Project Number:	IST-2001-33570
Project Acronym:	INFOMIX
Project Full Name:	Infomix: Boosting Information Integration
Deliverable Number:	D3.3
Title:	**Techniques for handling inconsistency and incompleteness in the IIM**
Workpackage:	WP3
Document Type:	Deliverable
Distribution:	Infomix Consortium
Status:	final
Document file:	WP4_T3_D3.3_0200.ps
Version:	2.0
Number of pages:	35
Due Date:	September 30, 2003
Delivery Date:	October 10, 2003
Partners contributed:	UNIROME1
Partners owning:	UNIROME1
Man Months:	10

Short Description:
In this document, we deal with the integration of incomplete and inconsistent data in the Information Integration Model (IIM). More specifically, we study query answering in the IIM and investigate the interaction among different kinds of constraints that can be specified on the global schema of a data integration system. We first identify cases in which query answering is undecidable. Then, we study the computational complexity of the problem for the decidable cases, and provide complexity lower bounds for different combinations of dependencies.

Document Change Record		
Version	**Date**	**Reason for Change**
v.1.0	September 22, 2003	First draft
	October 5, 2003	Review from the project partners
v.2.0	October 10, 2003	Final version

Contents

1 Introduction

In this document, we analyze the problem of dealing with inconsistency and incompleteness in the Information Integration Model (IIM). More specifically, we study the problem of query answering in the presence of incomplete and inconsistent data, according to the formalism adopted for the specification of the IIM. Such a formalism has to be expressive enough to allow for the specification of very rich forms of knowledge about the information to be integrated. On the other hand, a thorough study of the decidability and the complexity of the subsequent information integration activities is needed in order to verify the impact of the adopted choices.

As described in our previous deliverable D3.1 [25], 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. In our framework, $\mathcal{M}$ can express both the approaches for specifying the mapping commonly adopted in the theory and practice of data integration [26]: 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.

In principle, the formalism adopted for modelling the global schema allows the system designer to specify complex dependencies between global relations, by making use of powerful forms of integrity constraints expressed on $\mathcal{G}$. Furthermore, different semantics assumptions on the mapping enable for the specification of different way of interpreting the data at the sources with respect to data satisfying the corresponding portion of the global schema. Notably, such assumptions allow for properly modelling data incompleteness, i.e., situations in which data sources do not provide all information needed to answer user queries. Furthermore, they enable for specifying that retrievable data are complete but not all sound, or that they are both sound and complete. On the other hand, in general, 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.

Despite the rich expressiveness of the language used to specify the relational representation of the global schema $\mathcal{G}$, the interaction between the classes of constraints allowed on $\mathcal{G}$ makes query answering difficult under the different semantics considered, and may lead to non-decidable cases [25]. Hence, it is in general necessary to impose some restrictions on the simultaneous presence of different kinds of constraints on the global schema in order to obtain correct answers to user queries. Furthermore, the computational complexity of the decidable cases has to be established.

In this document, we study query answering under the different semantics, and investigate the interaction among different kinds of constraints allowed on the global schema, for a given language for the queries on the global schema, and a language for the views in the mapping. More specifically, 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, and we fix user queries to be the union of conjunctive queries, and views in the mapping to be non-recursive Datalog queries for GAV systems and conjunctive queries for LAV systems.

We identify cases in which query answering is undecidable for both strict and loose semantics. For the decidable cases, we study computational complexity of the query answering problem, and provide the lower bounds for different combinations of dependencies. We point out that the query answering problem in the different settings addressed is actually complete for the complexity classes mentioned in this report. However, establishing the memberships in these classes is inherently related to the problem of providing algorithms for query answering, which will be the subject of our forthcoming deliverable D5.2.

The present document is organized as follows. In Section 2, we present some theoretical background, i.e., we provide the basic notions of the relational model and of the computational complexity theory. In Section 3 we describe our formal framework for information integration which underlies the Information Integration Model (IIM). Section 4 provides an in-depth investigation on the decidability and complexity of query answering in our framework. For the sake of clarity, we first abstract from the integration context, and address the problem in a framework of a single relational database (Section 4.1). Decidability and complexity results obtained in this setting are then extended to data integration in Section 4.3.3. Finally, Section 5 concludes the paper.

We point out that the results provided in this document, together with the results in our next deliverable D5.1, extend to the presence of exclusion dependencies on the global schema our previous studies on inclusion dependencies and key dependencies recently published in [8], and partially described in our previous deliverable D3.1.

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 [19, 1, 31].

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 dependencies $r : \mathbf{A} \rightarrow B$ 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 the form $key(r) = \mathbf{A}$, 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 $key(r) = \mathbf{A}$ 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 $t_1[\mathbf{A}] = t_2[\mathbf{B}]$.

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 $\mathcal{RS} =$

[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 [11, 30] 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.

$\langle \Psi, \Sigma_I, \Sigma_K, \Sigma_E, \Sigma_F \rangle$ 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 [31, 18]. 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 [32] 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 [35]:

- 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 Formal framework for information integration

Informally, a data integration system consists of a (virtual) global schema, which specifies the global (user) elements, a source schema, which describes the structure of the sources in the system, and a mapping, which specifies the relationship between the sources and the global schema. User queries are posed on the global schema, and the system provides

[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

the answers to such queries by exploiting the information supplied by the mapping and accessing the sources that contain relevant data. Thus, from the syntactic viewpoint, the specification of an integration system depends on the following parameters:

- The form of the global schema, i.e., the formalism used for expressing global elements and relationships between global elements. Several settings have been considered in the literature, where, for instance, the global schema can be relational [20], object-oriented [4], semi-structured [29], based on Description Logics [23, 10], etc.;

- The form of the source schema, i.e., the formalism used for expressing data at the sources and relationships between such data. In principle, formalisms commonly adopted for the source schema are the same mentioned for the global schema;

- The form of the mapping. Two basic approach have been proposed in the literature, called respectively *global-as-view* (GAV) and *local-as-view* (LAV) [27, 33, 26] . The GAV approach requires that the global schema is defined in terms of the data sources: more precisely, every element of the global schema is associated with a view, i.e., a query, over the sources, so that its meaning is specified in terms of the data residing at the sources. Conversely, in the LAV approach, the meaning of the sources is specified in terms of the elements of the global schema: more exactly, the mapping between the sources and the global schema is provided in terms of a set of views over the global schema, one for each source element;

- The language of the mapping, i.e., the query language used to express views in the mapping;

- The language of the user queries, i.e., the query language adopted by users to issue queries on the global schema.

Let us now turn our attention on the semantics. According to [26], the semantics of a data integration system is given in terms of instances of the elements of the global schema (e.g., one set of tuples for each global relation if the global schema is relational, one set of objects for each global class if it is object-oriented, etc.). Such instances have to satisfy *(i)* the integrity constraints expressed between elements of the global schema, and *(ii)* the mapping specified between the global and the source schema.

Roughly speaking, the notion of satisfying the mapping depends on how the data that can be retrieved from the sources are interpreted with respect to the data that satisfy the corresponding portion of the global schema. Three different assumptions have been 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.

In the following we formally define the data integration framework underlying the INFOMIX IIM. We consider here a relational setting, i.e., the global and the source schema are expressed in the relational model. However, our framework can be easily generalized to different data models.

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., $\mathcal{S} = \langle \Psi_{\mathcal{S}}, \emptyset \rangle$. Dealing with only relational sources is not restrictive, since we can always assume that suitable wrappers present sources in the relational format. Furthermore, assuming that no integrity constraint is specified on $\mathcal{S}$ is equivalent to assuming that data satisfy constraints expressed on the sources in which they are stored. This is a common assumption in data integration, because sources are in general autonomous and external to the integration system, and satisfaction of constraints at the sources should be guaranteed by local data management systems;
- $\mathcal{M}$ 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 on $\mathcal{M}$, and denote it with $as(\mathcal{M})$, and pose $as(\mathcal{M}) = s$, c, or e, for the sound, complete, and exact assumption, respectively.

Based on $\mathcal{D}$ and $as(\mathcal{M})$, we now specify which is the information content of the global schema $\mathcal{G}$. 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) $r_{\mathcal{S}}^{\mathcal{D}} \supseteq q_{\mathcal{G}}^{\mathcal{D}}$ if $as(\mathcal{M}) = c$;
  - (c) $r_{\mathcal{S}}^{\mathcal{B}} = q_{\mathcal{G}}^{\mathcal{D}}$ 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 [34].

