Query rewriting and answering under constraints in data integration systems

Andrea Call Domenico Lembo Riccardo Rosati
Dipartimento di Informatica e Sistemistica
Universita di Roma "La Sapienza"
Via Salaria 113,1-00198 Roma, Italy
{cali,lembo, rosati}@dis.uniromal.it

Abstract

In this paper we address the problem of query answering and rewriting in global-as-view data integration systems, when key and inclusion dependencies are expressed on the global integration schema. In the case of *sound* views, we provide sound and complete rewriting techniques for a maximal class of constraints for which decidability holds. Then, we introduce a semantics which is able to cope with violations of constraints, and present a sound and complete rewriting technique for the same dec idable class of constraints. Finally, we consider the decision problem of query answering and give decidability and complexity results.

1 Introduction

The task of a data integration system is to combine data residing at different sources, providing the user with a unified view of them, called *global schema.* User queries are formulated over the global schema, and the system suitably queries the sources, providing an answer to the user, who is not obliged to have any information about the sources. The problem of data integration is a crucial issue in many application domains, e.g., re-engineering legacy systems, data warehousing, data mining, data exchange.

A central aspect of query processing is the specification of the relationship between the global schema and the sources; such a specification is given in the form of a so-called *mapping.* There are basically two approaches for specifying the mapping. The first approach, called *global-as-view* (GAV), requires that a view over the sources is associated with every element of the global schema. Conversely, the second approach, called *local-as-view* (LAV), requires the sources to be defined as views over the global schema [Lenzerini, 2002; Duschka and Levy, 1997].

The global schema is a representation of the domain of interest of the data integration system: integrity constraints are expressed on such a schema to enhance its expressiveness, thus improving its capability of representing the real world.

Since sources are in general autonomous, the data provided by the sources are likely not to satisfy the constraints on the global schema. Integrity constraints have to be taken into account during query processing; otherwise, the system may return incorrect answers to the user iFagin *et al.,* 2003; Call *et al,* 2002].

Another significant issue is that the sources may not provide exactly the data that satisfy the corresponding portion of the global schema; in particular, they may provide either a subset or a superset of the data satisfying the mentioned portion, and the mapping is to be considered *sound* or *complete* respectively. Mappings that are both sound and complete are called *exact.*

In this paper, we restrict our analysis to the GAV approach, which is the most used in the context of data integration. In particular, we study a relational data integration framework in which key dependencies (KDs) and inclusion dependencies (IDs) are expressed on the global schema, and the mapping is considered sound. The main contributions of this paper are the following:

1. After showing that query answering in the general case in undecidable, we provide a sound and complete query rewriting technique first for the case of IDs alone, and then for the case of KDs together with the maximal class of IDs for which the problem is decidable, called *non-key-conflicting IDs,* or simply NKCIDs (Section 3).
2. Since it is likely that data retrieved at different, autonomous sources violate the KDs, we introduce a novel semantics that is a "relaxation" of the sound semantics, and that allows minimal repairs of the data (Section 4). We then present a sound and complete query rewriting technique in the case where KDs and NKCIDs are expressed on the global schema (Section 5).
3. Finally, we present decidability and complexity results of the (decision) problem of query answering in the different cases (Section 6).

2 Formal framework for data integration

In this section we define a logical framework for data integration, based on the relational model with integrity constraints.

Syntax We consider to have an infinite, fixed alphabet T of constants (also called values) representing real world objects, and will take into account only databases having *T* as domain. We adopt the so-called *unique name assumption,* i.e., we assume that different constants denote different objects.

Formally, a data integration system I is a triple (G, S, *M),* where:

