| | |
|---|---|
| Project Number: | IST-2001-33570 |
| Project Acronym: | INFOMIX |
| Project Full Name: | Infomix: Boosting Information Integration |
| | |
| Deliverable Number: | D5.2 |
| Title: | **Methods and techniques for query rewriting** |
| Authors: | Domenico Lembo, Maurizio Lenzerini, Riccardo Rosati |
| Workpackage: | WP5 |
| Document Type: | Deliverable |
| Distribution: | Infomix Consortium |
| Status: | final |
| Document file: | WP5_T2_D5.2_0200.ps |
| Version: | 1.0 |
| Number of pages: | 38 |
| | |
| Due Date: | March 31, 2004 |
| Delivery Date: | April 7, 2004 |
| | |
| Partners contributed: | UNIROME1,TUWIEN |
| Partners owning: | UNIROME1 |
| Man Months: | 9 |

Short Description:
In this document we provide a set of *query rewriting* techniques that solve the problem of query answering within the INFOMIX framework. Query answering by rewriting is divided in two phases, where the first one re-expresses a user query posed over the global schema in terms of the relations at the sources, and the second evaluates the rewriting over the source extensions. Query rewriting depends on several choices adopted when modelling the integration system, e.g., form (LAV vs GAV) and semantics of the mapping, constraints on the global schema, language for user queries and mapping views, etc. In this document we provide sound and complete query rewriting algorithms defined for the different combinations of the parameters that are considered within the INFOMIX framework. We point out that such algorithms allow for an *intensional* processing of queries, where the computation is carried out in a way as much as possible independent of the data residing at the sources. This strongly augments the feasibility of the approach, and allows for interesting optimizations.

| **Document Change Record** | | |
|---|---|---|
| **Version** | **Date** | **Reason for Change** |
| v.1.0 | March 26, 2004 | First draft |
| | April 5, 2004 | Review from the project partners |
| v.2.0 | April 7, 2004 | Final version |

# Contents

**1 Introduction** **4**
1.1 Query rewriting . . . . . 4
1.2 Results . . . . . 5
1.3 Organization of the report . . . . . 6

**2 Theoretical Background** **6**
2.1 Query Languages . . . . . 6
2.2 The Relational Model . . . . . 8

**3 Formal framework** **9**

**4 Query Rewriting in GAV Data Integration Systems** **11**
4.1 Query Rewriting in the presence of IDs . . . . . 12
4.1.1 Preliminary definitions . . . . . 13
4.1.2 The algorithm ID-rewrite . . . . . 14
4.1.3 Soundness and completeness of ID-rewrite . . . . . 16
4.1.4 Data Integration Setting . . . . . 19
4.2 Query Rewriting in the presence of IDs, KDs, and EDs . . . . . 20
4.2.1 Semantics for inconsistent data sources . . . . . 22
4.2.2 Query Rewriting under the Loosely-sound Semantics . . . . . 24
4.3 Discussion . . . . . 26

**5 Query Rewriting in LAV Data Integration Systems** **26**
5.1 Query Rewriting in the presence of IDs and EDs . . . . . 27
5.1.1 Compiling LAV into GAV . . . . . 27
5.1.2 Query Rewriting in the Presence of IDs and EDs . . . . . 32
5.2 Query Rewriting in the Presence of KDs . . . . . 33
5.3 Discussion . . . . . 34

**6 Related Work** **34**

**7 Conclusions** **36**

# 1 Introduction

The query answering problem in data integration is the problem of computing answers to queries posed over the global reconciled schema of an integration system. In this document we provide a set of *query rewriting* techniques that solve the problem of query answering within the INFOMIX framework. Roughly speaking, in query rewriting, query answering is divided in two steps, where the first re-expresses the query in terms of the alphabet of the source relations, and the second evaluates the rewriting over the source extensions. Notice that, in generic query answering, no limits are posed on the way in which a query is processed, and the only goal is to compute the answer to the query by exploiting all possible information, in particular the source extensions. In Deliverable D5.1 [20] we have proposed several algorithms for query answering in both Global-As-View (GAV) and Local-As-View (LAV) data integration systems. In the following we concentrate on query rewriting and provide new and effective algorithms for such a matter.

## 1.1 Query rewriting

In the data integration framework adopted within the INFOMIX project and introduced in Deliverable D3.1 [19], query rewriting corresponds to the following task: given a query $q$ over the global schema of the data integration system, compute a query $q'$ over the source schema such that the semantics of the two queries is the same, i.e., the set of certain answers of $q$ corresponds to the evaluation of $q'$ over the sources. If the query $q'$ satisfies the above property for every instance of the sources (i.e., for every state of the data sources), then $q'$ is called *perfect rewriting* of $q$.

Intuitively, the reformulated query $q'$ encodes all the knowledge of the data integration system which is relevant for answering the query $q$. In particular, the main elements of the system in which such knowledge resides are: (i) the mapping between the global schema and the sources; and (ii) the integrity constraints expressed over the global schema.

More precisely, query rewriting depends on:

- the type of the mapping between the global schema and the source schema (GAV, LAV);
- the semantics of the mapping;
- the kinds of constraints allowed in the global schema;
- the language of the input query;
- the language of the output query.

Therefore, different query rewriting algorithms must be defined for different combinations of the above parameters.

**Remark.** Within the INFOMIX project, query rewriting plays a crucial role. In fact, in the INFOMIX prototype system, we want to base query processing on query rewriting algorithms. This choice is due to the fact that, in principle, query rewriting offers the possibility of exploiting *intensional* processing of queries. This is a very important aspect in data integration applications, in which the access to data sources is typically the main system bottleneck. Thus, the possibility of processing queries in a way as much as possible independent of the data residing at the sources, appears a very promising approach.

We point out that some of the results on query rewriting in GAV data integration systems presented in this document have been published in [8]. We also underline that most of the techniques provided here will be implemented within the INFOMIX prototype system.

## 1.2 Results

The main contributions of the present document can be summarized as follows:

- for GAV systems:
  1. we provide a sound and complete query rewriting technique under the sound semantics, a semantics for data integration systems which properly captures incompleteness of data stored at the sources. We first consider the case in which integrity constraints on the global schema are only inclusion dependencies (IDs), and then address the case in which also, exclusion dependencies (EDs) and key dependencies (KDs) are allowed;
  2. we present a sound and complete query rewriting technique in the presence of the same constraints under the loosely-sound semantics, a suitable relaxation of the sound semantics that allows for consistent query answering when data retrieved from the sources may violate EDs and KDs.
- for LAV systems:
  1. we present a compilation technique which allows for transforming a LAV system $\mathcal{I}$ into a GAV system $\mathcal{I}'$. Such a transformation is query-preserving, i.e., for every union of conjunctive queries (UCQ) $Q$, the answers to $Q$ posed to the system $\mathcal{I}$ are the same as the answers to $Q$ posed to $\mathcal{I}'$;
  2. based on the above compilation technique, we present a sound and complete query rewriting technique for LAV systems under the sound semantics in the presence of IDs and EDs.

These results advance the state of the art in data integration in terms of:

1. new algorithms: to the best of our knowledge, our methods are the first that solve query answering in the presence of EDs, KDs and cyclic IDs, for both incomplete and inconsistent data;

2. *effective* techniques for data integration which exploit intensional processing of queries. Indeed, our algorithms are able to intensionally carry out query processing, since treatment of integrity constraints is essentially done at the query and schema level. This strongly augments the feasibility of the approach, and allows for interesting optimizations, which will be presented in Deliverable D5.3 of the project.

### 1.3 Organization of the report

The present document is organized as follows. In Section 2 we recall some background on query languages and the relational model. In Section 3, we recall the formal framework for data integration adopted by the INFOMIX project and present the sound semantics for data integration systems. In Section 4 we present algorithms for computing the perfect rewriting of a UCQ in GAV integration systems: we first study the case in which only IDs are expressed on the global schema; then, we deal with the simultaneous presence of IDs, KDs and EDs; finally, we provide the loosely-sound semantics for inconsistent data, and present algorithms for computing the perfect rewriting of a UCQ in the presence of IDs, KDs, and EDs under the loosely-sound semantics. In Section 5 we first present our compilation technique for turning a LAV system into a GAV one, then we provide a sound and complete query rewriting technique under the sound semantics, for the case of IDs and EDs, and finally we discuss the problems introduced by allowing KDs in the global schema. Section 7 concludes the document.

## 2 Theoretical Background

In this section we recall some theoretical notions that will be used in this report. More specifically, we illustrate here the classes of queries that we deal with in the following and that are of interest in the INFOMIX context, and provide a brief formal description of the relational model with integrity constraints. For further background on logic programming and relational database theory, we refer the reader to [13, 1].

### 2.1 Query Languages

A *term* is either a variable or a constant. An *atom* is an expression $p(T_1, \ldots, T_n)$, where $p$ is a predicate (relation) of arity $n$ and $T_1, \ldots, T_n$ are terms.

A *Datalog*$^\neg$ rule $\rho$ is an expression of the form

$$r(\vec{\mathbf{u}}) \leftarrow r_1(\vec{\mathbf{u}}_1), \ldots, r_k(\vec{\mathbf{u}}_k),\ not\ r_{k+1}(\vec{\mathbf{u}}_{k+1}), \ldots,\ not\ r_{k+m}(\vec{\mathbf{u}}_{k+m}) \tag{1}$$

where $k, m \geq 0$, $r(\vec{\mathbf{u}}), r_1(\vec{\mathbf{u}}_1), \cdots, r_{k+m}(\vec{\mathbf{u}}_{k+m})$ are atoms, and there do not exist $1 \leq i \leq k$ and $k+1 \leq j \leq k+m$ such that $r_i = r_j$ (i.e., the rule is *consistent*). Each variable occurring in $\vec{\mathbf{u}}$ must occur in at least one of $\vec{\mathbf{u}}_1, \ldots, \vec{\mathbf{u}}_{k+m}$ (i.e., the rule is *safe*). $r(\vec{\mathbf{u}})$ is the *head* of the rule, denoted $head(\rho)$, while the conjunction $r_1(\vec{\mathbf{u}}_1), \ldots,\ not\ r_{k+m}(\vec{\mathbf{u}}_{k+m})$ constitutes the *body*, denoted $body(\rho)$. The number of terms in $\vec{\mathbf{u}}$ is the arity of $\rho$. If $\vec{\mathbf{u}}$ is empty, the arity is 0 and the rule is *boolean*. Variables in $head(\rho)$ are called *distinguished variables*, whereas variables appearing only in $body(\rho)$ are called *non-distinguished variables*. If $k+m=0$, the rule is called *fact*, and we omit the "$\leftarrow$" sign. If $k \neq 0$ and $m=0$ then $\rho$ is a *positive Datalog* (or simply a Datalog) rule, also called *conjunctive query (CQ)*. A *literal* $\ell$ is either an atom $p$, or a negated atom *not* $p$. In the former case $\ell$ is *positive*, in the latter $\ell$ is *negative*.

A *Datalog*$^\neg$ *program* $\mathcal{P}$ is a finite set of Datalog$^\neg$ rules. $\mathcal{P}$ is a *positive* Datalog (or simply, a Datalog) program, if all its rules are positive. We can associate the program

$\mathcal{P}$ with a labelled graph $G = \langle V, E \rangle$, where the set of vertices $V$ coincides with the set of predicates appearing in $\mathcal{P}$, and $E$ is the set of edges. An edge from $r$ to $s$ belongs to $E$ if and only if there exists a rule $\rho$ in $\mathcal{P}$ such that $r$ is the predicate in $head(\rho)$ and $s$ appears in a literal $\ell_s$ in $body(\rho)$. If $\ell_s$ is positive the edge is labelled with $\mathsf{p}$, whereas if $\ell_s$ is negative the edge is labelled with $\mathsf{n}$. If every cycle in $G$ comprises only edges labelled with $\mathsf{p}$, the program $\mathcal{P}$ is said *stratified Datalog*$^{\neg}$ program, denoted Datalog$^{\neg_s}$. Furthermore, if the graph $G$ is acyclic, $\mathcal{P}$ is said *non-recursive*

A (positive) non-recursive Datalog program in which all the rules present the same head predicate is also called *union of conjunctive queries (UCQ)*, and is often written in the form

$$r(\vec{\mathbf{u}}) \;\leftarrow\; conj_1(\vec{\mathbf{u}}, \vec{\mathbf{w}}_1) \vee \cdots \vee conj_m(\vec{\mathbf{u}}, \vec{\mathbf{w}}_m) \tag{2}$$

where for each $i \in \{1, \ldots, m\}$ $conj_i(\vec{\mathbf{u}}, \vec{\mathbf{w}}_i)$ is a conjunction of atoms.

Predicate symbols in $\mathcal{P}$ can be either *extensional* (*EDB predicates*), i.e., defined by the facts of a database (see Section 2.2), or *intensional* (*IDB predicates*), i.e., defined by the rules of the program.

For any program $\mathcal{P}$, let $\mathcal{U}_{\mathcal{P}}$ (the Herbrand Universe) be the set of all constants appearing in $\mathcal{P}$ (if no constants appear in $\mathcal{P}$, $\mathcal{U}_{\mathcal{P}}$ contains an arbitrary constant). A *substitution* of variables $\sigma$ is a finite set of the form $\sigma = \{X_1 \to T_1, \ldots, X_n \to T_n\}$ where each $X_i$ is a variable and each $T_i$ is a term, and the variables $X_1, \ldots, X_n$ are distinct. For each $i$ we say that $X_i$ and $T_i$ unify in $\sigma$. Given a set of atoms $D$ (resp. a rule $\rho$, a program $\mathcal{P}$) we indicate with $\sigma(D)$ (resp. $\sigma(\rho)$, $\sigma(\mathcal{P})$) the set obtained from $D$ (resp. $\rho$, $\mathcal{P}$) by simultaneously replacing each occurrence of $X_i$ with $T_i$. A term (resp. an atom, a literal, a rule or a program) is *ground*, if no variables occur in it. For any rule $\rho$ in $\mathcal{P}$, $ground(\rho)$ denotes the set of rules obtained by applying all possible substitutions from the variables in $\rho$ to elements of $\mathcal{U}_{\mathcal{P}}$. Let $B_{\mathcal{P}}$ be the set of all ground literals constructible from the predicate symbols appearing in $\mathcal{P}$ and the constants of $\mathcal{U}_{\mathcal{P}}$. An interpretation for $\mathcal{P}$ is any subset of $\mathcal{B}_{\mathcal{P}}$. The value of a ground positive literal $g$ w.r.t. an interpretation $I$, $value_I(g)$, is $true$ if $g \in I$ and $false$ otherwise. The value of a ground negative literal not $g$ is *not* $value_I(g)$. The truth value of a conjunction of ground literals $C = g_1, \ldots, g_n$ is the minimum over the values of the $g_i$, i.e. $value_I(C) = min(\{value_I(g_i) \mid 1 \leq i \leq n\})$. If $n = 0$, $value_I(C) = true$. A ground rule $\rho$ is *satisfied* by $I$ if $head(\rho) \geq body(\rho)$. An interpretation $M$ for $\mathcal{P}$ is a model of $\mathcal{P}$ if $M$ satisfies all ground rules of $\mathcal{P}$, denoted by $ground(\mathcal{P})$.

Let $\mathcal{P}$ be a positive Datalog program, the model-theoretic semantics assigns to $\mathcal{P}$ the set MM($\mathcal{P}$) of its *minimal models*, where a model $M$ for $\mathcal{P}$ is minimal, if no proper subset of $M$ is a model for $\mathcal{P}$.

The stable model semantics applies also to programs with negation. The *reduct* of a ground program $\mathcal{P}$ w.r.t. a set $X \subseteq B_{\mathcal{P}}$ is the positive ground program $\mathcal{P}^X$, obtained from $\mathcal{P}$ by

- deleting each rule having a ground literal *not* $p$, such that $p$ is in $X$;
- deleting the negative body from the remaining rules.