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 $\mathcal{I}_G = \langle \mathcal{G}_0, \mathcal{M}_G, \mathcal{S}_G\rangle$, where $\mathcal{S}_G$ contains three binary relations s_1, s_2 and s_3, and the mapping $\mathcal{M}_G$ is as follows:

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

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 query answering under the exact semantics is meaningless, since every possible fact is a logical consequence

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

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 $\mathcal{I}_L = \langle \mathcal{G}_0, \mathcal{M}_L, \mathcal{S}_L \rangle$, where $\mathcal{G}_0$ is the same of 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 of 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*(a, b) and *player*(a, d) that are not consistent with the key dependency, and it is not possible to make $\mathcal{B}$ satisfying such constraint by adding other facts to $\mathcal{B}$. On the other hand, each such $\mathcal{B}$ should contain also *team*(b, c) and *player*(e, f), where *team*(b, c) is consistent with the dependencies in the schema, whereas the inconsistency caused by *player*(e, f) can be resolved under the sound semantics by adding a suitable fact to $\mathcal{B}$ of the form *player*(f, α), 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 [15, 28, 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, $\mathcal{R}_1 <_{\mathcal{D}} \mathcal{R}_2$ stands for $\mathcal{R}_1 \leq_{\mathcal{D}} \mathcal{R}_2 \wedge \mathcal{R}_2 \not\leq_{\mathcal{D}} \mathcal{R}_1$.

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 (as illustrated in Section 3.2) $sem_c(\mathcal{I}, \mathcal{D}) \neq \emptyset$ for each $\mathcal{I}$ and for each $\mathcal{D}$, 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 $ans_{lc}(q, \mathcal{I}_0, \mathcal{D}_0) = \emptyset$. ■

Our loose semantics follows the principles that underlie the notion of *database repair* introduced by previous works in the area of inconsistent databases that have proposed several semantics relying either on set-containment or cardinality-based ordering [15, 14, 7, 28, 2, 3]. However, such studies basically apply to a single database setting [15, 7, 2, 3], and to GAV integration systems under the exact assumption on the mapping [28, 14, 21]. Conversely, our approach considers a pure data integration framework, and our semantics seriously takes into account the interpretation of the mapping. Nonetheless, with this consideration in place, we can say that our loosely-sound semantics is similar in the spirit to the semantics described in [15], whereas the loosely-exact semantics has been thoroughly studied in [21, 2, 3][5].

More recently, data inconsistency in a LAV data integration setting has been studied in [5] and [6]. The semantics proposed in [5] and [6] turns out to be different from the loose semantics for LAV of Section 3.3. Indeed, whereas our proposal focuses on the mapping and define a suitable relaxation of it in the presence of inconsistency, [5, 6] characterize the semantics in terms of the repairs of the different global databases that can be obtained populating the global schema according to the LAV mapping. More specifically, [5, 6] assume that the sources are sound, and consider the set $min(\mathcal{G})$ of the minimal (w.r.t. set inclusion) global databases that satisfy the mapping. Then, the repairs are the global databases consistent with the constraints that are minimal w.r.t. $\leq_{\mathcal{DB}}$ for some $\mathcal{DB} \in min(\mathcal{G})$, where $\mathcal{B} \leq_{\mathcal{DB}} \mathcal{B}'$ if $\triangle(\mathcal{B}, \mathcal{DB}) \subseteq \triangle(\mathcal{B}', \mathcal{DB})$, where in turn $\triangle(X, Y)$ indicates the symmetric difference between X and Y.

Notice that, in this semantics, even if the sources are assumed to be sound, the repairs are computed as if the retrieved data were exact. Under such assumption, the repair semantics can be different from the sound semantics even when the latter is not empty.In other words, one resorts to repair the system even when the system actually does not need to be repaired. This is shown in the following example.

Example 3.2 Consider a simple LAV system $\mathcal{I}$ in which there are two global relations $g_1/2$ and $g_2/2$, an inclusion dependency $g_1[1,2] \subseteq g_2[1,2]$, a source relation $s/2$, and a mapping assertion $s(x, y) \leftarrow g_1(x, y)$. Then, let $\mathcal{D} = \{s(\mathsf{a}, \mathsf{b})\}$ be a source database for $\mathcal{I}$. In this case we have $min(\mathcal{G}) = \{g_1(\mathsf{a}, \mathsf{b})\}$. This unique minimal global database is inconsistent w.r.t. the ID expressed on the global schema. According to [5, 6], two repairs have to be considered in this situation $\mathcal{R} = \emptyset$ and $\mathcal{R}' = \{g_1(\mathsf{a}, \mathsf{b}), g_2(\mathsf{a}, \mathsf{b})\}$. Hence, every query over the global schema has an empty consistent answer. ∎

Notice that, in our framework, when we assume the sources to be sound, we are able to solve the inconsistency in the above example without actually relaxing the mapping, since in this case $sem_s(\mathcal{I}, \mathcal{D}) = sem_{ls}(\mathcal{I}, \mathcal{D})$. Furthermore, given a simple query $q(X, Y) \leftarrow g_1(X, Y)$ we have $ans_s(q, \mathcal{I}, \mathcal{D}) = \{\langle \mathsf{a}, \mathsf{b} \rangle\}$, rather than the empty set.

We point out that, in all mentioned works, an in-depth investigation of the role of inclusion dependencies expressed on the database schema is actually missing, and the methods proposed to get consistent answers from inconsistent databases basically apply only to the class of "universally quantified constraints". Two exceptions are [21], where a preliminary study of this setting is presented, and [6], where, however, the problem is tackled by taking into account only a particular class of finite global databases, viz. the global repairs that can be computed by using only constants appearing in the source database.

Conversely, in the following we provide a thoroughly study of query answering under the different semantics, when inclusion, key and exclusion dependencies are expressed on the global schema.

[5] A more detailed description of the works on inconsistent databases can be found in our previous project deliverable D1.1.

4 Decidability and Complexity of Query Answering

In this section we focus on the relationship between the expressiveness of the global schema and the decidability and complexity of query answering. The interaction between the classes of constraints allowed on the global schema, namely, inclusion, key and exclusion dependencies, makes query answering difficult under the different semantics described, and may lead to non-decidable cases. Hence, it is in general necessary to impose some restrictions on the simultaneous presence of different kinds of constraints on the global schema in order to obtain correct answers to user queries.

For the sake of simplicity, we study here query answering in the setting of a single database. Such a setting allow 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. Undecidability and complexity results obtained in this setting immediately extended to GAV and LAV data integration systems.

Briefly, in the following:

(i) we identify cases in which query answering is undecidable for both strict and loose semantics;

(ii) for the decidable cases, we study computational complexity of the query answering problem, and provide the lower bounds for different combinations of dependencies.

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 [22], while the results for the loose semantics extend known results in the field of inconsistent databases [21, 13], 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 $sem_{l_s}(\mathcal{RS}, \mathcal{D})$, $sem_{l_c}(\mathcal{RS}, \mathcal{D})$ and $sem_{l_e}(\mathcal{RS}, \mathcal{D})$ 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, $\mathcal{R}_1 <_{\mathcal{D}} \mathcal{R}_2$ stands for $\mathcal{R}_1 \leq_{\mathcal{D}} \mathcal{R}_2 \wedge \mathcal{R}_2 \not\leq_{\mathcal{D}} \mathcal{R}_1$.

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 investigate the decidability and complexity of 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, we point out 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 strictly-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 strictly-sound semantics.

Inclusion and key dependencies: an undecidable case. Let us now study the remarkable case in which IDs and KDs are expressed over a relational schema, i.e., consider the schema $\mathcal{RS} = \langle \Psi, \Sigma \rangle$, where $\Sigma = \Sigma_I \cup \Sigma_K$. We now show that query answering in such a setting is not decidable in all cases.

We start by defining a restricted class of dependencies under which query answering is decidable.

Definition 4.1 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 $key(r_2) = \mathbf{A_2}$.