1. G is the *global schema* expressed in the relational model with integrity constraints. In particular, G = $\langle \Psi, \Sigma_I, \Sigma_K \rangle$, where *(i)* Ψ is a set of relations, each with an associated arity that indicates the number of its attributes. The attributes of a relation r of arity n are represented by the integers $1, \ldots, n$. *(ii)* Σ_I is a set of *inclusion dependencies* (IDs), i.e. a set of assertions of the form $r_1[\mathbf{A}] \subseteq r_2[\mathbf{B}]$, where r_1, r_2 are relations in Ψ, $\mathbf{A} = A_1, \ldots, A_n$ $(n \geq 0)$ is a sequence of attributes of r_1, and $\mathbf{B} = B_1, \ldots, B_n$ is a sequence of attributes of r_2. *(iii)* Σ_K is a set of *key dependencies* (KDs), i.e., a set of assertions of the form $key(r) = \mathbf{A}$, where r is a relation in the global schema, and $\mathbf{A} = A_1, \ldots, A_n$ is a sequence of attributes of r such that for each $i \in \{1, \ldots, n-1\}$ $a_i < a_{i+1}$. We assume, without loss of generality, that the attributes in A are the first *n* attributes of *r.* Moreover, we assume that at most one KD is specified for each relation.
2. *S* is the *source schema,* constituted by the schemas of the various sources that are part of the data integration system. We assume that the sources are relational, and that integrity constraints expressed on *S* are satisfied data at the sources. Hence, we do not take such constraints into account in our framework.
3. *M* is the *mapping* between the global and the source schema. In our framework the mapping is defined in the GAV approach, i.e., each relation in Ψ is associated with a *view,* i.e., a query, over the sources. We indicate the mapping as a set of assertions of the form (r, V), where *r* is a relation and *V* is the associated view over the source schema. We assume that the language used to express queries in the mapping is *positive Datalog* [Abiteboul *et al,* 1995], over the alphabet of the relation symbols in S. A Datalog query (or program) q of arity *n* is a collection of rules of the form $h(\vec{\mathbf{x}}) \leftarrow conj(\vec{\mathbf{x}}, \vec{\mathbf{y}})$, where $conj(\vec{\mathbf{x}}, \vec{\mathbf{y}})$ is a set of atoms whose predicate symbols are either relation symbols in S or the head symbol h, and involve $\vec{\mathbf{x}} = X_1, \ldots, X_n$ and $\vec{\mathbf{y}} = Y_1, \ldots, Y_m$, where X_i and Y_j are either variables or values of Γ. We call $h(\vec{\mathbf{x}})$ the *head* of the rule, and $conj(\vec{\mathbf{x}}, \vec{\mathbf{y}})$ the *body*.

Finally, a *query* over the global schema *q* is a formula that is intended to extract a set of tuples of elements of T. The language used to express queries over G is *union of conjunctive queries* (UCQ) [Abiteboul *et al,* 1995], i.e., a Datalog program such that each rule head uses the same predicate of the same arity, and only relation symbols of G occur in each rule body.

Semantics A *database instance* (or simply *database*) $\mathcal{C}$ for a relational schema $\mathcal{DB}$ is a set of facts of the form $r(t)$ where r is a relation of arity n in $\mathcal{DB}$ and t is an n-tuple of values of the domain alphabet Γ. We denote as $r^{\mathcal{C}}$ the set $\{t \mid r(t) \in \mathcal{C}\}$; moreover, given a Datalog query q, we denote as $q^{\mathcal{C}}$ the evaluation of q over $\mathcal{C}$, i.e., the minimal fixpoint model of q and $\mathcal{C}$ [Abiteboul *et al.*, 1995].

In order to specify the semantics of a data integration system I, we start by considering a *source database* for I, i.e., a database P for the source schema *S.* Based on P, we now specify which is the information content of the global schema G. We call *global database* for I any database for G. Formally, given a source database P for I = *(G,S, M),* the semantics of I wrt P, denoted *sern(I,D),* is a set of global databases for I, where a global database *B* is in *sem(I,V)* if:

1. $\mathcal{B}$ is consistent with $\mathcal{G}$, i.e., it satisfies the IDs in Σ_I and the KDs in Σ_K. More formally: *(i)* $\mathcal{B}$ satisfies an inclusion dependency $r_1[\mathbf{A}] \subseteq r_2[\mathbf{B}]$ if for each tuple t_1 in $r_1^{\mathcal{B}}$ there exists a tuple t_2 in $r_2^{\mathcal{B}}$ such that $t_1[\mathbf{A}] = t_2[\mathbf{B}]$, where $t[\mathbf{A}]$ is the projection of the tuple t over $\mathbf{A}$. If $\mathcal{B}$ satisfies all inclusion dependencies expressed on $\mathcal{G}$ we say that $\mathcal{B}$ is consistent with Σ_I; *(ii)* $\mathcal{B}$ satisfies a key dependency $key(r) = \mathbf{A}$ if for each $t_1, t_2 \in r^{\mathcal{B}}$ with $t_1 \neq t_2$ we have $t_1[\mathbf{A}] \neq t_2[\mathbf{A}]$. If $\mathcal{B}$ satisfies all key dependencies expressed on $\mathcal{G}$ we say that $\mathcal{B}$ is consistent with Σ_K.
2. $\mathcal{B}$ satisfies the mapping $\mathcal{M}$ wrt $\mathcal{D}$, i.e., it satisfies each pair $\langle r, V \rangle$ in $\mathcal{M}$ wrt $\mathcal{D}$. In particular, we say that $\mathcal{B}$ satisfies the pair $\langle r, V \rangle$ wrt $\mathcal{D}$, if all the tuples satisfying V in $\mathcal{D}$ satisfy r in $\mathcal{B}$, i.e. $V^{\mathcal{D}} \subseteq r^{\mathcal{B}}$. Note that the above definition amounts to consider any view V as *sound*, which means that the data retrieved from sources satisfy the global schema, but are not necessarily complete.