An interpretation $M$ is a stable model of $\mathcal{P}$ if and only if $M \in \mathrm{MM}(\mathcal{P}^M)$. The set of stable models of $\mathcal{P}$ is denoted by $\mathrm{SM}(\mathcal{P})$

It is well-known that for positive programs minimal models and stable models coincide, and that positive (resp. stratified) programs have a unique minimal (resp. stable) model.

In the following, given a Datalog$^\neg$ program $\mathcal{P}$, and a set of facts $\mathcal{D}$ that represent the extension of EDB predicates (i.e., facts of a database), we indicate with $\mathcal{P}^\mathcal{D}$ the union $\mathcal{P} \cup \mathcal{D}$, and say that $\mathcal{P}^\mathcal{D}$ is the evaluation of $\mathcal{P}$ over $\mathcal{D}$.

## 2.2 The Relational Model

We consider to have an infinite, fixed database domain $\mathcal{U}$ whose elements can be referenced by constants $c_1$,..., $c_n$ 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:

- $\Psi$ is a set of predicates or 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 $1, \ldots, n$;
- $\Sigma$ is a set of *integrity constraints* expressed on the relations in $\Psi$, i.e., assertions on the relations in $\Psi$ 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 $\Psi$ 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)*, *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 $\Psi$, $\mathbf{A} = A_1, \ldots, A_n$ ($n \geq 0$) 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 *key dependency* is an assertion the form $\mathit{key}(r) = \mathbf{A}$, where $r$ is a relation in $\Psi$, and $\mathbf{A} = A_1, \ldots, A_n$ is a sequence of distinct attributes of $r$. A database $\mathcal{DB}$ for $\mathcal{RS}$ satisfies a key dependency $\mathit{key}(r) = \mathbf{A}$ if for each $t_1, t_2 \in r^{\mathcal{DB}}$ with $t_1 \neq t_2$ we have $t_1[\mathbf{A}] \neq t_2[\mathbf{A}]$. We assume that at most one key dependency is specified for each relation;

[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 study in this report. See [9, 23] for more details on these dependencies.

- 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 $\Psi$, $\mathbf{A} = A_1, \ldots, A_n$ 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 $t_1[\mathbf{A}] = t_2[\mathbf{B}]$.

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

Finally, a *relational query* (or simply *query*) over $\mathcal{RS}$ is a formula (generally expressed in a language which amounts to a fragment of Datalog$^{\neg}$) whose EDB predicates are relations in $\Psi$. Each query $q$ is intended to extract a set of tuples of constants of $\mathcal{U}$, thus $q$ has an associated arity that indicates the number of constants of each tuple.

# 3 Formal framework

We now define a formal framework for data integration, based on the relational model with integrity constraints described in Section 2.2. We present both the syntax and the semantics of a data integration system. For a more detailed description of the framework we refer to Deliverable D3.3 [21].

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

- $\mathcal{G}$ is the *global schema* expressed in the relational model with inclusion, key and exclusion dependencies, i.e., $\mathcal{G} = \langle \Psi, \Sigma \rangle$, where $\Sigma = \Sigma_I \cup \Sigma_K \cup \Sigma_E$;

- $\mathcal{S}$ is the *source schema* expressed in the relational model without integrity constraints, i.e., $\mathcal{S} = \langle \Psi_{\mathcal{S}}, \emptyset \rangle$;

- $\mathcal{M}$ is the *mapping* between $\mathcal{G}$ and $\mathcal{S}$. In our framework we consider both the GAV and the LAV mapping. More precisely,

  - the GAV mapping is a set of assertions of the form $\langle r_{\mathcal{G}}, q_{\mathcal{S}} \rangle$, where $r_{\mathcal{G}}$ is a global relation and $q_{\mathcal{S}}$ is the associated query over the source schema $\mathcal{S}$. In this document, we study the setting in which the language used to express queries in the GAV mapping is non-recursive Datalog$^{\neg}$;
  - 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.

In order to specify the semantics of 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 $\mathcal{S}$. Based on $\mathcal{D}$, we specify which is the information content of the global schema $\mathcal{G}$. We call *global database* for $\mathcal{I}$ any database for $\mathcal{G}$. Formally, given a source database $\mathcal{D}$ for $\mathcal{I} = \langle \mathcal{G}, \mathcal{S}, \mathcal{M} \rangle$, the semantics of $\mathcal{I}$ w.r.t. $\mathcal{D}$, denoted $sem(\mathcal{I}, \mathcal{D})$, is the set of global databases for $\mathcal{I}$ such that:

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

(*ii*) $\mathcal{B}$ satisfies $\mathcal{M}$ w.r.t. $\mathcal{D}$. More specifically, $\mathcal{B}$ satisfies a GAV mapping $\mathcal{M}$ w.r.t. $\mathcal{D}$ if for each assertion $\langle r_{\mathcal{G}}, q_{\mathcal{S}} \rangle \in \mathcal{M}$ all the tuples satisfying $q_{\mathcal{S}}$ in $\mathcal{D}$ satisfy the global relation $r_{\mathcal{G}}$ in $\mathcal{B}$, i.e., $r_{\mathcal{G}}^{\mathcal{B}} \supseteq q_{\mathcal{S}}^{\mathcal{D}}$. Conversely, $\mathcal{B}$ satisfies a LAV mapping $\mathcal{M}$ w.r.t. $\mathcal{D}$ if for each assertion $\langle r_{\mathcal{S}}, q_{\mathcal{G}} \rangle \in \mathcal{M}$ all the tuples satisfying $r_{\mathcal{S}}$ in $\mathcal{D}$ satisfy $q_{\mathcal{G}}$ in $\mathcal{B}$, i.e., $r_{\mathcal{S}}^{\mathcal{D}} \subseteq q_{\mathcal{G}}^{\mathcal{B}}$. Note that the above definition amounts to consider any view in the mapping as *sound*, but not necessarily complete, which means that the data that can be retrieved at the sources are a subset of the data that satisfy the corresponding portion of the global schema. Other possible possible choices have been considered in [19, 21, 20].

According to the assumption on the views in the mapping above described, in the following we will refer to $sem(\mathcal{I}, \mathcal{D})$ as the *sound semantics*.

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

$$ans(Q, \mathcal{I}, \mathcal{D}) = \{ \langle c_1, \ldots, c_n \rangle \mid \langle c_1, \ldots, c_n \rangle \in Q^{\mathcal{B}} \text{ for each } \mathcal{B} \in sem(\mathcal{I}, \mathcal{D}) \ \}$$

**Example 3.1** Consider the global schema $\mathcal{G} = \langle \Psi, \Sigma_I, \Sigma_K, \Sigma_E \rangle$, where:

$\Psi$ contains the following relations:

$$\begin{array}{l} employee(Ename, Salary, Dep, Boss) \\ manager(Mname, DOB) \\ department(Code, Director); \end{array}$$

$\Sigma_I$ contains the following inclusion dependencies:

$$\begin{array}{rcll} employee[4] & \subseteq & manager[1] & (I_1) \\ manager[1] & \subseteq & department[2] & (I_2) \\ department[1] & \subseteq & employee[3] & (I_3), \end{array}$$

where $I_1$ states that a boss has to be also a manager, $I_2$ states that a manager has to be the director of a department, and $I_3$ states that at least one employee works in each department;

$\Sigma_K$ contains the following keys:

$$\begin{array}{rcl} key(employee) & = & \{Ename\} \\ key(manager) & = & \{Mname\} \\ key(department) & = & \{Code\} \end{array}$$

and $\Sigma_E$ contains the exclusion dependency

$$(department[2,1] \cap employee[1,3]) = \emptyset.$$

stating that a director is not an employee of the department that he directs.

We first construct a GAV data integration system $\mathcal{I}_G = \langle \mathcal{G}, \mathcal{S}_G, \mathcal{M}_G \rangle$ where the source schema $\mathcal{S}_G$ comprises the relations

$$\begin{array}{l} s_1(EmpName, EmpSalary, EmpDepName) \\ s_2(DepName, DepCode, DepDirector) \\ s_3(ManName, ManDOB, ManSalary), \end{array}$$

where $s_1$, $s_2$, and $s_3$ store information about employees, departments and managers respectively, and the mapping $\mathcal{M}_G$ is defined as follows:

$$\begin{array}{rcl} employee(N,S,D,B) & \leftarrow & s_1(N,S,Dn), s_2(Dn,D,B) \\ manager(N,Db) & \leftarrow & s_3(N,Db,S) \\ department(C,Dr) & \leftarrow & s_2(Dn,C,Dr). \end{array}$$

Let us now construct a LAV data integration system $\mathcal{I}_L = \langle \mathcal{G}, \mathcal{S}_L, \mathcal{M}_L \rangle$. To keep things simple, we consider a different source schema $\mathcal{S}_L$ that comprises $s_3$ that is the same as in $\mathcal{S}_G$, and $s_4(EmpName, EmpBoss)$ that stores information about employees and their boss. In such a system, we define the LAV mapping $\mathcal{M}_L$ as follows:

$$\begin{array}{rcl} s_3(N,Db,S) & \leftarrow & manager(N,Db), employee(N,S,D,N) \\ s_4(E,B) & \leftarrow & employee(E,S,D,B) \end{array}$$

Finally, we issue on the global schema the query

$$q(Dr) \leftarrow department(C,Dr), employee(Dr,S,D,B)$$

that asks for directors of departments that are also employees. ∎

In this document, we address the query answering problem, that is the problem of computing the set $ans(q, \mathcal{I}, \mathcal{D})$, by making use of query rewriting techniques. Hence, our aim is to exploit the mapping $\mathcal{M}$ in order to reformulate the query $Q$ into another query $Q_r$, the *rewriting*, that can be evaluated on the source database $\mathcal{D}$. We say that $Q_r$ is a *perfect rewriting* of $Q$ w.r.t. $\mathcal{I}$ iff $Q_r^{\mathcal{D}} = ans(Q, \mathcal{I}, \mathcal{D})$ for each source database $\mathcal{D}$ for $\mathcal{I}$.

# 4 Query Rewriting in GAV Data Integration Systems

In this section we present algorithms for computing the perfect rewriting of a UCQ in GAV integration systems. We first study the case in which only IDs are expressed on the global schema, then we deal with the simultaneous presence of IDs, KDs and EDs. Finally, we present a different characterization of the semantics of a GAV data integration system that allows us to properly deal with the situation in which data stored at the sources may result inconsistent with respect to KDs and EDs expressed over the global schema. In this respect, we first provide a general semantics for inconsistent data, which we call *loosely-sound* for being it a suitable relaxation of the sound semantics described in Section 3. Then, we present algorithms for computing the perfect rewriting of a UCQ in the presence of IDs, KDs, and EDs under the loosely-sound semantics.

Before proceeding, we introduce an example that we will use throughout this section.

**Example 4.1** Let $\mathcal{I}_0 = \langle \mathcal{G}_0, \mathcal{S}_0, \mathcal{M}_0 \rangle$ be a data integration system referring to the context of football teams. The global schema $\mathcal{G}_0 = \langle \Psi_0, \Sigma_0 \rangle$ consists of the three relations

$$\begin{array}{c} player(Pcode, Pname, Pteam) \\ team(Tcode, Tname, Tleader) \\ coach(Ccode, Cname, Cteam) \end{array}$$

and the following constraints:

$$\begin{array}{rcl} key(player) & = & \{Pcode\} \\ key(team) & = & \{Tcode\} \\ key(coach) & = & \{Ccode\} \\ team[Tleader] & \subseteq & player[Pcode] \\ (player[Pcode] & \cap & coach[Ccode]) = \emptyset \end{array}$$

where the last two constraints state that a team leader has to be a player, and that a coach can neither be a player nor a team leader.

The source schema $\mathcal{S}_0$ comprises the relation $s_1$ of arity 4, and the relations $s_2$, $s_3$ and $s_4$, of arity 3. The mapping $\mathcal{M}_0$ is defined by the three assertions

$$\begin{array}{rl} \langle player, & player(X,Y,Z) \leftarrow s_1(X,Y,Z,W) \rangle \\ \langle team, & team(X,Y,Z) \leftarrow s_2(X,Y,Z) \\ & team(X,Y,Z) \leftarrow s_3(X,Y,Z) \rangle \\ \langle coach, & coach(X,Y,Z) \leftarrow s_4(X,Y,Z) \rangle \end{array}$$

Finally, let $Q_0$ be the user query of the form $q(X) \leftarrow player(X,Y,Z)$ that asks for the codes of all players. ■

## 4.1 Query Rewriting in the presence of IDs

We start by studying query rewriting when only IDs are expressed on the global schema. To this aim, we present an algorithm that computes the perfect rewriting of a UCQ in a single database with inclusion dependencies under the sound semantics. The notion of perfect rewriting in the setting of a single database schema is analogous to the one given in the data integration context. More precisely, given a query $Q$ over a relational schema $\mathcal{RS}$, the perfect rewriting of $Q$ is another query $Q_r$ over $\mathcal{RS}$ such that $Q_r^{\mathcal{D}} = ans_s(Q, \mathcal{RS}, \mathcal{D})$ for each database instance $\mathcal{D}$ for $\mathcal{RS}$.

Before proceeding, we need some preliminary definitions.

### 4.1.1 Preliminary definitions

Given a conjunctive query $q$ over a relational schema $\mathcal{RS} = \langle\Psi, \Sigma_I\rangle$, we say that a variable $X$ is *unbound* in $q$ if it occurs only once in $q$, otherwise we say that $X$ is *bound* in $q$. Notice that variables occurring in the head of the query are necessarily bound, since each of them must also occur in the query body. A *bound term* is either a bound variable or a constant.

Following the standard notation used in deductive databases, we adopt a special symbol for all unbound variables in the query $q$: specifically, we use the special term "$\xi$" for such variables (deductive database systems use the symbol "_" to this purpose).

**Definition 4.2** Given an atom $g_1 = r(X_1, \ldots, X_n)$ and an atom $g_2 = r(Y_1, \ldots, Y_n)$, we say that $g_1$ *and* $g_2$ *unify* if there exists a variable substitution $\sigma$ such that $\sigma(g_1) = \sigma(g_2)$. Each such a $\sigma$ is called *unifier*. Moreover, if $g_1$ and $g_2$ unify, we denote as $\mathsf{mgu}(g_1, g_2)$ *a most general unifier of* $g_1$ *and* $g_2$.

Informally, two atoms unify if they can be made equal through a substitution of the variable symbols with other terms (either variables or constants) occurring in the atoms. We recall that each occurrence of $\xi$ in the two atoms actually represents a different, new variable symbol, therefore each occurrence of $\xi$ in $g_1$ and $g_2$ can be assigned to a different term.

For example, let $g_1 = r(X, \xi, c, \xi)$ and an atom $g_2 = r(\xi, Y, Z, \xi)$. Actually, $g_1$ and $g_2$ correspond to $g_1 = r(X, \xi_1, c, \xi_2)$, $g_2 = r(\xi_3, Y, Z, \xi_4)$. Then, $g_1$ and $g_2$ unify, and $\mathsf{mgu}(g_1, g_2) = \{\xi_3 \to X,\ \xi_1 \to Y,\ Z \to c,\ \xi_2 \to \xi_4\}$.

**Definition 4.3** Given an atom $g = s(X_1, \ldots, X_n)$ and an inclusion dependency $I \in \Sigma_I$ of the form $r[i_1, \ldots, i_k] \subseteq s[j_1, \ldots, j_k]$, we say that $I$ *is applicable to* $g$ if the following conditions hold:

1. for each $\ell$ such that $1 \leq \ell \leq n$, if $X_\ell \neq \xi$ then there exists $h$ such that $j_h = \ell$;
2. there exists a variable substitution $\sigma$ such that $\sigma(s[j_h]) = \sigma(s[j_{h'}])$ for each $h, h'$ such that $i_h = i_{h'}$. If such a $\sigma$ exists, we denote with $\sigma_{g,I}$ the most general substitution that verifies this condition (we recall that $\sigma$ is a most general substitution if for every other substitution $\sigma'$ that verifies this condition there exists a substitution $\gamma$ such that $\sigma' = \gamma\sigma$[2]).

