = \langle \Psi, \Sigma \rangle$ is a set of facts of the form $r(t)$ where r is a relation of arity n in Ψ and t is an n-tuple of constants of $\mathcal{U}$. We denote as $r^{\mathcal{DB}}$ the set $\{t \mid r(t) \ \in \mathcal{DB}\}$. A database $\mathcal{DB}$ for a schema $\mathcal{RS}$ is said to be *consistent* with $\mathcal{RS}$ if it satisfies all constraints expressed on $\mathcal{RS}$. The notion of satisfaction depends on the type of constraints defined over the schema.

The integrity constraints that we consider are *inclusion dependencies (IDs)*, *functional dependencies (FDs)*, *key dependencies (KDs)* and *exclusion dependencies (EDs)*. More specifically,

- an *inclusion dependency* is an assertion of the form $r_1[\mathbf{A}] \subseteq r_2[\mathbf{B}]$, where r_1, r_2 are relations in Ψ, $\mathbf{A} = A_1, \ldots, A_n$ $(n \geq 0)$ is a sequence of attributes of r_1, and $\mathbf{B} = B_1, \ldots, B_n$ is a sequence of distinct attributes of r_2. Therefore, we allow for repetition of attributes in the left-hand side of the inclusion[1]. A database $\mathcal{DB}$ for $\mathcal{RS}$ satisfies an inclusion dependency $r_1[\mathbf{A}] \subseteq r_2[\mathbf{B}]$ if for each tuple $t_1 \in r_1^{\mathcal{DB}}$ there exists a tuple $t_2 \in r_2^{\mathcal{DB}}$ such that $t_1[\mathbf{A}] = t_2[\mathbf{B}]$, where $t[\mathbf{A}]$ indicates the projection of the tuple t over $\mathbf{A}$;
- a *functional dependency* is an assertion of the form $r : \mathbf{A} \rightarrow B$[2], where r is a relation in Ψ, $\mathbf{A} = A_1, \ldots, A_n$, is a sequence of distinct attributes of r, and B is an attribute of r. A database $\mathcal{DB}$ for $\mathcal{RS}$ satisfies a functional dependency $r : \mathbf{A} \rightarrow B$ if for each $t_1, t_2 \in r^{\mathcal{DB}}$ such that $t_1[\mathbf{A}] = t_2[\mathbf{A}]$ we have that $t_1[B] = t_2[B]$;
- a *key dependency* is an assertion of the form $\mathit{key}(r) = \mathbf{A}$, where r is a relation in Ψ, and $\mathbf{A} = A_1, \ldots, A_n$ is a sequence of distinct attributes of r. A database $\mathcal{DB}$ for $\mathcal{RS}$ satisfies a key dependency $\mathit{key}(r) = \mathbf{A}$ if for each $t_1, t_2 \in r^{\mathcal{DB}}$ with $t_1 \neq t_2$ we have $t_1[\mathbf{A}] \neq t_2[\mathbf{A}]$. We assume that at most one key dependency is specified for each relation;
- an *exclusion dependency* is an assertion of the form $(r_1[\mathbf{A}] \cap r_2[\mathbf{B}]) = \emptyset$, where r_1 and r_2 are relations in Ψ, $\mathbf{A} = A_1, \ldots, A_n$ and $\mathbf{B} = B_1, \ldots, B_n$ are sequences of attributes of r_1 and r_2, respectively. A database $\mathcal{DB}$ for $\mathcal{RS}$ satisfies an exclusion dependency $(r_1[\mathbf{A}] \cap r_2[\mathbf{B}]) = \emptyset$ if there do not exist two tuples $t_1 \in r_1^{\mathcal{DB}}$ and $t_2 \in r_2^{\mathcal{DB}}$ such that $t_1[\mathbf{A}] = t_2[\mathbf{B}]$.

[1]Repetitions in the right-hand side force equalities between attributes of the relation in left-hand side, hence they imply constraints that we do not consider in this report. See [6, 19] for more details on these dependencies.

[2]Functional dependencies are given in normal form [1], i.e., with a single attribute in the right-hand side. Non-normal form FDs can be always expressed by means of a set of FDs in normal form.

Notice that KDs are particular FDs, and thus they could be expressed in the form that we use for FDs. However, since the class of KDs plays a crucial role in our framework, we make use of the different notations described above.

Sets of inclusion, functional, key and exclusion dependencies expressed on the database schema are denoted by Σ_I, Σ_F, Σ_K, and Σ_E, respectively. Furthermore, we use $\mathcal{RS} = \langle \Psi, \Sigma_I, \Sigma_K, \Sigma_E, \Sigma_F \rangle$ as a shortcut for $\mathcal{RS} = \langle \Psi, \Sigma_I \cup \Sigma_K \cup \Sigma_E \cup \Sigma_F \rangle$. In the absence of some kind of dependencies, we disregard the corresponding symbol. When a database $\mathcal{DB}$ satisfies all dependencies in Σ (resp. Σ_I, Σ_F, Σ_K, or Σ_E), we say that $\mathcal{DB}$ is consistent with Σ (resp. Σ_I, Σ_F, Σ_K, or Σ_E).

A *relational query* (or simply *query*) over $\mathcal{RS}$ is a formula that specifies a set of data to be retrieved from a database. In this document, we consider the classes of conjunctive queries, union of conjunctive queries and DATALOG queries. A *conjunctive query* (CQ) q of arity n over the schema $\mathcal{RS}$ is written in the form

$$q(\vec{\mathbf{x}}) \ \leftarrow \ conj(\vec{\mathbf{x}}, \vec{\mathbf{y}})$$

where

- q belongs to a new set of symbols $\mathcal{Q}$ (the alphabet of queries, that is disjoint from both $\mathcal{U}$ and Ψ);
- $q(\vec{\mathbf{x}})$ is the *head* of the conjunctive query;
- $conj(\vec{\mathbf{x}}, \vec{\mathbf{y}})$ is the *body* of the conjunctive query, which is a conjunction of atoms that involve $\vec{\mathbf{x}} = X_1, \ldots, X_n$ and $\vec{\mathbf{y}} = Y_1, \ldots, Y_m$, where all X_i and Y_j are either variables or constants of $\mathcal{U}$;
- the predicate symbols of the atoms in $conj(\vec{\mathbf{x}}, \vec{\mathbf{y}})$ are in Ψ;
- the number of variables of $\vec{\mathbf{x}}$ is called the *arity* of q, and is the arity of the relation denoted by the query q.

Given a database $\mathcal{DB}$, the answer to q over $\mathcal{DB}$, denoted $q^{\mathcal{DB}}$, is the set of n-tuples of constants $\langle c_1, \ldots, c_n \rangle$, such that, when substituting each c_i for X_i, the formula

$$\exists \vec{\mathbf{y}}. conj(\vec{\mathbf{x}}, \vec{\mathbf{y}})$$

evaluates to true in $\mathcal{DB}$.

A set of conjunctive queries with the same head predicate is a *Union of Conjunctive Queries* (UCQ). More formally, a UCQ is written in the form

$$q(\vec{\mathbf{x}}) \ \leftarrow \ conj_1(\vec{\mathbf{x}}, \vec{\mathbf{y}}_1) \vee \cdots \vee conj_m(\vec{\mathbf{x}}, \vec{\mathbf{y}}_m)$$

The answer to a UCQ q over a database $\mathcal{DB}$, as usually denoted $q^{\mathcal{DB}}$, is the set of n-tuples of constants $\langle c_1, \ldots, c_n \rangle$, such that, when substituting each c_i for X_i, the formula

$$\exists \vec{\mathbf{y}}_1. conj_1(\vec{\mathbf{x}}, \vec{\mathbf{y}}_1) \vee \cdots \vee \exists \vec{\mathbf{y}}_m. conj_m(\vec{\mathbf{x}}, \vec{\mathbf{y}}_m)$$

evaluates to true in $\mathcal{DB}$.

Finally, a DATALOG query is a collection of rules, each having the same form as a conjunctive query, except that predicate symbols in the body of the rules can be in $\mathcal{Q}$ as well. In a DATALOG query, each head predicate of the rules refers to an intermediate relation, and has not to contain predicates referring to database relations. The intermediate predicates are called *Intensional DataBase* (IDB) predicates, whereas predicates referring to stored relations are called *Extensional DataBase* (EDB) predicates. Given a DATALOG query q and a database $\mathcal{DB}$, the answer $q^{\mathcal{DB}}$ of q over $\mathcal{DB}$ is the minimal fixpoint model of q and $\mathcal{DB}$ [1].

Given a CQ, UCQ, or DATALOG query q, we also say that $q^{\mathcal{DB}}$ denotes the set of tuples that *satisfy* q over $\mathcal{DB}$.

2.2 Complexity Classes

We assume that the reader is familiar with the basic notions of computational complexity, and NP-completeness [20, 10]. P^A (NP^A) is the class of problems that are solved in polynomial time by a deterministic (nondeterministic) Turing machine using an oracle for A, i.e., that solves in constant time any problem in A. Furthermore, co-A is the class of problems that are complement of a problem in A. The classes Σ_k^p and Π_k^p of the *Polynomial Hierarchy* are defined as follows:

$$\Sigma_0^p = \Pi_0^p = P \text{ and for all } k \geq 1, \Sigma_k^p = NP^{\Sigma_{k-1}^p} \text{ and } \Pi_k^p = \text{co-}\Sigma_k^p$$

In particular $\Sigma_2^p = NP^{NP}$, and $\Pi_2^p = \text{co-}\Sigma_2^p$, i.e., Σ_2^p is the class of problems that are solved in polynomial time by a nondeterministic Turing machine that uses an NP-oracle, and Π_2^p is the class of problems that are complement of a problem in Σ_2^p.

Finally, PSPACE (NPSPACE) is the class of problems that can be solved by a deterministic (nondeterministic) Turing machine that uses a polynomially bounded amount of memory. By Savitch's theorem [21] it follows that PSPACE=NPSPACE (the theorem actually applies to a much more general class of space bounds).

In this report we consider the complexity of the query languages described in Section 2.1 in different relational settings. In particular we address two kinds of complexity [23]:

- the *data complexity*, which is the complexity with respect to the size of the underlying database instance, i.e., when the relational query and the schema are considered fixed whereas the database instance is considered as an input to the problem, and
- the *combined complexity*, which is the complexity with respect to the size of the database instance, the query and the schema, i.e., when the query, the schema and the database are considered as the input to the problem[3].

[3]The query (or schema) complexity, which is the complexity with respect to the query and the schema (but not the database) is not addressed in this report

3 Formal framework for information integration

We now define a formal framework for data integration, based on the relational model with integrity constraints described in the previous section. We first present the syntax, and then the semantics of a data integration system. For a more detailed description of the framework we refer to our previous deliverable [16].

3.1 Syntax

A data integration system $\mathcal{I}$ is a triple $\langle \mathcal{G}, \mathcal{S}, \mathcal{M} \rangle$, where:

- $\mathcal{G}$ is the *global schema* expressed in the relational model with inclusion, key and exclusion dependencies, i.e., $\mathcal{G} = \langle \Psi, \Sigma \rangle$, where $\Sigma = \Sigma_I \cup \Sigma_K \cup \Sigma_E$;
- $\mathcal{S}$ is the *source schema* expressed in the relational model without integrity constraints, i.e., $\mathcal{S} = \langle \Psi_{\mathcal{S}}, \emptyset \rangle$;
- $\mathcal{M}$ is the *mapping* between $\mathcal{G}$ and $\mathcal{S}$. In our framework we consider both the GAV and the LAV mapping. More precisely,
  - the GAV mapping is a set of assertions of the form $\langle r_{\mathcal{G}}, q_{\mathcal{S}} \rangle$, where $r_{\mathcal{G}}$ is a global relation and $q_{\mathcal{S}}$ is the associated query over the source schema $\mathcal{S}$. In this document, we study the setting in which the language used to express queries in the GAV mapping is non-recursive Datalog;
  - the LAV mapping is a set of assertions of the form $\langle r_{\mathcal{S}}, q_{\mathcal{G}} \rangle$, where $r_{\mathcal{S}}$ is a source relation and $q_{\mathcal{G}}$ is the associated query over the global schema $\mathcal{G}$. We assume that queries in the LAV mapping are Conjunctive Queries.

Finally, a *query over* $\mathcal{I}$ (also called *user query* in the following) is a formula that specifies which data to extract from the integration system. Each user query is issued over the global schema $\mathcal{G}$, and we assume that the language used to specify user queries is Union of Conjunctive Queries.

3.2 Semantics

Intuitively, to define the semantics of a data integration system, we have to start with a set of data at the sources, and we have to specify which are the data that satisfy the global schema with respect to such data at the sources, according to the assumption adopted for the mapping. Thus, in order to assign the semantics to a data integration system $\mathcal{I} = \langle \mathcal{G}, \mathcal{S}, \mathcal{M} \rangle$, we start by considering a *source database for* $\mathcal{I}$, i.e., a database $\mathcal{D}$ for the source schema $\mathcal{S}$. Then, we consider the assumption that can be adopted on the mapping $\mathcal{M}$. As already said in the introduction, we consider here three different assumptions: sound, complete and exact, that allows for different interpretations of data that can be retrieved from the sources with respect to data that satisfy the global schema. We denote the assumption on $\mathcal{M}$ with $as(\mathcal{M})$, and pose $as(\mathcal{M}) = s$, c, or e, for sound, complete, and exact, respectively.

Based on $\mathcal{D}$ and $as(\mathcal{M})$, we now specify what the information content of the global schema $\mathcal{G}$ is. We call any database $\mathcal{B}$ for $\mathcal{G}$ a *global database* for $\mathcal{I}$. Formally, the semantics of $\mathcal{I}$ w.r.t. $\mathcal{D}$ and $as(\mathcal{M})$, is the set of global databases $\mathcal{B}$ for $\mathcal{I}$ such that:

(i) $\mathcal{B}$ is consistent with $\mathcal{G}$;

(ii) $\mathcal{B}$ satisfies $as(\mathcal{M})$ with respect to $\mathcal{D}$, i.e.,

- if $\mathcal{M}$ is GAV, $\mathcal{B}$ satisfies $as(\mathcal{M})$ if for each assertion $\langle r_\mathcal{G}, q_\mathcal{S}\rangle \in \mathcal{M}$ we have that
  (a) $r_\mathcal{G}^\mathcal{B} \supseteq q_\mathcal{S}^\mathcal{D}$ if $as(\mathcal{M}) = s$;
  (b) $r_\mathcal{G}^\mathcal{B} \subseteq q_\mathcal{S}^\mathcal{D}$ if $as(\mathcal{M}) = c$;
  (c) $r_\mathcal{G}^\mathcal{B} = q_\mathcal{S}^\mathcal{D}$ if $as(\mathcal{M}) = e$;
- if $\mathcal{M}$ is LAV, $\mathcal{B}$ satisfies $as(\mathcal{M})$ if for each assertion $\langle r_\mathcal{S}, q_\mathcal{G}\rangle \in \mathcal{M}$ we have that
  (a) $r_\mathcal{S}^\mathcal{D} \subseteq q_\mathcal{G}^\mathcal{B}$ if $as(\mathcal{M}) = s$;
  (b) $r_\mathcal{S}^\mathcal{D} \supseteq q_\mathcal{G}^\mathcal{B}$ if $as(\mathcal{M}) = c$;
  (c) $r_\mathcal{S}^\mathcal{D} = q_\mathcal{G}^\mathcal{B}$ if $as(\mathcal{M}) = e$;

Intuitively, in GAV (and in LAV) the three different assumptions allow us to model different situations in which queries over $\mathcal{S}$ (resp. relations in $\mathcal{S}$) provide (a) any subset of tuples that satisfy the corresponding relation in $\mathcal{G}$ (resp. query over $\mathcal{G}$), (b) any superset of such tuples, or (c) exactly such tuples.

The semantics of $\mathcal{I}$ w.r.t. $\mathcal{D}$ and $as(\mathcal{M})$, is denoted with $sem_{as(\mathcal{M})}(\mathcal{I}, \mathcal{D})$, where $as(\mathcal{M}) = s, c, e$ respectively for the sound, complete, or exact assumption. Obviously, $sem_{as(\mathcal{M})}(\mathcal{I}, \mathcal{D})$ contains in general several global databases for $\mathcal{I}$.

Finally, we give the semantics of queries. Formally, given a source database $\mathcal{D}$ for $\mathcal{I}$ and an assumption $as(\mathcal{M})$ on $\mathcal{M}$, we call *certain answers* (or simply *answers*) to a query q of arity n with respect to $\mathcal{I}$, $\mathcal{D}$ and $as(\mathcal{M})$, the set

$$ans_{as(\mathcal{M})}(q, \mathcal{I}, \mathcal{D}) = \{\langle c_1, \ldots, c_n\rangle \mid \langle c_1, \ldots, c_n\rangle \in q^\mathcal{B} \text{ for each } \mathcal{B} \in sem_{as(\mathcal{M})}(\mathcal{I}, \mathcal{D})\ \}$$

The problem of *query answering* is the problem of computing the set $ans_{as(\mathcal{M})}(q, \mathcal{I}, \mathcal{D})$. It should be easy to see that query answering in data integration systems is essentially a form of reasoning in the presence of incomplete information [22].

Example 3.1 Consider the relational schema $\mathcal{G}_0 = \langle \Psi_0, \Sigma_0\rangle$ where Ψ_0 contains the two relations[4] $player(Pname, Pteam)$ and $team(Tname, Tcity)$, and Σ_0 contains the ID $player[Pteam] \subseteq team[Tname]$, stating that every player is enrolled in a team of a city.

GAV mapping. Let us first construct a GAV data integration system in which $\mathcal{G}_0$ is the global schema, i.e., let us consider $\mathcal{I}_G = \langle \mathcal{G}_0, \mathcal{S}_G, \mathcal{M}_G\rangle$, where $\mathcal{S}_G$ contains three binary relations s_1, s_2 and s_3, and the mapping $\mathcal{M}_G$ is as follows:

$$\begin{array}{rl} \langle player, & player(X,Y) \leftarrow s_1(X,Y) \\ & player(X,Y) \leftarrow s_2(X,Y)\rangle \\ \langle team, & team(X,Y) \leftarrow s_3(X,Y)\rangle. \end{array}$$

[4]For the sake of clarity, in the examples we use names to denote attributes, rather than integers.

Assume to have the source database $\mathcal{D}_G = \{s_1(\mathsf{a},\mathsf{b}), s_1(\mathsf{e},\mathsf{f}), s_2(\mathsf{a},\mathsf{d}), s_3(\mathsf{b},\mathsf{c})\}$ for $\mathcal{I}_G$, where $\mathsf{a},\mathsf{b},\mathsf{c},\mathsf{d},\mathsf{e},\mathsf{f}$ are constants of $\mathcal{U}$. It should be easy to see that $sem_e(\mathcal{I}_G,\mathcal{D}_G) = \emptyset$, since s_3 does not store the cities of the teams f and d. This in turn implies that in this example query answering under the exact semantics is meaningless, since every possible fact is a logical consequence of $\mathcal{I}$ and $\mathcal{D}$: for instance, the answer to the query that asks for all team names in *team*, i.e., $q(x) \leftarrow team(x,y)$, is the whole interpretation domain $\mathcal{U}$ (that is, every possible constant belongs to the extension of the query).