By simply evaluating each view over the source database P, we obtain a global database, called *retrieved global database ret (I,d*), that actually satisfies the sound mapping (but that is not necessarily consistent with G*).*

We give now the semantics of queries. Formally, given a source database $\mathcal{D}$ for $\mathcal{I}$ we call *answers* to a query q of arity n w.r.t. $\mathcal{I}$ and $\mathcal{D}$, the set $ans(q, \mathcal{I}, \mathcal{D})$ defined as follows: $ans(q, \mathcal{I}, \mathcal{D}) = \{\langle c_1, \ldots, c_n \rangle \mid \text{for each } \mathcal{B} \in sem(\mathcal{I}, \mathcal{D}), \langle c_1, \ldots, c_n \rangle \in q^{\mathcal{B}}\}$.

In this paper, we address the query answering problem, that is the problem of computing the set *ans(q,I,D).* To this aim, we make use of query rewriting techniques, i.e., we exploit the mapping *M* 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. $\mathcal{I}$ if $q_r^{\mathcal{D}} = ans(q, \mathcal{I}, \mathcal{D})$ for each $\mathcal{D}$. Furthermore, with regard to decidability and complexity results, we will refer to the decision problem associated to query answering, that is, given a data integration system $\mathcal{I} = \langle \mathcal{G}, \mathcal{S}, \mathcal{M} \rangle$, a source database $\mathcal{D}$, a query q of arity n over $\mathcal{G}$ and a n-tuple $\bar{t}$ of values of Γ, to establish whether $\bar{t} \in ans(q, \mathcal{I}, \mathcal{D})$.

Example 2.1 Consider a data integration system $\mathcal{I}_0 = \langle \mathcal{G}_0, \mathcal{S}_0, \mathcal{M}_0 \rangle$, referring to the context of football teams. The global schema $\mathcal{G}_0$ consists of the relation predicates $player(Pname, Pcountry, Pteam)$ and $team(Tacronym, Tname, Tleader)$, and the following constraints: $key(player) = \{Pname\}$, $key(team) = \{Tacronym\}$, $team[Tleader] \subseteq player[Pname]$.

The source schema *So* consists of the schemas of three sources comprising the relation s_1 of arity 4, and the relations s_2 and ,s_3, both of arity 3. Finally, the mapping M_0 is defined by the two assertions

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

Consider the source database $\mathcal{D}_0 = \{s_1(Totti, ITA, RM, 27), s_1(Beckham, ENG, MU, 28), s_2(RM, Roma, Totti), s_3(MU, Man.Utd., Giggs)\}$. Then, $ret(\mathcal{I}_0, \mathcal{D}_0) = \{player(Totti, ITA, RM), player(Beckham, ENG, MU), team(RM, Roma, Totti), team(MU, Man.Utd., Giggs)\}$. Notice that the facts in $ret(\mathcal{I}_0, \mathcal{D}_0)$ together with the foreign key constraint $team[Tleader] \subseteq player[Pname]$ impose that *Giggs* is a player. Since the views are sound, the semantics for the integration system has to account for all the global databases that provide the country and the team of the player. Hence, $sem(\mathcal{I}_0, \mathcal{D}_0)$ contains all database instances that can be obtained by adding to $ret(\mathcal{I}_0, \mathcal{D}_0)$ (among others) at least one fact of the form $player(Giggs, \alpha, \beta)$, where α and β are values of the domain Γ. Given the query $q(X) \leftarrow player(X, Y, Z)$, we have that $ans(q, \mathcal{I}_0, \mathcal{D}_0) = \{Totti, Beckham, Giggs\}$. ∎

3 Query rewriting

In this section we present algorithms for computing the perfect rewriting of a UCQ query in GAV integration systems with KDs and IDs. We first study the case in which only IDs are expressed on the global schema, then we deal with the simultaneous presence of both IDs and KDs.

Query rewriting under IDs only We start by studying query rewriting when only IDs are expressed on the global schema. To this aim, we need some preliminary definitions.

Given a conjunctive query *q,* we say that a variable A" 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.

In the following, we assume that all unbound variables in the query q are represented by the special term ξ.

Definition 3.1 Given an atom $g = s(X_1, \ldots, X_n)$ and an inclusion $I = r[i_1, \ldots, i_k] \subseteq s[j_1, \ldots, j_k]$, we say that I *is applicable to* g if, for each ℓ such that $1 \leq \ell \leq n$, if $X_\ell \neq \xi$ then there exists h such that $j_h = \ell$. Moreover, we denote with $gr(g, I)$ the atom $s(Y_1, \ldots, Y_m)$ (m is the arity of s in Ψ) where for each ℓ such that $1 \leq \ell \leq m$, $Y_\ell = X_{j_h}$ if there exists h such that $i_h = \ell$, otherwise $Y_\ell = \xi$.