Moreover, if $I$ is applicable to $g$, we denote with $gr(g, I)$ the atom $r(Y_1, \ldots, Y_m)$ ($m$ is the arity of $r$ in $\Psi$) where for each $\ell$ such that $1 \leq \ell \leq m$, $Y_\ell = \sigma_{g,I}(X_{j_h})$ if there exists $h$ such that $i_h = \ell$, otherwise $Y_\ell = \xi$.

Roughly speaking, an inclusion $I$ is applicable to an atom $g$ if the relation symbol of $g$ corresponds to the symbol in the right-hand side of $I$ and if all the attributes for which bound terms appear in $g$ are propagated by the inclusion $I$ (condition 1 of the above definition). Moreover, if the left-hand side of the inclusion contains repeated attributes, then the inclusion is applicable only if the corresponding terms in $g$ unify (condition 2 of the above definition). When $I$ is applicable to $g$, $gr(g, I)$ denotes the atom obtained from $g$ by using $I$ as a rewriting rule, whose direction is right-to-left.

[2]Given two substitutions $\sigma_1$ and $\sigma_2$ we use $\sigma_1\sigma_2(D)$ as a shortcut for $\sigma_1(\sigma_2(D))$.

For instance, let $I = r[1,2] \subseteq s[1,3]$, $g = s(X,\xi,c)$ and let $r$ be of arity 4. $I$ is applicable to $g$, since the attributes of $s$ which contain bound terms (i.e., attributes 1 and 3) are propagated by the inclusion, and $gr(g,I) = r(X,c,\xi,\xi)$. As another example, consider now $I = r[1,1] \subseteq s[1,3]$, $g = s(X,\xi,c)$ and let $r$ be of arity 4. $I$ is applicable to $g$ since the terms $X$ and $c$ unify and $\sigma_{g,I} = \{X \rightarrow c\}$. Therefore, $gr(g,I) = r(c,\xi,\xi,\xi)$.

### 4.1.2 The algorithm ID-rewrite

In Figure 1 we define the algorithm ID-rewrite that computes the perfect rewriting of a UCQ $Q$ over a relational schema $\mathcal{RS} = \langle \Psi, \Sigma_I \rangle$. In the algorithm, it is assumed that unbound variables in the input query $Q$ are represented by the symbol $\xi$.

Specifically, the algorithm generates a set of conjunctive queries that constitute a perfect rewriting of $Q$, by computing the closure of the set $Q$ with respect to the following two rules:

1. if there exists a conjunctive query $q \in Q$ such that $q$ contains two atoms $g_1$ and $g_2$ that unify, then the algorithm adds to $Q$ the conjunctive query $\mathsf{reduce}(q, g_1, g_2)$ obtained from $q$ by the following algorithm[3]:

   **Algorithm** $\mathsf{reduce}(q, g_1, g_2)$
   **Input:** conjunctive query $q$, atoms $g_1, g_2 \in body(q)$
   such that $g_1$ and $g_2$ unify
   **Output:** reduced conjunctive query $q'$
   $q' = q$;
   $\sigma := \mathsf{mgu}(g_1, g_2)$;
   $body(q') := body(q') - \{g_2\}$;
   $q' := \sigma(q')$;
   $q' := \tau(q')$;
   **return** $q'$

   Informally, the above algorithm reduce starts by eliminating $g_2$ from the query body; then, the substitution $\mathsf{mgu}(g_1, g_2)$ is applied to the whole query (both the head and the body), and finally the function $\tau$ is applied. Such a function replaces with $\xi$ each variable symbol $X$ such that there is a single occurrence of $X$ in the query. The use of $\tau$ is necessary in order to guarantee that, in the generated query, each unbound variable is represented by the symbol $\xi$.

   For instance, let $q = q(x) \leftarrow g_1(x,y), g_2(x,y)$ be a conjunctive query, and let $I = g_1[1,2] \subseteq g_2[1,2]$ be an inclusion dependency applicable to the atom $g_2(x,y)$. According to step 2 below, the algorithm atom-rewrite applied to $q$ and $I$ produces the query $q' = q(x) \leftarrow g_1(x,y), g_1(x,y)$. It is easy to see that the two atoms in $q'$ unify. Hence, the algorithm $\mathsf{reduce}(q, g_1(x,y), g_1(x,y))$ first produces the query $q(x) \leftarrow g_1(x,y)$, by applying the most general unifier (the identity function in this case), then generates the query $q(x) \leftarrow g_1(x,\xi)$ by means of the operator $\tau$.

[3] With a little abuse, in the algorithms we use set operators on the body of conjunctive queries.

**Algorithm** ID-rewrite($\mathcal{RS}, Q$)
**Input:** relational schema $\mathcal{RS} = \langle \Psi, \Sigma_I \rangle$,
where $\Sigma_I$ is a set of inclusion dependencies, UCQ $Q$
**Output:** perfect rewriting of $Q$
$Q' := Q$;
**repeat**
  $Q_{aux} := Q'$;
  **for each** $q \in Q_{aux}$ **do**
  (a) **for each** $g_1, g_2 \in body(q)$ **do**
    **if** $g_1$ and $g_2$ unify
    **then** $Q' := Q' \cup \{\mathsf{reduce}(q, g_1, g_2)\}$;
  (b) **for each** $g \in body(q)$ **do**
    **for each** $I \in \Sigma_I$ **do**
      **if** $I$ is applicable to $g$
      **then** $Q' := Q' \cup \{\mathsf{atom\text{-}rewrite}(q, g, I)\}$
**until** $Q_{aux} = Q'$;
**return** $Q'$

Figure 1: Algorithm ID-rewrite

2. if there exists an inclusion $I$ and a conjunctive query $q \in Q$ containing an atom $g$ such that $I$ is applicable to $g$, then the algorithm adds to $Q$ the query obtained from $q$ by replacing $g$ with $gr(g, I)$ and by applying the substitution $\sigma_{g,I}$ to $q$. Namely, this step adds new queries obtained by applying inclusion dependencies as rewriting rules (applied from right to left), through the following algorithm atom-rewrite:

   **Algorithm** atom-rewrite($q, g, I$)
   **Input:** conjunctive query $q$, atom $g \in body(q)$,
   ID $I$ such that $I$ is applicable to $g$
   **Output:** rewritten conjunctive query $q'$
   $q' = q$;
   $body(q') := body(q') - \{g\}$;
   $q' := \sigma_{g,I}(q')$;
   $body(q') := body(q') \cup \{gr(g, I)\}$;
   **return** $q'$

The above two rules correspond respectively to steps (a) and (b) of the algorithm.

Termination of the algorithm is immediately implied by the fact that the number of conjunctive queries that can be generated by the algorithm is finite, since the maximum length (i.e., the number of atoms) of a generated query is equal to the maximum length of the queries in $Q$, and the number of different atoms that can be generated by the algorithm is finite, since the alphabet of relation symbols used is finite (and corresponds to the relation symbols occurring in $Q$ and in $\Sigma_I$), as well as the set of terms used (corresponding to the set of variable names and constants occurring in the query $Q$ plus the symbol $\xi$).

**Example 3.1 (contd.)** Consider now only the global schema $\mathcal{G}$ and disregard the mapping and the source schema, and assume that only the IDs are specified over $\mathcal{G}$, i.e., $\mathcal{G} = \langle \Psi, \Sigma_I \rangle$, where $\Sigma_I$ contains:

$$\begin{array}{rcll} employee[4] & \subseteq & manager[1] & (I_1) \\ manager[1] & \subseteq & department[2] & (I_2) \\ department[1] & \subseteq & employee[3] & (I_3). \end{array}$$

Let us call $Q$ the user query

$$q(Dr) \leftarrow department(C, Dr), employee(Dr, S, D, B).$$

For ease of exposition we pose $Q = \{q_0\}$ where $q_0$ indicates the unique CQ in $Q$. Then we execute $\mathsf{ID\text{-}rewrite}(\mathcal{G}, Q)$. At the beginning, $Q_{aux} = Q' = Q = \{q_0\}$. It is easy to see that step (a) has not to be performed for $q_0$, whereas, for step (b), the ID $I_2$ is applicable to the atom $g_1 = department(\xi, Dr)$. Hence, $g_r(g_1, I_2) = manager(Dr, \xi)$ and $Q' = \{q_0\} \cup \{q_1\}$, where $q_1$ has the form $q(Dr) \leftarrow manager(Dr, \xi), employee(Dr, \xi, \xi, \xi)$. As for the query $q_1$, $I_1$ is applicable to the atom $g_2 = manager(Dr, \xi)$, and $g_r(g_2, I_1) = employee(Dr, \xi)$. Hence, we have that $Q' = \{q_0, q_1\} \cup \{q_2\}$, where $q_2$ has the form $q(Dr) \leftarrow employee(\xi, \xi, \xi, Dr), employee(Dr, \xi, \xi, \xi)\}$. Since condition in step (a) holds for the two atoms of $q_2$, ID-rewrite suitably executes the algorithm reduce that returns the query $q_3$ having the form $q(Dr) \leftarrow employee(Dr, \xi, \xi, Dr)$, which is added to $Q'$. Since there are no other IDs applicable to atoms in $Q'$, ID-rewrite returns the following UCQ:

$$\begin{array}{rcl} q(Dr) & \leftarrow & department(C, Dr), employee(Dr, S, D, B) \\ q(Dr) & \leftarrow & manager(Dr, D'), employee(Dr, S, D, B) \\ q(Dr) & \leftarrow & employee(N, S, D, Dr), employee(Dr, S', D', B) \\ q(Dr) & \leftarrow & employee(Dr, S, D, Dr) \end{array}$$

where, for the sake of clarity, the symbols $\xi$ have been replaced by the new fresh variable symbols $B, C, C'D, D', N, S, S'$. ■

**Example 4.1 (contd.)** Let us assume that only the inclusion dependency $team[Tleader] \subseteq player[Pcode]$, is expressed on the global schema $\mathcal{G}_0$, and disregard the KDs and the ED. Such inclusion dependency is applicable to the unique atom in the user query $q(X) \leftarrow player(X, Y, Z)$. By running the algorithm ID-rewrite, we obtain the following UCQ:

$$\begin{array}{rcl} q(X) & \leftarrow & player(X, Y, Z) \\ q(X) & \leftarrow & team(V, W, X) \end{array}$$

■

#### 4.1.3 Soundness and completeness of ID-rewrite

We now prove correctness of the algorithm ID-rewrite. To this aim, we first introduce the notion of *retrieved global database* $ret_G(\mathcal{I}, \mathcal{D})$ of a GAV integration system $\mathcal{I}$ and a source database $\mathcal{D}$, which is obtained by simply evaluating each view in the mapping $\mathcal{M}$ over $\mathcal{D}$.

In order to prove correctness of the algorithm ID-rewrite, we need to make use of the notions of semantics and answer set to queries in the case of a single relational database with inclusion dependencies. Such notions correspond to the ones defined in Section 3 for data integration systems (respectively, the definition of *sem* and *ans*), when we consider the retrieved global database as the database instance, and disregard the mapping. Therefore, in the following we denote the semantics of a relational schema $\mathcal{RS} = \langle \Psi, \Sigma_I \rangle$, with database instance $\mathcal{D}$ as $sem(\mathcal{RS}, \mathcal{D})$, and the answer set of a query $Q$ over such a database as $ans(Q, \mathcal{RS}, \mathcal{D})$. Based on the above notions, the notion of perfect rewriting of a query over a database with IDs is analogous to the one given in Section 3 in the case of a data integration system[4].

We also need to recall the notion of *chase* introduced by Johnson and Klug [16], and widely discussed in [21] and [20]. We denote as $chase(\mathcal{RS}, \mathcal{D})$ a (possibly infinite) database instance that is a representative of all the databases in $sem(\mathcal{RS}, \mathcal{D})$. The database $chase(\mathcal{RS}, \mathcal{D})$ is constructed, starting from $\mathcal{D}$, by repeatedly applying, as long as it is applicable, the following rule:

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

Hereinafter, for ease of exposition, we denote as $\Pi_{ID}$ the UCQ returned by ID-rewrite$(\mathcal{RS}, Q)$.

We are now ready to prove the main result of this section.

**Theorem 4.4** *Let $\mathcal{RS} = \langle \Psi, \Sigma_I \rangle$ be a relational schema, where $\Sigma_I$ is a set of inclusion dependencies, and let $Q$ be a UCQ over $\mathcal{RS}$. Then, $\Pi_{ID}$ is a perfect rewriting of $Q$ w.r.t. $\mathcal{RS}$.*

*Proof.* In the proof we make use of the following definition: let $q$ be a conjunctive query of arity $n$, and let $\bar{t}$ be an $n$-tuple; a set of facts $G$ is an *image of* $q(\bar{t})$ if there exists a substitution $\sigma$ such that $\sigma(head(q)) = q(\bar{t})$ and $\sigma(body(q)) = G$. It is easy to see that a tuple $\bar{t}$ belongs to the evaluation of the query $q$ over a database instance $\mathcal{D}$, i.e., $\bar{t} \in q^{\mathcal{D}}$, iff there exists $G \subseteq \mathcal{D}$ such that $G$ is an image of $q(\bar{t})$.

First, we prove that the algorithm is sound, i.e., for each $q \in \Pi_{ID}$ and for each instance $\mathcal{D}$, $q^{\mathcal{D}} \subseteq ans(Q, \mathcal{RS}, \mathcal{D})$. The proof is by induction on the structure of the set $\Pi_{ID}$, which can be seen as inductively defined from the initial set $Q$ by the two operations (a) and (b) in the algorithm in Figure 1. Let $\mathcal{D}$ be a database for $\mathcal{RS}$. The base step is immediately proven since, for each $q \in Q$ and for each $\mathcal{D}$, $q^{\mathcal{D}} \subseteq ans(Q, \mathcal{RS}, \mathcal{D})$. As for the inductive step, let $q$ be a conjunctive query such that $q^{\mathcal{D}} \subseteq ans(Q, \mathcal{RS}, \mathcal{D})$ for each instance $\mathcal{D}$ (inductive hypothesis), and let $q_1$ be a query obtained from $q$ by

[4]For more details on the relationship between a single database schema and a data integration system, see [21, 20].

applying step (a) of the algorithm. Then, $q_1 = \mathsf{reduce}(q, g_1, g_2)$, i.e., there exist two atoms $g_1, g_2 \in \mathit{body}(q)$ such that $g_1$ and $g_2$ unify. Then, if there exists a tuple $\bar{t}$ and a set of facts $G \subseteq \mathcal{D}$ such that $G$ is an image of $q_1(\bar{t})$, then $G$ is also an image of $q(\bar{t})$. Consequently, $q_1^{\mathcal{D}} \subseteq q^{\mathcal{D}}$, which implies that $q_1^{\mathcal{D}} \subseteq \mathit{ans}(Q, \mathcal{RS}, \mathcal{D})$. Under the same inductive hypothesis, now let $q_1$ be a query obtained from $q$ by applying step (b) of the algorithm. Suppose $\bar{t} \in q_1^{\mathcal{D}}$. Let $g$ be the atom of $q$ transformed by step (b) of the algorithm, i.e., $q_1 = \mathsf{atom\text{-}rewrite}(q, g, I)$. It is immediate to see that there exists a fact in $\mathcal{D}$ to which the ID chase rule triggered by $I$ is applicable, which implies that there exists an image of $q(\bar{t})$ in $\mathit{chase}(\mathcal{RS}, \mathcal{D})$. Therefore, $\bar{t} \in \mathit{ans}(q, \mathcal{RS}, \mathcal{D})$, which implies that $q_1^{\mathcal{D}} \subseteq \mathit{ans}(q, \mathcal{RS}, \mathcal{D})$, consequently $q_1^{\mathcal{D}} \subseteq \mathit{ans}(Q, \mathcal{RS}, \mathcal{D})$.