On the other hand, $sem_c(\mathcal{I}_G,\mathcal{D}_G) = \{\{player(\mathsf{a},\mathsf{b}), team(\mathsf{b},\mathsf{c})\}, \{team(\mathsf{b},\mathsf{c})\}, \emptyset\}$. Furthermore, let be $\overline{\mathcal{B}} = \{player(\mathsf{a},\mathsf{b}), player(\mathsf{a},\mathsf{d}), player(\mathsf{e},\mathsf{f}), team(\mathsf{b},\mathsf{c})\}$, $sem_s(\mathcal{I}_G,\mathcal{D}_G)$ contains all global databases that can be obtained by adding to $\overline{\mathcal{B}}$ (among others) at least one fact of the form $team(\mathsf{d},\alpha)$ and one fact of the form $team(\mathsf{f},\beta)$, where α and β are constants of the domain $\mathcal{U}$. Notice that, since $\emptyset \in sem_c(\mathcal{I}_G,\mathcal{D}_G)$, $ans_c(q,\mathcal{RS},\mathcal{D}) = \emptyset$, i.e., there is no answer to the query in the complete semantics, whereas $ans_s(q,\mathcal{I}_G,\mathcal{D}_G) = \{\mathsf{b},\mathsf{d},\mathsf{f}\}$.

LAV mapping. Let us now construct a LAV data integration system $\mathcal{I}_L = \langle \mathcal{G}_0, \mathcal{S}_L, \mathcal{M}_L\rangle$, where $\mathcal{G}_0$ is the same as the GAV system $\mathcal{I}_G$, $\mathcal{S}_L$ contains the binary relation s_3 and s_4 and the mapping $\mathcal{M}_L$ is as follows:

$$\begin{array}{ll}\langle s_4, & s_4(X,Y) \leftarrow player(X,Y)\rangle \\ \langle s_3, & s_3(X,Y) \leftarrow team(X,Y)\rangle.\end{array}$$

Consider the source database $\mathcal{D}_L = \{s_3(\mathsf{b},\mathsf{c}), s_4(\mathsf{a},\mathsf{b}), s_4(\mathsf{e},\mathsf{f}), s_4(\mathsf{a},\mathsf{d}),\}$. It is easy to see that $sem_e(\mathcal{I}_L,\mathcal{D}_L) = sem_e(\mathcal{I}_G,\mathcal{D}_G) = \emptyset$, $sem_s(\mathcal{I}_L,\mathcal{D}_L) = sem_s(\mathcal{I}_G,\mathcal{D}_G)$, and $sem_c(\mathcal{I}_L,\mathcal{D}_L) = sem_c(\mathcal{I}_G,\mathcal{D}_G)$. Obviously, the answers to the query q, under the sound, complete, and exact semantics are the same as the GAV case.

■

3.3 Semantics for Inconsistent Data Sources

Since sources are in general autonomous and heterogeneous, data retrieved from them are likely not to satisfy the constraints expressed on the global schema $\mathcal{G}$. According to the semantics above defined, the situation may arise in which data retrieved from the sources cannot be reconciled in the global schema in such a way that both the constraints expressed on $\mathcal{G}$ and the assumption on the mapping are satisfied, thus the integration system should be regarded as *inconsistent*. In particular, the simple violation of a single dependency (under the sound and exact semantics) may lead to the non-interesting case in which $sem_e(\mathcal{I},\mathcal{D}) = \emptyset$ (see Example 3.1), and $sem_s(\mathcal{I},\mathcal{D}) = \emptyset$, for both the GAV and the LAV mapping. On the other hand, since most of the data could satisfy global constraints, it seems unreasonable to consider the entire system inconsistent and unable to provide significant answers to queries. If we do not want to conclude in these cases that no global database exists in the semantics of a data integration system $\mathcal{I} = \langle \mathcal{G}, \mathcal{S}, \mathcal{M}\rangle$ with respect to a source database $\mathcal{D}$ and an assumption $as(\mathcal{M})$, we need a different characterization of the mapping.

Example 3.1 (contd.) Assume now to specify the key dependency $key(player) = \{Pname\}$ on $\mathcal{G}_0$, stating that a player cannot be enrolled in more than one team. For both the GAV and the LAV approach (hence we adopt now the same symbol $\mathcal{I}_0$ for both $\mathcal{I}_G$ and $\mathcal{I}_L$, and $\mathcal{D}_0$ for both $\mathcal{D}_G$ and $\mathcal{D}_L$), it is now also $sem_s(\mathcal{I}_0, \mathcal{D}_0) = \emptyset$, other than $sem_e(\mathcal{I}_0, \mathcal{D}_0) = \emptyset$. Indeed, every database $\mathcal{B}$ in $sem_s(\mathcal{I}_0, \mathcal{D}_0)$ should contain the facts $player(\mathsf{a}, \mathsf{b})$ and $player(\mathsf{a}, \mathsf{d})$ that are not consistent with the key dependency, and it is not possible to make $\mathcal{B}$ satisfy this constraint by adding other facts to $\mathcal{B}$. On the other hand, each such $\mathcal{B}$ should contain also $team(\mathsf{b}, \mathsf{c})$ and $player(\mathsf{e}, \mathsf{f})$, where $team(\mathsf{b}, \mathsf{c})$ is consistent with the dependencies in the schema, whereas the inconsistency caused by $player(\mathsf{e}, \mathsf{f})$ can be resolved under the sound semantics by adding a suitable fact to $\mathcal{B}$ of the form $team(\mathsf{f}, \alpha)$, where α is a constant of $\mathcal{U}$. Therefore, rather than the whole domain $\mathcal{U}$, the query $q(x) \leftarrow team(x, y)$ should return the answer set $\{\mathsf{b}, \mathsf{f}\}$ under the sound semantics. For the exact assumption, we can show in an analogous way that, taking into account those tuples that can be retrieved from the sources and that do not violate the constraints, the answer set to our query might be $\{\mathsf{b}\}$, for both the LAV and the GAV system. ■

A possible solution to this problem is to characterize the semantics of $\mathcal{I}$ in terms of those databases that

- satisfy the integrity constraints on $\mathcal{G}$, and
- approximate "at best" the satisfaction of the assumptions on $\mathcal{M}$, i.e., are *as close as possible* to the semantics interpretation of $\mathcal{M}$.

In other words, the integrity constraints of $\mathcal{G}$ are considered "hard", whereas the assumptions on $\mathcal{M}$ are considered "soft", and we resort to satisfy the mapping at best when we are not able to satisfy it in rigorous way.

In the spirit of a common approach in the literature on inconsistent databases [9, 18, 2], we now propose a modified definition of the semantics that reflects the above idea. Given a source database $\mathcal{D}$ for $\mathcal{I}$ and an assumption $as(\mathcal{M})$ on the mapping $\mathcal{M}$, we define an ordering on the set of all global databases for $\mathcal{I}$ that are consistent with $\mathcal{G}$. If $\mathcal{B}_1$ and $\mathcal{B}_2$ are two such databases, we define $\mathcal{B}_1 \leq_{\mathcal{D}} \mathcal{B}_2$ as follows:

- if $\mathcal{M}$ is GAV, for each assertion $\langle r_{\mathcal{G}}, q_{\mathcal{S}} \rangle \in \mathcal{M}$
  - (a) $r_{\mathcal{G}}^{\mathcal{B}_1} \cap q_{\mathcal{S}}^{\mathcal{D}} \supseteq r_{\mathcal{G}}^{\mathcal{B}_2} \cap q_{\mathcal{S}}^{\mathcal{D}}$ if $as(\mathcal{M}) = s$;
  - (b) $r_{\mathcal{G}}^{\mathcal{B}_1} - q_{\mathcal{S}}^{\mathcal{D}} \subseteq r_{\mathcal{G}}^{\mathcal{B}_2} - q_{\mathcal{S}}^{\mathcal{D}}$ if $as(\mathcal{M}) = c$;
  - (c) $r_{\mathcal{G}}^{\mathcal{B}_1} \cap q_{\mathcal{S}}^{\mathcal{D}} \supseteq r_{\mathcal{G}}^{\mathcal{B}_2} \cap q_{\mathcal{S}}^{\mathcal{D}}$ and $r_{\mathcal{G}}^{\mathcal{B}_1} - q_{\mathcal{S}}^{\mathcal{D}} \subseteq r_{\mathcal{G}}^{\mathcal{B}_2} - q_{\mathcal{S}}^{\mathcal{D}}$ if $as(\mathcal{M}) = e$;
- if $\mathcal{M}$ is LAV, for each assertion $\langle r_{\mathcal{S}}, q_{\mathcal{G}} \rangle \in \mathcal{M}$
  - (a) $q_{\mathcal{G}}^{\mathcal{B}_1} \cap r_{\mathcal{S}}^{\mathcal{D}} \supseteq q_{\mathcal{G}}^{\mathcal{B}_2} \cap r_{\mathcal{S}}^{\mathcal{D}}$ if $as(\mathcal{M}) = s$;
  - (b) $q_{\mathcal{G}}^{\mathcal{B}_1} - r_{\mathcal{S}}^{\mathcal{D}} \subseteq q_{\mathcal{G}}^{\mathcal{B}_2} - r_{\mathcal{S}}^{\mathcal{D}}$ if $as(\mathcal{M}) = c$;
  - (c) $q_{\mathcal{G}}^{\mathcal{B}_1} \cap r_{\mathcal{S}}^{\mathcal{D}} \supseteq q_{\mathcal{G}}^{\mathcal{B}_2} \cap r_{\mathcal{S}}^{\mathcal{D}}$ and $q_{\mathcal{G}}^{\mathcal{B}_1} - r_{\mathcal{S}}^{\mathcal{D}} \subseteq q_{\mathcal{G}}^{\mathcal{B}_2} - r_{\mathcal{S}}^{\mathcal{D}}$ if $as(\mathcal{M}) = e$;

Furthermore, $\mathcal{B}_1 <_{\mathcal{D}} \mathcal{B}_2$ stands for $\mathcal{B}_1 \leq_{\mathcal{D}} \mathcal{B}_2 \wedge \mathcal{B}_2 \not\leq_{\mathcal{D}} \mathcal{B}_1$.

With this notion in place, we can modify the definition of semantics of a data integration system $\mathcal{I} = \langle \mathcal{G}, \mathcal{S}, \mathcal{M} \rangle$ with respect to a source database $\mathcal{D}$ for $\mathcal{I}$ and an assumption $as(\mathcal{M})$ on $\mathcal{M}$. In order to distinguish between the semantics of Section 3.2 and their modified version, in the following we refer to the former as *strict* semantics, while we call the latter *loose* semantics, and denote them with $sem_{l\,as(\mathcal{M})}(\mathcal{I}, \mathcal{D})$ (where, as usually, $as(\mathcal{M}) \in \{s, c, e\}$). Namely, we call *strictly-sound*, *strictly-complete*, and *strictly-exact* the sound, complete, and exact semantics, whereas we respectively call *loosely-sound*, *loosely-complete*, and *loosely-exact* the three corresponding loose semantics. Formally, a global database $\mathcal{B}$ for $\mathcal{I}$ is in $sem_{l\,as(\mathcal{M})}(\mathcal{I}, \mathcal{D})$, if

(*i*) $\mathcal{B}$ is consistent with $\mathcal{G}$;

(*ii*) $\mathcal{B}$ is minimal with respect to $\leq_{\mathcal{D}}$, i.e., for no other global database $\mathcal{B}'$ for $\mathcal{I}$ consistent with $\mathcal{G}$, we have that $\mathcal{B}' <_{\mathcal{D}} \mathcal{B}$.

The notion of answers to a user query q coincides with the one given for the strict semantics, and we indicate the set of answers to q under the loose semantics with $ans_{l\,as(\mathcal{M})}(q, \mathcal{I}, \mathcal{D})$. It is immediate to verify that, if $sem_{as(\mathcal{M})}(\mathcal{I}, \mathcal{D}) \neq \emptyset$ for any $as(\mathcal{M}) \in \{s, c, e\}$, then the strict semantics and the loose one coincide, in the sense that, for each query q, $ans_{as(\mathcal{M})}(q, \mathcal{I}, \mathcal{D}) = ans_{l\,as(\mathcal{M})}(q, \mathcal{I}, \mathcal{D})$. Consequently, since $sem_c(\mathcal{I}, \mathcal{D}) \neq \emptyset$ for each $\mathcal{I}$ and for each $\mathcal{D}$[5], it follows that the strictly-complete and the loosely-complete semantics always coincide.

Moreover, notice that the loose semantics is never empty, i.e., it always holds that $sem_{l\,as(\mathcal{M})}(\mathcal{I}, \mathcal{D}) \neq \emptyset$ for any $as(\mathcal{M}) \in \{s, c, e\}$, even if $sem_{as(\mathcal{M})}(\mathcal{I}, \mathcal{D}) = \emptyset$.

Example 3.1 (contd.) With regard to our ongoing example, for both the LAV and GAV system we have that:

1. $sem_{le}(\mathcal{I}_0, \mathcal{D}_0)$ contains the database $\mathcal{B}_1 = \{player(\mathsf{a}, \mathsf{b}),\ team(\mathsf{b}, \mathsf{c})\}$, and all the databases of the form $\mathcal{B}_2 = \{player(\mathsf{a}, \mathsf{d}),\ team(\mathsf{b}, \mathsf{c}),\ team(\mathsf{d}, \alpha)\}$ for each $\alpha \in \mathcal{U}$, $\mathcal{B}_3 = \{player(\mathsf{a}, \mathsf{b}),\ player(\mathsf{e}, \mathsf{f}),\ team(\mathsf{b}, \mathsf{c}),\ team(\mathsf{f}, \alpha)\}$ for each $\alpha \in \mathcal{U}$, and $\mathcal{B}_4 = \{player(\mathsf{a}, \mathsf{d}),\ player(\mathsf{e}, \mathsf{f}),\ team(\mathsf{b}, \mathsf{c}),\ team(\mathsf{d}, \alpha),\ team(\mathsf{f}, \beta)\}$ for each $\alpha, \beta \in \mathcal{U}$;

2. $sem_{ls}(\mathcal{I}_0, \mathcal{D}_0)$ contains the databases of the form $\mathcal{B}_3$ and $\mathcal{B}_4$, and each global database for $\mathcal{I}_0$ consistent with $\mathcal{G}_0$ that can be obtained by consistently adding facts to a database of the form $\mathcal{B}_3$ or $\mathcal{B}_4$;

3. $sem_{lc}(\mathcal{I}_0, \mathcal{D}_0) = sem_c(\mathcal{I}_0, \mathcal{D}_0)$.

Therefore, under the three semantics, the answers to the query $q(x) \leftarrow team(x, y)$ are respectively $ans_{le}(q, \mathcal{I}_0, \mathcal{D}_0) = \{\mathsf{b}\}$, $ans_{ls}(q, \mathcal{I}_0, \mathcal{D}_0) = \{\mathsf{b}, \mathsf{f}\}$ and $ans_{lc}(q, \mathcal{I}_0, \mathcal{D}_0) = \emptyset$. ∎

[5] It should be easy to see that the empty database always satisfies the complete assumption on the mapping mapping and the integrity constraints on $\mathcal{G}$ considered in our framework, for each $\mathcal{I}$ and for each $\mathcal{D}$.

4 Query Answering

We now address the problem of query answering in the presence of integrity constraints under the different semantics considered in our framework. As widely discussed in our previous deliverables D3.1 and D3.3 ([15, 16]), the interaction between the classes of constraints allowed on the global schema, namely, inclusion, key and exclusion dependencies, makes in general query answering a non-trivial task, and may lead to non-decidable cases.

In this section we recall the cases for which query answering is undecidable and present algorithms for decidable cases. Together with complexity lower bounds provided in [16], such algorithms allow us to establish the computational complexity of the query answering problem under both strict and loose semantics.

For the sake of simplicity, we first study query answering in the setting of a single database. Such a setting allows us to overlook here the mapping, and concentrate on the formalism adopted for the global schema, i.e., on the interaction between inclusion, key and exclusion dependencies. Query answering algorithms for GAV and LAV integration systems that exploit the algorithms defined in this section, as well as complexity characterization of query answering in data integration systems, will be then provided in Section 4.5.

We remark that the results we have obtained for the strictly-sound semantics in a single database setting extend previous studies on query containment under integrity constraints [13], while the results for the loose semantics extend known results in the field of inconsistent databases [12, 7], by taking into account inclusion dependencies, which add significant complexity to the problem. In particular, the key issue in our work is that we are able to deal with infinite models for a database schema, that are to be taken into account when cyclic inclusion dependencies are present in the schema.

Before proceeding, we formalize in the following the connection between a single database and a data integration system, and provide both strict and loose semantics definition in this setting.

4.1 Single Database

Actually, a database schema $\mathcal{RS}$ corresponds to the global schema $\mathcal{G}$ of a data integration system $\mathcal{I} = \langle \mathcal{G}, \mathcal{S}, \mathcal{M} \rangle$ in which relations in $\mathcal{G}$ are in a one-to-one correspondence with relations in $\mathcal{S}$, i.e., if $s_1/h_1, \ldots, s_n/h_n$ are the source relations, then the global relations are $g_1/h_1, \ldots, g_n/h_n$, and the GAV (or LAV) mapping is given by the n assertions $\langle g_i, g_i(X_1, \ldots, X_{h_i}) \leftarrow s_i(X_1, \ldots, X_{h_i}) \rangle$ (resp. $\langle s_i, s_i(X_1, \ldots, X_{h_i}) \leftarrow g_i(X_1, \ldots, X_{h_i}) \rangle$) for each i, $1 \leq i \leq n$. Let us now construct from a source database $\mathcal{D}_\mathcal{S}$ for $\mathcal{I}$ a global database for $\mathcal{I}$, namely, a database instance $\mathcal{D}$ for the schema $\mathcal{RS}$ such that $g_i^{\mathcal{D}} = s_i^{\mathcal{D}_\mathcal{S}}$. It is easy now to shift the assumptions that can be adopted on the mapping $\mathcal{M}$ directly to the database instance $\mathcal{D}$, denoted $as(\mathcal{D})$. More precisely,

- if $\mathcal{M}$ is sound, then $\mathcal{D}$ can be assumed sound, that is the semantics of $\mathcal{RS}$ with respect to $\mathcal{D}$, denoted by $sem_s(\mathcal{RS}, \mathcal{D})$, is the set of database instances $\mathcal{B}$ for $\mathcal{RS}$ such that $\mathcal{B} \supseteq \mathcal{D}$;

- if $\mathcal{M}$ is complete, then $\mathcal{D}$ can be assumed complete, that is the semantics of $\mathcal{RS}$ with respect to $\mathcal{D}$, denoted by $sem_c(\mathcal{RS}, \mathcal{D})$, is the set of database instances $\mathcal{B}$ for $\mathcal{RS}$ such that $\mathcal{B} \subseteq \mathcal{D}$;

- if $\mathcal{M}$ is exact, then $\mathcal{D}$ can be assumed exact, that is the semantics of $\mathcal{RS}$ with respect to $\mathcal{D}$, denoted by $sem_e(\mathcal{RS}, \mathcal{D})$, is actually the single database instances $\mathcal{D}$, i.e., $sem_e(\mathcal{RS}, \mathcal{D}) = \{\mathcal{D}\}$.