Roughly speaking, an inclusion / is applicable to an atom *g* if the relation symbol of *g* corresponds to the symbol in the right-hand side of / and if all the attributes for which bound terms appear in *g* are propagated by the inclusion *1.* When *I* is applicable to *g, gr(g, I)* denotes the atom obtained from *g* by using / as a rewriting rule whose direction is right-to-left.

Definition 3.2 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 for each i such that $1 \leq i \leq n$, either $X_i = Y_i$ or $X_i = \xi$ or $Y_i = \xi$. Moreover, if g_1 and g_2 unify, we denote as $U(g_1, g_2)$ the atom $r(Z_1, \ldots, Z_n)$ where, for each i, if $X_i = Y_i$ or $Y_i = \xi$ then $Z_i = X_i$, otherwise $Z_i = Y_i$.

Informally, two atoms unify if they can be made equal through a substitution of each instance of the special symbol ξ with other terms.

Below we define the algorithm ID-rewrite to compute the perfect rewriting of a union of conjunctive queries *Q.* Informally, the algorithm computes the closure of the set of conjunctive queries Q with respect to the following two rules:
(i) if there exists a query $q \in Q$ such that $body(q)$ contains two atoms g_1 and g_2 that unify, then the algorithm computes the query $reduce(q, g_1, g_2)$, which is obtained from q by replacing g_1 and g_2 with $U(g_1, g_2)$ in the query body, and then by applying the substitution obtained in the computation of $U(g_1, g_2)$ to the whole query. Such a new query is then transformed by the function τ, which replaces with ξ each variable symbol X such that there is a single occurrence of X in q. The use of τ is necessary in order to guarantee that each unbound variable is represented by the symbol ξ. Such a query is then added to Q.
(ii) if there exists an inclusion I and a 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 $gr(g, I)$ in its body (denoted in the algorithm as $q[g/gr(g, I)]$). Namely, this step adds new conjunctions obtained by applying inclusion dependencies as rewriting rules (applied from right to left).

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

Algorithm ID-rewrite(Ψ, Σ_I, q)
Input: relational schema Ψ, inclusion dependencies Σ_I,
union of conjunctive queries 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 $Q' := Q' \cup \{\tau(reduce(q, g_1, g_2))\}$;
  (b) **for each** $g \in body(q)$ **do**
    for each $I \in \Sigma_I$ **do**
      if I is applicable to g
      then $Q' := Q' \cup \{ q[g/gr(g, I)] \}$
until $Q_{aux} = Q'$;
return Q'

Termination of the algorithm is immediately implied by the fact that the number of conjunctions that can be generated by the algorithm is finite, since the maximum length of a generated conjunction is equal to the maximum length of a conjunction in the body of the initial query (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 Σ_I), as well as the set of terms used (corresponding to the set of variable and constant names occurring in the query Q plus the symbol ξ).

Henceforth, we denote as Π_{ID} the UCQ returned by ID-rewrite(Ψ, Σ_I, Q). Moreover, we define the Datalog program $\Pi_{\mathcal{M}} = \{V | \langle r, V \rangle \in \mathcal{M}\}$.

Theorem 3.3 *Let $\mathcal{I} = \langle \mathcal{G}, \mathcal{S}, \mathcal{M} \rangle$ be an integration system and let Q be a UCQ query over $\mathcal{G}$. Then, $\Pi_{ID} \cup \Pi_{\mathcal{M}}$ is a perfect rewriting of Q w.r.t. $\mathcal{I}$.*

Query rewriting under KDs and IDs Now we address the problem of query rewriting in the case where KDs and IDs are defined on the global schema. Unfortunately, KDs and IDs interact reciprocally so that the (decision) problem of query answering in this setting becomes undecidable. The following theorem is a consequence of a similar property proved in [Calì *et ai,* 2003] in the context of a single database.

Theorem 3.4 *Consider a data integration system* $\mathcal{I} = \langle \mathcal{G}, \mathcal{S}, \mathcal{M} \rangle$, *with* $\mathcal{G} = \langle \Psi, \Sigma_I, \Sigma_K \rangle$, *where* Σ_I *and* Σ_K *are sets of IDs and KDs respectively. Given a source database for* $\mathcal{I}$, *a query* q *over* $\mathcal{G}$, *and a tuple* $\bar{t}$ *of values of* Γ, *the problem of calculating* $ans(q, \mathcal{I}, \mathcal{D})$ *is undecidable.*

Undecidability of calculating the certain answers to a query immediately implies undecidability of calculating the perfect rewriting [Calì *et* a/., 20031. 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 3.5 Consider a data integration system $\mathcal{I} = \langle \mathcal{G}, \mathcal{S}, \mathcal{M} \rangle$, with $\mathcal{G} = \langle \Psi, \Sigma_I, \Sigma_K \rangle$. An ID $r_1[\mathbf{A}_1] \subseteq r_2[\mathbf{A}_2]$ is a *non-key-conflicting ID (NKCID)* w.r.t. $\mathbf{K}$ if either: *(i)* no KD is defined on r_2; *(ii)* the KD $key(r_2) = \mathbf{K}$ is in Σ_K and $\mathbf{A}_2$ is not a strict superset of $\mathbf{K}$, i.e., $\mathbf{A}_2 \not\supset \mathbf{K}$. If all IDs in Σ_I are NKCIDs w.r.t. Σ_K, the system $\mathcal{I}$ is said *non-key-conflicting (NKC).*

We point out that the class of NKC IDs 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 $key(r_2) = \mathbf{A}_2$.