Then, we prove completeness of the algorithm. Let $\mathcal{D}$ be a database and $\bar{t}$ an $n$-tuple such that $\bar{t} \in \mathit{ans}(Q, \mathcal{RS}, \mathcal{D})$. We prove that there exists $q \in \Pi_{ID}$ such that $\bar{t} \in q^{\mathcal{D}}$. First, since $\bar{t} \in \mathit{ans}(Q, \mathcal{RS}, \mathcal{D})$, it follows that there exists $q_0 \in Q$ such that $\bar{t} \in \mathit{ans}(q_0, \mathcal{RS}, \mathcal{D})$. Therefore, from Lemma 5 of [16], it follows that there exists a finite number $k$ such that there is an image $G_k$ of $q_0(\bar{t})$ in $\mathit{chase}_k(\mathcal{RS}, \mathcal{D})$, where $\mathit{chase}_k(\mathcal{RS}, \mathcal{D})$ represents the database obtained from $\mathcal{D}$ after $k$ applications of the ID chase rule. Moreover, without loss of generality we can assume that every application of the ID chase rule is necessary in order to generate such an image $G_k$: i.e., $\mathit{chase}_k(\mathcal{RS}, \mathcal{D})$ can be seen as a forest (set of trees) where: (i) the roots correspond to the facts of $\mathcal{D}$ ; (ii) there are exactly $k$ edges, where each edge corresponds to an application of the ID chase rule; (iii) each leaf is either a fact in $\mathcal{D}$ or a fact in $G_k$. In the following, for each $i$ such that $0 \le i \le k$, we denote as $G_i$ the *pre-image* of $q_0(\bar{t})$ in $\mathit{chase}_i(\mathcal{RS}, \mathcal{D})$, i.e.,

$$\begin{aligned} G_i = \{ f \in \mathit{chase}_i(\mathcal{RS}, \mathcal{D}) \mid & \text{ there exists } f' \in G_k \text{ s.t. } f \text{ is an ancestor of } f' \text{ in} \\ & \mathit{chase}_k(\mathcal{RS}, \mathcal{D}) \text{ and in } \mathit{chase}_i(\mathcal{RS}, \mathcal{D}) \text{ there exists no successor of} \\ & f \text{ which is an ancestor of } f' \text{ in } \mathit{chase}_k(\mathcal{RS}, \mathcal{D}) \} \end{aligned}$$

Now we prove that, starting from $G_k$, we can "go back" through the rule application and find a query $q$ in $\Pi_{ID}$ such that the pre-image $G_0$ of $q_0(\bar{t})$ in $\mathit{chase}_0(\mathcal{RS}, \mathcal{D}) = \mathcal{D}$ is also an image of $q(\bar{t})$, i.e., such that $\bar{t} \in q^{\mathcal{D}}$. To this aim, we prove that there exists $q \in \Pi_{ID}$ such that $G_0$ is an image of $q(\bar{t})$ and $|\mathit{body}(q)| = |G_0|$. The proof is by induction on the structure of $\mathit{chase}_k(\mathcal{RS}, \mathcal{D})$.

Base step: There exists $q' \in \Pi_{ID}$ such that $G_k$ is an image of $q'(\bar{t})$ and $|\mathit{body}(q')| = |G_k|$. This is an immediate consequence of the facts that: (i) $q_0 \in \Pi_{ID}$; (ii) $\Pi_{ID}$ is closed with respect to step (a) of the algorithm. Indeed, if $|G_k| < |\mathit{body}(q_0)|$ then there exist two atoms $g_1, g_2 \in \mathit{body}(q_0)$ and a fact $f$ in $G_k$ such that $f$ and $g_1$ unify and $f$ and $g_2$ unify, which implies that $g_1$ and $g_2$ unify, therefore by step (a) of the algorithm it follows that there exists a query $q_1 \in \Pi_{ID}$ (with $q_1 = \mathsf{reduce}(q_0, g_1, g_2)$) such that $G_k$ is an image of $q_1(\bar{t})$ and $|\mathit{body}(q_1)| = |\mathit{body}(q_0)| - 1$. Now, if $|G_k| < |\mathit{body}(q_1)|$, we can iterate the above proof, thus we conclude that there exists $q' \in \Pi_{ID}$ such that $G_k$ is an image of $q'(\bar{t})$ and $|\mathit{body}(q')| = |G_k|$.

Inductive step: suppose that there exists $q \in \Pi_{ID}$ such that $G_i$ is an image of $q(\bar{t})$ and $|\mathit{body}(q)| = |G_i|$. Let $I = r_1[\mathbf{X}] \subseteq r_2[\mathbf{Y}]$ be the ID applied to $\mathit{chase}_{i-1}(\mathcal{RS}, \mathcal{D})$ to obtain $\mathit{chase}_i(\mathcal{RS}, \mathcal{D})$. Since non-propagated attributes in $r_2$ contain new constants, i.e., constants not appearing elsewhere in $G_i$, and since $|\mathit{body}(q)| = |G_i|$, it follows that the corresponding variables of the atom $r_2$ in $body(q)$ are unbound variables (represented by $\xi$); moreover, if $X_i = X_j$ then $r_2[Y_i] = r_2[Y_j]$, by the ID chase rule. Therefore, there exists an atom $g = r_2(\overline{Z}) \in body(q_i)$ such that $I$ is applicable to $g$. Therefore, by step (b) of the algorithm it follows that there exists a query $q_1 \in \Pi_{ID}$ (with $q_1 = \mathsf{atom\text{-}rewrite}(q, g, I)$) such that $G_{i-1}$ is an image of $q_1(\overline{t})$. Now, there are two possible cases: either $|body(q_1)| = |G_{i-1}|$, and in this case the thesis follows; or $|body(q_1)| = |G_{i-1}|+1$. This last case arises if and only if the fact of the form $f = r_1(\overline{t}')$ to which the ID chase rule is applied is both in $G_{i-1}$ and in $G_i$. This implies that there exist two atoms $g_1, g_2 \in body(q_1)$ such that $f$ and $g_1$ unify and $f$ and $g_2$ unify, hence $g_1$ and $g_2$ unify, therefore by step (a) of the algorithm (applied to $q_1$) it follows that there exists $q_2 \in \Pi_{ID}$ (with $q_2 = \mathsf{reduce}(q_1, g_1, g_2)$) such that $G_{i-1}$ is an image of $q_2(\overline{t})$ and $|body(q_2)| = |G_{i-1}|$, which proves the thesis. □

### 4.1.4 Data Integration Setting

We now go back to the data integration setting. According Theorem 4.4, the algorithm ID-rewrite provides a perfect rewriting $\Pi_{ID}$ of a UCQ $Q$ with respect to the global schema $\mathcal{G}$. However, since data are stored at the sources, we are interested in reformulating the user query in terms of the source relations in $\mathcal{S}$ rather than in another query on the global relations in $\mathcal{G}$. To this aim, we make use of the mapping $\mathcal{M}$. Our use of $\mathcal{M}$ corresponds in principle to *unfold*, i.e., replace, each atom in $\Pi_{ID}$ with a query over the sources. More precisely, we define the non-recursive Datalog$^\neg$ program $\Pi_\mathcal{M} = \{q_\mathcal{S} \mid \langle r_\mathcal{G}, q_\mathcal{S}\rangle \in \mathcal{M}\}$, and use it to produce the perfect rewriting of $Q$.

**Theorem 4.5** *Let $\mathcal{I} = \langle \mathcal{G}, \mathcal{S}, \mathcal{M}\rangle$ be a data integration system and let $Q$ be a user query over $\mathcal{I}$. Then, $\Pi_{ID} \cup \Pi_\mathcal{M}$ is a perfect rewriting of $Q$ w.r.t. $\mathcal{I}$.*

*Proof.* It is easy to see that the evaluation of $\Pi_\mathcal{M}$ over each source database $\mathcal{D}$ for $\mathcal{I}$ produces the retrieved global database $ret_G(\mathcal{I}, \mathcal{D})$. Then, the thesis follows immediately from Theorem 4.4. □

**Example 3.1 (contd.)** Let us now to take into account the mapping of the GAV system $\mathcal{I}_G = \langle \mathcal{G}, \mathcal{S}_G, \mathcal{M}_G\rangle$. The perfect rewriting for the query $Q$ w.r.t. $\mathcal{I}_G$, is as follows

$$\begin{array}{rcl}
q(Dr) & \leftarrow & department(C, Dr), employee(Dr, S, D, B)\\
q(Dr) & \leftarrow & manager(Dr, D'), employee(Dr, S, D, B)\\
q(Dr) & \leftarrow & employee(N, S, D, Dr), employee(Dr, S', D', B)\\
q(Dr) & \leftarrow & employee(Dr, S, D, Dr)\\
employee(N, S, D, B) & \leftarrow & s_1(N, S, Dn), s_2(Dn, D, B)\\
manager(N, Db) & \leftarrow & s_3(N, Db, S)\\
department(C, Dr) & \leftarrow & s_2(Dn, C, Dr).
\end{array}$$

■

**Example 4.1 (contd.)** Analogously, by taking into account the mapping $\mathcal{M}_0$, we easily obtain the perfect rewriting for the query $Q_0$ w.r.t. $\mathcal{I}_0$, which is as follows

$$\begin{array}{rcl} q(X) & \leftarrow & player(X,Y,Z) \\ q(X) & \leftarrow & team(V,W,X) \\ player(X,Y,Z) & \leftarrow & s_1(X,Y,Z,W) \\ team(X,Y,Z) & \leftarrow & s_2(X,Y,Z) \\ team(X,Y,Z) & \leftarrow & s_3(X,Y,Z) \\ coach(X,Y,Z) & \leftarrow & s_4(X,Y,Z) \end{array}$$

■

## 4.2 Query Rewriting in the presence of IDs, KDs, and EDs

We now address the problem of query rewriting in GAV systems in the case where IDs, EDs and KDs are defined on the global schema. First of all, we consider interaction between IDs and KDs. As widely discussed in [21, 20], KDs and IDs interact reciprocally so that the (decision) problem of query answering in this setting becomes undecidable.

In the following, we recall some results and definitions already presented in [21] and [20]. To this aim, we need to maintain distinct the set of constants of the domain $\mathcal{U}$ that occur in the source database $\mathcal{D}$, that is the *active domain* of $\mathcal{D}$, from the other constants of $\mathcal{U}$, which can be used, for instance, in the construction of the chase (see Section 4.1.3). Hereinafter, we indicate with $\mathcal{U}_\mathcal{D}$ the active domain of a source database $\mathcal{D}$.

In particular, we recall the following property, (see also Theorem 4.10 of [20]).

**Theorem 4.6** *Let $\mathcal{I} = \langle \mathcal{G}, \mathcal{S}, \mathcal{M} \rangle$ be a data integration system, with $\mathcal{G} = \langle \Psi, \Sigma_I, \Sigma_K \rangle$, where $\Sigma_I$ and $\Sigma_K$ are sets of IDs and KDs respectively. Let $\mathcal{D}$ be a source database for $\mathcal{I}$, let $Q$ be a UCQ over $\mathcal{G}$, and let $t$ be a tuple of constants of $\mathcal{U}_\mathcal{D}$. Then, the problem of establishing whether $t \in ans(Q, \mathcal{I}, \mathcal{D})$ is undecidable.*

Undecidability of query answering in the presence of IDs and KDs immediately implies undecidability of query answering in the presence of also EDs, and of computing the perfect rewriting in such a setting.

Interestingly, as shown [21, 20], the problem of query answering becomes decidable if we restrict the IDs to be in a particular class, so that they do not interact with KDs.

**Definition 4.7** Given a relational schema $\mathcal{RS} = \langle \Psi, \Sigma_I, \Sigma_K \rangle$, an inclusion dependency in $\Sigma_I$ 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 $\Sigma_K$ if either: $(i)$ no KD is defined on $r_2$, or $(ii)$ the KD $key(r_2) = \mathbf{K}$ is in $\Sigma_K$, 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 $\Sigma_I$ are NKCIDs with respect to $\Sigma_K$.

Informally, a set of dependencies is NKC if no ID in $\Sigma_I$ propagates a proper superset 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 correspond to IDs of the form $r_1[\mathbf{A}_1] \subseteq r_2[\mathbf{A}_2]$ such that $key(r_2) = \mathbf{A}_2$.

In the following, we simply call NKC integration systems those data integration systems in which the global schema is NKC.

The most important feature of a NKC data integration system is the *separation* between IDs and KDs; in such a case, in fact, we can take IDs into account as if the KDs were not expressed on $\mathcal{G}$. More precisely, if $ret_G(\mathcal{I},\mathcal{D})$[5] satisfies KDs on $\mathcal{G}$, we can compute the certain answers to a query $Q$ only by reasoning on the IDs specified over $\mathcal{G}$ (see also Theorem 4.5 of [20]).

A similar "separation" result has been provided in [20] also for EDs and IDs. In such a case, no restrictions have to be imposed to the form of IDs or EDs, since query answering in this setting is always decidable. On the other hand, in order to overlook the presence of EDs when computing certain answers to $Q$, we need to consider explicitly the set of exclusion dependencies that are logical consequence of the IDs and EDs originally specified over $\mathcal{G}$, and verify whether $ret_G(\mathcal{I},\mathcal{D})$ is consistent with them (see also Theorem 4.12 of [20]).

By combining together the two separation properties described above we obtain the following theorem, in which $\Sigma_E^*$ indicates the set of EDs that are logically implied by the set of dependencies $\Sigma_E \cup \Sigma_I$.

**Theorem 4.8 [Theorem 4.29 of [20]]**
*Let $\mathcal{I} = \langle \mathcal{G}, \mathcal{S}, \mathcal{M} \rangle$ be a NKC GAV system, where $\mathcal{G} = \langle \Psi, \Sigma_I, \Sigma_K, \Sigma_E \rangle$, let $\mathcal{I}' = \langle \mathcal{G}', \mathcal{S}, \mathcal{M} \rangle$, where $\mathcal{G}' = \langle \Psi, \Sigma_I \rangle$, be the system obtained by $\mathcal{I}$ by eliminating the KDs and the EDs of $\mathcal{G}$; let $\mathcal{D}$ be a source database for $\mathcal{I}$ and $\mathcal{I}'$. Moreover, let $Q$ be a query of arity $n$ over $\mathcal{G}$ and $\mathcal{G}'$, and $t$ an $n$-tuple of constants of $\mathcal{U}_\mathcal{D}$. We have that $t \notin ans(Q,\mathcal{I},\mathcal{D})$ iff $ret_G(\mathcal{I},\mathcal{D})$ is consistent with $\Sigma_K \cup \Sigma_E^*$ and $t \notin ans(Q,\mathcal{I}',\mathcal{D})$.*

We now go back to query rewriting. According to the above result, in the case of a NKC GAV integration system, we can apply the same technique developed for IDs alone, provided that data at the sources are consistent with the key and the exclusion dependencies on the global schema, i.e., the retrieved global database $ret_G(\mathcal{I},\mathcal{D})$ does not violate KDs, and the closure of EDs w.r.t. to implication by IDs and EDs. This is formally enounced in the following Theorem.

**Theorem 4.9** *Let $\mathcal{I} = \langle \mathcal{G}, \mathcal{S}, \mathcal{M} \rangle$ be a NKC GAV system, where $\mathcal{G} = \langle \Psi, \Sigma_I, \Sigma_K, \Sigma_E \rangle$, let $\mathcal{D}$ be a source database for $\mathcal{I}$ such that $ret_G(\mathcal{I},\mathcal{D})$ is consistent with $\Sigma_K \cup \Sigma_E^*$, and let $Q$ be a user query over $\mathcal{I}$. Then, $(\Pi_{ID} \cup \Pi_\mathcal{M})^\mathcal{D} = ans(Q, I, \mathcal{D})$.*

*Proof.* Follows directly from Theorems 4.5 and 4.8. □

Observe, however, that, once $ret_G(\mathcal{I},\mathcal{D})$ is computed, the unfolding step, i.e., the use of $\Pi_\mathcal{M}$ becomes unnecessary, since it is possible to obtain the answers to $Q$ by directly evaluating $\Pi_{ID}$ over $ret_G(\mathcal{I},\mathcal{D})$.