We also indicate with $ans_{as(\mathcal{D})}(q, \mathcal{RS}, \mathcal{D})$ the set of *answers* to a query q posed to $\mathcal{RS}$, where $as(\mathcal{D}) = s, c$ or e for the sound, complete and exact semantics, respectively. As usual, a tuple t is in the answer to q iff $t \in q^{\mathcal{B}}$ for each $\mathcal{B} \in sem_{as(\mathcal{D})}(\mathcal{RS}, \mathcal{D})$. Furthermore, we denote with $sem_{ls}(\mathcal{RS}, \mathcal{D})$, $sem_{lc}(\mathcal{RS}, \mathcal{D})$ and $sem_{le}(\mathcal{RS}, \mathcal{D})$ respectively the loosely-sound, loosely-complete and loosely-exact semantics, defined analogously to the loose semantics for a data integration system described in Section 3.3. More precisely, given a possibly inconsistent database $\mathcal{D}$ for $\mathcal{RS}$ an assumption $as(\mathcal{D})$ on $\mathcal{D}$, and two databases $\mathcal{B}_1$ and $\mathcal{B}_2$ consistent with $\mathcal{RS}$, we define $\mathcal{B}_1 \leq_{\mathcal{D}} \mathcal{B}_2$ if:

- $\mathcal{B}_1 \cap \mathcal{D} \supseteq \mathcal{B}_2 \cap \mathcal{D}$, if $as(\mathcal{D}) = s$;

- $\mathcal{B}_1 - \mathcal{D} \subseteq \mathcal{B}_2 - \mathcal{D}$, if $as(\mathcal{D}) = c$;

- $\mathcal{B}_1 \cap \mathcal{D} \supseteq \mathcal{B}_2 \cap \mathcal{D}$ and $\mathcal{B}_1 - \mathcal{D} \subseteq \mathcal{B}_2 - \mathcal{D}$, if $as(\mathcal{D}) = e$.

As usual, $\mathcal{B}_1 <_{\mathcal{D}} \mathcal{B}_2$ stands for $\mathcal{B}_1 \leq_{\mathcal{D}} \mathcal{B}_2 \wedge \mathcal{B}_2 \not\leq_{\mathcal{D}} \mathcal{B}_1$.

Hence, a database $\mathcal{B}$ for $\mathcal{RS}$ is in $sem_{l\,as(\mathcal{D})}(\mathcal{RS}, \mathcal{D})$, if

(*i*) $\mathcal{B}$ is consistent with $\mathcal{RS}$;

(*ii*) $\mathcal{B}$ is minimal with respect to $\leq_{\mathcal{D}}$.

Example 3.1 (contd.) Disregard now the mapping and focus only on the relational schema $\mathcal{G}_0$. Consider the database instance

$$\mathcal{D} = \{player(\mathsf{a}, \mathsf{b}), player(\mathsf{a}, \mathsf{d}), player(\mathsf{e}, \mathsf{f}), team(\mathsf{b}, \mathsf{c})\}$$

where $\mathsf{a}, \mathsf{b}, \mathsf{c}, \mathsf{d}, \mathsf{e}, \mathsf{f}$ are constants of $\mathcal{U}$. It is easy to see that the strict and the loose semantics are the same of the integration setting.

■

In the following, we address the decision problem associated to query answering, that is, given a relational schema $\mathcal{RS} = \langle \Psi, \Sigma \rangle$, where $\Sigma = \Sigma_I \cup \Sigma_K \cup \Sigma_E$, a database instance $\mathcal{D}$ for $\mathcal{RS}$, an assumption $as(\mathcal{D})$ on $\mathcal{D}$, a query q of arity n and an n-tuple t of constants of the domain $\mathcal{U}$, to establish whether $t \in ans_{as(\mathcal{D})}(q, \mathcal{RS}, \mathcal{D})$.

4.2 Query Answering under the Strict Semantics

We focus here on query answering under the strictly-sound, complete and exact semantics. As illustrated in Section 3.2, when the data are considered complete the empty database always belongs to $sem_c(\mathcal{RS}, \mathcal{D})$, independently of Σ_I, Σ_K and Σ_E; therefore, for any query q we have immediately $ans_c(q, \mathcal{RS}, \mathcal{D}) = \emptyset$; hence, the answer to the decision problem is always negative for any tuple t. When the data are considered exact, we have two cases:

1. $\mathcal{D}$ satisfies Σ, therefore $sem_e(\mathcal{RS}, \mathcal{D}) = \{\mathcal{D}\}$ and $ans_e(q, \mathcal{RS}, \mathcal{D}) = q^{\mathcal{D}}$. So, query answering reduces to classical query processing in relational databases, and it is immediate to establish whether $t \in ans_e(q, \mathcal{RS}, \mathcal{D})$;

2. $\mathcal{D}$ violates at least one constraint in Σ, therefore $sem_e(\mathcal{RS}, \mathcal{D}) = \emptyset$ and $ans_e(q, \mathcal{RS}, \mathcal{D})$ consists of all tuples of the same arity as q; the answer to the decision problem is therefore affirmative, independently of q and t.

The case where the data are considered sound is more interesting: in fact, if the inclusion dependencies in Σ_I are not satisfied, we may think of adding suitable facts to $\mathcal{D}$ in order to satisfy them (according to the strictly-sound semantics, we are not allowed to repair such violations by deleting facts). In this case, if $\mathcal{D}$ satisfies Σ_K and Σ_E, the semantics of $\mathcal{RS}$ is constituted in general by several (possibly infinite) databases, each of which may have infinite size, since there are several ways of adding facts to $\mathcal{D}$. Query answering with sound data is therefore a difficult task, that is not decidable in all cases. Therefore, in the following we concentrate on the strictly-sound semantics and study query answering in the presence of different combinations of IDs, KDs, and EDs.

4.2.1 Query Answering under the Strictly-sound Semantics

First of all, it is worth noticing that, if Σ contains only KDs and EDs, a situation arises that is analogous to the one described for the exact semantics. Indeed, according to the sound semantics, we are only allowed to add facts to $\mathcal{D}$ in order to satisfy constraints, but KDs and EDs cannot be satisfied by adding facts. Hence, if $\mathcal{D}$ is not consistent with constraints in Σ_K or Σ_E, then $sem_s(\mathcal{RS}, \mathcal{D}) = \emptyset$, thus query answering is meaningless, whereas if $\mathcal{D}$ satisfies $\Sigma_K \cup \Sigma_E$, then $ans_s(q, \mathcal{RS}, \mathcal{D}) = q^{\mathcal{D}}$. Furthermore, KDs and EDs do not interfere with each other, where interfering means that satisfying a constraint may lead to the violation of other constraints. This immediately allows us to "repair" KDs and EDs separately. Notice that also different KDs (or different EDs) do not interfere with each other, whereas this is not true for IDs. Hence, we can assert that the presence of IDs significantly complicates query answering under the sound semantics.

For the sake of clarity, we next separately study query answering in the presence of IDs only, in the presence of IDs and EDs, and in the presence of IDs and KDs. Finally, we tackle the general setting.

Query Answering in the presence of IDs In this section, we extend previous studies on query containment under integrity constraints, and show that query answering in the presence of IDs is decidable. To this aim, we need some preliminary results, presented in the milestone paper of Johnson and Klug [13], which addresses the problem of conjunctive query containment in a relational schema, in the presence of inclusion dependencies[6]. According to this paper, given a relational schema $\mathcal{RS} = \langle \Psi, \Sigma_I \rangle$, and two conjunctive queries of the same arity q_1 and q_2 expressed over $\mathcal{RS}$, to test whether $q_1 \subseteq q_2$, we first have to "freeze" the body of q_1, considering its atoms as facts in a database instance $\mathcal{D}$ for $\mathcal{RS}$, and then applying the *chase* procedure to such a database. The resulting (possibly infinite) database, denoted as $chase(\mathcal{RS}, \mathcal{D})$, is constructed by repeatedly applying, as long as it is applicable, the following rule:

> INCLUSION DEPENDENCY CHASE RULE.
> Suppose there is a tuple t in $r^{chase(\mathcal{RS},\mathcal{D})}$, and there is an ID in Σ_I of the form $r[\mathbf{X}_r] \subseteq s[\mathbf{X}_s]$. If there is no tuple t' in $s^{chase(\mathcal{RS},\mathcal{D})}$ such that $t'[\mathbf{X}_s] = t[\mathbf{X}_r]$, then we add a new tuple t_{chase} in $s^{chase(\mathcal{RS},\mathcal{D})}$ such that $t_{chase}[\mathbf{X}_s] = t[\mathbf{X}_r]$, and for any attribute A of s, such that $A \notin \mathbf{X}_s$, $t_{chase}[A]$ is a fresh constant from $\mathcal{U}$, not appearing elsewhere in $chase(\mathcal{RS}, \mathcal{D})$.

It should be easy to see that $chase(\mathcal{RS}, \mathcal{D})$ may be infinite. In particular, this happens if IDs in Σ_I are cyclic. Nonetheless, it enjoys a fundamental property, as stated by the following Lemma (adapted from [3]) .

Lemma 4.1 *Let $\mathcal{RS} = \langle \Psi, \Sigma_I \rangle$ be a relational schema, and $\mathcal{D}$ a database instance for $\mathcal{RS}$, then $chase(\mathcal{RS}, \mathcal{D})$ does not violate any inclusion dependencies in Σ_I.*

Proof. Suppose by contradiction that the inclusion dependency $r[\mathbf{X}] \subseteq s[\mathbf{Y}]$ is violated in $chase(\mathcal{RS}, \mathcal{D})$. This implies that there is a tuple t in $r^{chase(\mathcal{RS},\mathcal{D})}$ such that, for no tuple t' in $s^{chase(\mathcal{RS},\mathcal{D})}$, $t'[\mathbf{Y}] = t[\mathbf{X}]$. This would imply that we can apply the chase rule and insert a new tuple t_{chase} in $s^{chase(\mathcal{RS},\mathcal{D})}$ such that $t_{chase}[\mathbf{Y}] = t[\mathbf{X}]$, and for each attribute A of s such that $A \notin \mathbf{Y}$, $t_{chase}[A]$ is a fresh constant; but this contradicts the assumption. $\square$

Johnson and Klug have proved that $q_1 \subseteq q_2$ if and only if the tuple t_f in the frozen head of q_1 belongs to $q_2^{chase(\mathcal{RS},\mathcal{D})}$. Moreover, they have shown that, to check whether $t_f \in q_2^{chase(\mathcal{RS},\mathcal{D})}$, only a finite portion of $chase(\mathcal{RS}, \mathcal{D})$ needs to be considered. Based on this property, they have defined a PSPACE algorithm that checks if $t_f \in q_2^{chase(\mathcal{RS},\mathcal{D})}$[7].

To this aim we need to maintain distinct the set of constants of the domain $\mathcal{U}$ that occur in the database $\mathcal{D}$, that is the *active domain* of $\mathcal{D}$, from the other constants of $\mathcal{U}$, which can be used in the construction of the chase. Hereinafter, we indicate with $\mathcal{U}_\mathcal{D}$ the active domain of a database $\mathcal{D}$.

In the case of query answering, we are able to exploit the technique of Johnson and Klug. More specifically, we make use of the notion of chase as specified by the following result.

[6] In [13], results on query containment of CQs under IDs are also generalized to the case in which there is a limited interaction between IDs and FDs. This generalization is actually not interesting for our discussion.

[7] Actually, Johnson and Klug have considered the case of boolean queries, for which $q_1 \subseteq q_2$ iff $q_2^{chase(\mathcal{RS},\mathcal{D})}$ is non-empty. According to our version of the algorithm, this is equivalent to check that the empty tuple $\langle\rangle$, which represents the frozen head of q_1, is contained in $q_2^{chase(\mathcal{RS},\mathcal{D})}$.

Theorem 4.2 *Let $\mathcal{RS} = \langle \Psi, \Sigma_I \rangle$ be a relational schema and $\mathcal{D}$ a database instance for $\mathcal{RS}$; let q be a conjunctive query of arity n, and t an n-tuple of constants of $\mathcal{U}_\mathcal{D}$. We have that $t \in ans_s(q, \mathcal{RS}, \mathcal{D})$ if and only if $t \in q^{chase(\mathcal{RS},\mathcal{D})}$.*

Proof (sketch).
"$\Rightarrow$" Since $chase(\mathcal{RS}, \mathcal{D})$ satisfies $\mathcal{I}$, it belongs to $sem_s(\mathcal{RS}, \mathcal{D})$. From the definition of $ans_s(q, \mathcal{RS}, \mathcal{D})$, it follows that $t \in q^{chase(\mathcal{RS},\mathcal{D})}$.
"$\Leftarrow$" Analogously to [3], it can be proved by induction on the structure of $chase(\mathcal{RS}, \mathcal{D})$ that, for any database instance $\mathcal{B} \in sem_s(\mathcal{RS}, \mathcal{D})$, there exists a homomorphism μ that sends the tuples of $chase(\mathcal{RS}, \mathcal{D})$ to the tuples of $\mathcal{B}$. By hypothesis $t \in q^{chase(\mathcal{RS},\mathcal{D})}$, so there exists a homomorphism λ from the atoms of q to the facts of $chase(\mathcal{RS}, \mathcal{D})$; the composition $\lambda \circ \mu$ witnesses that $t \in ans_s(q, \mathcal{RS}, \mathcal{D})$. $\square$

Based on the above property, we can apply the algorithm of [13] for query answering, i.e., to check whether a tuple t belongs to $ans_s(q, \mathcal{RS}, \mathcal{D})$. More specifically, given an UCQ $q(\vec{\mathbf{x}}) \leftarrow conj_1(\vec{\mathbf{x}}, \vec{\mathbf{y}}_1) \vee \ldots \vee conj_m(\vec{\mathbf{x}}, \vec{\mathbf{y}}_m)$ of arity n, an n-tuple t of constants of $\mathcal{U}_\mathcal{D}$, and a database instance $\mathcal{D}$ for $\mathcal{RS}$, we construct a query q_1 of the form $q_1(t) \leftarrow d_1, \ldots, d_k$ where $d_1, \ldots, d_k$ are all facts in $\mathcal{D}$. Then, $t \in ans_s(q, \mathcal{RS}, \mathcal{D})$ iff q_1 is contained in at least one CQ of the form $q(\vec{\mathbf{x}}) \leftarrow conj_i(\vec{\mathbf{x}}, \vec{\mathbf{y}}_i)$, for each i, $1 \leq i \leq n$. We indicate with $\mathsf{Answer}_{JK}(\mathcal{RS}, \mathcal{D}, q, t)$ the algorithm for query answering that makes use of the algorithm for query containment of [13] as above described. From the results of [13] it easily follows that Answer_{JK} is a PSPACE algorithm.

Query Answering in the presence of IDs and EDs Let us now add EDs to the set of integrity constraints specified on $\mathcal{RS}$, i.e., let be $\Sigma = \Sigma_I \cup \Sigma_E$. It should be easy to see that, in general, $chase(\mathcal{RS}, \mathcal{D})$ is not a database in $sem_s(\mathcal{RS}, \mathcal{D})$.

Example 4.3 Consider the relational schema $\mathcal{RS}_0 = \langle \Psi_0, \Sigma_0 \rangle$, where Ψ_0 contains three unary relations r_1, r_2 and r_3, and $\Sigma_0 = \Sigma_{I_0} \cup \Sigma_{E_0}$ contains the constraints

$$\begin{array}{rl} r_1[1] & \subseteq r_2[1], \text{ and} \\ (r_2[1] & \cap\, r_3[1]) = \emptyset, \end{array}$$

and assume that $\mathcal{D}_0 = \{r_1(a), r_3(a)\}$ is a database instance for $\mathcal{RS}_0$. Such database is consistent with Σ_{E_0}, but it is not consistent with Σ_{I_0}. By applying just once the inclusion dependency chase rule we obtain $chase(\mathcal{RS}_0, \mathcal{D}_0) = \{r_1(a), r_2(a), r_3(a)\}$. $chase(\mathcal{RS}_0, \mathcal{D}_0)$ is now consistent with Σ_{I_0}, but it is not consistent with Σ_{E_0}. Furthermore, no database satisfying both Σ_0 and the soundness of $\mathcal{D}_0$ exists, i.e., $sem_s(\mathcal{RS}_0, \mathcal{D}_0) = \emptyset$.

Consider now $\mathcal{D}_1 = \{r_1(a)\}$. It is easy to see that $chase(\mathcal{RS}_0, \mathcal{D}_1) = \{r_1(a), r_2(a)\}$ is consistent with both Σ_{I_0} and Σ_{E_0}, and that all databases in $sem_s(\mathcal{RS}_0, \mathcal{D}_1)$ can be obtained by consistently adding facts to $chase(\mathcal{RS}_0, \mathcal{D}_1)$. $\blacksquare$

From the above example, we can argue that it should still be possible to answer a query posed on $\mathcal{RS}$ by looking only at $chase(\mathcal{RS}, \mathcal{D})$, provided some mechanism for checking if $sem_s(\mathcal{RS}, \mathcal{D})$ is not empty.

Our feeling is that, in order to assert that $sem_s(\mathcal{RS}, \mathcal{D}) \neq \emptyset$, we should consider explicitly the set of exclusion dependencies that are logical consequences of Σ_I and Σ_E

and then verify wether $\mathcal{D}$ is consistent with it. We recall that a constraint δ is a logical consequence of (or is logically implied by) a set of constraints Σ, denoted $\Sigma \models \delta$, if all database instances that satisfy Σ satisfy also δ. The set of EDs that are logically implied by the set $\Sigma = \Sigma_I \cup \Sigma_E$ is denoted with $\Sigma_E^{*,\Sigma}$, or simply Σ_E^* when Σ is clear from the context.

For instance, in Example 4.3, $\Sigma_{E_0}^* = \Sigma_{E_0} \cup \{(r_1[1] \cap r_3[1]) = \emptyset\}$, and $\mathcal{D}_0$ is not consistent with $\Sigma_{E_0}^*$, whereas $\mathcal{D}_1$ is consistent with it.

In the following we formally prove that we can make use of the chase to answer queries in the presence of IDs and EDs.

Lemma 4.4 *Let $\mathcal{RS} = \langle \Psi, \Sigma_I, \Sigma_E \rangle$ be a relational schema, and $\mathcal{D}$ a database instance for $\mathcal{RS}$. Then chase$(\mathcal{RS}, \mathcal{D})$ satisfies $\Sigma_I \cup \Sigma_E$ if and only if $\mathcal{D}$ is consistent with Σ_E^*.*

Proof. "$\Rightarrow$"We first show that if $\mathcal{D}$ violates any exclusion dependency in Σ_E^*, then *chase*$(\mathcal{RS}, \mathcal{D})$ is not consistent with $\Sigma_I \cup \Sigma_E$. By contradiction, let us assume that *chase*$(\mathcal{RS}, \mathcal{D})$ satisfies $\Sigma_I \cup \Sigma_E$. This would imply that *chase*$(\mathcal{RS}, \mathcal{D})$ satisfies also Σ_E^*, but, since in the construction of *chase*$(\mathcal{RS}, \mathcal{D})$ facts are only added and never removed, also $\mathcal{D}$ would satisfy Σ_E^*, and this contradicts the assumption.