The most important feature of a NKC data integration system is the *separation* between the IDs and the KDs; in such a case, in fact, we can take IDs into account as if the KDs were not expressed on $\mathcal{G}$.

Theorem 3.6 (Separation) *Given a NKC data integration system* $\mathcal{I} = \langle \mathcal{G}, \mathcal{S}, \mathcal{M} \rangle$, *with* $\mathcal{G} = \langle \Psi, \Sigma_I, \Sigma_K \rangle$, *let* $\mathcal{I}' = \langle \mathcal{G}', \mathcal{S}, \mathcal{M} \rangle$, *with* $\mathcal{G}' = \langle \Psi, \Sigma_I, \emptyset \rangle$, *the system obtained by* $\mathcal{I}$ *by eliminating the KDs 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* $\bar{t}$ *an* n*-tuple of values. We have that* $\bar{t} \notin ans(q, \mathcal{I}, \mathcal{D})$ *iff* $\mathcal{D}$ *is consistent with* Σ_K *and* $\bar{t} \notin ans(q, \mathcal{I}', \mathcal{D})$.

Proof (sketch). We say that $\mathcal{D}$ is consistent with Σ_K iff $ret(\mathcal{I}, \mathcal{D})$ is consistent with Σ_K. An important result, immediately derived from [Johnson and Klug, 1984], states that $ans(q, \mathcal{I}, \mathcal{D})$ is obtained by evaluating q over a (possibly infinite) database, called *chase* and denoted with $chase(ret(\mathcal{I}, \mathcal{D}))$. The chase is obtained by adding tuples to $ret(\mathcal{I}, \mathcal{D})$ in a way that the added tuples repair violations of IDs. The chase satisfies the IDs over $\mathcal{G}$, and it is a representative of all databases in $sem(\mathcal{I}, \mathcal{D})$ [Calì *et al.*, 2002].

"⇒"Since by hypothesis $\bar{t} \notin ans(q, \mathcal{I}, \mathcal{D})$, there exists a global database $\mathcal{B} \in sem(\mathcal{I}, \mathcal{D})$ such that $\bar{t} \notin q^{\mathcal{B}}$. *A fortiori*, $\mathcal{B}$ satisfies Σ_I, therefore $\mathcal{B} \in sem(\mathcal{I}', \mathcal{D})$, The claim follows immediately.

"⇐"By hypothesis, $\mathcal{D}$ is consistent with Σ_K and $\bar{t} \notin ans(q, \mathcal{I}', \mathcal{D})$. Therefore, $\bar{t} \notin q^{chase(ret(\mathcal{I}', \mathcal{D}))}$; note that $chase(ret(\mathcal{I}', \mathcal{D})) = chase(ret(\mathcal{I}, \mathcal{D}))$. It can be shown, by induction on the number of added tuples in the construction of $chase(ret(\mathcal{I}, \mathcal{D}))$, that if $\mathcal{I}$ is a NKC system, and $ret(\mathcal{I}, \mathcal{D})$ satisfies Σ_K, also $chase(ret(\mathcal{I}, \mathcal{D}))$ satisfies Σ_K. It follows that $chase(ret(\mathcal{I}, \mathcal{D}))$ is a representative for all databases in $sem(\mathcal{I}, \mathcal{D})$, and $ans(q, \mathcal{I}, \mathcal{D}) = q^{chase(ret(\mathcal{I}, \mathcal{D}))}$. The claim follows straightforwardly. □

We now go back to query rewriting. In the case of a NKC data integration system, we can apply the same technique developed for IDs alone, provided that we take into account the KDs with suitable rules. Indeed, observe that if *ret(I, V)* violates Σ_K, any tuple is in the answer to any query. Therefore, with regard to this issue, we first introduce a unary global relation *val:* the idea is that *val* stores all values occurring in $\mathcal{D}$. We construct a set of rules Π_{val} as follows: denoting with $\{r_1, \ldots, r_{N_r}\}$ the set of all relations in $\mathcal{G}$,

$$val(X_j) \leftarrow \quad r_i(X_1, \ldots, X_{n_i})$$

with $1 \leq i \leq N_r$ and $1 \leq j \leq n_i$. Then, consider a KD of the form $key(r) = \mathbf{K}$; without loss of generality, we suppose that r has arity m and $\mathbf{K} = \{1, \ldots, k\}$ $(k < m)$. We introduce a set of $m - k$ rules; in particular, for $k + 1 \leq i \leq m$:

$$\begin{aligned} q(Y_1, \ldots, Y_n) \quad \leftarrow \quad & r(X_1, \ldots, X_k, X_{k+1}, \ldots, X_m), \\ & r(X_1, \ldots, X_k, X'_{k+1}, \ldots, X'_m), \\ & X_i \neq X'_i, val(Y_1), \ldots, val(Y_n) \end{aligned}$$

We denote with Π_{KD} the set of rules introduced as described above. From the results of Section 3 and from the above observation, we derive the following result.