**Example 3.1 (contd.)** Let us now consider also the sets $\Sigma_K$ and $\Sigma_E$ of key and exclusion dependencies expressed on $\mathcal{G}$ that have been introduced in Section 3:

$$\begin{array}{rcl} key(employee) & = & \{Ename\} \\ key(manager) & = & \{Mname\} \\ key(department) & = & \{Code\} \\ (department[2,1] & \cap & employee[1,3]) = \emptyset. \end{array}$$

[5]The definition of $ret_G(\mathcal{I},\mathcal{D})$ has been given in Section 4.1.3.

It is immediate to verify that the global schema $\mathcal{G} = \langle \Psi, \Sigma_I, \Sigma_K, \Sigma_E \rangle$, is a NKC system, since no ID in $\Sigma_I$ contains a proper superset of the key of the relation in its right-hand side. In such a situation, given a source database $\mathcal{D}$ such that the retrieved global database $ret_G(\mathcal{I}, \mathcal{D})$ satisfies $\Sigma_K \cup \Sigma_E^*$, the answer to $Q$ is computed by evaluating over $ret_G(\mathcal{I}, \mathcal{D})$ the same rewritten query $\mathsf{ID\text{-}rewrite}(\mathcal{G}, Q)$ that we have computed in the presence of inclusion dependencies only. ∎

**Example 4.1 (contd.)** As discussed in the above example, also in this case the the global schema $\mathcal{G}_0 = \langle \Psi_0, \Sigma_0 \rangle$, is an NKC system. Therefore, also in this case $ans(Q_0, \mathcal{I}_0, \mathcal{D}_0) = \mathsf{ID\text{-}rewrite}(\mathcal{G}_0, Q_0)^{ret_G(\mathcal{I}_0, \mathcal{D}_0)}$ for each source database $\mathcal{D}_0$ for $\mathcal{I}_0$ such that $ret_G(\mathcal{I}_0, \mathcal{D}_0)$ is consistent with $\Sigma_{K_0} \cup \Sigma_{E_0}^*$. ∎

Conversely, when data at the sources are such that $ret_G(\mathcal{I}, \mathcal{D})$ is not consistent $\Sigma_K \cup \Sigma_E^*$, by the semantics defined in Section 3, any tuple is in the answer to any query, i.e., the answer to the query is meaningless. To overcome this problem, a possibility may be to adopt different semantics for the system, which allow to cope with violations of inclusion, key and exclusion dependencies. Roughly speaking, as in the case of violations of IDs, which "automatically repaired" under the sound semantics, such semantics allow for a repair of data violating key and exclusion dependencies. In the next subsection we present the *loosely-sound* semantics for GAV data integration systems, a relaxation of the sound semantics that allows us to deal with inconsistent data and to obtain certain answers to user queries rather than meaningless answers. Other material concerning the problem of inconsistency in data integration can be found in [21, 20], where we have studied this problem under different semantics, both in the GAV and in the LAV framework.

### 4.2.1 Semantics for inconsistent data sources

As already said, in the sound semantics, violations of IDs are treated "automatically" because of the nature of the semantics; instead, the violation of a single KD or ED leads to the non-interesting case in which $sem(\mathcal{I}, \mathcal{D}) = \emptyset$.

According to a common approach in the literature on inconsistent databases [12, 22, 2, 14], we now introduce the *loosely-sound* semantics, opposed to the one adopted so far (that we will call *strictly-sound* in the rest of the section), in which the soundness assumption is suitably relaxed. The intuition is that in the "relaxed" semantics we are allowed to delete tuples from $ret_G(\mathcal{I}, \mathcal{D})$ to repair violations of KDs and EDs , as long as we "minimize" such deletions; violations of IDs are treated as in the sound semantics.

Given a source database $\mathcal{D}$ for $\mathcal{I} = \langle \mathcal{G}, \mathcal{S}, \mathcal{M} \rangle$, we define an ordering on the set of all global databases for $\mathcal{I}$ that are consistent with $\mathcal{G}$. Given two such global databases $\mathcal{B}_1$ and $\mathcal{B}_2$, we pose $\mathcal{B}_1 \leq_{\mathcal{D}} \mathcal{B}_2$ iff

$$\mathcal{B}_1 \cap ret_G(\mathcal{I}, \mathcal{D}) \supset \mathcal{B}_2 \cap ret_G(\mathcal{I}, \mathcal{D}).$$

That is, the portion of $ret_G(\mathcal{I}, \mathcal{D})$ contained in the global database is greater in $\mathcal{B}_1$ than in $\mathcal{B}_2$, i.e., $\mathcal{B}_1$ approximates the sound mapping better than $\mathcal{B}_2$. Furthermore, $\mathcal{B}_1 <_{\mathcal{D}} \mathcal{B}_2$ stands for $\mathcal{B}_1 \leq_{\mathcal{D}} \mathcal{B}_2 \wedge \mathcal{B}_2 \not\leq_{\mathcal{D}} \mathcal{B}_1$.

We call *maximal* w.r.t. $\leq_{\mathcal{D}}$ a global database $\mathcal{B}$ for $\mathcal{I}$ consistent with $\mathcal{G}$, such that there exists no global database $\mathcal{B}'$ consistent with $\mathcal{G}$ such that $\mathcal{B}' <_{\mathcal{D}} \mathcal{B}$. Based on this notion, we define the loosely-sound semantics $sem_l$ as follows:

$$sem_l(\mathcal{I}, \mathcal{D}) = \{\mathcal{B} \mid \mathcal{B} \text{ is consistent with } \mathcal{G} \text{ and } \mathcal{B} \text{ is maximal w.r.t. } \leq_{\mathcal{D}}\}.$$

Finally, we denote with $ans_l(Q, \mathcal{I}, \mathcal{D})$ the set of certain answers to a user query $Q$ under the loosely-sound semantics.

**Example 4.1 (contd.)** Assume that the information content of the sources is given by the database $\mathcal{D}_0 = \{\ s_1(\mathsf{10}, \mathsf{Totti}, \mathsf{RM}, \mathsf{27}),\ s_1(\mathsf{9}, \mathsf{Giggs}, \mathsf{MU}, \mathsf{28}),\ s_2(\mathsf{RM}, \mathsf{Roma}, \mathsf{10}),\ s_3(\mathsf{MU}, \mathsf{Man.\ Utd.}, \mathsf{8}),\ s_3(\mathsf{RM}, \mathsf{Real\ Madrid}, \mathsf{10}),\ s_4(\mathsf{7}, \mathsf{Ferguson}, \mathsf{MU})\}$. Then, the retrieved

$player^{ret_G(\mathcal{I}_0,\mathcal{D}_0)}$:

| | | |
|---|---|---|
| 10 | Totti | RM |
| 9 | Giggs | MU |

$team^{ret_G(\mathcal{I}_0,\mathcal{D}_0)}$:

| | | |
|---|---|---|
| RM | Roma | 10 |
| MU | Man. Utd. | 8 |
| RM | Real Madrid | 10 |

$coach^{ret_G(\mathcal{I}_0,\mathcal{D}_0)}$:

| | | |
|---|---|---|
| 7 | Ferguson | MU |

Figure 2: Retrieved global database for the football teams scenario.

global database $ret_G(\mathcal{I}_0, \mathcal{D}_0)$ shown in Fig. 2 violates

1. the inclusion dependency $team[Tleader] \subseteq player[Pcode]$, witnessed by the fact $team(\mathsf{MU}, \mathsf{Man.\ Utd.}, \mathsf{8})$, which has $\mathsf{8}$ as leader's code that does not appear as first component of *player*, and

2. the key constraint on *team*, witnessed by the facts $team(\mathsf{RM}, \mathsf{Roma}, \mathsf{10})$ and $team(\mathsf{RM}, \mathsf{Real\ Madrid}, \mathsf{10})$, which coincide on *Tcode* but differ on *Tname*.

A repair of such violations can be achieved by removing exactly one of these last two facts, and by adding a tuple to *palyer* of the form $\langle \mathsf{8}, \alpha, \beta\rangle$, where $\alpha$ and $\beta$ are constants of the domain $\mathcal{U}$. hence, $sem_l(\mathcal{I}_0, \mathcal{D}_0)$ contains databases of the form $\mathcal{B}_1$ and $\mathcal{B}_2$ shown in Figure 3, and databases obtained from them by consistently adding other facts.

$player^{\mathcal{B}_1}$:

| | | |
|---|---|---|
| 10 | Totti | RM |
| 9 | Giggs | MU |
| 8 | $\alpha$ | $\beta$ |

$team^{\mathcal{B}_1}$:

| | | |
|---|---|---|
| RM | Roma | 10 |
| MU | Man. Utd. | 8 |

$coach^{\mathcal{B}_1}$:

| | | |
|---|---|---|
| 7 | Ferguson | MU |

$player^{\mathcal{B}_2}$:

| | | |
|---|---|---|
| 10 | Totti | RM |
| 9 | Giggs | MU |
| 8 | $\alpha$ | $\beta$ |

$team^{\mathcal{B}_2}$:

| | | |
|---|---|---|
| MU | Man. Utd. | 8 |
| RM | Real Madrid | 10 |

$coach^{\mathcal{B}_2}$:

| | | |
|---|---|---|
| 7 | Ferguson | MU |

Figure 3: Forms of global databases in $sem_l(\mathcal{I}_0, \mathcal{D}_0)$.

For the query $q(X) \leftarrow player(X, Y, Z)$, we thus obtain that $ans_l(q, \mathcal{I}_0, \mathcal{D}_0) = \{\mathsf{8}, \mathsf{9}, \mathsf{10}\}$. If we consider the query $q'(Y) \leftarrow team(X, Y, Z)$, we have that

$ans_l(q', \mathcal{I}_0, \mathcal{D}_0) = \{\mathsf{Man.\ Utd.}\}$, while considering $q''(X,Z) \leftarrow team(X,Y,Z)$, we have that $ans_l(q'', \mathcal{I}_0, \mathcal{D}_0) = \{\langle \mathsf{RM}, 10\rangle, \langle \mathsf{MU}, 8\rangle\}$. ■

It is immediate to verify that, if $sem(\mathcal{I}, \mathcal{D}) \neq \emptyset$, then $sem_l(\mathcal{I}, \mathcal{D}) = sem(\mathcal{I}, D)$, i.e., if there exists a global database that satisfies both the constraints on $\mathcal{G}$ and the mapping assertions in $\mathcal{M}$ w.r.t. a source database $\mathcal{D}$, then the strictly-sound and the loosely-sound semantics coincide.

### 4.2.2 Query Rewriting under the Loosely-sound Semantics

We now address the problem of query answering under the loosely-sound semantics. Specifically, we present a sound and complete rewriting technique to compute certain answers to queries posed to NKC systems.

Our method relies on Theorem 4.8 stating that for NKC systems it is possible to first deal with inclusion dependencies and then with key and exclusion dependencies, provided the computation of $\Sigma_E^*$: actually, for the IDs we exploit the algorithm ID-rewrite, whereas for the other dependencies we make use of a set of Datalog$^\neg$ rules, constructed according to KDs and EDs in $\Sigma_K$ and $\Sigma_E^*$. Intuitively, such rules, together with the rules in the mapping, allow us to compute the maximal subsets of $ret_G(\mathcal{I}, \mathcal{D})$ that are consistent with $\Sigma_K \cup \Sigma_E^*$, in such a way that we can then evaluate the program $\Pi_{ID}$ produced by the algorithm ID-rewrite over each such subset.

For the sake of clarity, we first consider the case of KDs, and then we address the general setting of IDS, KDs, and EDs. In this case, we define a Datalog$^\neg$ program $\Pi_{KD}$ that, for each relation $r \in \mathcal{G}$, comprises the rules

$$
\begin{array}{rcl}
r(\vec{\mathbf{x}}, \vec{\mathbf{y}}) & \leftarrow & r_{\mathcal{D}}(\vec{\mathbf{x}}, \vec{\mathbf{y}})\ ,\ not\ \overline{r}(\vec{\mathbf{x}}, \vec{\mathbf{y}}) \\
\overline{r}(\vec{\mathbf{x}}, \vec{\mathbf{y}}) & \leftarrow & r_{\mathcal{D}}(\vec{\mathbf{x}}, \vec{\mathbf{y}})\ ,\ r(\vec{\mathbf{x}}, \vec{\mathbf{z}})\ ,\ Y_1 \neq Z_1 \\
 & \cdots & \\
\overline{r}(\vec{\mathbf{x}}, \vec{\mathbf{y}}) & \leftarrow & r_{\mathcal{D}}(\vec{\mathbf{x}}, \vec{\mathbf{y}})\ ,\ r(\vec{\mathbf{x}}, \vec{\mathbf{z}})\ ,\ Y_m \neq Z_m
\end{array}
$$

where: in $r(\vec{\mathbf{x}}, \vec{\mathbf{y}})$ the variables in $\vec{\mathbf{x}}$ correspond to the attributes constituting the key of the relation $r$; $\vec{\mathbf{y}} = Y_1, \ldots, Y_m$ and $\vec{\mathbf{z}} = Z_1, \ldots, Z_m$[6].

Informally, for each relation $r$, $\Pi_{KD}$ contains (i) a relation $r_{\mathcal{D}}$ that represents $r^{ret_G(\mathcal{I}, \mathcal{D})}$; (ii) a relation $r$ that represents a subset of $r^{ret_G(\mathcal{I}, \mathcal{D})}$ that is consistent with the KD for $r$; (iii) an auxiliary relation $\overline{r}$. The above rules force each stable model $M$ of $\Pi_{KD}$, i.e., $M \in \mathrm{SM}(\Pi_{KD})$, to be such that $r^M$ is a maximal subset of tuples from $r^{ret_G(\mathcal{I}, \mathcal{D})}$ that are consistent with the KD for $r$.

Let us now consider the situation in which also exclusion dependencies are specified on the global schema, i.e., $\mathcal{G} = \langle \Psi, \Sigma_I, \Sigma_K, \Sigma_E \rangle$. We have now to force each stable model $M$ of our program to be such that $r^M$ is a maximal subset of tuples from $r^{ret_G(\mathcal{I}, \mathcal{D})}$ consistent with both the KD for $r$ and the EDs in $\Sigma_E^*$ that involve $r$.

To this aim, for each exclusion dependency $(r[\mathbf{A}] \cap\ s[\mathbf{B}]) = \emptyset$ in $\Sigma_E^*$ we add to $\Pi_{KD}$ the set of rules:

$$
\overline{r}(\vec{\mathbf{x}}, \vec{\mathbf{y}}) \ \leftarrow \ r_{\mathcal{D}}(\vec{\mathbf{x}}, \vec{\mathbf{y}}) s(\vec{\mathbf{x}}, \vec{\mathbf{z}})\ ,
$$

[6] Without loss of generality we assume that the attributes in the key precede all other attributes in $r$.

$$\overline{s}(\vec{\mathbf{x}},\vec{\mathbf{y}}) \ \leftarrow \ s_{\mathcal{D}}(\vec{\mathbf{x}},\vec{\mathbf{y}}) \ , \ r(\vec{\mathbf{x}},\vec{\mathbf{z}})$$

where in $r(\vec{\mathbf{x}},\vec{\mathbf{z}})$ the variables in $\vec{\mathbf{x}}$ correspond to the sequence of attributes $\mathbf{A}$ of $r$, and in $s(\vec{\mathbf{x}},\vec{\mathbf{z}})$ the variables in $\vec{\mathbf{x}}$ correspond to the sequence of attributes $\mathbf{B}$ of $s$[7]. Let us call $\Pi_{ED}$ such set of rules.