"$\Leftarrow$"We now prove that if $\mathcal{D}$ is consistent with Σ_E^*, then *chase*$(\mathcal{RS}, \mathcal{D})$ satisfies $\Sigma_I \cup \Sigma_E$. From Lemma 4.1 it follows that *chase*$(\mathcal{RS}, \mathcal{D})$ does not violate any inclusion dependency in Σ_I, hence it remains to prove that, *chase*$(\mathcal{RS}, \mathcal{D})$ satisfies Σ_E. We first show that *chase*$(\mathcal{RS}, \mathcal{D})$ satisfies Σ_E^*. The proof is by induction on the structure of *chase*$(\mathcal{RS}, \mathcal{D})$. In the following we indicate with $chase_i(\mathcal{RS}, \mathcal{D})$ the portion of *chase*$(\mathcal{RS}, \mathcal{D})$ obtained by applying i times the chase rule.

As for the base step, $chase_0(\mathcal{RS}, \mathcal{D}) = \mathcal{D}$, then, by the assumption, it satisfies Σ_E^*. As for the inductive step, assume by contradiction that $chase_i(\mathcal{RS}, \mathcal{D})$ is consistent with Σ_E^*, but $chase_{i+1}(\mathcal{RS}, \mathcal{D})$ is not. Let $r_1[\mathbf{X}] \subseteq r_2[\mathbf{Y}]$ be an ID in Σ_I that is violated by $chase_i(\mathcal{RS}, \mathcal{D})$, $t_1 \in r_1^{chase_i(\mathcal{RS},\mathcal{D})}$ is a tuple that violates such ID, $t_2 \in r_2^{chase_{i+1}(\mathcal{RS},\mathcal{D})}$ is the tuple generated by the chasing rule, i.e., let be $t_2[\mathbf{Y}] = t_1[\mathbf{X}]$, and let $(r_2[\mathbf{Z}] \cap r_3[\mathbf{W}]) = \emptyset$ be the ED in Σ_E violated in $chase_{i+1}(\mathcal{RS}, \mathcal{D})$. Then, there must exist $t_3 \in r_3^{chase_{i+1}(\mathcal{RS},\mathcal{D})}$ such that $t_3[\mathbf{W}] = t_2[\mathbf{Z}]$. According to the chase rule, for any attribute A of r_2 such that $A \notin \mathbf{Y}$, $t_2[A]$ is a fresh constant of $\mathcal{U}$ that does not appear in $chase_i(\mathcal{RS}, \mathcal{D})$. Hence, since also $t_3 \in r_3^{chase_i(\mathcal{RS},\mathcal{D})}$, we have that $\mathbf{Z} \subseteq \mathbf{Y}$. It should be easy to see now that Σ_E^* contains the ED $(r_1[\mathbf{X}'] \cap r_3[\mathbf{W}]) = \emptyset$, where $\mathbf{X}'$ is a sequence of attributes of r_1 such that $\mathbf{X}' \subseteq \mathbf{X}$ and $t_1[\mathbf{X}'] = t_2[\mathbf{Z}]$. Hence, $t_1[\mathbf{X}'] = t_3[\mathbf{W}]$, and so $chase_i(\mathcal{RS}, \mathcal{D})$ does not satisfy Σ_E^*, but this contradicts the inductive assumption.

From the fact that *chase*$(\mathcal{RS}, \mathcal{D})$ satisfies Σ_E^*, it immediately follows that *chase*$(\mathcal{RS}, \mathcal{D})$ satisfies Σ_E. □

With this result in place, we are able to extend the result of [13] to the case of IDs and EDs.

Theorem 4.5 (IDs-EDs Separation) *Let $\mathcal{RS} = \langle \Psi, \Sigma_I, \Sigma_E \rangle$ be a relational schema, and let $\mathcal{RS}_1 = \langle \Psi, \Sigma_I \rangle$ be the relational schema obtained from $\mathcal{RS}$ by removing the EDs. Let $\mathcal{D}$ be a database instance for $\mathcal{RS}$ and $\mathcal{RS}_1$, q a query of arity n over $\mathcal{RS}$, and t an n-tuple of constants of $\mathcal{U}_\mathcal{D}$. We have that $t \notin ans_s(q, \mathcal{RS}, \mathcal{D})$ iff $\mathcal{D}$ is consistent with Σ_E^* and $t \notin ans_s(q, \mathcal{RS}_1, \mathcal{D})$.*

Proof. "⇒"By hypothesis $t \notin ans_s(q, \mathcal{RS}, \mathcal{D})$; this means that there exists $\mathcal{B} \in sem_s(\mathcal{RS}, \mathcal{D})$, such that $t \notin q^{\mathcal{B}}$. Since $\mathcal{B}$ satisfies $\Sigma_I \cup \Sigma_E$, and $\mathcal{D} \subseteq \mathcal{B}$, it easily follows that $\mathcal{D}$ satisfies Σ_E^*. It is also immediate to verify that $\mathcal{B} \in sem_s(\mathcal{RS}_1, \mathcal{D})$, therefore $t \notin ans_s(q, \mathcal{RS}_1, \mathcal{D})$, thus proving the claim.

"⇐"By hypothesis $t \notin ans_s(q, \mathcal{RS}_1, \mathcal{D})$ and $\mathcal{D}$ satisfies Σ_E^*. From the former condition, according to Theorem 4.2, it follows that $t \notin q^{chase(\mathcal{RS}_1, \mathcal{D})}$, whereas from the latter condition, according to Lemma 4.4, it follows that $chase(\mathcal{RS}, \mathcal{D})$ satisfies $\Sigma_I \cup \Sigma_E$. Since $chase(\mathcal{RS}, \mathcal{D}) = chase(\mathcal{RS}_1, \mathcal{D})$ by construction, then $chase(\mathcal{RS}_1, \mathcal{D}) \in sem_s(\mathcal{RS}, \mathcal{D})$. The claim follows immediately. □

Before concluding, we provide now a set of inference rules $\mathcal{I}_{ED}$ that allows for the computation of $\Sigma_E^{*,\Sigma}$, where $\Sigma = \Sigma_I \cup \Sigma_E$, and we prove that such inference rules are correct and complete with respect to the problem of inferring logical implication of EDs by IDs and EDs.

In the following set $\mathcal{I}_{ED}$ of inference rules, r, s, r_1, r_2 are relations, whereas $\mathbf{X}$, $\mathbf{Y}$, $\mathbf{Z}$, $\mathbf{W}$, $\mathbf{X}'$ and $\mathbf{Y}'$ range over sequence of attributes of such relations.

> *ID-ED1*: **If** $(r[\mathbf{X}] \cap r[\mathbf{Y}]) = \emptyset$, where $\mathbf{X} = X_1, \ldots, X_n$ and $\mathbf{Y} = Y_1, \ldots, Y_n$, **then** $(r[\mathbf{X}'] \cap s[\mathbf{Y}']) = \emptyset$, where $\mathbf{X}'$ is every set of attributes in r such that $\mathbf{X}' = X_1, \ldots, X_n, X_{n+1}, \ldots, X_m$ and $\mathbf{Y}'$ is every set of attributes in s such that $\mathbf{Y}' = Y_1, \ldots, Y_n, Y_{n+1}, \ldots, Y_m$.
>
> *ID-ED2*: **If** $(r[\mathbf{Z}] \cap r[\mathbf{Z}]) = \emptyset$, **then** $(r[\mathbf{X}] \cap s[\mathbf{Y}]) = \emptyset$ for all relations s and for all sequences of attributes $\mathbf{X}$ and $\mathbf{Y}$[8].
>
> *ID-ED3*: **If** $r_1[\mathbf{X}] \subseteq r_2[\mathbf{Y}]$ and $(r_2[\mathbf{Z}] \cap r_3[\mathbf{W}]) = \emptyset$ such that $\mathbf{Z} \subseteq \mathbf{Y}$, **then** $(r_1[\mathbf{X}'] \cap r_3[\mathbf{W}]) = \emptyset$ where $\mathbf{X}' \subseteq \mathbf{X}$ is such that $r_1[\mathbf{X}'] \subseteq r_2[\mathbf{Z}]$.

We now prove soundness and completeness of the above rules, i.e., we show that $\Sigma \vdash_{\mathcal{I}_{ED}} \delta$ if and only if $\Sigma \models \delta$, where δ is an ED, and $\Sigma \vdash_{\mathcal{I}_{ED}} \delta$ indicates that δ is provable from Σ using $\mathcal{I}_{ED}$ (we omit the subscript $\mathcal{I}_{ED}$ when clear from the context). We say that the ED δ is *provable* from Σ using $\mathcal{I}_{ED}$ if there exists a sequence of IDs and EDs (called *proof*) $\delta_1, \ldots, \delta_n = \delta$, $(n \geq 1)$ such that for each $i \in \{1, \ldots, n\}$ either

(a) $\delta_i \in \Sigma$, or

(b) there is a substitution for some rule $\rho \in \mathcal{I}_{ED}$ such that δ_i corresponds to the consequence of ρ, and such that for each ID and ED in the antecedent of ρ, the corresponding ID and ED is in the set $\{\delta_j \mid 1 \leq j < i\}$.

Theorem 4.6 *The set $\mathcal{I}_{ED} = \{$ID-ED1,ID-ED2,ID-ED3$\}$ is sound and complete for logical implication of EDs by IDs and EDs.*

Proof. Soundness of the rules is easily verified. For completeness, let $\mathcal{RS} = \langle \Psi, \Sigma \rangle$ be a relational schema where $\Sigma = \Sigma_I \cup \Sigma_E$ is a set of IDs and EDs over $\mathcal{RS}$, and let δ be the exclusion dependency $(r[\mathbf{X}] \cap s[\mathbf{Y}]) = \emptyset$ such that $\Sigma \models \delta$.

[8]Notice that, $(r[\mathbf{Z}] \cap r[\mathbf{Z}]) = \emptyset$ is satisfied by a database instance $\mathcal{B}$ if $r^{\mathcal{B}}$ is empty.

We construct a database instance $\mathcal{DB}$ for $\mathcal{RS}$ such that $\mathcal{DB} \not\models \delta$ and we use it to demonstrate that $\Sigma \vdash \delta$.

Let r/n and s/m, be two relations in Ψ, and $\mathbf{X} = X_1, \ldots, X_k$ and $\mathbf{Y} = Y_1, \ldots, Y_k$ be two sequences of attributes of r and s, respectively. Suppose by contradiction that $\Sigma \not\vdash \delta$, and consider a database instance $\mathcal{D}$ for $\mathcal{RS}$ such that $\mathcal{D} = \{r(a_1, \ldots, a_k, b_1, \ldots, b_{n-k}), s(a_1, \ldots, a_k, c_1, \ldots, c_{m-k})\}$[9], where $a_i \neq a_l$, $a_i \neq b_j$, $a_i \neq c_h$ and $b_j \neq c_h$ for $i = 1, \ldots, k$, $l = 1, \ldots, k$ and $l \neq i$, $j = 1, \ldots, n-k$, and $h = 1, \ldots, m-k$. It's easy to see that $\mathcal{D}$ does not satisfy δ, and that $chase(\mathcal{RS}, \mathcal{D})$ does not satisfy δ as well.

We next show that $chase(\mathcal{RS}, \mathcal{D})$ is consistent with Σ (thus contradicting the assumption that $\Sigma \models \delta$).

As for IDs, from Lemma 4.1 it follows that $chase(\mathcal{RS}, \mathcal{D})$ is consistent with Σ_I.

We consider now the EDs. Let Σ' be the set of constraints obtained by adding to Σ the EDs inferred by Σ using $\mathcal{I}_{ED}$. We first show that $chase(\mathcal{RS}, \mathcal{D})$ does not violate EDs in Σ'. The proof is by induction on the construction of $chase(\mathcal{RS}, \mathcal{D})$. We indicate with $chase(\mathcal{RS}, \mathcal{D})_i$ the portion of $chase(\mathcal{RS}, \mathcal{D})$ obtained by applying i times the chase rule.

As for the base step, $chase(\mathcal{RS}, \mathcal{D})_0 = \mathcal{D}$, hence, according to the construction of $\mathcal{D}$, $chase(\mathcal{RS}, \mathcal{D})_0$ does not violate (i) EDs that do not involve r and s, (ii) EDs of the form $(r[\mathbf{X}'] \cap s[\mathbf{Y}']) = \emptyset$ where $\mathbf{X}' \not\subseteq \mathbf{X}$ or $\mathbf{Y}' \not\subseteq \mathbf{Y}$, and (iii) EDs of the form $(r[\mathbf{Z}] \cap r[\mathbf{W}]) = \emptyset$ (resp. $(s[\mathbf{Z}] \cap s[\mathbf{W}]) = \emptyset$) where $\mathbf{Z} = Z_1, \ldots, Z_k$ and $\mathbf{W} = W_1, \ldots, W_k$ and there exists $i \in \{1, \ldots, k\}$ such that $Z_i \neq W_i$.

It remains to prove that $chase(\mathcal{RS}, \mathcal{D})_0$ does not violate

(iv) EDs of the form $(r[\mathbf{X}'] \cap s[\mathbf{Y}']) = \emptyset$, where $\mathbf{X}'$ is every set of attributes in r such that $\mathbf{X}' = X_1, \ldots, X_k, X_{k+1}, \ldots, X_h$ and $\mathbf{Y}'$ is every set of attributes in s such that $\mathbf{Y}' = Y_1, \ldots, Y_k, Y_{k+1}, \ldots, Y_h$;

(v) EDs of the form $(r[\mathbf{X}] \cap r[\mathbf{X}]) = \emptyset$ and $(s[\mathbf{X}] \cap s[\mathbf{X}]) = \emptyset)$.

However, constraints of the form (iv) (reps. (v)) cannot be in Σ', otherwise according to rule ID-ED1 (resp. ID-ED2) it should be $\Sigma' \vdash \delta$.

As for the inductive step, assume by contradiction that $chase_i(\mathcal{RS}, \mathcal{D})$ is consistent with Σ', but $chase_{i+1}(\mathcal{RS}, \mathcal{D})$ is not. Let $r_1[\mathbf{X}] \subseteq r_2[\mathbf{Y}]$ be an ID in Σ' that is violated by $chase_i(\mathcal{RS}, \mathcal{D})$, $t_1 \in r_1^{chase_i(\mathcal{RS},\mathcal{D})}$ is a tuple that violates such ID, $t_2 \in r_2^{chase_{i+1}(\mathcal{RS},\mathcal{D})}$ is the tuple generated by the chasing rule, i.e., let be $t_2[\mathbf{Y}] = t_1[\mathbf{X}]$, and let $(r_2[\mathbf{Z}] \cap r_3[\mathbf{W}]) = \emptyset$ be the ED in Σ' violated in $chase_{i+1}(\mathcal{RS}, \mathcal{D})$. Then, there must exist $t_3 \in r_3^{chase_{i+1}(\mathcal{RS},\mathcal{D})}$ such that $t_3[\mathbf{W}] = t_2[\mathbf{Z}]$. According to the chase rule, for any attribute A of r_2 such that $A \notin \mathbf{Y}$, $t_2[A]$ is a fresh constant of $\mathcal{U}$ that does not appear in $chase_i(\mathcal{RS}, \mathcal{D})$. Hence, since also $t_3 \in r_3^{chase_i(\mathcal{RS},\mathcal{D})}$, we have that $\mathbf{Z} \subseteq \mathbf{Y}$. It should be easy to see now that, according to the ID-ED3 rule, from the above ID and ED we can prove the exclusion dependency $(r_1[\mathbf{X}'] \cap r_3[\mathbf{W}]) = \emptyset$, where $\mathbf{X}'$ is a sequence of attributes of r_1 such that $\mathbf{X}' \subseteq \mathbf{X}$ and $t_1[\mathbf{X}'] = t_2[\mathbf{Z}]$. Hence, $t_1[\mathbf{X}'] = t_3[\mathbf{W}]$, and so $chase_i(\mathcal{RS}, \mathcal{D})$ does not satisfy $(r_1[\mathbf{X}'] \cap r_3[\mathbf{W}]) = \emptyset$, but this contradicts the inductive assumption.

[9] Without loss of generality we assume that $\mathbf{X}$ and $\mathbf{Y}$ are the first k attributes in r and s, respectively.

Since $\Sigma_E \subseteq \Sigma'$, it immediately follows that $chase(\mathcal{RS}, \mathcal{D})$ is consistent with Σ_E, but since it does not satisfy δ, we can conclude that δ is not a logical consequence of Σ. □

Finally, we provide a complexity result for logical implication of EDs by IDs and EDs.

Theorem 4.7 *Let $\mathcal{RS} = \langle \Psi, \Sigma \rangle$ be a relational schema, where $\Sigma = \Sigma_I \cup \Sigma_E$, and let δ be an exclusion dependency. Then, the problem of deciding wether $\Sigma \models \delta$ is PSPACE complete.*

Proof. As for the membership, we show that there exist a non-deterministic polynomial space algorithm for deciding logical implication of δ by Σ. Then, by Savitch's theorem [21] this can be transformed into a deterministic algorithm that runs in polynomial space. More precisely, in order to prove that $\Sigma \models \delta$, we check if $\Sigma \vdash_{\mathcal{I}_{ED}} \delta$, i.e., we look for a proof $\delta_1, \ldots, \delta_n$, such that $\delta_n = \delta$. Without loss of generality, we assume that no useless dependencies are in the proof, i.e., every dependency, except δ_n, appears in the antecedent of at least one inference rule of $\mathcal{I}_{ED}$. According to the set $\mathcal{I}_{ED}$, to generate in the proof δ_i, such that $1 \leq i \leq n$ and $\delta_i \notin \Sigma$, we need only δ_{i-1} (if δ_i is generated by applying ID-ED1 or ID-ED2) or δ_{i-1} and δ_j such that $1 \leq j < i-1$ (if δ_i is generated by applying ID-ED3). Since in this last situation one among δ_{i-1} and δ_j is an ID belonging to Σ_I, the additional space required in both cases is bounded to the space required for storing a single ED, rather than the complete proof at the $(i-1)$-th step. Moreover, the number of EDs that are logical consequence of Σ is bounded to $O(2^{n \log n})$, where n is the maximum between the number of relations in Ψ and the maximum arity of the relations in Ψ. Hence, a counter of $n \log n$ bits allows us to stop the non-deterministic computation after $2^{n \log n}$ steps if δ has not still been proved.

Hardness easily follows from the fact that deciding logical implication of EDs by IDs and EDs actually subsumes deciding logical implication of IDs. Since logical implication of IDs is a PSPACE complete problem [1], the hardness follows. □

Query Answering in the presence of IDs and KDs Let us now study the case in which IDs and KDs are expressed over a relational schema, i.e., consider the schema $\mathcal{RS} = \langle \Psi, \Sigma \rangle$, where $\Sigma = \Sigma_I \cup \Sigma_K$. As shown in [15, 16], query answering in such a setting is not decidable in all cases. More specifically, in [15, 16] we have introduced a particular class of IDs, called *non-key-conflicting inclusion dependency* (NKCID), under which query answering is decidable, and we have shown that, as soon as we extend IDs beyond NKIDs, query answering becomes undecidable. For the sake of completeness, in the following we recall the formal definitions of [15, 16], and the theorems establishing undecidability of query answering. For the proofs of such theorems we refer the reader to [15, 16].

Definition 4.8 Given a relational schema $\mathcal{RS} = \langle \Psi, \Sigma_I, \Sigma_K \rangle$, an inclusion dependency in Σ_I of the form $r_1[\mathbf{A}_1] \subseteq r_2[\mathbf{A}_2]$ is a *non-key-conflicting inclusion dependency* (NKCID) with respect to Σ_K if either: (i) no KD is defined on r_2, or (ii) the KD $key(r_2) = \mathbf{K}$ is in Σ_K, and $\mathbf{A}_2$ is not a strict superset of $\mathbf{K}$, i.e., $\mathbf{A}_2 \not\supset \mathbf{K}$. Moreover, the schema $\mathcal{RS}$ is *non-key-conflicting* (NKC) if all the IDs in Σ_I are NKCIDs with respect to Σ_K.