Theorem 3.7 *Consider a data integration system* $\mathcal{I} = \langle \mathcal{G}, \mathcal{S}, \mathcal{M} \rangle$, *and a query* q *of arity* n *over* $\mathcal{G}$. *Then,* $\Pi_{\mathrm{ID}} \cup \Pi_{\mathrm{KD}} \cup \Pi_{val} \cup \Pi_{\mathcal{M}}$ *is a perfect rewriting of* q.

4 Semantics for inconsistent data sources

In the sound semantics, violations of IDs are treated "automatically"because of the nature of the semantics; instead, the violation of a single KD 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 [Fagin *et al.*, 1983; Lin and Mendelzon, 1998; Arenas *et al.*, 1999], we now introduce the *loosely-sound* semantics, opposed to the previous one (that we will call *strictly-sound*), in which the soundness assumption is suitably relaxed. The intuition is that in the "relaxed" semantics we are allowed to delete tuples from $ret(\mathcal{I}, \mathcal{D})$ to repair violations of KDs, as long as we "minimize" such deletions; violations of IDs are treated as in the sound semantics.

Given a source database $\mathcal{D}$, we define the following ordering $\gg_{(\mathcal{I}, \mathcal{D})}$ over the global databases for $\mathcal{I}$ that are consistent with $\mathcal{G}$. Given two such global databases $\mathcal{B}_1$ and $\mathcal{B}_2$, we write $\mathcal{B}_1 \gg_{(\mathcal{I}, \mathcal{D})} \mathcal{B}_2$, iff $\mathcal{B}_1 \cap ret(\mathcal{I}, \mathcal{D}) \supset \mathcal{B}_2 \cap ret(\mathcal{I}, \mathcal{D})$. That is, the portion of $ret(\mathcal{I}, \mathcal{D})$ 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 $\mathcal{B}_2$.

We call *maximal* w.r.t. (J, *V)* a global database *B* for I consistent with G, such that there exists no global database *B'* consistent with G such that *B'* >>(I,D) *B.* Based on this notion, we define the loosely-sound semantics sera/ as follows: *semi(I,D)* = *{B* | *B* is consistent with G and *B* is maximal w.r.t. (I , *V)}.* Finally, we denote with *ans₁{q,I,D)* the set of answers to queries under the loosely-sound semantics.

Example 2.1 (cont.) Consider now the source database $\mathcal{D}'$ obtained by adding to $\mathcal{D}_0$ the fact $s_2(RM, Roma, Beckham)$. Then, $ret(\mathcal{I}_0, \mathcal{D}') = ret(\mathcal{I}_0, \mathcal{D}_0) \cup \{team(RM, Roma, Beckham)\}$ We have now that the tuples in $ret(\mathcal{I}_0, \mathcal{D}')$ violate also the KD $key(team) = \{Tacronym\}$; hence, $sem_l(\mathcal{I}_0, \mathcal{D}')$ contains the databases of the forms $\{player(Totti, ITA, RM), player(Beckham, ENG, MU), team(MU, Man.Utd., Giggs), team(RM, Roma, Totti), player(Giggs, \alpha, \beta)\}$ and $\{player(Totti, ITA, RM), player(Beckham, ENG, MU), team(MU, Man.Utd., Giggs), team(RM, Roma, Beckham), player(Giggs, \alpha, \beta)\}$, for each $\alpha, \beta \in \Gamma$.
Notice that for the query $q(X) \leftarrow player(X, Y, Z)$ $ans_l(q, \mathcal{I}_0, \mathcal{D}') = ans(q, \mathcal{I}_0, \mathcal{D})$. On the other hand, given the query $q'(X, Z) \leftarrow team(X, Y, Z)$ we have that $ans_l(q', \mathcal{I}_0, \mathcal{D}') = \{\langle MU, Giggs\rangle\}$ whereas $ans(q', \mathcal{I}_0, \mathcal{D}) = \{\langle MU, Giggs\rangle, \langle RM, Totti\rangle\}$. ∎

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.