Then, given a NKC GAV integration system $\mathcal{I} = \langle \mathcal{G}, \mathcal{S}, \mathcal{M} \rangle$ where $\mathcal{G} = \langle \Psi, \Sigma_I, \Sigma_K, \Sigma_E \rangle$ and given a user query $Q$, we consider the Datalog$^\neg$ program $\Pi = \Pi_{ID} \cup \Pi_{\mathcal{MD}} \cup \Pi_{KD} \cup \Pi_{ED}$, where $\Pi_{ID}$ is obtained through $\textsf{ID-rewrite}(\mathcal{G}, Q)$, and $\Pi_{\mathcal{MD}}$ is obtained from $\Pi_{\mathcal{M}}$ by replacing each symbol $r$ with $r_{\mathcal{D}}$.

**Example 4.1 (contd.)** According to the above discussion, the program $\Pi$ in our example, is as follows:

$$\begin{array}{rcl}
q(X) & \leftarrow & player(X,Y,Z) \\
q(X) & \leftarrow & team(V,W,X) \\
player_{\mathcal{D}}(X,Y,Z) & \leftarrow & s_1(X,Y,Z,W) \\
team_{\mathcal{D}}(X,Y,Z) & \leftarrow & s_2(X,Y,Z) \\
team_{\mathcal{D}}(X,Y,Z) & \leftarrow & s_3(X,Y,Z) \\
coach_{\mathcal{D}}(X,Y,Z) & \leftarrow & s_4(X,Y,Z) \\
player(X,Y,Z) & \leftarrow & player_{\mathcal{D}}(X,Y,Z) \ , \ not \ \overline{player}(X,Y,Z) \\
\overline{player}(X,Y,Z) & \leftarrow & player(X,V,W) \ , \ player_D(X,Y,Z) \ , \ Y \neq V \\
\overline{player}(X,Y,Z) & \leftarrow & player(X,V,W) \ , \ player_D(X,Y,Z) \ , \ Z \neq W \\
team(X,Y,Z) & \leftarrow & team_{\mathcal{D}}(X,Y,Z) \ , \ not \ \overline{team}(X,Y,Z) \\
\overline{team}(X,Y,Z) & \leftarrow & team(X,V,W) \ , \ team_{\mathcal{D}}(X,Y,Z) \ , \ Y \neq V \\
\overline{team}(X,Y,Z) & \leftarrow & team(X,V,W) \ , \ team_{\mathcal{D}}(X,Y,Z) \ , \ Z \neq W \\
coach(X,Y,Z) & \leftarrow & coach_{\mathcal{D}}(X,Y,Z) \ , \ not \ \overline{coach}(X,Y,Z) \\
\overline{coach}(X,Y,Z) & \leftarrow & coach(X,V,W) \ , \ coach_D(X,Y,Z) \ , \ Y \neq V \\
\overline{coach}(X,Y,Z) & \leftarrow & coach(X,V,W) \ , \ coach_D(X,Y,Z) \ , \ Z \neq W \\
\overline{player}(X,Y,Z) & \leftarrow & player_{\mathcal{D}}(X,Y,Z) \ , \ coach(X,V,W) \\
\overline{coach}(X,Y,Z) & \leftarrow & coach_{\mathcal{D}}(X,Y,Z) \ , \ team(V,W,X) \\
\overline{coach}(X,Y,Z) & \leftarrow & coach_{\mathcal{D}}(X,Y,Z) \ , \ player(X,V,W) \\
\overline{team}(X,Y,Z) & \leftarrow & team_{\mathcal{D}}(X,Y,Z) \ , \ coach(Z,V,W)
\end{array}$$

where $\Pi_{ID}$ is constituted by the first two rules, $\Pi_{\mathcal{MD}}$ by the rules ranging from the 3rd to the 6th, $\Pi_{KD}$ by the rules ranging from the 7th to the 15th, and $\Pi_{ED}$ by the last four rules. Notice that in the rules of $\Pi_{ED}$ we considered also the exclusion dependency $team[Tleader] \cap coach[Ccode] = \emptyset$ that is a logical consequence of the ID and the ED expressed on $\mathcal{G}_0$. ■

[7]Without loss of generality, we assume that $\mathbf{A}$ and $\mathbf{B}$ precede all other attributes in $r$ and $s$, respectively.

Soundness and completeness of our technique, with respect to the problem of computing the certain answers to a user query under the loosely-sound semantics, can be easily proved by considering that: *(i)* the evaluation of $\Pi_{\mathcal{MD}}$ over a source database $\mathcal{D}$ for $\mathcal{I}$ produces $ret(\mathcal{I}, \mathcal{D})$; *(ii)* the maximal subsets of $ret(\mathcal{I}, \mathcal{D})$ that are consistent with $\Sigma_E^* \cup \Sigma_K$ coincide with the stable models of the program $\Pi_{KD} \cup \Pi_{ED}$ evaluated over $ret(\mathcal{I}, \mathcal{D})$ *(iii)* the evaluation of $\Pi_{KD}$ over such subsets returns the certain answers to the user query $Q$ (according to the separation property stated in Theorem 4.8). Therefore, we are able to provide the following theorem.

**Theorem 4.10** *Let $\mathcal{I} = \langle \mathcal{G}, \mathcal{S}, \mathcal{M} \rangle$ be a NKC system, and $Q$ be a query over $\mathcal{I}$. Then, $\Pi_{KD} \cup \Pi_{ED} \cup \Pi_{ID} \cup \Pi_{\mathcal{MD}}$ is a perfect rewriting of $Q$ w.r.t. $\mathcal{I}$.*

## 4.3 Discussion

In this Section we have presented techniques for query rewriting in GAV data integration systems under the strictly-sound and the loosely-sound semantics, in the relational setting, when inclusion, key, and exclusion dependencies are expressed on the global schema. We underline that our approach is strongly intensional, since treatment of constraints is carried out at the query and schema level, and query answering is solved by first computing the perfect rewriting of a query issued on the global schema, and then evaluating the rewritten query. This is an important aspect also for the realization of the INFOMIX system, which implements (among others) our algorithms for query rewriting under the loosely-sound semantics in GAV. Indeed, in the INFOMIX system architecture, components for query processing can be decoupled from the extensional layer devoted to the extraction of the data from the sources, which tipically represent the main bottleneck in data integration applications. With regard to query answering under the strictly-sound semantics, the feasibility of our approach is also witnessed by the fact that, since perfect rewritings are expressed in terms of union of conjunctive queries, they can be easily handled by standard database query evaluation techniques. Furthermore, the tractability of our algorithms can be easily proved, since evaluating a UCQ over a database is in PTIME in data complexity. Finally, our overall technique resorts to more complex rewritings expressed in Datalog$^\neg$ only when needed, i.e., only in the case in which data at the sources are inconsistent with KDs or EDs, and classical methods developed under the strict semantics would not be able to provide significative answers to queries.

# 5 Query Rewriting in LAV Data Integration Systems

We now focus our attention on LAV integration systems, and provide a sound and complete query rewriting technique under the strictly-sound semantics in the presence of integrity constraints. More precisely, our method is based on an off-line compiling technique that, starting from a LAV system specification, produces the specification of a GAV system which is "query-equivalent" to the original LAV system. After such a compilation, we can reuse the query rewriting technique defined for GAV systems also in the LAV case.

We point out that, analogously to the GAV setting, our technique allows for a modular treatment of *(i)* the integration systems in the presence of integrity constraints, through the compilation of LAV into GAV, and of *(ii)* KDs, EDs and IDs, through the separability property of NKCIDs, EDs and KDs. Furthermore, the compilation of a LAV system into a GAV one can be executed off-line, only at each change of the system specification. Based on such a modularity, we exploit both off-line and intensional processing of queries in order to provide an effective approach to the problem of query answering.

In the following, we first address the case in which only IDs and EDs are specified over the global schema, and then consider also the presence of KDs. This is motivated by the fact that the presence of KDs complicates the task of query rewriting in LAV data integration systems. Indeed, we will show that in such a case the necessity arises of resorting to recursive rewritings.

## 5.1 Query Rewriting in the presence of IDs and EDs

We now present a procedure for computing the perfect rewriting of a UCQ $Q$ posed to a LAV integration system under the strictly-sound semantics in the presence of IDs and EDs. The basic idea of our approach is to first compile the LAV system into an equivalent GAV one, and then make use of the algorithm developed for the GAV case. Let us describe separately the above steps.

### 5.1.1 Compiling LAV into GAV

The notion of equivalence between data integration systems is given in terms of queries, and it is the same as in [6]. In particular, given two integration systems $\mathcal{I} = \langle \mathcal{G}, \mathcal{S}, \mathcal{M} \rangle$ and $\mathcal{I}' = \langle \mathcal{G}', \mathcal{S}, \mathcal{M}' \rangle$ over the same source schema $\mathcal{S}$ and such that all relations of $\mathcal{G}$ are also relations of $\mathcal{G}'$, we say that $\mathcal{I}'$ is *query-preserving* with respect to $\mathcal{I}$, if for every query $Q$ over $\mathcal{I}$ and for every source database $\mathcal{D}$ for $\mathcal{I}$, we have that $ans(Q, \mathcal{I}, \mathcal{D}) = ans(Q, \mathcal{I}', \mathcal{D})$. In other words, we say that $\mathcal{I}'$ is query-preserving with respect to $\mathcal{I}$ if, for any query over the global schema of $\mathcal{I}$, and for any source database $\mathcal{D}$, the answers we get for the query on the two integration systems are identical.

Our compiling technique extends a similar procedure presented in [6], which does not allow for query processing and is not able to deal with integrity constraints. Furthermore, in [6] only mapping queries without constants are considered. Here, we improve this technique as follows:

*(i)* we extend the class of LAV systems that can be compiled into GAV ones, allowing the presence of IDs and EDs on the global schema;

*(ii)* we allow mapping queries having constants of the domain $\mathcal{U}$ appearing in the body and/or in the head;

*(iii)* we avoid the use of the *simple equality-generating dependencies* introduced in the transformation of [6] together with IDs: we only make use of IDs where attributes may appear more than once in the left-hand side.

We point out that the simplification indicated at point *(iii)* allows for adopting, in the compiled system, the algorithm ID-rewrite for query processing in GAV under IDs that has been described in Section 4.1. Indeed, such an algorithm is able to deal also with IDs with repetition in their left-hand side[8].

We now present the actual transformation of a LAV system into a query-preserving GAV one. Let $\mathcal{I} = \langle \mathcal{G}, \mathcal{S}, \mathcal{M} \rangle$ be the initial LAV system, with $\mathcal{G} = \langle \Psi, \Sigma_I, \Sigma_E \rangle$, and $\mathcal{I}' = \langle \mathcal{G}', \mathcal{S}, \mathcal{M}' \rangle$, with $\mathcal{G}' = \langle \Psi', \Sigma_I', \Sigma_E \rangle$, be the transformed GAV one. The transformation is performed as follows.

- The set of sources $\mathcal{S}$ remains unchanged.
- The global schema $\mathcal{G}'$ is obtained from $\mathcal{G}$ by introducing in $\Psi'$, for each relation $s/n$ in $\mathcal{S}$:
  - (i) a new relation $\mathsf{image\_}s/(n+n_c)$, where $n_c$ is the number of constants appearing in $body(V_s)$, where $V_s$ is the view associated to $s$[9];
  - (i) a new relation $\mathsf{expand\_}s/(n + n_c + n_n)$, where $n_n$ is the number of non-distinguished variables appearing in $V_s$.
- The GAV mapping $\mathcal{M}'$ associates to each global relation $\mathsf{image\_}s$ the query

  $$\mathsf{image\_}s(X_1, \ldots, X_{n_d}, e_1, \ldots, e_{n_c}) \ \leftarrow \ s(X_1, \ldots, X_{n_d}, c_1, \ldots, c_{n_{c,h}})$$

  where $X_1^h, \ldots, X_{n_d}^h$ are the distinguished variables of $V_s$, $e_1, \ldots, e_{n_c}$ are constants in $body(V_s)$, and $c_1, \ldots, c_{n_{c,h}}$ are constants in $head(V_s)$. We do not associate any query to the remaining global relations.
- The set of inclusion dependencies $\Sigma_I'$ contains the dependencies in $\Sigma_I$ plus the IDs constructed as follows. We suppose to enumerate distinguished variables, constants and non-distinguished variables in $V_s$ as $Z_1, \ldots, Z_{n_d+n_c+n_n}$, where $Z_1, \ldots, Z_{n_d}$ are distinguished variables and $Z_{n_d+1}, \ldots, Z_{n_d+n_c}$ are constants.
  - (i) For each relation $\mathsf{image\_}s/(n + n_c)$ in $\Psi'$ we add the inclusion dependency

    $$\mathsf{image\_}s[1, \ldots, n + n_c] \subseteq \mathsf{expand\_}s[1, \ldots, n + n_c];$$

  - (ii) for each relation $\mathsf{expand\_}s$ in $\Psi'$ and for each atom $g(Z_{i_1}, \ldots, Z_{i_{k+\ell+m}})$ occurring in $body(V_s)$, we add the inclusion dependency

    $$\mathsf{expand\_}s[i_1, \ldots, i_{k+\ell+m}] \subseteq g[1, \ldots, k + \ell + m].$$

    Note that the inclusion dependencies introduced here may have repeated attributes on the left-hand side, since the same variable may occur more than once in each atom.
- The set of EDs $\Sigma_E$ remains unchanged.

[8]Notice that the algorithm ID-rewrite presented in this document extends the one presented in [8] that cannot handle such form of IDs.

[9]Hereinafter $V_s$ indicates the view that the mapping associates to $s$.

It is immediate to verify that, given a LAV integration system $\mathcal{I}$ , the corresponding GAV integration system $\mathcal{I}'$ defined as above can be constructed in time that is linear in the size of $\mathcal{I}$.

**Example 3.1 (contd.)** Consider now the LAV system $\mathcal{I}_L = \langle \mathcal{G}, \mathcal{S}_L, \mathcal{M}_L \rangle$ described in Section 3, and assume that only IDs are specified over $\mathcal{G}$[10]. We also recall here that $\mathcal{S}_L$ comprises the two relations $s_3$ and $s_4$ and that $\mathcal{M}_L$ is as follows:

$$\begin{array}{rcl} s_3(N, Db, S) & \leftarrow & manager(N, Db), employee(N, S, D, N) \\ s_4(E, B) & \leftarrow & employee(E, S, D, B) \end{array}$$

Let us now Compile $\mathcal{I}_L$ into a query equivalent GAV system $\mathcal{I}_{LG} = \langle \mathcal{G}', \mathcal{S}_L, \mathcal{M}' \rangle$, where $\mathcal{G}' = \langle \Psi', \Sigma'_I \rangle$. According to the algorithm above described, $\Psi'$ is obtained by adding to $\Psi$ the relations $\mathsf{image\_}s_3/3$, $\mathsf{image\_}s_4/2$, $\mathsf{expand\_}s_3/4$ and $\mathsf{expand\_}s_4/4$, whereas $\Sigma'_I$ is obtained by adding to $\Sigma_I = \{I_1, I_2, I_3\}$ the following inclusion dependencies

$$\begin{array}{rcll} \mathsf{image\_}s_3[1,2,3] & \subseteq & \mathsf{expand\_}s_3[1,2,3] & (I_4) \\ \mathsf{image\_}s_4[1,2] & \subseteq & \mathsf{expand\_}s_4[1,4] & (I_5) \\ \mathsf{expand\_}s_3[1,2] & \subseteq & manager[1,2] & (I_6) \\ \mathsf{expand\_}s_3[1,3,4,1] & \subseteq & employee[1,2,3,4] & (I_7) \\ \mathsf{expand\_}s_4[1,2,3,4] & \subseteq & employee[1,2,3,4] & (I_8). \end{array}$$

Moreover, the mapping $\mathcal{M}'$ is

$$\begin{array}{rcl} \mathsf{image\_}s_3(N, Db, S) & \leftarrow & s_3(N, Db, S) \\ \mathsf{image\_}s_4(E, B) & \leftarrow & s_4(E, B) \end{array}$$

■

We now prove that our transformation is query-preserving. To this aim we need some preliminary notions and properties that have been given in [20] where query answering in LAV systems under the sound semantics has been addressed. For the sake of completeness we recall here the notions and the theorems that are needed for proving the result that we provide in this section. We omit the proofs of theorems, and refer to [20] for formal demonstrations of these results. Let us start with the notion of retrieved global database for LAV systems without constraints.