Informally, a set of dependencies is NKC if no ID in Σ_I propagates a proper subset of the key of the relation in its right-hand side. We point out that the class of NKCIDs comprises the well-known class of *foreign key dependencies*, which corresponds to IDs of the form $r_1[\mathbf{A_1}] \subseteq r_2[\mathbf{A_2}]$ such that $key(r_2) = \mathbf{A_2}$.

Definition 4.9 Given a relational schema $\mathcal{RS} = \langle \Psi, \Sigma_I, \Sigma_K \rangle$, an inclusion dependency in Σ_I of the form $r_1[\mathbf{A}_1] \subseteq r_2[\mathbf{A}_2]$ is a *1-key-conflicting inclusion dependency* (1KCIDs) with respect to Σ_K if the KD $key(r_2) = \mathbf{K}$ is in Σ_K, $\mathbf{A}_2$ is allowed to cover $\mathbf{K}$ plus at most one attribute of r_2. Moreover, the schema $\mathcal{RS}$ is *1-key-conflicting* (1KC) if at least one IDs in Σ_I is a 1KCID with respect to Σ_K.

We now recall our first result: query answering is undecidable in the presence of KDs and 1KCIDs.

Theorem 4.10 *Let $\mathcal{RS} = \langle \Psi, \Sigma_I, \Sigma_K \rangle$ be a 1KC schema, $\mathcal{D}$ a database instance for $\mathcal{RS}$, q a query of arity n over $\mathcal{RS}$, and t an n-tuple of constants of $\mathcal{U}_\mathcal{D}$. The problem of establishing whether $t \in ans_s(q, \mathcal{RS}, \mathcal{D})$ is undecidable.*

Now we come to query answering in the case of NKCIDs and KDs, and prove that this problem is decidable. The most relevant property of NKCIDs is that they do not interfere with KDs, so that we can operate with NKCIDs just as if the KDs were not defined in the schema. In particular, we can again make use of the chase to answer queries in the presence of IDs and KDs. This property is expressed by the following result.

Lemma 4.11 *Let $\mathcal{RS} = \langle \Psi, \Sigma_I, \Sigma_K \rangle$ be a NKC schema, and $\mathcal{D}$ a database instance for $\mathcal{RS}$. Then $chase(\mathcal{RS}, \mathcal{D})$ satisfies Σ_I and Σ_K if and only if $\mathcal{D}$ is consistent with Σ_K.*

Proof. "$\Rightarrow$" We show that if $\mathcal{D}$ violates any key dependency in Σ_K, then $chase(\mathcal{RS}, \mathcal{D})$ is not consistent with $\Sigma_I \cup \Sigma_K$. By contradiction, let us assume that $chase(\mathcal{RS}, \mathcal{D})$ satisfies $\Sigma_I \cup \Sigma_K$. Since in the construction of $chase(\mathcal{RS}, \mathcal{D})$ facts are only added and never removed, also $\mathcal{D}$ would satisfy Σ_K, and this contradicts the assumption.

"$\Leftarrow$" We now prove that if $\mathcal{D}$ is consistent with Σ_K, then $chase(\mathcal{RS}, \mathcal{D})$ satisfies $\Sigma_I \cup \Sigma_K$. From Lemma 4.1 it follows that $chase(\mathcal{RS}, \mathcal{D})$ does not violate any inclusion dependency in Σ_I, hence it remains to prove that, $chase(\mathcal{RS}, \mathcal{D})$ satisfies Σ_K. The proof is by induction on the structure of $chase(\mathcal{RS}, \mathcal{D})$. First, by hypothesis $\mathcal{D}$ is consistent with Σ_K, and this verifies the base step. For the induction step, suppose we insert in $chase(\mathcal{RS}, \mathcal{D})$ a tuple t into a relation r, on which a key dependency $key(r) = \mathbf{K}$ is defined, according to the ID $s[\mathbf{A}] \subseteq r[\mathbf{B}]$. We will show that there is no violation of the key dependencies on r, by showing that t does not agree on $\mathbf{K}$ with any pre-existing tuple t in $r^{chase^*(\mathcal{RS},\mathcal{D})}$, where $chase^*(\mathcal{RS}, \mathcal{D})$ is the portion of $chase(\mathcal{RS}, \mathcal{D})$ constructed until the insertion of t.

According to the definition of NKCIDs, the possible cases are the following.

1. $\mathbf{B} = \mathbf{K}$. In this case we have a foreign key dependency; t and $\bar{t}$ cannot agree on $\mathbf{K}$, because in that case t wouldn't have been added.

2. $\mathbf{B} \subset \mathbf{K}$. The two tuples differ on the constants of $\mathbf{B}$ (otherwise only one of the two would have been added), so they differ also on $\mathbf{K}$.

3. $\mathbf{B} \cap \mathbf{K} \neq \emptyset$ and $\mathbf{B} - \mathbf{K} \neq \emptyset$. In this case $\mathbf{B}$ partially overlaps with $key(r)$; we necessarily have $\mathbf{K} - \mathbf{B} \neq \emptyset$, otherwise $\mathbf{B}$ would be a strict superset of $\mathbf{K}$. Therefore t and $\bar{t}$ differ in the constants in $\mathbf{K} - \mathbf{B}$, where t has fresh constants, thus they differ *a fortiori* on $\mathbf{K}$.
4. $\mathbf{B} \cap \mathbf{K} = \emptyset$. In this case the two tuples differ in the constants in $\mathbf{K}$, where t has fresh constants.

□

Analogously to the case of IDs and EDs, we exploit the above lemma to extend the result of [13] to the case of IDs and KDs.

Theorem 4.12 (IDs-KDs Separation) *Let $\mathcal{RS} = \langle \Psi, \Sigma_I, \Sigma_K \rangle$ be a NKC relational schema, and let $\mathcal{RS}_1 = \langle \Psi, \Sigma_I \rangle$ be the relational schema obtained from $\mathcal{RS}$ by removing the KDs. Let $\mathcal{D}$ be a database instance for $\mathcal{RS}$ and $\mathcal{RS}_1$, q a query of arity n over $\mathcal{RS}$, and t an n-tuple of constants of $\mathcal{U}_\mathcal{D}$. We have that $t \notin ans_s(q, \mathcal{RS}, \mathcal{D})$ iff $\mathcal{D}$ is consistent with Σ_K and $t \notin ans_s(q, \mathcal{RS}_1, \mathcal{D})$.*

Proof. Easily obtainable from the proof of Theorem 4.5 by replacing Σ_E and Σ_E^* with Σ_K, and Lemma 4.4 with Lemma 4.11. □

Query Answering in the presence of IDs, KDs, and EDs Finally, we consider the general case in which IDs, KDs and EDs are expressed over a relational schema $\mathcal{RS}$. Obviously, according to Theorem 4.10, in the presence of 1KCIDs with respect to the KDs on $\mathcal{RS}$, query answering is undecidable, being the case in which only IDs and KDs are expressed over $\mathcal{RS}$ a special case of the setting studied here.

As for the case of NKC schemas, the following theorem states that query answering is decidable. The theorem, whose proof follows straightforwardly from Theorem 4.5 and Theorem 4.12, integrates the results that we have presented separately for IDs and EDs, and for IDs and KDs.

Theorem 4.13 (IDs-EDs-KDs Separation) *Let $\mathcal{RS} = \langle \Psi, \Sigma_I, \Sigma_K, \Sigma_E \rangle$ be a NKC schema, and let $\mathcal{RS}_1 = \langle \Psi, \Sigma_I, \rangle$ be the relational schema obtained from $\mathcal{RS}$ by removing the KDs and EDs. Let $\mathcal{D}$ be a database instance for $\mathcal{RS}$ and $\mathcal{RS}_1$, q a query of arity n over $\mathcal{RS}$, and t an n-tuple of constants of $\mathcal{U}_\mathcal{D}$. We have that $t \notin ans_s(q, \mathcal{RS}, \mathcal{D})$ iff $\mathcal{D}$ is consistent with Σ_K and Σ_E^* and $t \notin ans_s(q, \mathcal{RS}_1, \mathcal{D})$.*

Based on the above theorem, we define the algorithm $\mathsf{Answer_S}$, that solves query answering in the case of NKCIDs, KDs and EDs.

Algorithm $\mathsf{Answer_S}(\mathcal{RS}, \mathcal{D}, q, t)$
Input: NKC database schema $\mathcal{RS} = \langle \Psi, \Sigma_I, \Sigma_K, \Sigma_E \rangle$, database instance $\mathcal{D}$,
query q of arity n over $\mathcal{RS}$, n-tuple t of constants of $\mathcal{U}_\mathcal{D}$;
Output: *true* if $t \in ans_s(q, \mathcal{RS}, \mathcal{D})$, *false* otherwise;
if $\mathcal{D}$ is not consistent with $\Sigma_K \cup \Sigma_E^*$
then return *true*
else return $\mathsf{Answer}_{JK}(\langle \Psi, \Sigma_I \rangle, \mathcal{D}, q, t)$

Soundness and completeness of the algorithm follow immediately from Theorem 4.13 and from soundness and completeness of Answer_{JK} [13].

To conclude the section, we present a complexity result for query answering in our setting under the sound semantics.

Theorem 4.14 *Let $\mathcal{RS} = \langle \Psi, \Sigma_I, \Sigma_K, \Sigma_E \rangle$ be a NKC schema, $\mathcal{D}$ a database instance for $\mathcal{RS}$, q a query of arity n over $\mathcal{RS}$, and t an n-tuple of constants of $\mathcal{U}_\mathcal{D}$. The problem of establishing whether $t \in ans_s(q, \mathcal{RS}, \mathcal{D})$ is PSPACE-complete with respect to combined complexity. Moreover, it is in PTIME in data complexity.*

Proof. As discussed From the results in [13], it follows directly that the problem in the case of IDs alone is PSPACE-complete in combined complexity; being such a case a particular case of NKCIDs, KDs and EDs (when no KD is defined, any ID is non-key-conflicting), PSPACE-hardness in our general case follows trivially.

Membership is proved by showing that the algorithm $\mathsf{Answer}_\mathsf{S}$ runs in PSPACE. Consider a database schema $\mathcal{RS} = \langle \Psi, \Sigma_I, \Sigma_K, \Sigma_E \rangle$ where Σ_I, Σ_K and Σ_E are sets of NKCIDs, KDs, and EDs, respectively. Our algorithm $\mathsf{Answer}_\mathsf{S}$ proceeds as follows. The first step, checks whether $\mathcal{D}$ satisfies $\Sigma_K \cup \Sigma_E^*$. Checking if $\mathcal{D}$ satisfies Σ_K is clearly feasible in PTIME (and *a fortiori* in PSPACE). Analogously, given an ED $\delta \in \Sigma_E^*$, checking if $\mathcal{D}$ satisfies δ can be done in PTIME. From Theorem 4.7, it follows that checking wether $\delta \in \Sigma_E^*$ is a PSPACE complete problem. Then, the first step of the algorithm is computable in PSPACE. If $\mathcal{D}$ does not satisfy $\Sigma_K \cup \Sigma_E^*$, the answer is trivial; if it does, we disregard Σ_K and Σ_E, and apply the PSPACE algorithm Answer_{JK}. Hence, All steps of $\mathsf{Answer}_\mathsf{S}$ are computable in PSPACE.

Membership in PTIME in data complexity follows immediately since Answer_{JK} runs in time polynomial in data complexity. □

It should be easy to see that the above complexity results also hold for the case in which only EDs and IDs are expressed over the relational schema, and for the case of only KDs and NKCIDs (proofs for these cases are analogous to the above proofs).

4.3 Query Answering under the Loose Semantics

In this section we analyze the problem of computing answers to queries under the loose semantics. In particular, since (as shown in Section 3.3) the loosely-complete and the strictly-complete semantics coincide, we study query answering under the loosely-sound semantics and the loosely-exact semantics.

4.3.1 Query Answering under the Loosely-sound Semantics

As we already said, differently from the strictly-sound semantics, given a relational schema $\mathcal{RS} = \langle \Psi, \Sigma_I, \Sigma_K, \Sigma_E \rangle$ and a database instance $\mathcal{D}$, it always holds that $sem_{ls}(\mathcal{RS}, \mathcal{D}) \neq \emptyset$, because we are now allowed to also eliminate facts from $\mathcal{D}$ in order to satisfy integrity constraints. Notice that, while to satisfy key and exclusion dependencies we are forced to delete facts from $\mathcal{D}$, we always try to satisfy inclusion dependencies by adding new facts, and resort to facts deletion also in this case only if there is no other chance to obtain a database consistent with $\mathcal{RS}$. This because databases in $sem_{ls}(\mathcal{RS}, \mathcal{D})$ are the ones that are "as sound as possible", thus we have to consider only databases consistent with the constraints that "minimize" elimination of facts from $\mathcal{D}$.

As shown in [15, 16], query answering under the loosely-sound semantics is undecidable for 1-key-conflicting relational schemas.

Theorem 4.15 *Let $\mathcal{RS} = \langle \Psi, \Sigma_I, \Sigma_K, \Sigma_E \rangle$ be a 1KC relational schema, $\mathcal{D}$ a database instance for $\mathcal{RS}$, q a query of arity n over $\mathcal{RS}$, and t a n-tuple of constants of $\mathcal{U}_\mathcal{D}$. The problem of establishing whether $t \in ans_{ls}(q, \mathcal{RS}, \mathcal{D})$ is undecidable.*

Proof. See proof of Theorem 4.7 of [16]. □

As for the class of non-key-conflicting relational schemas, we give a method for computing answers to a query q under the loosely-sound semantics that can be informally explained as follows: we first identify the maximal subsets of $\mathcal{D}$ that are consistent with $\Sigma_K \cup \Sigma_E^*$, then for each such database $\mathcal{D}'$ we make use of the algorithm $\mathsf{Answer_S}$ presented in Section 4.2.1. Indeed, it can be shown that a tuple t is a consistent answer to a query q with respect to $\mathcal{RS}$ and $\mathcal{D}$, i.e., $t \in ans_{ls}(q, \mathcal{RS}, \mathcal{D})$, iff $\mathsf{Answer_S}(\mathcal{RS}, \mathcal{D}', q, t)$ returns true for each such database $\mathcal{D}'$.

More specifically, we define the following algorithm:

Algorithm $\mathsf{Answer_{LS}}(\mathcal{RS}, \mathcal{D}, q, t)$
Input: non-key-conflicting relational schema $\mathcal{RS} = \langle \Psi, \Sigma_I, \Sigma_K, \Sigma_E \rangle$, database instance $\mathcal{D}$, query q of arity n over $\mathcal{RS}$, n-tuple t of constants of $\mathcal{U}_\mathcal{D}$;
Output: *true* if $t \in ans_{ls}(q, \mathcal{RS}, \mathcal{D})$, *false* otherwise;
if there exists $\mathcal{D}_1 \subseteq \mathcal{D}$
such that
(1) $\mathcal{D}_1$ is consistent with $\Sigma_K \cup \Sigma_E^*$;
(2) **for each** $r(t) \in \mathcal{D} - \mathcal{D}_1$,
$\mathcal{D}_1 \cup \{r(t)\}$ is not consistent with $\Sigma_K \cup \Sigma_E^*$;
(3) $\mathsf{Answer_S}(\mathcal{RS}, \mathcal{D}_1, q, t)$ returns *false*
then return *false*
else return *true*

Informally, conditions (1) and (2) together check that $\mathcal{D}_1$ is a maximal subset of $\mathcal{D}$ consistent with $\Sigma_K \cup \Sigma_E^*$; this implies the existence of a database $\mathcal{B} \in sem_{ls}(\mathcal{RS}, \mathcal{D})$ such that $\mathcal{B} \cap \mathcal{D} = \mathcal{D}_1$. Then, condition (3) verifies that $t \notin q^\mathcal{B}$.

Theorem 4.16 *Let $\mathcal{RS} = \langle \Psi, \Sigma_I, \Sigma_K, \Sigma_E \rangle$ be a NKC relational schema, $\mathcal{D}$ a database instance for $\mathcal{RS}$, q a query of arity n over $\mathcal{RS}$, and t a n-tuple of constants of $\mathcal{U}_\mathcal{D}$. Then, $t \in ans_{ls}(q, \mathcal{RS}, \mathcal{D})$ iff $\mathsf{Answer_{LS}}(\mathcal{RS}, \mathcal{D}, q, t)$ returns true.*

Proof. We first provide the following result that is needed in the proof

Lemma 4.17 *Let $\mathcal{RS} = \langle \Psi, \Sigma_I, \Sigma_K, \Sigma_E \rangle$ be a NKC relational schema and $\mathcal{D}$ a database instance for $\mathcal{RS}$. Then, $sem_{ls}(\mathcal{RS}, \mathcal{D}) = \bigcup_{\mathcal{D}'} sem_s(\mathcal{RS}, \mathcal{D}')$ for each $\mathcal{D}'$ maximal subset of $\mathcal{D}$ consistent with $\Sigma_K \cup \Sigma_E^*$.*

Proof. Given $\mathcal{B} \in sem_{ls}(\mathcal{RS}, \mathcal{D})$, then $\mathcal{D}' = \mathcal{B} \cap \mathcal{D}$ is a maximal subset of $\mathcal{D}$ consistent with $\Sigma_K \cup \Sigma_E^*$. Indeed, since $\mathcal{B}$ satisfies $\Sigma_K \cup \Sigma_E^*$, then every subset of $\mathcal{B}$ satisfies such constraints, and if there would exist $\mathcal{D}'' \subseteq \mathcal{D}$ such that $\mathcal{D}''$ satisfies $\Sigma_K \cup \Sigma_E^*$ and $\mathcal{D}'' \supset \mathcal{D}'$, $\mathcal{B}$ did not belong to $sem_{ls}(\mathcal{RS}, \mathcal{D})$. It easily follows that $\mathcal{B} \in sem_s(\mathcal{RS}, \mathcal{D}')$, since $\mathcal{B}$ is consistent with $\mathcal{RS}$, and $\mathcal{B} \supseteq \mathcal{D}'$. Consider now $\mathcal{D}'$ maximal subset of $\mathcal{D}$ consistent with $\Sigma_K \cup \Sigma_E^*$. From Theorem 4.13 it follows that $sem_s(\mathcal{RS}, \mathcal{D}')$ is not empty. Hence, for each $\mathcal{B}' \in sem_s(\mathcal{RS}, \mathcal{D}')$ we have also that $\mathcal{B}' \in sem_{ls}(\mathcal{RS}, \mathcal{D})$. Indeed, $\mathcal{B}'$ is consistent with $\mathcal{RS}$, and is minimal w.r.t. $\leq_{\mathcal{D}}$. The last property can be easily verified by assuming that there exist $\mathcal{B}'' <_{\mathcal{D}} \mathcal{B}'$, i.e., $\mathcal{B}''$ is consistent with $\mathcal{RS}$ and $\mathcal{B}'' \cap \mathcal{D} \supset \mathcal{B}' \cap \mathcal{D}$. Since $\mathcal{B}' \cap \mathcal{D} \supset \mathcal{B}' \cap \mathcal{D}' = \mathcal{D}'$, this would imply $\mathcal{B}'' \cap \mathcal{D} \supset \mathcal{D}'$, thus contradicting the assumption. □

We now provide the proof of the theorem.