We now show that the class of NKC dependencies plays a crucial role for decidability of query answering. More precisely, we show that, as soon as we extend the class of dependencies beyond the non-key-conflicting case, query answering is undecidable. In particular, we introduce, together with KDs, inclusion dependencies of the form $r_1[\mathbf{A}_1] \subseteq r_2[\mathbf{A}_2]$ such that, 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. We will call such IDs *1-key-conflicting IDs* (1KCIDs) with respect to Σ_K. A *1-key-conflicting* (1KC) relational schema is defined analogously to a NKC schema. We first show undecidability of implication of KDs and 1KCIDs.

Theorem 4.2 *The problem of implication*[6] *for KDs and 1KCIDs is undecidable.*

Proof. The proof is by reduction from the more general problem of implication of functional dependencies and inclusion dependencies. Consider a generic instance of this problem, i.e., given a relational schema $\mathcal{RS} = \langle \Psi, \Sigma_I, \Sigma_F \rangle$, where Σ_I is a set of IDs and Σ_F is a set of FDs, and an inclusion dependency δ. We assume that all FDs in Σ_F are in *normal form*, i.e. of the form $r : \mathbf{A} \rightarrow B$, where a single attribute B is in the right-hand side. We construct an *ad hoc* problem of implication of KDs and 1KCIDs, consisting of a relational schema $\mathcal{RS}_1 = \langle \Psi_1, \Sigma_{I_1}, \Sigma_{K_1} \rangle$, where Σ_{I_1} is a set of 1KCID with respect to Σ_{K_1}, and the same dependency δ. We will show that the two problems are equivalent, i.e. $(\Sigma_I \cup \Sigma_F) \models \delta$ if and only if $(\Sigma_{I_1} \cup \Sigma_{K_1}) \models \delta$. The dependencies Σ_{I_1} and Σ_{K_1}, defined in a new relational schema $\mathcal{RS}_1 = \langle \Psi_1, \Sigma_{I_1}, \Sigma_{K_1} \rangle$, are constructed as follows.

- The new set of relations Ψ_1 includes all relations in Ψ (plus those added as below).
- Σ_{I_1} includes all IDs in Σ_I (plus those added as below).
- For each FD φ in Σ_F of the form

  $$r : \mathbf{A} \rightarrow B,$$

  we add to Ψ_1 an auxiliary relation r_φ of arity $|\mathbf{A}| + 1$, and we add to Σ_{I_1} the dependencies

  $$\begin{array}{rrcl} \gamma_1 : & r_\varphi[\mathbf{A}, B] & \subseteq & r[\mathbf{A}, B] \\ \gamma_2 : & r[\mathbf{A}, B] & \subseteq & r_\varphi[\mathbf{A}, B], \end{array}$$

  and to Σ_{K_1} key dependency

  $$\varkappa : \quad key(r_\varphi) = \mathbf{A}.$$

[6]For the details about implication of database dependencies, we refer the reader to [1].

Note that all the IDs in Σ_{I_1} are 1KCIDs with respect to Σ_{K_1}, since keys in Σ_{K_1} are only those expressed on the auxiliary relations r_φ, and the only IDs on r_φ are of the form γ_2. The following result, will be used in the rest of the proof.

Lemma 4.3 *For any database $\mathcal{B}_1$ for $\mathcal{RS}_1$, we have that $\mathcal{B}_1$ satisfies $\{\varphi, \gamma_1, \gamma_2\}$ if and only if $\mathcal{B}_1$ satisfies $\{\varkappa, \gamma_1, \gamma_2\}$.*

Proof.

"$\Rightarrow$"By hypothesis, for any two tuples $t_1, t_2 \in r^{\mathcal{B}_1}$, it cannot happen that $t_1[\mathbf{A}] = t_2[\mathbf{A}]$ and $t_1[B] = t_2[B]$. Since by dependencies γ_1, γ_2 we have $r^{\mathcal{B}}[\mathbf{A}, B] = r_\varphi^{\mathcal{B}}[\mathbf{A}, B]$, we deduce that the same property holds for all tuples in $r_\varphi^{\mathcal{B}_1}$; therefore $\varkappa$ is satisfied by definition.

"$\Leftarrow$"By hypothesis, for any two tuples $t_1, t_2 \in r_\varphi^{\mathcal{B}_1}$, it cannot happen that $t_1[\mathbf{A}] = t_2[\mathbf{A}]$ and $t_1[B] = t_2[B]$. Since by dependencies γ_1, γ_2 we have $r^{\mathcal{B}_1}[\mathbf{A}, B] = r_\varphi^{\mathcal{B}_1}[\mathbf{A}, B]$, we deduce that the same property holds for all tuples in $r^{\mathcal{B}_1}$; therefore φ is satisfied by definition. □

From this result it follows that we are able to simulate general FDs by using KDs and 1KCIDs only. Now we end the reduction by showing that $(\Sigma_I \cup \Sigma_F) \models \delta$ if and only if $(\Sigma_{I_1} \cup \Sigma_{K_1}) \models \delta$.

"$\Rightarrow$"By contradiction, suppose $(\Sigma_{I_1} \cup \Sigma_{K_1}) \not\models \delta$; then there exists a database $\mathcal{B}_1$ for $\mathcal{RS}_1$ such that $\mathcal{B}_1$ satisfies $(\Sigma_{I_1} \cup \Sigma_{K_1})$ and violates δ. Consider a database $\mathcal{B}$ for $\mathcal{RS}$ obtained from $\mathcal{B}_1$ by removing the facts associated with the relations of the form r_φ introduced in the reduction. By Lemma 4.3, $\mathcal{B}$ satisfies $(\Sigma_I \cup \Sigma_F)$; moreover, $\mathcal{B}$ cannot satisfy δ because $\mathcal{B}$ coincides with $\mathcal{B}_1$ on the relations in Ψ.

"$\Leftarrow$"By contradiction, suppose $(\Sigma_I \cup \Sigma_F) \not\models \delta$; then there exists a database $\mathcal{B}$ for $\mathcal{RS}$ such that $\mathcal{B}$ satisfies $(\Sigma_I \cup \Sigma_F)$ and violates δ. We construct a database $\mathcal{B}_1$ for $\mathcal{RS}_1$ that coincides with $\mathcal{B}$ on the relations in Ψ, and such that the facts associated with the relations of the form r_φ, introduced in the reduction, are such that the dependencies of the form γ_1, γ_2 are satisfied. By Lemma 4.3, $\mathcal{B}_1$ satisfies $(\Sigma_{I_1} \cup \Sigma_{K_1})$; moreover, it cannot satisfy δ because $\mathcal{B}_1$ coincides with $\mathcal{B}$ on the relations in Ψ.

The reduction is clearly computable in a finite amount of time. Since implication of IDs and FDs is undecidable, the thesis follows. □

A similar result on undecidability of the implication problem for multiple key dependencies and foreign key dependencies has been established in [17, 16]. The connection between the two problems can be intuitively seen from the fact that a foreign key constraint that presents in its right-hand side a key $\mathbf{K}_1$, may also be a key conflicting inclusion dependency with respect to another key $\mathbf{K}_2$ of the same relation, if $\mathbf{K}_2 \subset \mathbf{K}_1$. However, our result (that has been developed independently of the one in [17, 16]) is slightly more general since it proves undecidability of implication for 1KCIDs.

We now show that query answering is undecidable in the presence of KDs and 1KCIDs.

Theorem 4.4 *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 $\mathcal{U}$. The problem of establishing whether $t \in ans_s(q, \mathcal{RS}, \mathcal{D})$ is undecidable.*

Proof. The proof is analogous to a proof of PSPACE-hardness of an analogous result (addressed in the context of query containment) proved by Vardi and published in [22]. We will show a counterexample in which the problem is undecidable. Let δ be the following inclusion dependency:

$$r[A_1, \ldots, A_k] \subseteq s[B_1, \ldots, B_k]$$

where r has arity n and s has arity m. Without loss of generality, δ involves the first k attributes of r and s respectively. We choose a database instance $\mathcal{D}$ for $\mathcal{RS}$ containing the single fact $r(c_1, \ldots, c_n)$. Then we consider the following boolean query:

$$q \leftarrow r(X_1, \ldots, X_n), s(X_1, \ldots, X_k, Y_{k+1}, \ldots, Y_m)$$

Note that the query q has a positive answer (i.e., $\langle\rangle \in ans_s(q, \mathcal{RS}, \mathcal{D})$) if and only if the fact $s(c_1, \ldots, c_k, d_{k+1}, \ldots, d_m)$ is in all databases in $sem_s(\mathcal{RS}, \mathcal{D})$. It is immediate to see that this is true if and only if $(\Sigma_I \cup \Sigma_K) \models \delta$. Since implication of 1KCIDs and KDs is undecidable, the thesis follows. $\square$

As an immediate consequence of this theorem, undecidability of query answering in the presence of KDs and general IDs follows. Moreover, the problem is still undecidable if we restrict to the class of instances consistent with the key dependencies.