**Definition 5.1** Let $\mathcal{I} = \langle \mathcal{G}, \mathcal{S}, \mathcal{M} \rangle$ be a LAV data integration system, and $\mathcal{D}$ a source database for $\mathcal{I}$. The *retrieved global database* $ret_L(\mathcal{I}, \mathcal{D})$ is defined constructively as follows. Consider a mapping assertion $\langle s, V_s \rangle \in \mathcal{M}$ where $s \in \mathcal{S}$ and $V_s$ is of the form

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

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

[10] Since EDs remain unchanged in the transformation, we consider a schema without EDs.

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

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

The following lemma shows that $ret_L(\mathcal{I}, \mathcal{D})$ is a good representative of all databases in the semantics of the system $\mathcal{I}$.

**Lemma 5.2** *Let $\mathcal{I} = \langle \mathcal{G}, \mathcal{S}, \mathcal{M} \rangle$ be a LAV data integration system, with $\mathcal{G} = \langle \Psi, \emptyset \rangle$ and $\mathcal{D}$ a source database for $\mathcal{I}$. We have that for any database $\mathcal{B} \in sem(\mathcal{I}, \mathcal{D})$ there exists a well-defined homomorphism from $ret_L(\mathcal{I}, \mathcal{D})$ to $\mathcal{B}$ that sends constants of the active domain $\mathcal{U}_{\mathcal{D}}$ to themselves and fresh constants to constants of $\mathcal{U}$, and such that each fact of $g^{ret_L(\mathcal{I},\mathcal{D})}$ is sent to a fact of $g^{\mathcal{B}}$.*

By exploiting the above lemma, it is possible to show that, analogously to the GAV case, in LAV systems without constraints the answer to a user query $Q$ can be calculated by evaluating $Q$ over the retrieved global database. Hence, in the case in which no dependency is specified on the global schema, query answering in LAV is analogous to query answering over a single relational schema, modulo the computation of $ret_L(\mathcal{I}, \mathcal{D})$.

With this notion in place, we can apply the results presented in [20], which show that, in the presence of IDs, the answer to a query $Q$ over a database is computed by evaluating $Q$ over the chase of the database[11]. Therefore, the following theorem can be easily proved.

**Theorem 5.3** *Let $\mathcal{I} = \langle \mathcal{G}, \mathcal{S}, \mathcal{M} \rangle$ be a LAV data integration system, with $\mathcal{G} = \langle \Psi, \Sigma_I \rangle$; let $\mathcal{D}$ be a source database for $\mathcal{I}$, and $Q$ a query over $\mathcal{G}$ of arity $n$. Moreover, let $t$ be an $n$-tuple of constants of $\mathcal{U}_{\mathcal{D}}$ Then, we have that $t \in ans(Q, \mathcal{I}, \mathcal{D})$ iff $t \in Q^{chase(\mathcal{G}, ret_L(\mathcal{I},\mathcal{D}))}$.*

Let us finally consider the simultaneous presence of IDs and EDs on the global schema. Also in this setting, the notion of retrieved global database still allows for query answering. Intuitively, this is possible since the introduction of fresh constants in $ret_L(\mathcal{I}, \mathcal{D})$ cannot lead to the violations of exclusion dependencies (constants are never equal), when $ret_L(\mathcal{I}, \mathcal{D})$ is consistent with the closure of EDs w.r.t. logical implication of IDs and EDs, and hence that ED violations can be caused only by the data stored at the sources.

**Theorem 5.4** *Let $\mathcal{I} = \langle \mathcal{G}, \mathcal{S}, \mathcal{M} \rangle$ be a LAV data integration system, with $\mathcal{G} = \langle \Psi, \Sigma_I, \Sigma_E \rangle$; let $\mathcal{D}$ be a source database for $\mathcal{I}$, and $Q$ a query of arity $n$ over $\mathcal{G}$. Moreover, let $t$ be an $n$-tuple of constants of $\mathcal{U}_{\mathcal{D}}$ Then, we have that $t \in ans(Q, \mathcal{I}, \mathcal{D})$ iff $ret_L(\mathcal{I}, \mathcal{D})$ is consistent with $\Sigma_E^*$ and $t \in Q^{chase(\mathcal{G}, ret_L(\mathcal{I},\mathcal{D}))}$.*

[11] The notion of chase has been given in Section 4.1.3

We are now able to prove that a GAV system $\mathcal{I}'$, obtained from a LAV one $\mathcal{I}$ with our technique, is query-preserving w.r.t. $\mathcal{I}$, by showing the equivalence of the chase of the two retrieved global databases.

**Theorem 5.5** *Let $\mathcal{I} = \langle \mathcal{G}, \mathcal{S}, \mathcal{M} \rangle$ be a LAV integration system, with $\mathcal{G} = \langle \Psi, \Sigma_I, \Sigma_E \rangle$ and let $\mathcal{I}' = \langle \mathcal{G}', \mathcal{S}, \mathcal{M}' \rangle$, with $\mathcal{G} = \langle \Psi', \Sigma'_I, \Sigma_E \rangle$ be the corresponding GAV integration system defined as above. Then $\mathcal{I}'$ is query-preserving with respect to $\mathcal{I}$, i.e., for every query $Q$ over $\mathcal{I}$ and for every source database $\mathcal{D}$ for $\mathcal{I}$, we have that $ans(Q, \mathcal{I}, \mathcal{D}) = ans(Q, \mathcal{I}', \mathcal{D})$.*

*Proof.*

Let us initially overlook the EDs in $\Sigma_E$, and consider the two systems $\mathcal{I} = \langle \mathcal{G}, \mathcal{S}, \mathcal{M} \rangle$ and $\mathcal{I}' = \langle \mathcal{G}', \mathcal{S}, \mathcal{M}' \rangle$, in which $\mathcal{G} = \langle \Psi, \Sigma_I \rangle$ and $\mathcal{G}' = \langle \Psi', \Sigma'_I \rangle$. Let $Q$ be a query over $\mathcal{G}$, and $\mathcal{D}$ a source database for $\mathcal{I}$ and $\mathcal{I}'$; we will show that $ans(Q, \mathcal{I}, \mathcal{D}) = ans(Q, \mathcal{I}', \mathcal{D})$.

As for the LAV system $\mathcal{I}$, from Theorem 5.3 it follows that $ans(Q, \mathcal{I}, \mathcal{D}) = Q^{chase(\langle \Psi, \Sigma_I \rangle, ret_L(\mathcal{I}, \mathcal{D}))}$ (where we exclude from the answer set the tuples that contain at least one of the newly introduced constants for the construction of $ret_L(\mathcal{I}, \mathcal{D})$).

Regarding the GAV system $\mathcal{I}'$, we have that $ans(Q, \mathcal{I}', \mathcal{D}) = Q^{chase(\langle \Psi', \Sigma'_I \rangle, ret_G(\mathcal{I}', \mathcal{D}))}$ (we exclude here the answers that contain at least one of the newly introduced constants for the construction of the chase).

Now, let $\Sigma_{new}$ be the IDs introduced in $\mathcal{I}'$ during the transformation, with $\Sigma'_I = \Sigma_I \cup \Sigma_{new}$; observe that $chase(\langle \Psi', \Sigma_{new} \rangle, ret_G(\mathcal{I}', \mathcal{D}))$ is finite, since the dependencies in $\Sigma_{new}$ are acyclic.

Moreover, it is straightforward to see that, modulo renaming of the fresh constants $z_i$ introduced in the construction of the retrieved global database for the LAV system $\mathcal{I}$ and in the construction of the chase for the GAV system $\mathcal{I}'$, we have that, for each relation symbol $g$ in $\mathcal{G}$ (and therefore in $\mathcal{G}'$), $g^{chase(\langle \Psi', \Sigma_{new} \rangle, ret_G(\mathcal{I}', \mathcal{D}))} = g^{ret_L(\mathcal{I}, \mathcal{D})}$. In fact, by applying the ID chase rule for $chase(\langle \Psi', \Sigma_{new} \rangle, ret_G(\mathcal{I}', \mathcal{D}))$, we populate the relations of $\mathcal{G}'$ that are also in $\mathcal{G}$ exactly in the same way in which they are populated in the construction of $ret_L(\mathcal{I}, \mathcal{D})$. The fresh variables are first generated by applying the chase rule for the IDs of the form $\mathsf{image\_}s[1, \ldots, n + n_c] \subseteq \mathsf{expand\_}s[1, \ldots, n + n_c]$, and then propagated to the global relations $g_i$ in the positions where the non-distinguished variables appear. The constants appearing in the body of the LAV mapping queries are properly introduced in the head of the GAV mapping queries, and then propagated to the global relations $g_i$ in the positions where the constants appear. Finally, constants of the source database $\mathcal{D}$, are introduced in $ret_G(\mathcal{I}', \mathcal{D})$ according to the mapping $\mathcal{M}'$, and then propagated to the global relations in the positions where the distinguished variables appear.

Now, observe that, since the dependencies in $\Sigma_I$ do not involve the relations introduced in the transformation from $\mathcal{I}$ to $\mathcal{I}'$, the application of the ID chase rule to $chase(\langle \Psi', \Sigma_{new} \rangle, ret_G(\mathcal{I}', \mathcal{D}))$ is independent of the newly introduced relations. Therefore, for each relation symbol in $\mathcal{G}$ (and also in $\mathcal{G}'$), we have $g^{chase(\langle \Psi, \Sigma_I \rangle, ret_L(\mathcal{I}, \mathcal{D}))} = g^{chase(\langle \Psi', \Sigma_I \rangle, chase(\langle \Psi', \Sigma_{new} \rangle, ret_G(\mathcal{I}', \mathcal{D})))} = g^{chase(\langle \Psi', \Sigma'_I \rangle, ret_G(\mathcal{I}', \mathcal{D}))}$.

Now, let us reintroduce the EDs $\Sigma_E$ both in $\mathcal{I}$ and in $\mathcal{I}'$. The difference with the above situation is that now, in order to exploit the construction of the chase, we need to verify that the retrieved global database satisfies $\Sigma^*_E$ both in the LAV and in the GAV system, according to Theorem 5.4 and Theorem 4.8. more precisely, we have to show that $ret_L(\mathcal{I},\mathcal{D})$ satisfies $\Sigma_E^{*,\Sigma_I\cup\Sigma_E}$ iff $ret_G(\mathcal{I},\mathcal{D})$ satisfies $\Sigma_E^{*,\Sigma'_I\cup\Sigma_E}$, where where $\Sigma'_I\cup\Sigma_E$ (resp. $\Sigma_I\cup\Sigma_E$) indicates the set of constraints with respect to which the logical implication of EDs by IDs and EDs is computed. First of all, we recall that, from results of [20] (see of Lemma 4.4), if a database instance $\mathcal{DB}$ for a relational schema $\mathcal{RS}=\langle\Psi,\Sigma_I,\Sigma_E\rangle$ satisfies $\Sigma_E^{*,\Sigma_I\cup\Sigma_E}$, then $chase(\mathcal{RS},\mathcal{DB})$ satisfies $\Sigma_E$. Conversely, if $\mathcal{DB}$ does not satisfy $\Sigma_E^{*,\Sigma_I\cup\Sigma_E}$, query answering is meaningless, indeed some ED in $\Sigma_E$ should be violated by $chase(\mathcal{RS},\mathcal{D})$. Notice now that EDs in $\Sigma_E$ in $\mathcal{G}'$ involve only global relation $g$ that belong also to $\mathcal{G}$. Hence, since $g^{chase(\langle\Psi,\Sigma_I\rangle,ret_L(\mathcal{I},\mathcal{D}))}=g^{chase(\langle\Psi',\Sigma'_I\rangle,ret_G(\mathcal{I}',\mathcal{D}))}$ it follows that *(a)* if $ret_L(\mathcal{I},\mathcal{D})$ does not satisfy $\Sigma_E^{*,\Sigma_I\cup\Sigma_E}$, then $ret_G(\mathcal{I}',\mathcal{D})$ does not satisfy $\Sigma_E^{*,\Sigma'_I\cup\Sigma_E}$, and *(b)* if $ret_L(\mathcal{I},\mathcal{D})$ satisfies $\Sigma_E^{*,\Sigma_I\cup\Sigma_E}$, then $ret_G(\mathcal{I}',\mathcal{D})$ satisfies $\Sigma_E^{*,\Sigma'_I\cup\Sigma_E}$. Notice that in the case *(a)* query answering is meaningless for both $\mathcal{I}$ and $\mathcal{I}'$, i.e., every tuple is in the answer to any query, whereas in case *(b)* query answering is analogous to the case in which no ED is specified on the global schema, i.e., $ans(Q,\mathcal{I},\mathcal{D})=Q^{chase(\langle\Psi,\Sigma_I\rangle,ret_L(\mathcal{I},\mathcal{D}))}$ and $ans(Q,\mathcal{I}',\mathcal{D})=Q^{chase(\langle\Psi',\Sigma'_I\rangle,ret_G(\mathcal{I}',\mathcal{D}))}$.

Since $Q$ involves only global relations that belong both to $\mathcal{G}$ and $\mathcal{G}'$, and the fresh constants are not considered in the evaluation of $Q$ over $chase(\langle\Psi',\Sigma'_I\rangle,ret_G(\mathcal{I}',\mathcal{D}))$ and over $chase(\langle\Psi,\Sigma_I\rangle,ret_L(\mathcal{I},\mathcal{D}))$, the claim follows immediately. □

### 5.1.2 Query Rewriting in the Presence of IDs and EDs

With the results of the previous section in place, we can now address the problem of computing the answers to a query posed to a LAV data integration system, in the presence of IDs and EDs on the global schema. Since we are able to transform a LAV system into a query-preserving GAV one, we can apply the same technique we developed for GAV systems.

**Theorem 5.6** *Let $\mathcal{I}=\langle\mathcal{G},\mathcal{S},\mathcal{M}\rangle$ be a LAV data integration system, where $\mathcal{G}=\langle\Psi,\Sigma_I,\Sigma_E\rangle$, $Q$ a query over $\mathcal{I}$, and $\mathcal{D}$ a source database for $\mathcal{I}$; moreover, let $\mathcal{I}'=\langle\mathcal{G}',\mathcal{S},\mathcal{M}'\rangle$ be the GAV system obtained by compiling $\mathcal{I}$ into an equivalent GAV system, where $\mathcal{G}'=\langle\Psi',\Sigma'_I,\Sigma_E\rangle$, and let $\Pi_{ID}$ be the UCQ returned by* ID-rewrite$(\langle\Psi',\Sigma'_I\rangle,Q)$ *and $\Pi_{\mathcal{M}'}$ the program $\{V_g\mid\langle g,V_g\rangle\in\mathcal{M}'\}$. Then, we have that $\Pi_{ID}\cup\Pi_{\mathcal{M}'}=ans(Q,\mathcal{I},\mathcal{D})$ iff $ret_L(\mathcal{I},\mathcal{D})$ is consistent with $\Sigma_E^{*,\Sigma_I\cup\Sigma_E}$, where $\Sigma_E^{*,\Sigma_I\cup\Sigma_E}$ indicates the set of EDs that are logically implied by the dependencies in $\Sigma_I\cup\Sigma_E$.*

*Proof.* The theorem is an immediate consequence of Theorem 5.5 and Theorem 4.5. □

**Example 3.1 (contd.)** Consider again the query $Q$ that contains the only CQ $q_0$ of the form $q(Dr)\leftarrow department(C,Dr),employee(Dr,S,D,B)$ asking for directors of departments that are also employees. After compiling the LAV system $\mathcal{I}_L=\langle\mathcal{G},\mathcal{S}_L,\mathcal{M}_L\rangle$ into the query-preserving GAV system $\mathcal{I}_{LG}=\langle\mathcal{G}',\mathcal{S}_L,\mathcal{M}'\rangle$, we are able to compute the answers to the query $Q$, by executing ID-rewrite$(\langle\Psi',\Sigma'_I\rangle,Q)$[12]. At the beginning,

[12] Since no ED is specified over $\mathcal{G}$ the retrieved global database of $\mathcal{I}_L$ trivially satisfies the EDs.

