Fixpoint Semantics for Query Answering in Data Integration Systems

Zoran Majkić

Dipartimento di Informatica e Sistemistica
University of Roma "La Sapienza"
Via Salaria 113, I-00198 Roma, Italy
majkic@dis.uniroma1.it

Abstract. Data integration consists in providing a uniform access to a set of data sources, through a unified representation of the data called *global schema*; a *mapping* specifies the relationship between the global schema and the sources. In this paper we introduce a very general framework for data integration based on category theory. We give a denotational semantics for the data integration framework in the case of the GLAV approach, where the integrity constraints on the global schema are tuple-generating dependencies. We revisit the query known rewriting algorithms for this kind of systems by giving a denotational coalgebra semantics for them. In particular, since the known algorithms for query rewriting are based on the notion of *chase* of a database w.r.t. a set of constraints, we introduce a *chase operator*, and we show, by giving a fixpoint semantics for such an operator, that only a finite portion of the (possibly infinite) chase is needed for query answering.

1 Introduction

The task of a data integration system [10] is to provide the user with a unified view, called *global schema*, of a set of heterogeneous data sources. Once the user issues a query over the global schema, the system carries out the task of suitably accessing the different sources and assemble the retrieved data into the final answer to the query. In this context, a crucial issue is the specification of the relationship between the global schema and the sources, which is called *mapping* [8, 10]. In this paper we use a more complex mapping, called GLAV [7, 6], consists in associating views over the global schema to views over the sources.

Since the global schema is a representation of the domain of interest of the system, it needs to be represented by means of a flexible an expressive formalism: to this aim, *integrity constraints* are expressed on it. The data at the sources may not satisfy the constraints on the global schema; in this case a common assumption (which is the one adopted in this paper) is to consider the sources as *sound*, i.e., they provide a *subset* of the data that satisfy the global schema.

This paper is based on the translation of data integration systems [10] into the denotational semantics based on category theory [11, 13]. Such a translation is useful in order to obtain fixpoint semantics for canonical solutions of database integration systems and simple commutative diagrams based on coalgebra homomorphisms for complex query rewriting algorithms.

First, we present a denotational semantics for both database mappings and integrity constraints where we introduce two kinds of database equivalences and their correspondent partial orders (Section 2). Second, we consider a specific framewok for data integration, where the mapping is GLAV, and the constraints on the global schema are *tuple-generating dependencies (TGDs)* on the global schema (Section 3); TGDs are an extension of *inclusion dependencies*, which are an important class of dependencies in database schemata [1, 9]. Note that GLAV assertions are logic formulae that are analogous to TGDs. The semantics of query answering in this kind of systems is formulated in terms of a *canonical model*, which is a representative of all minimal Herbrand models for the global schema of an integration system. Algorithms for query answering are proposed in [4, 2], where query answering is performed by *rewriting* the user query into a new query that can be evaluated over the data sources. Finally (Sections 4 and 5), we present a functorial translation of the logical theory expressing a data integration system into a denotational semantics, based on the database category, and formalise a fixpoint property for query answering. Moreover, we give a coalgebra semantics for query rewriting algorithms. paper.