Corollary 4.5 *Let $\mathcal{RS} = \langle \Psi, \Sigma_I, \Sigma_K \rangle$ be a 1KC schema, $\mathcal{D}$ a database instance for $\mathcal{RS}$ consistent with Σ_K, q a query of arity n over $\mathcal{RS}$, and t an n-tuple of constants of $\mathcal{U}$. The problem of establishing whether $t \in ans_s(q, \mathcal{RS}, \mathcal{D})$ is undecidable.*

Proof. The case where $\mathcal{D}$ does not satisfy Σ_K is clearly decidable, since in that case the answer to the problem is always affirmative. The claim follows immediately. $\square$

Obviously, undecidability holds even in the presence of also EDs. In the following we use the terms NKC and 1KC schemas even if also EDs are issued over the global schema.

Complexity lower bounds in the strictly-sound semantics As already said, in the cases in which only EDs and KDs are specified on the relational schema, query answering basically reduces to classical query processing in relational databases, whose computational complexity is in PTIME w.r.t. data complexity and NP-complete (and a fortiori NP-hard) in combined complexity [1]. More involved is the case of NKC relational schemas. From the results in [22], it follows immediately that the problem in the case of IDs alone is in PTIME in data complexity, and PSPACE-complete (hence PSPACE-hard) in combined complexity. Of course, since this case is a particular case of NKCIDs, KDs and EDs (when no KD is defined, any ID is non-key-conflicting), we can easily prove the following theorem:

Theorem 4.6 *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 $\mathcal{U}$. The problem of establishing whether $t \in ans_s(q, \mathcal{RS}, \mathcal{D})$ is PSPACE-hard with respect to combined complexity.*

Obviously, PSPACE-hardness holds also for the case of only EDs and IDs, and for the case of only KDs and NKCIDs.

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 $\mathcal{D}$.

Undecidability of 1KC schemas We now show undecidability of query answering for 1-key-conflicting relational schemas under the loosely-sound semantics.

Theorem 4.7 *Let $\mathcal{RS} = \langle \Psi, \Sigma_I, \Sigma_K \rangle$ 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 $\mathcal{U}$. The problem of establishing whether $t \in ans_{ls}(q, \mathcal{RS}, \mathcal{D})$ is undecidable.*

Proof. Consider the case in which $\mathcal{D}$ is consistent with Σ_K. Undecidability in this case follows from Corollary 4.5 since, in the case in which $\mathcal{D}$ is consistent with Σ_K, $t \in ans_{ls}(q, \mathcal{RS}, \mathcal{D})$ iff $t \in ans_s(q, \mathcal{RS}, \mathcal{D})$. □

Obviously, the result holds in the presence of EDs, i.e., when $\mathcal{RS} = \langle \Psi, \Sigma_I, \Sigma_K, \Sigma_E \rangle$.

Complexity lower bounds in the loosely-sound semantics We first point out that in the presence of only IDs, the loosely-sound coincides with the strictly-sound semantics. Hence hardness results established for the strictly-sound semantics hold also in the case (in particular, for the results in [22], query answering under IDs is PSPACE-hard in combined complexity).

On the other hand, the complexity characterization of the query answering problem turns out to be different in the presence of KDs or EDs, as showed by the following theorems:

Theorem 4.8 *Let $\mathcal{RS} = \langle \Psi, \Sigma_K \rangle$ 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 $\mathcal{U}$. The problem of establishing whether $t \in ans_{ls}(q, \mathcal{RS}, \mathcal{D})$ is coNP-hard with respect to data complexity.*

Proof. We prove coNP-hardness by reducing the 3-colorability problem to the complement of our problem. Consider a graph $G = \langle V, E \rangle$ with a set of vertices V and edges E. We define a relational schema $\mathcal{RS} = \langle \Psi, \Sigma_K \rangle$ where Ψ consists of the two binary relations *edge* and *col*, and Σ_K contains the dependency $key(col) = \{1\}$. The instance $\mathcal{D}$ is defined as follows:

$$\begin{aligned} \mathcal{D} \;=\; & \{col(n,i) | i \in \{1,2,3\} \text{ and } n \in V\} \cup \\ & \{edge(x,y) | \langle x,y \rangle \in E\} \end{aligned}$$

Finally, we define the query

$$q \leftarrow edge(X,Y), col(X,Z), col(Y,Z)$$

We prove that G is 3-colorable (i.e., for each pair of adjacent vertices, the vertices are associated with different colors) if and only if $\langle\rangle \notin ans_{ls}(q, \mathcal{RS}, \mathcal{D})$ (i.e., the boolean query q has a negative answer)[7]. In fact, it is immediate to verify that, for each possible coloring C of the graph (i.e., a set of pairs of vertices and colors, where the three colors are represented by the values 1,2,3) there exists $\mathcal{B} \in sem_{ls}(\mathcal{RS}, \mathcal{D})$ that exactly corresponds to C, i.e., $col^{\mathcal{B}}$ is exactly the set of pairs in the coloring C. Therefore, if there exists a coloring that is a 3-coloring, then $\langle\rangle \notin q^{\mathcal{B}}$ for some $\mathcal{B} \in sem_{ls}(\mathcal{RS}, \mathcal{D})$, consequently $\langle\rangle \notin ans_{ls}(q, \mathcal{RS}, \mathcal{D})$. Conversely, it is immediate to verify that, for each $\mathcal{B} \in sem_{ls}(\mathcal{RS}, \mathcal{D})$, $col^{\mathcal{B} \cap \mathcal{D}}$ corresponds to a possible coloring of the graph. Hence, if each possible coloring is not a 3-coloring, then $\langle\rangle \in q^{\mathcal{B}}$, therefore $\langle\rangle \in ans_{ls}(q, \mathcal{RS}, \mathcal{D})$. □

Let us now consider EDs: in this case a result analogous to the previous one holds.

Theorem 4.9 *Let $\mathcal{RS} = \langle \Psi, \Sigma_E \rangle$ 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 $\mathcal{U}$. The problem of establishing whether $t \in ans_{ls}(q, \mathcal{RS}, \mathcal{D})$ is coNP-hard with respect to data complexity.*

Proof. In order to show coNP-hardness, also in this case we reduce the 3-colorability problem to our problem. More precisely, we construct a reduction of the 3-colorability problem to the complement of the problem of query answering under the loosely-sound semantics in the presence of EDs only. Consider a graph $G = \langle V, E \rangle$ with a set of vertices V and edges E. We define a relational schema $\mathcal{RS} = \langle \Psi, \Sigma_E \rangle$ where Ψ consists of the binary relation *edge* and the three unary relations *col1*, *col2* and *col3*. Then, Σ_E contains the dependencies

$$\begin{aligned} col1[1] \;\cap\; & col2[1] = \emptyset; \\ col1[1] \;\cap\; & col3[1] = \emptyset; \\ col2[1] \;\cap\; & col3[1] = \emptyset. \end{aligned}$$

The instance $\mathcal{D}$ is defined as follows:

$$\begin{aligned} \mathcal{D} \;=\; & \{col1(n), col2(n), col3(n) | n \in V\} \cup \\ & \{edge(x,y) | \langle x,y \rangle \in E\} \end{aligned}$$

[7] $\langle\rangle$ represents the empty tuple. According to a common notation in relational databases [1], given a boolean query q, $\langle\rangle \in q^{\mathcal{B}}$ iff q evaluates to true over a database instance $\mathcal{B}$.