$Q_{aux} = Q' = Q = \{q_0\}$, and inclusion dependencies $I_1, I_2, I_3$ are applicable to the atoms of the queries in $Q'$ exactly as performed by the execution of $\mathsf{ID\text{-}rewrite}(\langle\Psi, \Sigma_I\rangle, Q)$ described in Section 4.1. Hence, we can describe the execution of $\mathsf{ID\text{-}rewrite}(\langle\Psi', \Sigma'_I\rangle, Q)$ starting from the output of $\mathsf{ID\text{-}rewrite}(\langle\Psi, \Sigma_I\rangle, Q)$, i.e., when $Q' = \{q_0, q_1, q_2, q_3\}$.

As an example consider $q_3$ and its unique atom $g_3 = employee(Dr, \xi, \xi, Dr)$. It is easy to see that $I_7$ is applicable to the atom in $g_3$ and that $gr(I_7, g_3) = \mathsf{expand\_}s_3(Dr, \xi, \xi, \xi)$. Then, by applying $I_4$ to such an atom we obtain $\mathsf{image\_}s_3(Dr, \xi, \xi)$ to which no ID in $\Sigma'_I$ is applicable. The algorithm proceeds analogously for the other queries in $Q'$. The result of the computation is

$$\begin{array}{rcl}
q(Dr) & \leftarrow & \mathsf{image_}s_3(Dr, Db, S) \\
q(Dr) & \leftarrow & \mathsf{image_}s_3(Dr, Db, S), \mathsf{image_}s_4(Dr, B) \\
q(Dr) & \leftarrow & \mathsf{image_}s_4(Dr, Dr) \\
q(Dr) & \leftarrow & \mathsf{image_}s_3(Dr, Db, S), \mathsf{image_}s_4(E, Dr) \\
\mathsf{image_}s_3(N, Db, S) & \leftarrow & s_3(N, Db, S) \\
\mathsf{image_}s_4(E, B) & \leftarrow & s_4(E, B)
\end{array}$$

■

## 5.2 Query Rewriting in the Presence of KDs

We conclude this section by considering query rewriting for LAV systems in the presence of NKCIDs, KDs and EDs, i.e., NKC LAV systems. As widely discussed in [20], query answering in this setting is decidable, and it is possible to separately deal with IDs and KDs. In such a deliverable, in order to take into account the presence of KDs on the global schema, we have introduced a new definition of retrieved global database which refines Definition 5.1. Roughly speaking, the retrieved global database in NKC LAV systems, denoted with $ret_{LK}(\mathcal{I}, \mathcal{D})$, is obtained starting from the database $ret_L(\mathcal{I}, \mathcal{D})$, by repeatedly applying the so-called KD-RULE that unifies fresh constants, i.e., the new constant symbols introduced in the construction of $ret_L(\mathcal{I}, \mathcal{D})$ (see [20] for more details). Hence, an actual procedure for processing a query $Q$ over a NKC LAV system $\mathcal{I} = \langle\mathcal{G}, \mathcal{S}, \mathcal{M}\rangle$, where $\mathcal{G} = \langle\Psi, \Sigma_I, \Sigma_K, \Sigma_E\rangle$, consists of computing $\mathsf{ID\text{-}rewrite}(\Psi, \Sigma_I, Q)$, then computing $ret_{LK}(\mathcal{I}, \mathcal{D})$, and finally evaluating $\mathsf{ID\text{-}rewrite}(\Psi, \Sigma_I, Q)$ over $ret_{LK}(\mathcal{I}, \mathcal{D})$. If $ret_{LK}(\mathcal{I}, \mathcal{D})$ is not consistent with $\Sigma_K$, then query answering is meaningless, i.e., every tuple is in the answer of any query.

Comparing the two query processing techniques presented for GAV and LAV systems with KDs and IDs, we note that, while in the GAV case we can evaluate the query $\Pi_{ID} \cup \Pi_{\mathcal{M}}$ over the source database $\mathcal{D}$, where $\Pi_{ID}$ is the output of $\mathsf{ID\text{-}rewrite}$ and $\Pi_{\mathcal{M}}$ is the program associated with the mapping, in the LAV case we necessarily have to evaluate $\Pi_{ID}$ over $ret_{LK}(\mathcal{I}, \mathcal{D})$. This procedure cannot be considered as a rewriting technique, because of the need of computing $ret_{LK}(\mathcal{I}, \mathcal{D})$, which forces us to heavily deal with data, i.e., extensions of the sources.

Now the question arises if, also in the presence of KDs, it is possible to first compile a LAV system into a query equivalent GAV one, and then proceed as GAV, i.e., to have a sound and complete query rewriting algorithm. However, such a possibility does not actually hold, since in LAV systems with KDs it is not possible to obtain a perfect rewriting of a UCQ expressed in terms of a UCQ.

Indeed, through the following example we prove that *recursive* queries are needed to answer a UCQ in a LAV system with key dependencies on the global schema. The example is adapted from the one used in [10] to show that a recursive query rewriting is needed in the presence of *functional dependencies*, i.e., a more general class of dependencies than KDs.[13]

**Example 5.7** Let $\mathcal{I} = \langle \mathcal{G}, \mathcal{S}, \mathcal{M} \rangle$ be a system in which $\mathcal{G}$ comprises the relations $pilot(Name, Airline)$ and $aircraft(Code, Airline)$ where the key dependencies $key(pilot) = \{1\}$, and $key(aircraft) = \{1\}$ are expressed on $\mathcal{G}$. Consider the source relation $s(Pilot, Aircraft)$ and the LAV mapping $s(P, C) \leftarrow pilot(P, A), aircraft(C, A)$. Finally, let us pose the following query $Q$ on the global schema: $q(P) \leftarrow pilot(\mathsf{Mike}, A), pilot(P, A)$, asking for pilots of the same company of Mike. Since $s$ does not maintain information about which airlines pilots work for, it is not possible to answer the query disregarding the KDs on $\mathcal{G}$. Moreover, it can be shown that no UCQ is a perfect rewriting of the query, and that the perfect rewriting must necessarily be expressed in terms of a recursive query: for instance, the following recursive query is a perfect rewriting of $Q$:

$$\begin{array}{lcl} q(P) & \leftarrow & s(\mathsf{Mike}, C), s(P, C) \\ q(P) & \leftarrow & q(P_1), s(P_1, C), s(P, C) \end{array}$$

■

### 5.3 Discussion

In this section we have addressed the problem of query answering in LAV integration systems, and have presented techniques for query answering, when key, inclusion and exclusion dependencies are expressed on the global schema. In the presence of only IDs and EDs our approach is purely intensional, since query answering is solved by first computing the perfect rewriting of a query issued on the global schema, and then evaluating the rewritten query. On the other hand, in the presence of KDs, we resort to query answering, i.e., we use constructively data at the sources in order to provide answers to the user query. As said in the above subsection, this is needed in order to take into account KDs in the construction of the retrieved global database, which may force fresh constants introduced during such a process to be equal.

## 6 Related Work

Several works in the literature address the problem of query rewriting in data integration in the presence of integrity constraints on the global schema. In this respect, query rewriting under integrity constraints has been first studied in the LAV setting. In particular, the inverse rules algorithm presented in [10] suitably deals with functional dependencies

[13] Actually, [10] refers to maximally contained rewritings: however, when queries in the mapping are conjunctive queries, the maximally-contained rewriting coincides with the perfect rewriting.

expressed over the global schema. The algorithm computes the perfect rewriting in the case of queries and mapping expressed in terms of conjunctive queries, and "maximally contained" rewritings in the case of recursive queries[14]. It is worth noticing that, even in the case in which no integrity constraint is specified over the global schema, the perfect rewriting produced by the algorithm in [10] is a logic program with Skolem functions representing existential variables in the head of the rules. In order to evaluate such a rewriting over a database, a non-trivial procedure to eliminate Skolem function from the program has to be performed for each query issued on the system. Conversely, by applying our technique for query rewriting in LAV, which is based on the transformation of the system into an equivalent GAV one, we directly obtain a UCQ that can be easily handled without resorting to other pre-processing procedures. Therefore, although developed for dealing with integrity constraints, our technique for query rewriting in LAV turns out to be simpler than the Duschka inverse rules algorithm [10], even in the absence of inclusion and exclusion dependencies. Furthermore, as pointed out in Section 5, the compilation of a LAV system into a GAV one can be executed off-line, only at each change of the system specification.

Then, in [15] Gryz analyzes query rewriting under inclusion dependencies in LAV systems. Moreover, a method is presented which is able to deal simultaneously with acyclic IDs and functional dependencies. It is worth noticing that, in that paper, an algorithm for computing the rewriting of a conjunctive query in a database with inclusion dependencies is presented. The algorithm is based on a brilliant intuition, i.e., rewriting a conjunctive query by computing the rewriting of each atom in a way "almost independent" of the other atoms. This can be obtained if the body of the initial query $q$ is preliminarily transformed ("minimized"). However, we have found out that Gryz's algorithm does not actually compute the perfect rewriting, in the sense that some conjunctions of the perfect rewriting are missing. Indeed, the algorithm ID-rewrite presented in Section 4.1 is certainly inspired by Gryz's main intuition, but overcomes the above mentioned incompleteness through a new technique for generating the conjunctions of the rewriting.

In GAV integration systems, the role played by integrity constraints for query processing has been in general overlooked. A first attempt to deal with integration under constraints in the GAV approach has been addressed in [7]. Compared to our approach, this work is able to handle a more restricted class of integrity constraints (KDs and foreign key dependencies). Moreover, such an approach is technically more involved than the one presented in this paper, since it makes use of Skolem functions and SLD-refutation for computing the perfect rewriting of a CQ[15].

Moreover, [11] presents an approach for dealing with integrity constraints in the context of data exchange, where the mapping is GLAV (a generalization of LAV and GAV). However, in data exchange the global schema (called *target schema*) has to be materialized, therefore the class of dependencies that is considered in this work (and that is incomparable with the classes we consider in this paper) is such that the chase of the retrieved global database is always finite.

We point out that all the above mentioned approaches do not address the problem of inconsistent data, hence they can be compared only with our algorithms for query

[14]See Deliverable D5.1 [18] for a description of the inverse rules algorithm and its extensions.
[15]See Deliverable D5.1 [18] for more details on this paper.

answering under the strictly-sound semantics.

On the other hand, works that deal with data inconsistencies have been developed mainly in a context of a single database. Furthermore, techniques for computing consistent answers proposed so far in the literature are able to deal only with universally quantified constraints, or treat existentially quantified constraints under restrictive assumptions [2, 14, 3]. Furthermore, they consider repair semantics that are different from the loosely-sound semanitcs considered in this document. A more detailed description of these works has been presented in [18].

More recently, data inconsistency in a purely data integration setting has been studied in [4] and [5]. Differently from our approach, both these papers address a LAV framework and hence their results cannot be directly compared with the technique described in this report. Furthermore, as discussed in Deliverable D5.1 [20], the semantics proposed in [4] and [5] is different from the loose semantics for LAV introduced in Section 4.2.1.

In more detail, [4] extends to LAV integration systems the method for consistent query answering originally presented in [2] in a single database context. Such result is achieved by combining the query reformulation technique of [2] with the Duschka inverse rules algorithm [10], that is extended here to the case in which the query over the global database is a non-recursive Datalog$^{\neg}$ query. The obtained rewriting is shown to be a maximally contained rewriting. As in [2], only binary universally quantified constraints and non-existentially quantified global queries are considered.

Conversely, [5] takes into account a very general class of constraints and queries, namely almost all constraints expressed as first order logic formulae, and stratified Datalog$^{\neg}$ queries. However, the authors introduce the assumption that the constants used for repairing are only the constants in the active domain, i.e., the constants occurring in the source database. Hence, they consider only the cases in which the chase is finite. Repairs in [5] can be obtained from the stable models of a suitable Datalog$^{\vee,\neg}$ program, i.e., Datalog$^{\neg}$ where disjunction is allowed in the head of the rules.

In conclusion, we point out that none of the above mentioned works provides a solution to the query answering problem for the case of cyclic inclusion dependencies under the semantics (both strict and loose) considered in this document.

# 7 Conclusions

In this document we have provided techniques for query rewriting within the INFOMIX framework. In particular, for the GAV framework we have defined algorithms for query rewriting under the sound and the loosely-sound semantics, in the presence of inclusion, key, and exclusion dependencies expressed over the global schema of data integration systems. For the LAV approach, we have focused on the sound semantics, and have provided a query rewriting algorithm for the case in which IDs and EDs are specified on the global schema, and discussed the problems arising in the presence of KDs. Interestingly, our technique in LAV is based on an off-line transformation of a LAV system into a query-equivalent GAV one, that allows us to re-apply in the LAV framework some of the techniques developed for the GAV case.

To the best of our knowledge, our studies on query rewriting under the sound and loosely-sound semantics, which properly capture the incomplete and inconsistent nature of data stored at the sources of data integration systems, are the first studies that provide a thorough analysis of intensional data integration in the presence of KDs, EDs, and cyclic IDs.

## References

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

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

[3] Pablo Barceló and Leopoldo Bertossi. Reapairing databases with annotated predicate logic. In *Proc. the 10th Int. Workshop on Non-Monotonic Reasoning (NMR 2002)*, pages 160–170, 2002.

[4] 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.

[5] 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)*, pages 10 – 15, 2003.

[6] Andrea Calì, Diego Calvanese, Giuseppe De Giacomo, and Maurizio Lenzerini. On the expressive power of data integration systems. In *Proceedings of the Twentyfirst International Conference on Conceptual Modeling (ER 2002)*, 2002.

[7] Andrea Calì, Diego Calvanese, Giuseppe De Giacomo, and Maurizio Lenzerini. Data integration under integrity constraints. *Information Systems*, 29:147–163, 2004.

[8] 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)*, pages 16–21, 2003.

[9] 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.

[10] Oliver M. Duschka, Michael R. Genesereth, and Alon Y. Levy. Recursive query plans for data integration. *Journal of Logic Programming*, 43(1):49–73, 2000.

[11] Ronald Fagin, Phokion G. Kolaitis, Renée J. Miller, and Lucian Popa. Data exchange: Semantics and query answering. In *Proceedings of the Ninth International Conference on Database Theory (ICDT 2003)*, pages 207–224, 2003.

[12] 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.

[13] 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.

[14] 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.

[15] Jarek Gryz. Query rewriting using views in the presence of functional and inclusion dependencies. *Information Systems*, 24(7):597–612, 1999.

[16] 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.

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

[18] Domenico Lembo, Maurizio Lenzerini, and Riccardo Rosati. Review on models and systems for information integration. Technical Report D1.1, Infomix Consortium, July 2002.

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

[20] Domenico Lembo, Maurizio Lenzerini, and Riccardo Rosati. Methods and techniques for query rewriting. Technical Report D5.1, Infomix Consortium, October 2003.

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

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

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