5 Query rewriting in loosely-sound semantics

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

Our method relies on Theorem 3.6 stating that for NKC systems it is possible to "separately" deal with inclusion and key dependencies: actually, for the first ones we exploit the algorithm ID-rewrite(Ψ, Σ_I, Q) presented in Section 3, whereas for the second ones we make use of Datalog$^\neg$ under stable model semantics , a well-known extension of Datalog that allows for using negation in the body of program rules [Kolaitis and Papadimitriou, 1991].

More specifically, we define a Datalog$^\neg$ program Π_{lKD} that allows us to compute the maximal subsets of $ret(\mathcal{I}, \mathcal{D})$ that are consistent with Σ_K. Π_{lKD} is obtained by taking, for each relation $r \in \mathcal{G}$, the rules

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

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 $\vec{\mathbf{z}} = Z_1, \ldots, Z_m$.

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

Then, we consider the Datalog$^\neg$ program $\Pi_{lKD} \cup \Pi_{ID} \cup \Pi_{\mathcal{M}\mathcal{D}}$, where Π_{ID} is obtained through ID-rewrite(Ψ, Σ_I, Q), and $\Pi_{\mathcal{M}\mathcal{D}}$ is obtained from $\Pi_{\mathcal{M}}$ by replacing each symbol r with $r_{\mathcal{D}}$.

Theorem 5.1 *Let $\mathcal{I} = (\mathcal{G}, \mathcal{S}, \mathcal{M})$ be a NKC system, and Q be a UCQ of arity n over $\mathcal{G}$. Then, $\Pi_{lKD} \cup \Pi_{ID} \cup \Pi_{\mathcal{M}\mathcal{D}}$ is a perfect rewriting of Q w.r.t. $\mathcal{I}$.*

6 Summary of complexity results

Strictly-sound semantics. Query answering is undecidable even if we allow a slightly more general class of IDs than the NKCIDs; let us define a 1-key-conflicting (1KC) data integration system as a system such that for each ID $r_1[\mathbf{A}_1] \subseteq r_2[\mathbf{A}_2]$, $\mathbf{A}_2$ can be a strict superset of $key(r_2)$ (if defined), but containing at most one attribute more than $key(r_2)$.

Theorem 6.1 *The problem of query answering in 1KC integration systems, under the strictly-sound semantics, is undecidable.*

In the strictly-sound semantics, the complexity of the decision problem of query answering is immediately derived from the rewriting of Section 3.

Theorem 6.2 *The problem of query answering in NKC integration systems, under the strictly-sound semantics, is in PTIME in data complexity.*

Proof. Trivial, since the perfect rewriting $\Pi_{ID} \cup \Pi_{KD} \cup \Pi_{val} \cup \Pi_{\mathcal{M}}$ can be evaluated in PTIME w.r.t. $\mathcal{D}$. □

Loosely-sound semantics Since, as we already said, when $sem(\mathcal{I}, \mathcal{D}) \neq \emptyset$ the strict semantics and the loose ones coincide, it is easy to see that the above properties of query answering under the strictly-sound semantics can be easily generalized.

Theorem 6.3 *The problem of query answering in 1KC integration systems, under the loosely-sound semantics, is undecidable.*

We now characterize the problem of query answering under the loosely-sound semantics in NKC systems.

Theorem 6.4 *The problem of query answering in NKC integration systems, under the loosely-sound semantics, is coNP-complete in data complexity.*

Proof (sketch). Membership in coNP follows from Theorem 5.1, and from the fact that query answering in Datalog$^\neg$ is coNP-complete in data complexity, while coNP-hardness can be easily proved by a reduction of the 3-COLORABILITY problem to our problem. □

The summary of the results we have obtained is reported in the table in Figure 1, which presents the complexity of query answering for both the strictly-sound and the loosely-sound semantics. Each row corresponds to a different class of dependencies (specified in the first two columns), while each cell of the table reports data complexity and combined complexity[1] of query answering for UCQs: for each decidable case, the complexity of the problem is complete w.r.t. the class reported. In the second column of the table, FK stands for "foreign key dependencies" (a well-known class of IDs) while GEN stands for "general IDs". We have marked with the symbol ♠ the cells corresponding either to already known results or to results immediately implied by known results.

[1] The results for combined complexity, which we cannot present in detail due to space limitations, hold under the assumption that the mapping is expressed in terms of UCQs.

KDs	IDs	strictly-sound	loosely-sound
no	GEN	PTIME/PSPACE$^\spadesuit$	PTIME/PSPACE$^\spadesuit$
yes	no	PTIME/NP$^\spadesuit$	coNP/$\Pi_2^p{}^\spadesuit$
yes	FK	PTIME/PSPACE	coNP/PSPACE
yes	NKC	PTIME/PSPACE	coNP/PSPACE
yes	1KC	undecidable	undecidable
yes	GEN	undecidable$^\spadesuit$	undecidable$^\spadesuit$

Figure 1: Complexity of query answering (decision problem)

7 Discussion and related work

In this paper we have presented techniques for query rewriting in data integration systems with integrity constraints, and analyzed the complexity of query answering. To this aim, we have exploited formalisms and methods both from the traditional database theory and from computational logic.