Finally, we define the query

$$
\begin{array}{rcl}
q & \leftarrow & edge(X,Y), col1(X), col1(Y) \ \vee \\
 & & edge(X,Y), col2(X), col2(Y) \ \vee \\
 & & edge(X,Y), col3(X), col3(Y).
\end{array}
$$

It is easy to see that, also in this case, G is 3-colorable if and only if $\langle\rangle \notin ans_{ls}(q, \mathcal{RS}, \mathcal{D})$. Indeed, for each possible coloring C of the graph there exists $\mathcal{B} \in sem_{ls}(\mathcal{RS}, \mathcal{D})$ that exactly corresponds to C, i.e., $col1^{\mathcal{B}}$, $col2^{\mathcal{B}}$ and $col3^{\mathcal{B}}$ store exactly the set of nodes associated in the coloring C with color 1, 2 and 3, respectively. Therefore, if there exists a coloring that is a 3-coloring, then $\langle\rangle \notin q^{\mathcal{B}}$ for some $\mathcal{B} \in sem_{ls}(\mathcal{RS}, \mathcal{D})$, consequently $\langle\rangle \notin ans_{ls}(q, \mathcal{RS}, \mathcal{D})$. Conversely, it is immediate to verify that, for each $\mathcal{B} \in sem_{ls}(\mathcal{RS}, \mathcal{D})$, $col1^{\mathcal{B}\cap\mathcal{D}}$, $col2^{\mathcal{B}\cap\mathcal{D}}$ and $col3^{\mathcal{B}\cap\mathcal{D}}$ correspond to a possible coloring of the graph. Hence, if each possible coloring is not a 3-coloring, then $\langle\rangle \in q^{\mathcal{B}}$, therefore $\langle\rangle \in ans_{ls}(q, \mathcal{RS}, \mathcal{D})$. □

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

Theorem 4.10 *Let $\mathcal{RS} = \langle\Psi, \Sigma\rangle$ be a relational schema, where either $\Sigma = \Sigma_K$ or $\Sigma = \Sigma_E$. 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 $\mathcal{U}$. The problem of establishing whether $t \in ans_{ls}(q, \mathcal{RS}, \mathcal{D})$ is Π^p_2-hard with respect to combined complexity.*

Proof. Consider first the case in which $\Sigma = \Sigma_K$. We prove hardness with respect to Π^p_2, by reducing to our problem the 2-QBF validity problem, i.e., the validity problem for quantified boolean formulae.

Let g be the 2-QBF formula $\forall x_1 \ldots x_n \exists y_1 \ldots y_m f(x_1, \ldots, x_n, y_1, \ldots, y_m)$ where f is a 3-CNF, i.e., a propositional formula of the form

$$(l_{1,1} \vee l_{1,2} \vee l_{1,3}) \wedge \ldots \wedge (l_{k,1} \vee l_{k,2} \vee l_{k,3}). \tag{1}$$

where each $l_{i,j}$ is a literal of the form a or $\overline{a}$, with $a \in \{x_1, \ldots, x_n, y_1, \ldots, y_m\}$.

We construct a relational schema $\mathcal{RS} = \langle\Psi, \Sigma_K\rangle$ as follows:

- Ψ consists of the relations (with associated arities): $x_i/2$ for $1 \leq i \leq n$, $y_i/1$ for $1 \leq i \leq m$, and $c_i/3$ for $1 \leq i \leq k$;
- Σ_K consists of the key dependencies $key(x_i) = \{1\}$ for $1 \leq i \leq n$.

Moreover, we build an instance $\mathcal{D}$ for $\mathcal{RS}$, containing the following facts:

1. for each i such that $1 \leq i \leq n$, the facts $x_i(c_0, 0), x_i(c_0, 1)$;
2. for each i such that $1 \leq i \leq m$, the facts $y_i(0), y_i(1)$;
3. for each i such that $1 \leq i \leq k$ the facts of the form $c_i(b_{i,1}, b_{i,2}, b_{i,3})$ such that $b_{i,j} \in \{0, 1\}$ for $1 \leq j \leq 3$, and when applying the substitution $\sigma\{l_{i,1} \to b_{i,1}, l_{i,2} \to b_{i,2}, l_{i,3} \to b_{i,3}\}$ to f, the conjunct $l_{i,1} \vee l_{i,2} \vee l_{i,3}$ evaluates to 1, i.e., to *true*.

Finally, let q be the query

$$q \leftarrow x_1(C_1, X_1), \dots x_n(C_n, X_n), y_1(Y_1) \dots y_m(Y_m), c_1(Z_{1,1}, Z_{1,2}, Z_{1,3}), \dots c_k(Z_{k,1}, Z_{k,2}, Z_{k,3})$$

where either

(i) $Z_{i,j} = X_r$ if $l_{i,j} = X_r$ or $l_{i,j} = \overline{X}_r$ for $r \in \{1, \dots n\}$, or

(ii) $Z_{i,j} = Y_r$ if $l_{i,j} = Y_r$ or $l_{i,j} = \overline{Y}_r$ for $r \in \{1, \dots m\}$.

We prove that g is valid if and only if $\langle\rangle \in ans_{ls}(q, \mathcal{RS}, \mathcal{D})$.

First, suppose that g is valid. Then, for each propositional evaluation μ of $x_1, \dots, x_n$ there exists a propositional evaluation μ' of $y_1, \dots, y_m$ such that the interpretation of f with respect to μ and μ' is true. Let $\mathcal{B}_\mu$ be the database obtained from $\mathcal{D}$ by removing each fact of the form $x_i(c_0, 1)$ if $\mu(x_i) = \mathit{false}$ and $x_i(c_0, 0)$ if $\mu(x_i) = \mathit{true}$. It is immediate to see that $\mathcal{B}_\mu \in sem_{ls}(\mathcal{RS}, \mathcal{D})$. Now, since the interpretation of f with respect to μ and μ' is true, it follows that each conjunct $l_{i,1} \vee l_{i,2} \vee l_{i,2}$ in f evaluates to true. Then, according to the construction of $\mathcal{D}$ (point 3), for $1 \leq i \leq n$, $\mathcal{B}_\mu$ contains the facts $c_i(b_{i,1}, b_{i,2}, b_{i,3})$, where $b_{i,j} = 1$ if $\nu(l_{i,j}) = \mathit{true}$, or $b_{i,j} = 0$ if $\nu(l_{i,j}) = \mathit{false}$, where $\nu = \mu \cup \mu'$. It should be easy to see that the query q evaluates to true over $\mathcal{B}_\mu$, i.e., $\langle\rangle \in q^{\mathcal{B}_\mu}$. Consider now the set $\{\mu_1, \dots, \mu_h\}$ of possible propositional evaluations of $x_1, \dots, x_n$ in g, then each $\mathcal{B} \in sem_{ls}(\mathcal{RS}, \mathcal{D})$ is such that $\mathcal{B} \supseteq \mathcal{B}_{\mu_i}$, for some $i \in \{1, \dots, h\}$. It follows that $\langle\rangle \in ans_{ls}(q, \mathcal{RS}, \mathcal{D})$.

Suppose now that g is not valid. Therefore, there exists an evaluation μ of $x_1, \dots, x_n$ such that for each evaluation μ' of $y_1, \dots, y_m$, the interpretation of f with respect to μ and μ' is false. Let $\mathcal{B}$ be the database obtained from $\mathcal{D}$ by removing each fact of the form $x_i(c_0, 1)$ if $\mu(x_i) = \mathit{false}$ and $x_i(c_0, 0)$ if $\mu(x_i) = \mathit{true}$ It is immediate to see that $\mathcal{B} \in sem_{ls}(\mathcal{RS}, \mathcal{D})$, since it corresponds to a maximal subset of $\mathcal{D}$ consistent with Σ_K. Since g is not valid, then at least one conjunct in f is not satisfied for each evaluation μ' of $y_1, \dots, y_m$. Hence, in the evaluation of the query q over $\mathcal{B}$, at least one atom of the form $c_i(Z_{i,1}, Z_{i,2}, Z_{i,3})$ does not unify with the facts in $\mathcal{B}$. Therefore, $\langle\rangle \notin q^{\mathcal{B}}$, which implies that $\langle\rangle \notin ans_{le}(q, \mathcal{RS}, \mathcal{D})$.

Consider now the case in which $\Sigma = \Sigma_E$. To prove Π^p_2-hardness, we use a reduction from 2-QBF validity problem similar to the one described above. Indeed, in this case we construct a relational schema $\mathcal{RS} = \langle \Psi, \Sigma_E \rangle$ as follows:

- Ψ is as in the above reduction;

- Σ_E consists of the exclusion dependencies $x_i[1] \cap x_i[2] = \emptyset$ for $1 \leq i \leq n$.