"⇒"If $t \in ans_{ls}(q, \mathcal{RS}, \mathcal{D})$ then $t \in q^{\mathcal{B}}$ for each $\mathcal{B} \in sem_{ls}(\mathcal{RS}, \mathcal{D})$. From Lemma 4.17 it follows that $t \in ans_s(q, \mathcal{RS}, \mathcal{D}_1)$ for each $\mathcal{D}_1$ maximal subset of $\mathcal{D}$ consistent with $\Sigma_K \cup \Sigma_E^*$, and from soundness and completeness of algorithm $\mathsf{Answer}_{\mathsf{S}}$, it follows that $\mathsf{Answer}_{\mathsf{S}}(\mathcal{RS}, \mathcal{D}_1, q, t)$ returns *true* for each such database $\mathcal{D}_1$. Hence, $\mathsf{Answer}_{\mathsf{LS}}(\mathcal{RS}, \mathcal{D}, q, t)$ returns *true*.

"⇐"Suppose by contradiction that $t \notin ans_{ls}(q, \mathcal{RS}, \mathcal{D})$ and $\mathsf{Answer}_{\mathsf{LS}}(\mathcal{RS}, \mathcal{D}, q, t)$ returns *true*. This means that for each $\mathcal{D}_1$ maximal subset of $\mathcal{D}$ consistent with $\Sigma_K \cup \Sigma_E^*$, $\mathsf{Answer}_{\mathsf{S}}(\mathcal{RS}, \mathcal{D}_1, q, t)$ returns *true*. From soundness and completeness of algorithm $\mathsf{Answer}_{\mathsf{S}}$, it follows that $t \in ans_s(q, \mathcal{RS}, \mathcal{D}_1)$ for each such database $\mathcal{D}_1$, i.e., $t \in q^{\mathcal{B}}$ for each $\mathcal{B} \in sem_s(\mathcal{RS}, \mathcal{D}_1)$. From Lemma 4.17 it follows that $t \in q^{\mathcal{B}}$ for each $\mathcal{B} \in sem_{ls}(\mathcal{RS}, \mathcal{D})$, but this contradicts the assumption. □

We give now the computational characterization of the problem of query answering under the loosely-sound semantics in the presence of NKCIDs, KDs, and EDs. Since hardness of the problem has been studied in [16], we concentrate here on the complexity upper bounds, in order to establish the complexity classes for which the query answering problem is complete. We start by analyzing data complexity.

Lemma 4.18 *Let $\mathcal{RS} = \langle \Psi, \Sigma_I, \Sigma_K, \Sigma_E \rangle$ be a NKC relational schema, $\mathcal{D}$ a database instance for $\mathcal{RS}$, q a query of arity n over $\mathcal{RS}$, and t an n-tuple of constants of $\mathcal{U}_{\mathcal{D}}$. The problem of establishing whether $t \in ans_{ls}(q, \mathcal{RS}, \mathcal{D})$ is in coNP with respect to data complexity.*

Proof. Membership in coNP follows from the algorithm $\mathsf{Answer}_{\mathsf{LS}}(\mathcal{RS}, \mathcal{D}, q, t)$ and from Theorem 4.14. Indeed, in the algorithm the problem of establishing whether $t \notin ans_{ls}(q, \mathcal{RS}, \mathcal{D})$, that is the complement of our problem, is carried out by guessing a database and checking conditions (1), (2), and (3) that can be verified in polynomial time. □

It is easy to see that the above complexity characterization holds also in the absence of IDs on the relational schema $\mathcal{RS}$, i.e., when $\Sigma_I = \emptyset$, and in all the cases in which at least one between Σ_K or Σ_E is not empty. It is also worth noticing that query answering in the presence of only IDs over $\mathcal{RS}$ under the loosely-sound semantics is analogous to the same problem under the strictly-sound semantics, which has been addressed in the previous section.

Let us now turn our attention to lower bounds. According to [16], query answering under the loosely-sound semantics in the presence of either KDs or EDs is coNP-hard. Hence, whenever KDs or EDs are specified on the relational schema, the problem of query answering is coNP-complete with respect data complexity. In particular, we can provide the following theorem.

Theorem 4.19 *Let $\mathcal{RS} = \langle \Psi, \Sigma_I, \Sigma_K, \Sigma_E \rangle$ be a NKC relational schema, $\mathcal{D}$ a database instance for $\mathcal{RS}$, q a query of arity n over $\mathcal{RS}$, and t an n-tuple of constants of $\mathcal{U}_{\mathcal{D}}$. The problem of establishing whether $t \in ans_{ls}(q, \mathcal{RS}, \mathcal{D})$ is coNP-complete with respect to data complexity.*

Proof. The claim follows from Lemma 4.18, and from Theorem 4.8 or Theorem 4.9 of [16]. □

Let us now deal with combined complexity. We first consider query answering in the absence of IDs, and then address the general setting.

Theorem 4.20 *Let $\mathcal{RS} = \langle \Psi, \Sigma \rangle$ be a relational schema, where either $\Sigma = \Sigma_K$ or $\Sigma = \Sigma_E$ (or $\Sigma = \Sigma_K \cup \Sigma_E$). Moreover, let $\mathcal{D}$ be a database instance for $\mathcal{RS}$, q a query of arity n over $\mathcal{RS}$, and t an n-tuple of constants of $\mathcal{U}_{\mathcal{D}}$. The problem of establishing whether $t \in ans_{ls}(q, \mathcal{RS}, \mathcal{D})$ is Π_2^p-complete with respect to combined complexity.*

Proof. To prove membership in Π_2^p, consider the algorithm $\mathsf{Answer_{LS}}(\mathcal{RS}, \mathcal{D}, q, t)$. Since there are no IDs specified over $\mathcal{RS}$, the algorithm $\mathsf{Answer_S}(\mathcal{RS}, \mathcal{D}_1, q, t)$ in condition (3) simply verifies if $t \notin q^{\mathcal{D}_1}$, i.e., it performs classical query answering in the absence of integrity constraints. Since classical query processing over relational databases is combined complete for NP, the claim easily follows. Notice that the above proof holds either if $\Sigma = \Sigma_K$ or if $\Sigma = \Sigma_E$ (or if $\Sigma = \Sigma_K \cup \Sigma_E$).

Hardness has been proved in [16] by a reduction to our problem of the 2-QBF validity problem, i.e., the validity problem for quantified boolean formulae. Two different reductions have been given in the two cases in which $\Sigma = \Sigma_K$ or $\Sigma = \Sigma_E$. □

As for the general case, we prove in the following that the presence of IDs significantly complicates the problem.

Theorem 4.21 *Let $\mathcal{RS} = \langle \Psi, \Sigma_I, \Sigma_K, \Sigma_E \rangle$ be a NKC relational schema, $\mathcal{D}$ a database instance for $\mathcal{RS}$, q a query of arity n over $\mathcal{RS}$, and t an n-tuple of constants of $\mathcal{U}_{\mathcal{D}}$. The problem of establishing whether $t \in ans_{ls}(q, \mathcal{RS}, \mathcal{D})$ is PSPACE-complete with respect to combined complexity.*

Proof.

We first point out that in the presence of only IDs, the loosely-sound coincides with the strictly-sound semantics. Hence PSPACE-hardness established for the strictly-sound semantics (see Theorem 4.6 of [16]) holds also in this case.

Membership in PSPACE follows from algorithm $\mathsf{Answer_{LS}}(\mathcal{RS}, \mathcal{D}, q, t)$ and Theorem 4.14. Indeed, it is easy to see that conditions (1), (2), and (3) can be verified in polynomial space. Then, by Savitch's theorem [21] $\mathsf{Answer_{LS}}$ can be transformed into a deterministic algorithm that runs in polynomial space. □

Notice that, since PSPACE-completeness of query answering under the strictly-sound semantics also holds in the presence of only IDs and EDs, and only IDs and NKCIDs on the relational schema, as said at end of Section 4.2.1, PSPACE-completeness of query answering under the loosely-sound semantics holds for NKC schemas of the form $\mathcal{RS} = \langle\Psi, \Sigma_I, \Sigma_K\rangle$ and of the form $\mathcal{RS} = \langle\Psi, \Sigma_I, \Sigma_E\rangle$.

4.3.2 Query Answering under the Loosely-exact Semantics

We now study the query answering problem under the loosely-exact semantics. We recall that, differently from the loosely-sound semantics, in this case IDs can be always satisfied by either adding or deleting facts. Hence, $sem_{le}(\mathcal{RS}, \mathcal{D})$ accounts for databases that minimize both elimination and insertion of facts, i.e., that are "as exact as possible".

As shown in [15, 16], query answering under the loosely-exact semantics is undecidable in the general case, i.e., when no restriction is imposed on the form of IDs, KDs and EDs.

Theorem 4.22 *Let $\mathcal{RS} = \langle\Psi, \Sigma_I, \Sigma_K, \Sigma_E\rangle$ be a relational schema, $\mathcal{D}$ a database instance for $\mathcal{RS}$, q a query of arity n over $\mathcal{RS}$, and t an n-tuple of constants of $\mathcal{U_D}$. The problem of establishing whether $t \in ans_{le}(q, \mathcal{RS}, \mathcal{D})$ is undecidable.*

Proof. See proof of Theorem 4.12 of [15]. □

Let us turn our attention on query answering under the loosely-exact semantics in the case of NKC relational schemas. To this aim, we first define a particular query $Q(\delta_i, t)$ associated with a tuple t and an inclusion dependency δ_i.

Definition 4.23 Let δ_i be an inclusion dependency of the form $r[i_1, \ldots, i_k] \subseteq s[j_1, \ldots, j_k]$, where r has arity n and s has arity m, and let t be an n-tuple. We denote as $Q(\delta_i, t)$ the boolean conjunctive query $q \leftarrow s(z_1, \ldots, z_m)$, where, for each ℓ such that $1 \leq \ell \leq m$, each z_ℓ is as follows: if there exists h such that $\ell = j_h$ then $z_\ell = t[i_h]$, otherwise $z_\ell = X_\ell$.

In the following, the query $Q(\delta_i, t)$ is used in order to verify whether a relational schema $\mathcal{RS}$ and an instance $\mathcal{D}$ imply the existence in all databases $\mathcal{B} \in sem_s(\mathcal{RS}, \mathcal{D})$ of a fact of the form $s(t')$ such that $t[i_1, \ldots, i_k] = t'[j_1, \ldots, j_k]$.

Below we define the algorithm $\mathsf{Answer_{LE}}$ for query answering under the loosely-exact semantics.

Algorithm $\mathsf{Answer_{LE}}(\mathcal{RS}, \mathcal{D}, q, t)$
Input: NKC relational schema $\mathcal{RS} = \langle\Psi, \Sigma_I, \Sigma_K, \Sigma_E\rangle$, database instance $\mathcal{D}$,
query q of arity n over $\mathcal{RS}$, n-tuple t of constants of $\mathcal{U_D}$
Output: *true* if $t \in ans_{le}(q, \mathcal{RS}, \mathcal{D})$, *false* otherwise
if there exists $\mathcal{D}' \subseteq \mathcal{D}$ **such that**
(a) $\mathcal{D}'$ is consistent with $\Sigma_K \cup \Sigma_E^*$ **and**
(b) $\mathsf{Answer_S}(\langle\Psi, \Sigma_I\rangle, \mathcal{D}', q, t)$ returns *false* **and**
(c) **for each** $\mathcal{D}''$ such that $\mathcal{D}' \subset \mathcal{D}'' \subseteq \mathcal{D}$
(c1) $\mathcal{D}''$ is not consistent with $\Sigma_K \cup \Sigma_E^*$ **or**
(c2) **there exists** $\delta_i \in \Sigma_I$ **and** $r(t_1) \in \mathcal{D}''$
such that

$\mathsf{Answer_S}(\langle\Psi, \Sigma_I\rangle, \mathcal{D}', Q(\delta_i, t_1), \langle\rangle)$ returns *false* **and**
$\mathsf{Answer_S}(\langle\Psi, \emptyset\rangle, \mathcal{D}'', Q(\delta_i, t_1), \langle\rangle)$ returns *false*
then return *false*
else return *true*

Intuitively, to return *false* the algorithm looks for the existence of a database $\mathcal{B}'$ in $sem_{le}(\mathcal{RS}, \mathcal{D})$ such that $t \notin q^{\mathcal{B}'}$. As in the algorithm $\mathsf{Answer_{LS}}$, the database $\mathcal{B}'$ is represented by its intersection with the initial instance $\mathcal{D}$ (denoted as $\mathcal{D}'$ in the algorithm): the fact that $t \notin q^{\mathcal{B}'}$ is verified by condition (b), while the fact that $\mathcal{B}' \in sem_{le}(\mathcal{RS}, \mathcal{D})$ is verified by conditions (a) and (c) of the algorithm. In particular, condition (c) verifies that, for each database $\mathcal{B}''$ (represented by its intersection with $\mathcal{D}$ denoted as $\mathcal{D}''$), it is not the case that $\mathcal{B}'' <_{\mathcal{D}} \mathcal{B}'$. In conditions (c1) and (c2), the symbol $\langle\rangle$ denotes the empty tuple.

Soundness and completeness of the algorithm is established by the following theorem.

Theorem 4.24 *Let $\mathcal{RS} = \langle\Psi, \Sigma_I, \Sigma_K, \Sigma_E\rangle$ be a NKC relational schema, $\mathcal{D}$ a database instance, q a query of arity n over Ψ, and t an n-tuple of constants of $\mathcal{U}_{\mathcal{D}}$. Then, $t \in ans_{le}(q, \mathcal{RS}, \mathcal{D})$ iff $\mathsf{Answer_{LE}}(\mathcal{RS}, \mathcal{D}, q, t)$ returns true.*

Proof. In order to prove correctness of the above algorithm, we need a preliminary lemma. In the following, given an instance $\mathcal{D}$ of a relational schema $\mathcal{RS} = \langle\Psi, \Sigma_I\rangle$, we denote as $chase_1(\mathcal{RS}, \mathcal{D})$ the set of new facts obtained by applying the chase rule to the facts in $\mathcal{D}$, i.e., the set of facts of the form $s(t_2)$ such that there exist $\delta_i = r[i_1, \ldots, i_k] \subseteq s[j_1, \ldots, j_k] \in \Sigma_I$ and $r(t_1) \in \mathcal{D}$ such that $t_1[i_1, \ldots, i_k] = t_2[j_1, \ldots, j_k]$ and there exists no $s(t_3) \in \mathcal{D}$ such that $t_1[i_1, \ldots, i_k] = t_3[j_1, \ldots, j_k]$.

Lemma 4.25 *Let $\mathcal{D}', \mathcal{D}''$ be instances of a relational schema $\mathcal{RS} = \langle\Psi, \Sigma_I\rangle$ such that $\mathcal{D}' \subset \mathcal{D}''$ and, for each $\delta_i \in \Sigma_I$ of the form $\delta_i = r[i_1, \ldots, i_k] \subseteq s[j_1, \ldots, j_k]$ and for each $r(t_1) \in \mathcal{D}''$, either $\mathsf{Answer_S}(\langle\Psi, \Sigma_I\rangle, \mathcal{D}', Q(\delta_i, t_1), \langle\rangle)$ returns true or $\mathsf{Answer_S}(\langle\Psi, \emptyset\rangle, \mathcal{D}'', Q(\delta_i, t_1), \langle\rangle)$ returns true. Then, $chase(\mathcal{RS}, \mathcal{D}'') - \mathcal{D}'' \subseteq chase(\mathcal{RS}, \mathcal{D}') - \mathcal{D}'$.*

Proof. It is straightforward to verify that the hypothesis implies that $chase_1(\mathcal{RS}, \mathcal{D}'') \subseteq chase_1(\mathcal{RS}, \mathcal{D}')$; this in turn implies that each new fact added in $chase(\mathcal{RS}, \mathcal{D}'')$ by an application of the chase rule in $chase(\mathcal{RS}, \mathcal{D}'')$ is also added by the chase rule in $chase(\mathcal{RS}, \mathcal{D}')$. Consequently, the thesis follows. □

We now prove the theorem.

"⇒"Suppose $\mathsf{Answer_{LE}}(\mathcal{RS}, \mathcal{D}, q, t)$ returns *false*. Then, there exists $\mathcal{D}' \subseteq \mathcal{D}$ such that conditions (a), (b) and (c) of the algorithm hold for $\mathcal{D}'$. Let $\mathcal{B}' = chase(\mathcal{RS}, \mathcal{D}')$. Now, suppose $\mathcal{B}' \notin sem_{le}(\mathcal{RS}, \mathcal{D})$: hence, there exists a database instance $\mathcal{B}''$ such that $\mathcal{B}''$ is consistent with Σ_K and $\mathcal{B}'' <_{\mathcal{D}} \mathcal{B}'$, which implies that $\mathcal{B}'' - \mathcal{D} \subseteq \mathcal{B}' - \mathcal{D}$. Since by hypothesis condition (c) holds for $\mathcal{D}'$, it follows that condition (c2) holds for $\mathcal{D}''$, i.e., there exists a fact $r(t_1) \in \mathcal{D}''$ and an inclusion $\delta_i = r[i_1, \ldots, i_k] \subseteq s[j_1, \ldots, j_k] \in \Sigma_I$ such that:

1. $\mathsf{Answer_S}(\langle\Psi, \Sigma_I\rangle, \mathcal{D}', Q(\delta_i, t_1), \langle\rangle)$ returns *false*, which implies that there is no fact in $\mathcal{B}'$ of the form $s(t_2)$ such that $t_1[i_1, \ldots, i_k] = t_2[j_1, \ldots, j_k]$;

2. $\mathsf{Answer_S}(\langle\Psi,\emptyset\rangle,\mathcal{D}'',Q(\delta_i,t_1),\langle\rangle)$ returns *false*, which implies that there is no fact in $\mathcal{D}''$ of the form $s(t_2)$ such that $t_1[i_1,\ldots,i_k]=t_2[j_1,\ldots,j_k]$. On the other hand, a fact of the form $s(t_2)$ such that $t_1[i_1,\ldots,i_k]=t_2[j_1,\ldots,j_k]$ must be present in $\mathcal{B}''$, due to the presence of $r(t_1)$ in $\mathcal{D}''$ and to the inclusion δ_i.

The two above conditions imply that there exists a fact of the form $s(t_2)$ in $\mathcal{B}''-\mathcal{D}$ which does not belong to $\mathcal{B}'$. Consequently, $\mathcal{B}''-\mathcal{D}\subseteq\mathcal{B}'-\mathcal{D}$ does not hold, thus contradicting the hypothesis that $\mathcal{B}''<_{\mathcal{D}}\mathcal{B}'$. Therefore, $\mathcal{B}'\in sem_{le}(\mathcal{RS},\mathcal{D})$, and since conditions (a) and (b) hold for $\mathcal{D}'$, it follows that $t\notin q^{\mathcal{B}'}$, hence $t\notin ans_{le}(q,\mathcal{RS},\mathcal{D})$.