Several works in the literature address the problem of data integration under constraints on the global schema. In this respect, query rewriting under integrity constraints has been first studied in the LAV setting. In particular, [Duschka and Genesereth, 1997] presents a method for query rewriting under functional dependencies in LAV systems, which is able to compute the perfect rewriting in the case of queries and mapping expressed through conjunctive queries, and "maximally contained" rewritings in the case of recursive mappings.

Then, [Gryz, 1999] analyzes query rewriting under inclusion dependencies in LAV systems, and presents a method which is able to deal simultaneously with acyclic IDs and functional dependencies, based on an algorithm for computing the rewriting of a conjunctive query in a database with inclusion dependencies. The algorithm is based on a very interesting idea: obtaining query rewriting 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 preliminarly "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. Our algorithm ID-rewrite presented in Section 3 is certainly inspired by Gryz's main intuitions, but overcomes the above mentioned incompleteness through a new technique for generating the rewriting.

Complexity of query answering in GAV under IDs alone is immediately derived by the results in [Johnson and Klug, 1984]; in this work, the same problem is solved for a restricted class of KDs and IDs, which, however, is significantly less general than the one treated in this paper. More recently, integration under constraints in GAV systems has been addressed in [Call *et al,* 20021, which presents a method for query rewriting in the presence of KDs and foreign key dependencies, under a semantics analogous to our strictly-sound semantics. Thus, the method does not deal with data inconsistencies w.r.t. KDs. Moreover, [Fagin *et al,* 2003] presents an approach for dealing with integrity constraints in a GLAV setting (a generalization of LAV and GAV).

In single database settings, [Arenas *et al,* 1999; Greco *et al.,* 2001] propose methods for consistent query answering in inconsistent databases, which are able to deal with universally quantified constraints. The semantics adopted in these works is different from the ones considered in the present paper.

Acknowledgments This research has been supported by the Projects INFOMIX (IST-2001-33570) and SEWASIE (IST-2001-34825) funded by the EU, and by the Project D2I funded by MIUR (Ministero per l'Istruzione, l'Universita e la Ricerca). We thank Maurizio Lenzerini and Jarek Gryz for precious discussions about this material.

References

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

[Arenas *et al.,* 1999] Marcelo Arenas, Leopoldo E. Bertossi, and Jan Chomicki. Consistent query answers in inconsistent databases. In *Proc. of PODS'99,* pages 68-79,1999.

[Call *et al.,* 2002] Andrea Call, Diego Calvanese, Giuseppe De Giacomo, and Maurizio Lenzerini. Data integration under integrity constraints. In *Proc. of CAiSE 2002,* volume 2348 of *LNCS,* pages 262-279. Springer, 2002.

[Cali *et al,* 2003] Andrea Call, Domenico Lembo, and Riccardo Rosati. On the decidability and complexity of query answering over inconsistent and incomplete databases. In *Proc. of PODS 2003,* 2003. To Appear.

[Duschka and Genesereth, 1997] Oliver M. Duschka and Michael R. Genesereth. Answering recursive queries using views. In *Proc. of PODS'97,* pages 109-116, 1997.

[Duschka and Levy, 1997] Oliver M. Duschka and Alon Y. Levy. Recursive plans for information gathering. In *Proc. of IJCAI'97,* pages 778-784, 1997.

[Fagin *et al,* 1983] Ronald Fagin, Jeffrey D. Ullman, and Moshe Y. Vardi. On the semantics of updates in databases. In *Proc. of PODS'83,* pages 352-365,1983.

[Fagin *et al.,* 2003] Ronald Fagin, Phokion G. Kolaitis, Renee J. Miller, and Lucian Popa. Data exchange: Semantics and query answering. In *Proc. of ICDT 2003,* pages 207-224, 2003.

[Greco *et al.,* 2001] Gianluigi Greco, Sergio Greco, and Ester Zumpano. A logic programming approach to the integration, repairing and querying of inconsistent databases. In *Proc. of ICLP'01,* volume 2237 of *LNAI,* pages 348-364. Springer, 2001.

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

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

[Kolaitis and Papadimitriou, 1991] Phokion G. Kolaitis and Christos H. Papadimitriou. Why not negation by fixpoint? *J. of Computer and System Sciences,* 43(1): 125-144,1991.

[Lenzerini, 2002] Maurizio Lenzerini. Data integration: A theoretical perspective. In *Proc. of PODS 2002,* pages 233-246,2002.

[Lin and Mendelzon, 1998] Jinxin Lin and Alberto O. Mendelzon. Merging databases under constraints. *Int. J. of Cooperative Information Systems,* 7(I):55-76,1998.