The instance $\mathcal{D}$ for $\mathcal{RS}$, contains the following facts:

1. for each i such that $1 \leq i \leq n$, the facts $x_i(1, 0), x_i(0, 1)$;

2. the same facts indicated at point 2 and 3 in the above reduction.

Finally, the query q is the same as above.

Proceeding as in the previous reduction, we can analogously prove that g is valid if and only if $\langle\rangle \in ans_{ls}(q, \mathcal{RS}, \mathcal{D})$.

□

Finally, we can straightforwardly generalize the above complexity results to the general framework.

Theorem 4.11 *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 $\mathcal{U}$. The problem of establishing whether $t \in ans_{ls}(q, \mathcal{RS}, \mathcal{D})$ is coNP-hard in data complexity and PSPACE-hard with respect to combined complexity.*

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".

Undecidability of 1KC schemas We first prove that query answering under the loosely-exact semantics is undecidable when no restriction is imposed on the form of IDs and KDs.

Theorem 4.12 *Let $\mathcal{RS} = \langle \Psi, \Sigma_I, \Sigma_K \rangle$ 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 $\mathcal{U}$. The problem of establishing whether $t \in ans_{le}(q, \mathcal{RS}, \mathcal{D})$ is undecidable.*

Proof. We reduce query answering in the loosely-sound semantics to query answering in the loosely-exact semantics. We can consider instances $\mathcal{D}$ consistent with Σ_K, since for this class of instances the problem of establishing whether $t \in ans_{ls}(q, \mathcal{RS}, \mathcal{D})$ is undecidable (see proof of Theorem 4.7). Starting from such a problem instance $(\mathcal{RS}, \mathcal{D}, q, t)$, we define a new problem instance $(\mathcal{RS}', \mathcal{D}', q', t')$ such that $t \in ans_{ls}(q, \mathcal{RS}, \mathcal{D})$ iff $t' \in ans_{le}(q', \mathcal{RS}', \mathcal{D}')$. Precisely:

- $\mathcal{RS}' = \langle \Psi', \Sigma'_I, \Sigma'_K \rangle$ is obtained from $\mathcal{RS}$ by:
  - defining Ψ' as the schema obtained from Ψ by adding an attribute to each relation in Ψ (in the last position);
  - changing each inclusion in order to propagate such a new attribute from r to s, i.e., Σ'_I is obtained from Σ_I by replacing each $I = r[i_1, \ldots, i_k] \subseteq s[j_1, \ldots, j_k]$ with $I' = r[i_1, \ldots, i_k, n] \subseteq s[j_1, \ldots, j_k, m]$, where n is the arity of r in Ψ' and m is the arity of s in Ψ';
- $\mathcal{D}'$ is the set $\mathcal{D}'_1 \cup \mathcal{D}'_2$, where $\mathcal{D}'_1 = \{\ r(\overline{u}, t_0) | r(\overline{u}) \in \mathcal{D}\ \}$ and

  $$\mathcal{D}'_2 = \{\ r(\overline{u}, t_1) \mid r \in \Psi \text{ and } \overline{u} \text{ is a tuple of values of } \mathcal{U}_{\mathcal{D}} \cup \{t_1\}\ \}$$

  where $\mathcal{U}_{\mathcal{D}}$ denotes the set of symbols from $\mathcal{U}$ appearing in $\mathcal{D}$, and t_0, t_1 are values not belonging to $\mathcal{U}_{\mathcal{D}}$. Notice that the set $\mathcal{D}'$ is finite;

- if the query q has the form

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

  the query q' is as follows:

$$q'(\vec{\mathbf{x}}, Y) \leftarrow conj_1(\vec{\mathbf{x}}, \vec{\mathbf{y}}_1, t_0) \vee \cdots \vee conj_k(\vec{\mathbf{x}}, \vec{\mathbf{y}}_k, t_0) \vee body'$$

  where $body'$ is the disjunction

$$\bigvee \{ r(\overline{u}, t_1) \ \mid \ r(\overline{u}) \in \mathcal{D} \text{ and there is a KD for } r \text{ in } \Sigma_K \}$$

- t' is obtained from t by adding the value t_0 at the end of the tuple t.

It can be shown that $t \in ans_{ls}(q, \mathcal{RS}, \mathcal{D})$ iff $t' \in ans_{le}(q', \mathcal{RS}', \mathcal{D}')$, since for each database $\mathcal{B}$ in $sem_{le}(\mathcal{RS}, \mathcal{D})$, there are two possible cases:

1. $\mathcal{B} \cap \mathcal{D} = \mathcal{D}$. In this case, due to the key dependencies Σ_K, $\mathcal{B}$ does not contain any tuple of the form $r(\overline{u}, t_1)$ such that $r(\overline{u}) \in \mathcal{D}$ and a key dependency for r is defined in Σ_K. Consequently, $t' \in q'^{\mathcal{B}}$ iff $t \in q^{\mathcal{B}}$. Moreover, it is immediate to verify that there exists at least one such $\mathcal{B}$ in $sem_{le}(\mathcal{RS}', \mathcal{D}')$;

2. $\mathcal{B} \cap \mathcal{D} \subset \mathcal{D}$. In this case, there exists at least one tuple in $\mathcal{B}$ of the form $r(\overline{u}, t_1)$ such that $r(\overline{u}) \in \mathcal{D}$ and a key dependency for r is defined in Σ_K, consequently $t' \in q'^{\mathcal{B}}$ for each such $\mathcal{B}$. In other words, this kind of databases does not affect $ans_{le}(q', \mathcal{RS}', \mathcal{D}')$, since in $\mathcal{B}$ every possible tuple is in the answer of q'.

Therefore, $t \in ans_{ls}(q, \mathcal{RS}, \mathcal{D})$ iff $t' \in ans_{le}(q', \mathcal{RS}', \mathcal{D}')$.

Finally, since the above reduction is effectively computable and since, by Theorem 4.7, establishing whether $t \in ans_{ls}(q, \mathcal{RS}, \mathcal{D})$ is undecidable, the thesis follows. □

As usual, undecidability holds also in the presence of EDs, i.e., when $\mathcal{RS} = \langle \Psi, \Sigma_I, \Sigma_K, \Sigma_E \rangle$.

Complexity lower bounds in the loosely-exact semantics We first point out that in the presence of only IDs, query answering under the loosely-exact semantics can be carried out by simply eliminating all the tuples that do not satisfy the IDs, and then evaluating the query on the obtained database. Computational complexity in this case is the complexity of standard query processing over a single database [1].

In the case of only EDs or KDs, the loosely-exact semantics coincides with the loosely-sound semantics, hence the computational characterization is as described in Section 4.3.1, for both data and combined complexity.

More complex, under the loosely-exact semantics, is the decidable case in which IDs interact with KDs, i.e., the case of NKC schemas.

Theorem 4.13 *Let $\mathcal{RS} = \langle \Psi, \Sigma_I, \Sigma_K \rangle$ 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 values of $\mathcal{U}$. The problem of establishing whether $t \in ans_{le}(q, \mathcal{RS}, \mathcal{D})$ is Π^p_2-hard with respect to data complexity and PSPACE-hard with respect to combined complexity.*

Proof. Hardness with respect to Π_2^p can be proved by a reduction from 2-QBF validity. To this aim, we consider here an instance of our problem in which no ED is specified on the relational schema $\mathcal{RS}$. Let g be the 2-QBF formula $\forall x_1 \ldots x_n \exists y_1 \ldots y_m f(x_1, \ldots, x_n, y_1, \ldots, y_m)$ where f is a 3-CNF, i.e., a propositional formula of the form $c_1 \wedge \ldots \wedge c_k$, in which each conjunct c_i is of the form $l_1 \vee l_2 \vee l_3$ and each l_i is a literal of the form a or $\overline{a}$, with $a \in \{x_1, \ldots, x_n, y_1, \ldots, y_m\}$. We construct a relational schema $\mathcal{RS} = \langle \Psi, \Sigma_I, \Sigma_K \rangle$ as follows:

1. Ψ consists of four relations (with associated arities): $r_1/3, r_2/2, r_3/2, s/1$;
2. Σ_I is constituted of four unary IDs:

$$\begin{array}{rcll} r_2[2] & \subseteq & r_1[2] & \text{(ID1)} \\ r_1[3] & \subseteq & r_3[1] & \text{(ID2)} \\ r_2[1] & \subseteq & r_3[1] & \text{(ID3)} \\ r_3[2] & \subseteq & r_2[1] & \text{(ID4)} \end{array}$$

3. Σ_K consists of the KD $key(r_1) = 1$.