"$\Leftarrow$" Suppose $t\notin ans_{le}(q,\mathcal{RS},\mathcal{D})$. Therefore, there exists $\mathcal{B}'\in sem_{le}(\mathcal{RS},\mathcal{D})$ such that $t\notin q^{\mathcal{B}'}$. Let $\mathcal{D}'=\mathcal{D}\cap\mathcal{B}'$. Since $\mathcal{B}'\in sem_{le}(\mathcal{RS},\mathcal{D})$, then $\mathcal{B}'$ satisfies $\Sigma_I\cup\Sigma_E^*$, and since $\mathcal{D}'\subseteq\mathcal{B}'$, condition (a) of the algorithm holds. From $t\notin q^{\mathcal{B}'}$ and from soundness and completeness of the algorithm $\mathsf{Answer_S}$ it follows that condition (b) holds for $\mathcal{D}'$. Now, suppose condition (c) does not hold for $\mathcal{D}'$: then, there exists $\mathcal{D}''$ such that conditions (c1) and (c2) do not hold for $\mathcal{D}'$ and $\mathcal{D}''$, i.e., $\mathcal{D}''$ is consistent with $\Sigma_K\cup\Sigma_E^*$ and, for each $\delta_i\in\Sigma_I$ of the form $\delta_i=r[i_1,\ldots,i_k]\subseteq s[j_1,\ldots,j_k]$ and for each $r(t_1)\in\mathcal{D}''$, either $\mathsf{Answer_S}(\langle\Psi,\Sigma_I\rangle,\mathcal{D}',Q(\delta_i,t_1),\langle\rangle)$ returns *true* or $\mathsf{Answer_S}(\langle\Psi,\emptyset\rangle,\mathcal{D}'',Q(\delta_i,t_1),\langle\rangle)$ returns *true*. By Lemma 4.25, it follows that $chase(\mathcal{RS},\mathcal{D}'')-\mathcal{D}''\subseteq chase(\mathcal{RS},\mathcal{D}')-\mathcal{D}'$. Now let $\mathcal{B}''=chase(\mathcal{RS},\mathcal{D}'')$: since $\mathcal{B}'\supseteq chase(\mathcal{RS},\mathcal{D}')$, it follows that $\mathcal{B}''-\mathcal{D}\subseteq\mathcal{B}'-\mathcal{D}$, and by hypothesis $\mathcal{D}''\supset\mathcal{D}'$, therefore $\mathcal{B}''\cap\mathcal{D}\supset\mathcal{B}'\cap\mathcal{D}$, hence $\mathcal{B}''<_{\mathcal{D}}\mathcal{B}'$. Moreover, since $\mathcal{D}''$ is consistent with $\Sigma_K\cup\Sigma_E^*$, $\mathcal{B}''$ is consistent with Σ_K, Σ_E and Σ_I, consequently $\mathcal{B}'\notin sem_{le}(\mathcal{RS},\mathcal{D})$, thus contradicting the hypothesis. Therefore, condition (c) holds for $\mathcal{D}'$, which implies that $\mathsf{Answer_{LE}}(\mathcal{RS},\mathcal{D},q,t)$ returns *false*. □

Finally, based on the above algorithm, we analyze the computational complexity of query answering under the loosely-exact semantics for NKC relational schemas.

Theorem 4.26 *Let $\mathcal{RS}=\langle\Psi,\Sigma_I,\Sigma_K,\Sigma_E\rangle$ be a NKC relational schema, $\mathcal{D}$ a database instance, q a query of arity n over $\mathcal{RS}$, and t an n-tuple of constants of $\mathcal{U}_{\mathcal{D}}$. The problem of establishing whether $t\in ans_{le}(q,\mathcal{RS},\mathcal{D})$ is Π_2^p-complete with respect to data complexity and PSPACE-complete with respect to combined complexity.*

Proof. The analysis of the algorithm $\mathsf{Answer_{LE}}$ shows that the problem is in Π_2^p w.r.t. data complexity. Indeed, it is immediate to verify that:

- condition (a) can be verified in polynomial time;
- condition (b) can be verified in polynomial time, as shown in Section 4.2.1;
- conditions (c1) and (c2) can be verified in polynomial time: therefore, condition (c) can be verified in nondeterministic polynomial time.

Consequently, if considered as a nondeterministic procedure, the algorithm runs in Σ_2^p w.r.t. data complexity, hence our problem (that is the complement of the problem solved by $\mathsf{Answer_{LE}}$) is in Π_2^p w.r.t. data complexity.

Hardness with respect to Π_2^p follows form Theorem 4.13 or Theorem 4.14 of [16], where Π_2^p-hardness of query answering under the loosely-exact semantics has been proved for the cases of IDs and KDs and IDs and EDs, respectively.

Data/Combined complexity for query answering:

EDs	KDs	IDs	strictly-sound	loosely-sound	loosely-exact
no	no	GEN	PTIME/PSPACE	PTIME/PSPACE	PTIME/NP
yes-no	yes	no	PTIME/NP	coNP/Π_2^p	coNP/Π_2^p
yes	yes-no	no	PTIME/NP	coNP/Π_2^p	coNP/Π_2^p
yes-no	yes	NKC	PTIME/PSPACE	coNP/PSPACE	Π_2^p/PSPACE
yes	no	GEN	PTIME/PSPACE	coNP/PSPACE	Π_2^p/PSPACE
yes-no	yes	1KC	undecidable	undecidable	undecidable
yes-no	yes	GEN	undecidable	undecidable	undecidable

Legenda: GEN = general IDs, NKC = non-key-conflicting IDs, 1KC = 1-key-conflicting IDs.

Figure 1: Complexity of query answering under EDs, KDs and IDs (decision problem)

As concerns combined complexity, it is immediate to verify that each of the conditions of the algorithm is computed in nondeterministic polynomial space, therefore the algorithm runs in nondeterministic polynomial space w.r.t. combined complexity, which proves membership in PSPACE of the problem. PSPACE-hardness can be proved by reducing query answering under loosely-sound semantics for databases without key dependencies to this problem, as discussed in [15] (proof of Theorem 4.13). □

According to the above theorem, and Theorems 4.13 and 4.14 of [16], the complexity of the query answering problem in the absence of EDs or in the absence of KDs, i.e., when $\mathcal{RS} = \langle \Psi, \Sigma_I, \Sigma_K \rangle$ or $\mathcal{RS} = \langle \Psi, \Sigma_I, \Sigma_E \rangle$, actually coincides with the complexity of the general case.

We finally point out that query answering under the loosely-exact semantics in the absence of IDs coincides with the same problem under the loosely-sound semantics, hence algorithms and complexity results are the same in the two cases.

4.4 Summary of Results

The summary of the results we have obtained is reported in Figure 1. Each column (with the exception of the first three) corresponds to a different semantics, while each row corresponds to a different combination of dependencies (specified in the first three columns, where *yes* indicates the presence of the dependencies, *no* the absence, and *yes-no* that the corresponding complexity is independent by these dependencies). Each cell of the table reports data complexity and combined complexity of query answering: for each decidable case, the complexity of the problem is complete with respect to the class reported (with the exception of the cases in which the complexity is PTIME).

Complexity under the strictly-sound semantics for the case in which no IDs are specified on the schema coincides with the complexity of standard query answering over relational databases [1].

We point out that, due to the correspondence between query answering and query containment illustrated in Section 4.2.1, all the complexity results established for the problem of query answering also hold for the conjunctive query containment problem.

4.5 Data integration setting

We now go back to the integration framework and generalize to this setting the results provided so far for the single database setting. To this aim, we simply reduce the problem of query answering over a relational schema to query answering in either GAV or LAV data integration systems. Indeed, as shown in Section 4.1, a relational schema can be actually seen as the global schema of a particular data integration system $\mathcal{I} = \langle \mathcal{G}, \mathcal{S}, \mathcal{M} \rangle$, and hence there is a tight connection between query answering in the two cases.

More precisely, given a relational schema $\mathcal{RS}$ that contains the relations $g_1/h_1, \ldots, g_n/h_n$, a database $\mathcal{D}'$ for $\mathcal{RS}$, a query q over $\mathcal{RS}$ and a tuple t of constants of the domain $\mathcal{U}$, we can construct, in polynomial time in the size of the schema and the query, a data integration system $\mathcal{I} = \langle \mathcal{G}, \mathcal{S}, \mathcal{M} \rangle$ as follows:

- $\mathcal{G} = \mathcal{RS}$;
- $\mathcal{S}$ contains the relations $s_1/h_1, \ldots, s_n/h_n$;
- if $\mathcal{I}$ is GAV, $\mathcal{M}$ is given by the n assertions of the form $\langle g_i, g_i(X_1, \ldots, X_k) \leftarrow s_i(X_1, \ldots, X_k) \rangle$, for each i, $1 \leq i \leq n$, whereas if $\mathcal{I}$ is LAV each assertion assumes the form $\langle s_i, s_i(X_1, \ldots, X_k) \leftarrow g_i(X_1, \ldots, X_k) \rangle$;
- a source database $\mathcal{D}$ for $\mathcal{I}$ is constructed by assigning to each source relation s_i the tuples that $\mathcal{D}'$ assigns to the corresponding global relation g_i, i.e., $s_i^{\mathcal{D}} = g_i^{\mathcal{D}'}$;
- q and t remain unchanged.

It should be easy to see that $sem_{as(\mathcal{D}')}(\mathcal{RS}, \mathcal{D}) = sem_{as(\mathcal{M})}(\mathcal{I}, \mathcal{D})$[10], hence, $t \in ans_{as(\mathcal{D}')}(\mathcal{RS}, \mathcal{D})$ iff $t \in ans_{as(\mathcal{M})}(\mathcal{G}, \mathcal{S}, \mathcal{M})$. The above reduction actually proves the following theorem.

Theorem 4.27 *Let $\mathcal{I} = \langle \mathcal{G}, \mathcal{S}, \mathcal{M} \rangle$ be a GAV data integration system, where $\mathcal{G} = \langle \Psi, \Sigma_I, \Sigma_K, \Sigma_E \rangle$, and Σ_I, Σ_K and Σ_E are sets of 1KCIDs w.r.t. Σ_K, KDs and EDs, respectively. Let $\mathcal{D}$ be a source database for $\mathcal{I}$, q a user query of arity n over $\mathcal{G}$, and t an n-tuple of constants of $\mathcal{U}$. Then, the problem of calculating $ans_{as(\mathcal{M})}(q, \mathcal{I}, \mathcal{D})$ is undecidable for the sound, the loosely-sound and the loosely-exact assumption on the mapping $\mathcal{M}$, i.e., for $as(\mathcal{M}) \in \{s, ls, le\}$, respectively.*

In the following we call 1-key-conflicting integration systems the systems where the global schema $\mathcal{G}$ is a 1KC relational schema, and non-key-conflicting integration systems the ones in which $\mathcal{G}$ is a NKC schema.

Notice that the above reduction allows us also to extend to both GAV and LAV integration frameworks all hardness results provided in the setting of a single relational schema. To conclude that also the same upper bounds hold in this setting, we need to

[10]Notice that in the left-hand side *sem* refers to a data integration system, whereas in the right hand side it refers to a database schema.

establish membership results for all the combinations of IDs, KDs and EDs under the different semantics considered in our framework. In the following we separately address this problem first for the GAV and then for the LAV framework.

4.5.1 GAV Framework

We start by providing the following definition:

Definition 4.28 Given a data integration system $\mathcal{I} = \langle \mathcal{G}, \mathcal{S}, \mathcal{M} \rangle$ and a source database $\mathcal{D}$ for $\mathcal{I}$, we call *retrieved global database*, denoted $ret(\mathcal{I}, \mathcal{D})$, the global database obtained by evaluating each query of the mapping $\mathcal{M}$ over $\mathcal{D}$, i.e., $ret(\mathcal{I}, \mathcal{D}) = \{r_{\mathcal{G}}(t) \mid t \in q_{\mathcal{S}}^{\mathcal{D}}$ for each $\langle r_{\mathcal{G}}, q_{\mathcal{S}} \rangle \in \mathcal{M}\}$.

Since queries in $\mathcal{M}$ are non-recursive Datalog queries, then $ret(\mathcal{I}, \mathcal{D})$ is unique.

A significant property of the retrieved global database is that it satisfies the mapping, whatever assumption on $\mathcal{M}$ is adopted. In particular, under the exact assumption $ret(\mathcal{I}, \mathcal{D})$ is the only database that satisfies the mapping, whereas under the sound or the complete assumption, it is in general one of several databases that satisfy the mapping, viz. all the databases that contain $ret(\mathcal{I}, \mathcal{D})$, or that are contained in $ret(\mathcal{I}, \mathcal{D})$, in the two cases, respectively. It should be easy now to generalize to each GAV data integration system the property holding for the particular case in which in $\mathcal{I}$ there is a one-to-one correspondence between relations in $\mathcal{G}$ and relations in $\mathcal{S}$. Indeed, the semantics of $\mathcal{I}$ with respect to a source database $\mathcal{D}$ and an assumption $as(\mathcal{M})$ on $\mathcal{M}$, coincides with the semantics of the global schema $\mathcal{G}$ with respect to the database instance $ret(\mathcal{I}, \mathcal{D})$ and the same assumption $as(ret(\mathcal{I}, \mathcal{D}))$ on $ret(\mathcal{I}, \mathcal{D})$, i.e., $as(\mathcal{M}) = as(ret(\mathcal{I}, \mathcal{D}))$. Hence, answering a user query q in such a setting can be tackled in two steps, where the first one aims at computing the retrieved global database, and the second one at calculating the certain answers to q over $ret(\mathcal{I}, \mathcal{D})$, taking into account the assumption on $ret(\mathcal{I}, \mathcal{D})$, i.e., $ans_{as(\mathcal{M})}(q, \mathcal{I}, \mathcal{D}) = ans_{as(ret(\mathcal{I},\mathcal{D}))}(q, \mathcal{G}, ret(\mathcal{I}, \mathcal{D}))$.

We are now able to provide the separation theorem for GAV integration systems thus extending to this setting the result established in Theorem 4.13.

Theorem 4.29 *Let $\mathcal{I} = \langle \mathcal{G}, \mathcal{S}, \mathcal{M} \rangle$ be a NKC GAV system, where $\mathcal{G} = \langle \Psi, \Sigma_I, \Sigma_K, \Sigma_E \rangle$, let $\mathcal{I}' = \langle \mathcal{G}', \mathcal{S}, \mathcal{M} \rangle$, where $\mathcal{G}' = \langle \Psi, \Sigma_I \rangle$, be the system obtained by $\mathcal{I}$ by eliminating the KDs and the EDs of $\mathcal{G}$; let $\mathcal{D}$ be a source database for $\mathcal{I}$ and $\mathcal{I}'$. Moreover, let q be a query of arity n over $\mathcal{G}$ and $\mathcal{G}'$, and t an n-tuple of constants of $\mathcal{U}_{\mathcal{D}}$. We have that $t \notin ans_s(q, \mathcal{I}, \mathcal{D})$ iff $ret(\mathcal{I}, \mathcal{D})$ is consistent with $\Sigma_K \cup \Sigma_E^*$ and $t \notin ans_s(q, \mathcal{I}', \mathcal{D})$, where Σ_E^* indicates the set EDs that are logically implied by the dependencies in $\Sigma_E \cup \Sigma_I$.*

Proof. The thesis is an immediate consequence of Theorem 4.13, and of $ans_{as(\mathcal{M})}(q, \mathcal{I}, \mathcal{D}) = ans_{as(ret(\mathcal{I},\mathcal{D}))}(q, \mathcal{G}, ret(\mathcal{I}, \mathcal{D}))$. □

Finally, we can conclude that all membership results given in the setting of a single relational schema hold in our framework for NKC GAV systems as well. Indeed, the computation of the retrieved global database is in PTIME in data complexity and it is PSPACE-complete in combined complexity, since evaluating non-recursive Datalog is

PTIME-complete in data complexity and PSPACE-complete in combined complexity[11] [8]. Obviously, this generalizes to GAV systems all complexity results reported in Figure 1.

4.5.2 LAV Framework

Consider now LAV systems: the question arises here of whether the separation property between NKCIDs and KDs holds also in this case, i.e., whether the analogous of Theorem 4.29 exists for the LAV framework. Furthermore, we would also establish the complexity classes for which the query answering problem for LAV systems is complete under the different semantics for different combinations of integrity constraints on the global schema.

Unfortunately, there does not exist a particular global database that plays for LAV systems the same role played by the retrieved global database for GAV systems. This can be easily seen, since LAV mappings provide in general only partial information about the data that satisfy the global schema, and several ways of populating the global schema according to the mapping may exist (even if the mapping is considered exact). Hence, it is not possible to characterize query answering in LAV under the strict and loose semantics as done for the GAV systems. Nonetheless, we are able to provide a definition of retrieved global database in LAV that is specific for the strictly-sound semantics, and that allows us to show that a separation theorem holds for NKC LAV systems under this semantics.

In order to achieve this final result, we first study query answering under the strictly-sound semantics in the presence of only IDs, then we consider IDs and EDs, and finally, we address the general setting of NKCIDs, KDs and EDs.

Our plan of attack is as follows. First, we consider LAV systems without constraints, and define the notion of retrieved global database for this kind of systems. We show that, analogously to the GAV case, in LAV systems without constraints the answer to a user query q can be calculated by evaluating q over the retrieved global database. With this notion in place, we can apply the results of Section 4.2, which show that, in the presence of IDs, the answer to a query q over a database $\mathcal{DB}$ is computed by evaluating q over the chase of $\mathcal{DB}$. We then consider the case of IDs and EDs, and show that the notion of retrieved global database still allows for query answering in this setting. Finally, for the case in which also KDs are specified on the global schema a different definition of retrieved global database will be taken into account.

In order to distinguish between the retrieved global database of GAV systems and LAV systems, hereinafter we denote the former with ret_G and the latter with ret_L.

Definition 4.30 Let $\mathcal{I} = \langle \mathcal{G}, \mathcal{S}, \mathcal{M} \rangle$ be a LAV data integration system, and $\mathcal{D}$ a source database for $\mathcal{I}$. The *retrieved global database* $ret_L(\mathcal{I}, \mathcal{D})$ is defined constructively as follows. Consider a mapping assertion $\langle s, V_s \rangle \in \mathcal{M}$ where $s \in \mathcal{S}$ and V_s is of the form

$$s(\vec{\mathbf{x}}, \vec{\mathbf{c}}) \leftarrow conj(\vec{\mathbf{x}}, \vec{\mathbf{e}}, \vec{\mathbf{y}})$$

where $conj$ is a conjunction of predicates, $\vec{\mathbf{x}} = X_1, \ldots, X_{n_d}$ is a sequence of n_d distinguished variables, $\vec{\mathbf{c}} = c_1, \ldots, c_{n_{c,h}}$ and $\vec{\mathbf{e}} = e_1, \ldots, e_{n_c}$ are constants of the domain $\mathcal{U}$, and

[11]Notice that allowing recursion in the mapping would lead to different combined complexity results, since Datalog is combined complete for EXPTIME [8].

$\vec{\mathbf{y}} = Y_1, \ldots, Y_{n_n}$ is a sequence of n_n non-distinguished variables. Without loss of generality, and for the sake of clearness of presentation, we assume that, for all predicates in *conj*, the symbols (when present) appear in the following order: first the distinguished variables, then the constants, and finally the non-distinguished variables.