Moreover, we build an instance $\mathcal{D}$ for $\mathcal{RS}$, containing the following facts:

1. for each i such that $1 \leq i \leq n$, the facts $r_1(x_i, x_i, c_0), r_1(x_i, \overline{x}_i, c_0)$;
2. for each i such that $1 \leq i \leq m$, the facts $r_1(y_i, y_i, c_1), r_1(y_i, \overline{y}_i, c_1)$;
3. for each i such that $1 \leq i \leq k$ and for each a_j occurring in c_i, the fact $r_2(c_i, a_j)$;
4. for each i such that $1 \leq i < k$, the fact $r_3(c_i, c_{i+1})$, plus the fact $r_3(c_k, c_1)$;
5. the facts $r_1(c_0, c_0, c_0), r_2(c_0, c_0), r_3(c_0, c_0)$;
6. the facts $s(c_i)$ for each i such that $1 \leq i \leq k$.

Finally, let q be the query $q \leftarrow s(X), r_3(X, Y)$. We prove that g is valid if and only if $\langle\rangle \in ans_{le}(q, \mathcal{RS}, \mathcal{D})$.

First, suppose that g is valid. Then, for each propositional evaluation μ of $x_1, \ldots, x_n$ there exists a propositional evaluation μ' of $y_1, \ldots, y_m$ such that the interpretation of f with respect to μ and μ' is true. Let $\mathcal{B}_\mu$ be the database obtained from $\mathcal{D}$ by:

(i) removing each fact of the form $r_1(x_i, x_i, c_0)$ if $\mu(x_i) = \mathit{false}$ and $r_1(x_i, \overline{x}_i, c_0)$ if $\mu(x_i) = \mathit{true}$;

(ii) removing each fact of the form $r_1(y_i, y_i, c_1)$ if $\mu'(y_i) = \mathit{false}$ and $r_1(y_i, \overline{y}_i, c_1)$ if $\mu'(y_i) = \mathit{true}$;

(iii) removing each fact of the form $r_2(c_i, x_j)$ (respectively, $r_2(c_i, \overline{x}_j)$) if $\mu(x_j) = \mathit{false}$ (respectively, $\mu(x_j) = \mathit{true}$) and removing all facts of the form $r_2(c_i, y_j)$ (respectively, $r_2(c_i, \overline{y}_j)$) if $\mu'(y_j) = \mathit{false}$ (respectively, $\mu'(y_j) = \mathit{true}$).

Now, since the interpretation of f with respect to μ and μ' is true, it follows that for each i such that $1 \leq i \leq k$ there is a fact of the form $r_2(c_i, V)$ in $\mathcal{B}_\mu$ (from now on the symbols V, V' represent generic values), which implies that $\mathcal{B}_\mu$ satisfies all the IDs in Σ_I; moreover, it is immediate to see that $\mathcal{B}_\mu$ is consistent with Σ_K. Then consider any $\mathcal{B}'$ such that $\mathcal{B}' \in sem_{le}(\mathcal{RS}, \mathcal{D})$. It can be immediately verified that, due to Σ_K and Σ_I, $\mathcal{B}'$ must be such that there exists an evaluation μ of $x_1, \ldots, x_n$ such that the set of facts in $\mathcal{B}'$ of the form $r_1(x_i, V, V')$ coincides with the set of facts in $\mathcal{B}_\mu$ of the form $r_1(x_i, V, V')$. Now suppose that in $\mathcal{B}'$ there is no fact of the form $r_3(c_i, V)$ such that $1 \leq i \leq k$. Due to (ID3) and to the fact that $\mathcal{B}'$ is consistent with Σ_I, it follows that in $\mathcal{B}'$ there is no fact of the form $r_2(c_i, V)$ such that $1 \leq i \leq k$. This in turn implies that, due to (ID4), in $\mathcal{B}'$ there is no fact of the form $r_1(V, V', c_1)$, consequently no fact in $\mathcal{D}$ of the form $r_1(y_i, V, V')$ is in $\mathcal{B}'$. Therefore, $\mathcal{B}' \cap \mathcal{D} \subset \mathcal{B}_\mu \cap \mathcal{D}$, and since $\mathcal{B}_\mu - \mathcal{D} = \emptyset$ and $\mathcal{B}_\mu$ is consistent with Σ_I and Σ_K, it follows that $\mathcal{B}' \notin sem_{le}(\mathcal{RS}, \mathcal{D})$, thus contradicting the hypothesis. Hence, for each $\mathcal{B}'$ such that $\mathcal{B}' \in sem_{le}(\mathcal{RS}, \mathcal{D})$, there is at least one fact of the form $r_3(c_i, V)$ such that $1 \leq i \leq k$, and since each such $\mathcal{B}'$ contains the fact $s(c_i)$ for each i such that $1 \leq i \leq k$, it follows that $\langle\rangle \in ans_{le}(\mathcal{RS}, \mathcal{D}, q)$.

Suppose now that g is not valid. Therefore, there exists an evaluation μ of $x_1, \ldots, x_n$ such that for each evaluation μ' of $y_1, \ldots, y_m$, the interpretation of f with respect to μ and μ' is false. Let $\mathcal{B}$ be the database obtained from $\mathcal{D}$ by:

(i) removing each fact of the form $r_1(x_i, x_i, c_0)$ if $\mu(x_i) = \mathit{false}$ and $r_1(x_i, \overline{x}_i, c_0)$ if $\mu(x_i) = \mathit{true}$;

(ii) removing all facts of the form $r_1(y_i, V, V')$;

(iii) removing all facts of the form $r_2(V, V')$ and each fact of the form $r_3(c_i, V)$ such that $1 \leq i \leq k$.

It is immediate to see that $\mathcal{B} \in sem_{le}(\mathcal{RS}, \mathcal{D})$, since it corresponds to a maximal subset of $\mathcal{D}$ consistent with Σ_I and Σ_K. Hence, in $\mathcal{B}$ there is no fact of the form $r_3(c_i, V)$ such that $1 \leq i \leq k$. Therefore, $\langle\rangle \notin q^{\mathcal{B}}$, which implies that $\langle\rangle \notin ans_{le}(\mathcal{RS}, \mathcal{D}, q)$.

As concerns combined complexity, PSPACE-hardness can be proved by reducing query answering under loosely-sound semantics for databases without key dependencies to this problem. The reduction is obtained by a slight modification of the reduction from query answering under loosely-sound semantics exhibited in the proof of Theorem 4.12, and observing that, if the original problem instance is such that, for each $I = r[\vec{\mathbf{A}}] \subseteq s[\vec{\mathbf{B}}] \in \Sigma_I$, $\vec{\mathbf{B}}$ does not cover the set of all the attributes of s, then the derived database schema $\mathcal{RS}'$ is a NKC schema. Moreover, it is immediate to verify that restricting to such a kind of problem instances does not affect PSPACE-hardness of the query answering problem under the loosely-sound semantics. Finally, the reduction is modified in a way such that the database instance $\mathcal{D}'$ obtained from the original instance $\mathcal{D}$ has size polynomial w.r.t. combined complexity. □

The following theorem completes the picture on the complexity lower bounds of the query answering problem under the loosely-exact semantics.

Theorem 4.14 *Let $\mathcal{RS} = \langle \Psi, \Sigma_I, \Sigma_E \rangle$ 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 values of $\mathcal{U}$. The problem of establishing whether $t \in ans_{le}(q, \mathcal{RS}, \mathcal{D})$ is Π_2^p-hard with respect to data complexity and PSPACE-hard with respect to combined complexity.*

Proof.

We prove the claim by a reduction from the problem of query answering under the loosely-exact semantics in the presence of IDs and KDs. According to the proof of Theorem 4.13, it is sufficient to consider instances of this problem in which no more than two tuples violate key dependencies on the schema.

Consider a relational schema $\mathcal{RS} = \langle \Psi, \Sigma_I, \Sigma_K \rangle$, a query q of arity n over $\mathcal{RS}$, an n-tuple of constants of the domain $\mathcal{U}$, and a database $\mathcal{D}$ for $\mathcal{RS}$. Then, we construct a relational schema $\mathcal{RS}' = \langle \Psi', \Sigma_I, \Sigma_E \rangle$ such that:

1. Ψ' comprises a relation $r'/n + 2$ for each $r/n \in \Psi$;
2. Σ_E comprises an exclusion dependency $r'[n+1] \cap r'[n+2] = 0$ for each $r' \in \Psi'$.

Moreover, we construct a database $\mathcal{D}'$ for $\mathcal{RS}$ such that:

- for each pair of facts $r(t_1, \ldots, t_n), r(c_1, \ldots, c_n) \in \mathcal{D}$ that violate the key dependency on r, we add to $\mathcal{D}'$ two facts of the form $r(t_1, \ldots, t_n, \alpha_1, \alpha_2), r(c_1, \ldots, c_n, \alpha_2, \alpha_1)$, where α_1 and α_2 are new fresh constants of the domain $\mathcal{U}$ that have not been previously introduced in the construction of $\mathcal{D}'$;
- for each fact $r(t_1, \ldots, t_n) \in \mathcal{D}$ such that there does not exist another fact in $\mathcal{D}$ with the same key, we add to $\mathcal{D}'$ a fact of the form $r(t_1, \ldots, t_n, \alpha_1, \alpha_2)$, where α_1 and α_2 are new fresh constants of the domain $\mathcal{U}$.

Finally, we construct a query q' having the same head of q, and whose body is obtained by replacing each predicate $r(X_1, \ldots X_n)$ in q with $r'(X_1, \ldots X_n, Y, Z)$, where Y and Z are new fresh variables that occur only once in the query.

Now, the above construction of the database $\mathcal{D}'$ implies that, for each pair of tuples of $\mathcal{D}$ that violates a key dependency in $\mathcal{RS}$, there exists a pair of tuples in $\mathcal{D}'$ that violates an exclusion dependency in $\mathcal{D}'$; moreover, these are the only EDs violations in $\mathcal{D}'$. Therefore, it follows that $t \in ans_{le}(q, \mathcal{RS}, \mathcal{D})$ iff $t \in ans_{le}(q', \mathcal{RS}', \mathcal{D}')$, thus proving the claim. □

4.3.3 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 showed 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 $sem_{as(\mathcal{D}')}(\mathcal{RS}, \mathcal{D}) = sem_{as(\mathcal{M})}(\mathcal{I}, \mathcal{D})$[8], 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 4.15 *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.*

Obviously, 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.

The table in Figure 1 summarizes the decidability and complexity results provided in this section. 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 of 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 hard with respect to the class reported (with the exception of the cases in which the complexity is PTIME).

5 Conclusions

In this document we have addressed query answering in the IIM in the presence of incomplete and inconsistent data. In particular, we have studied decidability and complexity of the problem under different semantics for different combinations of constraints expressed on the global schema.

A summary of the formalisms for the specification of the IIM studied in this report is presented in table below.

[8]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.

Data/Combined complexity lower bounds for *general* database instances:

EDs	KDs	IDs	strictly-sound	loosely-sound	loosely-exact
no	no	GEN	PTIME/PSPACE	PTIME/PSPACE	PTIME/NP
no	yes	no	PTIME/NP	coNP/Π^p_2	coNP/Π^p_2
yes	no	no	PTIME/NP	coNP/Π^p_2	coNP/Π^p_2
no	yes	NKC	PTIME/PSPACE	coNP/PSPACE	Π^p_2/PSPACE
yes	no	GEN	PTIME/PSPACE	coNP/PSPACE	Π^p_2/PSPACE
no	yes	1KC	undecidable	undecidable	undecidable
yes	yes	GEN	undecidable	undecidable	undecidable

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

Figure 1: Complexity and decidability of query answering under EDs, KDs and IDs

global schema	relational
source schema	relational
integrity constraints	KDs, IDs, EDs
form of the mapping	GAV, LAV
mapping assumption	sound, exact, complete
semantics	strict, loose
GAV mapping views	non-recursive DATALOG
LAV mapping views	Conjunctive Queries

Even if the formalisms adopted in the IIM allow for the specification of complex integration scenarios by using powerful constraints, we have shown that it is necessary to impose some restrictions on the simultaneous presence of different kinds of constraints on the global schema, in order to avoid undecidable cases.

We have also analyzed the complexity of query answering for decidable cases and provided complexity lower bounds for different combinations of dependencies.

Complexity upper bounds for the same combinations of integrity constraints will be given in our next Deliverable D5.2, where sound and complete algorithms for query answering will be provided.

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] Marcelo Arenas, Leopoldo E. Bertossi, and Jan Chomicki. Specifying and querying database repairs using logic programs with exceptions. In *Proceedings of the Fourth International Conference on Flexible Query Answering Systems (FQAS 2000)*, pages 27–41. Springer, 2000.

[4] D. Beneventano, S. Bergamaschi, S. Castano, A. Corni, R. Guidetti, G. Malvezzi, M. Melchiori, and M. Vincini. Information integration: the MOMIS project demonstration. In *Proceedings of the Twentysixth International Conference on Very Large Data Bases (VLDB 2000)*, 2000.

[5] Leopoldo Bertossi, Jan Chomicki, Alvaro Cortes, and C. Gutierrez. Consistent answers from integrated data sources. In *Proc. of the 6th Int. Conf. on Flexible Query Answering Systems (FQAS 2002)*, pages 71–85, 2002.

[6] Loreto Bravo and Leopoldo Bertossi. Logic programming for consistently querying data integration systems. In *Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence (IJCAI 2003)*, 2003. To appear.

[7] François Bry. Query answering in information systems with integrity constraints. In *IFIP WG 11.5 Working Conf. on Integrity and Control in Information System*. Chapman & Hall, 1997.

[8] 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.

[9] 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.

[10] Diego Calvanese, Giuseppe De Giacomo, and Maurizio Lenzerini. Answering queries using views over description logics knowledge bases. In *Proceedings of the Seventeenth National Conference on Artificial Intelligence (AAAI 2000)*, pages 386–391, 2000.

[11] 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.

[12] Ashok K. Chandra and Moshe Y. Vardi. The implication problem for functional and inclusion dependencies is undecidable. *SIAM Journal on Computing*, 14(3):671–677, 1985.

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

[14] Phan Minh Dung. Integrating data from possibly inconsistent databases. In *Proceedings of the Fourth International Conference on Cooperative Information Systems (CoopIS'96)*, pages 58–65, 1996.

[15] 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.

[16] Wenfei Fan and Leonid Libkin. On xml integrity constraints in the presence of dtds. *Journal of the ACM*, 43(3):368–406, 2002.

[17] Wenfei Fan and Jérôme Siméon. Integrity constraints for xml. In *Proceedings of the Nineteenth ACM SIGACT SIGMOD SIGART Symposium on Principles of Database Systems (PODS 2000)*, 2000.

[18] 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.

[19] 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.

[20] Michael R. Genereseth, Arthur M. Keller, and Oliver M. Duschka. Infomaster: An information integration system. In *ACM SIGMOD International Conference on Management of Data*, 1997.

[21] 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.

[22] 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.

[23] Thomas Kirk, Alon Y. Levy, Yehoshua Sagiv, and Divesh Srivastava. The Information Manifold. In *Proceedings of the AAAI 1995 Spring Symp. on Information Gathering from Heterogeneous, Distributed Enviroments*, pages 85–91, 1995.

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

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

[26] 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.

[27] Alon Y. Levy. Logic-based techniques in data integration. In Jack Minker, editor, *Logic Based Artificial Intelligence*. Kluwer Academic Publisher, 2000.

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

[29] Ioana Manolescu, Daniela Florescu, and Donald Kossmann. Answering XML queries on heterogeneous data sources. In *Proceedings of the Twentyseventh International Conference on Very Large Data Bases (VLDB 2001)*, pages 241–250, 2001.

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

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

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

[33] Jeffrey D. Ullman. Information integration using logical views. *Theoretical Computer Science*, 239(2):189–210, 2000.

[34] 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.

[35] 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.