To each tuple $(a_1, \ldots, a_{n_d}, c_1, \ldots, c_{n_{c,h}}) \in s^{\mathcal{D}}$ we associate the constant a_i to X_i, and a fresh constant z_i of $\mathcal{U}$ (not appearing elsewhere in the database, and that has never been introduced in the construction) to Y_i. Then, for each atom $g \in conj$ of the form $g(X_1, \ldots, X_k, e_1, \ldots, e_\ell, Y_1, \ldots, Y_m)$, where $\{X_1, \ldots, X_k\} \subseteq \vec{\mathbf{x}}$, $\{e_1, \ldots, e_\ell\} \subseteq \vec{\mathbf{e}}$, and $\{Y_1, \ldots, Y_m\} \subseteq \vec{\mathbf{y}}$, we add the fact $g(a_1, \ldots, a_k, e_1, \ldots, e_\ell, z_1, \ldots, z_m)$ in $ret_L(\mathcal{I}, \mathcal{D})$. Note that the join variables in the body of the view are assigned with the same fresh constants. Moreover, the construction process is finite.

Lemma 4.31 *Let $\mathcal{I} = \langle \mathcal{G}, \mathcal{S}, \mathcal{M} \rangle$ be a LAV data integration system, with $\mathcal{G} = \langle \Psi, \emptyset \rangle$ and $\mathcal{D}$ a source database for $\mathcal{I}$. We have that for any database $\mathcal{B} \in sem_s(\mathcal{I}, \mathcal{D})$ there exists a well-defined homomorphism from $ret_L(\mathcal{I}, \mathcal{D})$ to $\mathcal{B}$ that sends constants of the active domain $\mathcal{U}_{\mathcal{D}}$ to themselves and fresh constants to constants of $\mathcal{U}$, and such that each fact of $g^{ret_L(\mathcal{I},\mathcal{D})}$ is sent to a fact of $g^{\mathcal{B}}$.*

Proof. By definition of sound mapping $\mathcal{M}$, and by construction of $ret_L(\mathcal{I}, \mathcal{D})$, any database in $sem_s(\mathcal{I}, \mathcal{D})$ is forced to have a minimum set of facts $\mathcal{B}_m$, constituted by the facts obtained from $ret_L(\mathcal{I}, \mathcal{D})$ by replacing each fresh constant with some constant in $\mathcal{U}$. The homomorphism μ is easily constructed by mapping each fact of $ret_L(\mathcal{I}, \mathcal{D})$ to the corresponding fact of $\mathcal{B}_m$. □

Theorem 4.32 *Let $\mathcal{I} = \langle \mathcal{G}, \mathcal{S}, \mathcal{M} \rangle$ be a LAV data integration system, with $\mathcal{G} = \langle \Psi, \emptyset \rangle$; let $\mathcal{D}$ be a source database for $\mathcal{S}$, and q a query over $\mathcal{G}$ of arity n. Moreover, let t be an n-tuple of constants of $\mathcal{U}_{\mathcal{D}}$, i.e., t does not contain any of the fresh constants z_i introduced in the construction of $ret_L(\mathcal{I}, \mathcal{D})$. Then, we have that $t \in ans_s(q, \mathcal{I}, \mathcal{D})$ iff $t \in q^{ret_L(\mathcal{I},\mathcal{D})}$.*

Proof.

"⇒"This is trivial because $ret_L(\mathcal{I}, \mathcal{D})$ belongs to $sem_s(\mathcal{I}, \mathcal{D})$.

"⇐"Consider a tuple $t \in q^{ret_L(\mathcal{I},\mathcal{D})}$; we show that t is in $q^{\mathcal{B}}$ for *any* database $\mathcal{B} \in sem_s(\mathcal{I}, \mathcal{D})$. The existence of t witnesses the existence of a query homomorphism $\varkappa$ from the atoms of q to facts of $ret_L(\mathcal{I}, \mathcal{D})$, that sends the distinguished variables of q to the constants of t. Consider a generic database $\mathcal{B} \in sem_s(\mathcal{I}, \mathcal{D})$; by applying Lemma 4.31, we know that there exists a homomorphism μ from $ret_L(\mathcal{I}, \mathcal{D})$ to $\mathcal{B}$. The composition $\varkappa \circ \mu$ is a homomorphism from the atoms of q to $\mathcal{B}$, that preserves the image of the distinguished variables, and witnesses that $t \in ans_s(q, \mathcal{I}, \mathcal{D})$. □

Hence, in the case in which no dependency is specified on the global schema query answering in LAV is analogous to query answering over a single relational schema, modulo the computation of $ret_L(\mathcal{I}, \mathcal{D})$. Since such computation is polynomial both in data and combined complexity it easily turns out that computational complexity in LAV without constraints is in PTIME in data complexity and NP-complete in combined complexity [1].

The above Theorem can be easily extended to the case in which IDs are specified over the global schema.

Theorem 4.33 *Let $\mathcal{I} = \langle \mathcal{G}, \mathcal{S}, \mathcal{M} \rangle$ be a LAV data integration system, with $\mathcal{G} = \langle \Psi, \Sigma_I \rangle$; let $\mathcal{D}$ be a source database for $\mathcal{I}$, and q a query over $\mathcal{G}$ of arity n. Moreover, let t be an n-tuple of constants of $\mathcal{U}_\mathcal{D}$ Then, we have that $t \in ans_s(q, \mathcal{I}, \mathcal{D})$ iff $t \in q^{chase(\mathcal{G}, ret_L(\mathcal{I}, \mathcal{D}))}$.*

Proof.
"⇒"This follows from the fact that $chase(\mathcal{G}, ret_L(\mathcal{I}, \mathcal{D}))$ belongs to $sem_s(\mathcal{I}, \mathcal{D})$. Indeed, from Lemma 4.1 it follows that $chase(\mathcal{G}, ret_L(\mathcal{I}, \mathcal{D}))$ satisfies Σ_I and hence it is consistent with $\mathcal{G}$, whereas satisfaction of the mapping $\mathcal{M}$ can be easily seen from the fact that $ret_L(\mathcal{I}, \mathcal{D})$ satisfies the mapping, and adding new facts to $ret_L(\mathcal{I}, \mathcal{D})$ cannot make any mapping assertion to be violated.

"⇐"Consider a tuple $t \in q^{chase(\mathcal{G}, ret_L(\mathcal{I}, \mathcal{D}))}$; we show that t is in $q^\mathcal{B}$ for *any* database $\mathcal{B} \in sem_s(\mathcal{I}, \mathcal{D})$. Analogously to Theorem 4.32, the existence of t witnesses the existence of a query homomorphism $\varkappa$ from the atoms of q to facts of $chase(\mathcal{G}, ret_L(\mathcal{I}, \mathcal{D}))$, that sends the distinguished variables of q to the constants of t. Then, as in Theorem 4.2, it can be proved by induction on the structure of $chase(\mathcal{G}, ret_L(\mathcal{I}, \mathcal{D}))$ that, for any database instance $\mathcal{B} \in sem_s(\mathcal{G}, ret_L(\mathcal{I}, \mathcal{D}))$, there exists a homomorphism μ that sends the tuples of $chase(\mathcal{G}, ret_L(\mathcal{I}, \mathcal{D}))$ to the tuples of $\mathcal{B}$. It is also easy to see that $sem_s(\mathcal{G}, ret_L(\mathcal{I}, \mathcal{D})) = sem_s(\mathcal{I}, \mathcal{D})$. Indeed every $\mathcal{B} \in sem_s(\mathcal{G}, ret_L(\mathcal{I}, \mathcal{D})$ is consistent with $\mathcal{G}$ and contains $ret_L(\mathcal{I}, \mathcal{D})$, modulo renaming the fresh constants introduced in the construction of $ret_L(\mathcal{I}, \mathcal{D})$. Hence, each such $\mathcal{B}$ satisfies the mapping, since it contains necessarily the minimum set of facts that guarantee $s^\mathcal{B} = V_s^\mathcal{D}$, thus $sem_s(\mathcal{G}, ret_L(\mathcal{I}, \mathcal{D})) \subseteq sem_s(\mathcal{I}, \mathcal{D})$. In a similar way we can prove that $sem_s(\mathcal{I}, \mathcal{D}) \subseteq sem_s(\mathcal{G}, ret_L(\mathcal{I}, \mathcal{D}))$. Hence, the composition $\varkappa \circ \mu$ is a homomorphism from the atoms of q to $\mathcal{B} \in sem_s(\mathcal{I}, \mathcal{D})$, that preserves the image of the distinguished variables, and witnesses that $t \in ans_s(q, \mathcal{I}, \mathcal{D})$. □

Obviously, the above Theorem provides an actual procedure to answer a query over a LAV system in the presence of IDs on the global schema. Also in this case, the computational complexity is the same as in the framework of a single relational schema, i.e., it is in PTIME in data complexity and PSPACE-complete in combined complexity (see Section 4.4).

It should be easy to see that the above Theorem also holds in the case in which IDs and EDs are specified on the global schema, provided that $ret_L(\mathcal{I}, \mathcal{D})$ is consistent with the closure of EDs w.r.t. logical implication of IDs and EDs. Intuitively, this holds since the introduction of fresh constants in $ret_L(\mathcal{I}, \mathcal{D})$ cannot lead to the violation of exclusion dependencies (constants are never equal), and that ED violations can be caused only by the data stored at the sources.

Theorem 4.34 *Let $\mathcal{I} = \langle \mathcal{G}, \mathcal{S}, \mathcal{M} \rangle$ be a LAV data integration system, with $\mathcal{G} = \langle \Psi, \Sigma_I, \Sigma_E \rangle$; let $\mathcal{D}$ be a source database for $\mathcal{I}$, and q a query of arity n over $\mathcal{G}$. Moreover, let t be an n-tuple of constants of $\mathcal{U}_\mathcal{D}$ Then, we have that $t \in ans_s(q, \mathcal{I}, \mathcal{D})$ iff $ret_L(\mathcal{I}, \mathcal{D})$ is consistent with Σ_E^* and $t \in q^{chase(\mathcal{G}, ret_L(\mathcal{I}, \mathcal{D}))}$.*

Proof. Proceed as in Theorem 4.33 taking into account Lemma 4.4. □

From the above theorem it follows that query answering is in PTIME in data complexity and PSPACE-complete in combined complexity.

Now, we are able to answer our initial question and show that the analogous of Theorem 4.29 holds for LAV systems. In the following, we show that indeed a separation theorem also holds for NKC LAV systems, which in turn allows to establish decidability of query answering in such systems.

To this aim, we define the notion of *retrieved global database* for a LAV system $\mathcal{I} = \langle \mathcal{G}, \mathcal{S}, \mathcal{M} \rangle$, where $\mathcal{G} = \langle \Psi, \Sigma_I, \Sigma_K, \Sigma_E \rangle$. Such a database, denoted as $ret_{LK}(\mathcal{I}, \mathcal{D})$, is obtained starting from the database $ret_L(\mathcal{I}, \mathcal{D})$ formalized in Definition 4.30, by repeatedly applying the following rule that unifies fresh constants (the new constant symbols introduced in the construction of $ret_L(\mathcal{I}, \mathcal{D})$):

KD-RULE: if the KD $key(r) = \mathbf{A}$ is in Σ_K and there are two facts $r(t_1), r(t_2)$ in $ret_{LK}(\mathcal{I}, \mathcal{D})$ such that $t_1[\mathbf{A}] = t_2[\mathbf{A}]$, then, for each non-key attribute X of r: if $t_1[X]$ is a fresh constant z_i , then replace each occurrence of z_i with $t_2[X]$ in $ret_{LK}(\mathcal{I}, \mathcal{D})$; otherwise, if $t_2[X]$ is a fresh constant z_j, then replace each occurrence of z_j with $t_1[X]$ in $ret_{LK}(\mathcal{I}, \mathcal{D})$.

Informally, in the presence of key dependencies, the retrieved global database $ret_L(\mathcal{I}, \mathcal{D})$ of Definition 4.30 is not a good representative of the data that, through the mapping, the source data in $\mathcal{D}$ force to be present in the global relations, since, due the presence of key dependencies, some of the fresh constants introduced in the construction process of $ret_L(\mathcal{I}, \mathcal{D})$ could be made equal. Instead, by applying the KD-rule, we obtain a correct representative of such data, which are common to all databases in $sem_s(\mathcal{I}, \mathcal{D})$.

It is immediate to verify that $ret_{LK}(\mathcal{I}, \mathcal{D})$ can be computed in time polynomial in the size of $\mathcal{D}$ and of $\mathcal{I}$.

Analogously to the GAV case, when $ret_{LK}(\mathcal{I}, \mathcal{D})$ is consistent with $\Sigma_K \cup \Sigma_E^*$, we are able to separately deal with Σ_I and $\Sigma_K \cup \Sigma_E$.

Theorem 4.35 *Let $\mathcal{I} = \langle \mathcal{G}, \mathcal{S}, \mathcal{M} \rangle$ be a NKC LAV system, where $\mathcal{G} = \langle \Psi, \Sigma_I, \Sigma_K, \Sigma_E \rangle$, let $\mathcal{I}' = \langle \mathcal{G}', \mathcal{S}, \mathcal{M} \rangle$, where $\mathcal{G}' = \langle \Psi, \Sigma_I \rangle$, be the system obtained by $\mathcal{I}$ by eliminating the KDs and the EDs of $\mathcal{G}$; let $\mathcal{D}$ be a source database for $\mathcal{I}$ and $\mathcal{I}'$. Moreover, let q be a query of arity n over $\mathcal{G}$ and $\mathcal{G}'$, and t an n-tuple of constants of $\mathcal{U}_{\mathcal{D}}$. We have that $t \notin ans_s(q, \mathcal{I}, \mathcal{D})$ iff $ret_{LK}(\mathcal{I}, \mathcal{D})$ is consistent with $\Sigma_K \cup \Sigma_E^*$ and $t \notin ans_s(q, \mathcal{I}', \mathcal{D})$.*

Proof. As in the proof of Theorem 4.33, we can show that $sem_s(\mathcal{I}, \mathcal{D}) = sem_s(\mathcal{G}, ret_{LK}(\mathcal{I}, \mathcal{D}))$. Hence, the thesis follows from Theorem 4.13. $\square$

Therefore, the above theorem implies decidability of query answering in NKC LAV systems.

5 Conclusions

In this document, we have studied techniques, methods and complexity of query answering in the INFOMIX framework. In particular, we have recalled undecidability results provided in Deliverable D3.1 [15], and have defined algorithms for query answering under different semantics for different combinations of IDs, KDs and EDs on the global schema. By combining the hardness results shown in Deliverable D3.3 [16] with the membership results provided by the query answering algorithms of the present document, we have established the computational complexity of the query answering problem in the IIM. The table in Figure 1 of Section 4.4 summarizes the complexity results for query answering over inconsistent and incomplete databases provided in this document. Query answering algorithms developed in the context of a single relational schema have been then extended to GAV and LAV data integration frameworks in Section 4.5.

To the best of our knowledge, our studies on query answering over incomplete and inconsistent data are the first to provide a thorough analysis of data integration in the presence of KDs, EDs, and cyclic IDs.

References

[1] Serge Abiteboul, Richard Hull, and Victor Vianu. *Foundations of Databases*. Addison Wesley Publ. Co., Reading, Massachussetts, 1995.

[2] Marcelo Arenas, Leopoldo E. Bertossi, and Jan Chomicki. Consistent query answers in inconsistent databases. In *Proceedings of the Eighteenth ACM SIGACT SIGMOD SIGART Symposium on Principles of Database Systems (PODS'99)*, pages 68–79, 1999.

[3] Andrea Calì, Diego Calvanese, Giuseppe De Giacomo, and Maurizio Lenzerini. Data integration under integrity constraints. *Information Systems*, 2003. To appear.

[4] Andrea Calì, Domenico Lembo, and Riccardo Rosati. On the decidability and complexity of query answering over inconsistent and incomplete databases. In *Proceedings of the Twentysecond ACM SIGACT SIGMOD SIGART Symposium on Principles of Database Systems (PODS 2003)*, pages 260–271, 2003.

[5] Andrea Calì, Domenico Lembo, and Riccardo Rosati. Query rewriting and answering under constraints in data integration systems. In *Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence (IJCAI 2003)*, 2003. To appear.

[6] M. A. Casanova, R. Fagin, and C. H. Papadimitriou. Inclusion dependencies and their interaction with functional dependencies. *Journal of Computer and System Sciences*, 28(1):29–59, 1984.

[7] Jan Chomicki and Jerzy Marcinkowski. On the computational complexity of consistent query answers. Technical Report `arXiv:cs.DB/0204010v1`, 2002.

[8] Evgeny Dantsin, Thomas Eiter, Georg Gottlob, and Andrei Voronkov. Complexity and expressive power of logic programming. *ACM Computing Surveys*, 33(3):374–425, 2001.

[9] Ronald Fagin, Jeffrey D. Ullman, and Moshe Y. Vardi. On the semantics of updates in databases. In *Proceedings of the Second ACM SIGACT SIGMOD Symposium on Principles of Database Systems (PODS'83)*, pages 352–365, 1983.

[10] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W.H. Freeman and Company, San Francisco, Ca, 1979.

[11] M. Gelfond and V. Lifschitz. The stable model semantics for logic programming. In *Proceedings of the Fifth Logic Programming Symposium*, pages 1070–1080. The MIT Press, 1988.

[12] Gianluigi Greco, Sergio Greco, and Ester Zumpano. A logic programming approach to the integration, repairing and querying of inconsistent databases. In *Proceedings of the Seventeenth International Conference on Logic Programming (ICLP'01)*, volume 2237 of *Lecture Notes in Artificial Intelligence*, pages 348–364. Springer, 2001.

[13] David S. Johnson and Anthony C. Klug. Testing containment of conjunctive queries under functional and inclusion dependencies. *Journal of Computer and System Sciences*, 28(1):167–189, 1984.

[14] Domenico Lembo, Maurizio Lenzerini, and Riccardo Rosati. Functional specification of the INFOMIX system. Technical Report D2.1, Infomix Consortium, September 2002.

[15] Domenico Lembo, Maurizio Lenzerini, and Riccardo Rosati. Formalism for IIM specification. Technical Report D3.1, Infomix Consortium, April 2003.

[16] Domenico Lembo, Maurizio Lenzerini, and Riccardo Rosati. Techniques for handling inconsistency and incompleteness in the IIM. Technical Report D3.3, Infomix Consortium, September 2003.

[17] Maurizio Lenzerini. Data integration: A theoretical perspective. In *Proceedings of the Twentyfirst ACM SIGACT SIGMOD SIGART Symposium on Principles of Database Systems (PODS 2002)*, pages 233–246, 2002.

[18] Jinxin Lin and Alberto O. Mendelzon. Merging databases under constraints. *International Journal of Cooperative Information Systems*, 7(1):55–76, 1998.

[19] John C. Mitchell. The implication problem for functional and inclusion dependencies. *Information and Control*, 56:154–173, 1983.

[20] Christos H. Papadimitriou. *Computational Complexity*. Addison Wesley Publ. Co., Reading, Massachussetts, 1994.

[21] W. J. Savitch. Relationship between nondeterministic and deterministic tape complexities. *Journal of Computer and System Sciences*, 4(2):177–192, 1970.

[22] Ron van der Meyden. Logical approaches to incomplete information. In Jan Chomicki and Günter Saake, editors, *Logics for Databases and Information Systems*, pages 307–356. Kluwer Academic Publisher, 1998.

[23] Moshe Y. Vardi. The complexity of relational query languages. In *Proceedings of the Fourteenth ACM SIGACT Symposium on Theory of Computing (STOC'82)*, pages 137–146, 1982.