2 Denotational semantics for Database mappings

In this section we introduce concepts of denotational semantics, together with behavioral equivalence for databases, which will be used for the functorial translation of logical database theory into the database category.
Let introduce some basic categorial notions [11]. The *category* is an abstract structure composed by collection of objects and arrows (or *morphisms*) between them. Each arrow has a source and a target object and each object has at least its identity arrow. The composition of arrows (by operator $'\circ'$) satisfies the associative laws. A *functor* $F : \mathcal{K}_1 \longrightarrow \mathcal{K}_2$ is a mapping (a pair of functions: F^0 for objects and F^1 for arrows) from the category $\mathcal{K}_1$ to $\mathcal{K}_2$ that preserves the categorical structure, that is, the composition of arrows, and maps identity arrow of any object A into the identity arrow of the object $F^0(A)$. An *endofunctor* is a functor with the same source and target category. There exists the category **Cat**: take as objects all categories, as arrows all functors (the composition of functors is a composition of paired functions). Given any two categories $\mathcal{K}_1$ and $\mathcal{K}_2$, we can define the category $\mathcal{K}_2^{\mathcal{K}_1}$: each object is some functor from $\mathcal{K}_1$ to $\mathcal{K}_2$; the arrows between these objects (i.e., functors) are called *natural transformations*, and their composition $'\bullet'$ is called *vertical composition* of natural transformations. Let $F, G : \mathcal{K}_1 \longrightarrow \mathcal{K}_2$ be two functors, then a *transformation* (which is a function) $\varepsilon : F \longrightarrow G$ represents a family of arrows in $\mathcal{K}_2$, $\varepsilon(A) : F^0(A) \longrightarrow G^0(A)$ for each object A in $\mathcal{K}_1$; such transformation is *natural* if for each arrow $f : A \longrightarrow B$ in $\mathcal{K}_1$, $F^1(f) \circ \varepsilon(B) = \varepsilon(A) \circ G^1(f)$.

The Fixpoint semantics and coalgebra semantics for a query rewriting algorithms will be considered in the *base database category* DB [13, 12]. Such category for database mappings is at *instance level*: each object of this category is an extensional database which corresponds to a Herbrand model of some database logical theory; arrows in this category are mapping morphisms between extensional databases . The connection between logical (schema) level and this computation category is based on *interpretation* functors. Thus, each rule-based conjunctive query at schema level over

database $\mathcal{A}$ will be translated into an morphism from some database instance A (some model of the database schema $\mathcal{A}$) into a database instance composed by all views of the instance A.
The new approach based on the behavioral point of view for databases and the introduction of observations, which are computations without side-effects, defines also the fundamental (from Universal algebra [14]) monad [11,5] (T, η, μ) of the endofunctor $T : DB \longrightarrow DB$, such that for any object (database) A, the object TA denotes a database composed by the set of *all views* of A obtained by the set of all possible conjunctive+union queries (the free term algebra with carrier set A of the "select-project-join + union" query language (of SPJRU or its equivalent SPCU algebra, Chapter 4.5, 5.4 in [1]), or more precisely, TA is "generated" by this algebra). Morphisms of this category are all possible mappings between database instances *based on views*.
We introduce two functions, $\partial_0 \ and \ \partial_1$, such that for any view-map $q_{A_i} : A \longrightarrow TA$, we have that $\partial_0(q_{A_i}) = \{r_1, ..., r_k\} \subseteq A$ is a subset of relations of A used as arguments by this query q_{A_i} and $\partial_1(q_{A_i}) = v \in TA$ (resulting view of a query q_{A_i}) and for any morphism (database mapping)from a database A to B, $f : A \longrightarrow B$, defined as a set of queries q_{A_i} mapped from A into B in one of the following cases: 1. inclusion case: when $v = \partial_1(q_{A_i}) \subseteq b_j$, 2. inverse-inclusion case: when $v = \partial_1(q_{A_i}) \supseteq b_j$, 3. equal case: when $v = \partial_1(q_{A_i}) = b_j$, where $b_j \in B$ is some relation (table) in B, we have

$$f = \bigcup_{q_{A_i} \in S_f} \{q_{A_i}\} \ \ , \ \ \partial_0(f) = \bigcup_{q_{A_i} \in S_f} \partial_0(q_{A_i}) \ \ , \ \ \partial_1(f) = \bigcup_{q_{A_i} \in S_f} \{\partial_1(q_{A_i})\}$$

In what follows we omit the parenthesis for singletons $\{..\}$. When we define a mapping between two databases A and B, implicitly we define the "information flux" $\widetilde{f}$ [13], i.e, the set of views of A "transmitted" by this mapping into B. This concept can be intuitively explained in the following simplified way: each mapping from a database A into a database B defines a part of information in B which is passed by this mapping from a database A. Let r_i, $i = 1, .., n$, be the set of resulting relations of views over A used in the mapping f in order to define a part of information in B, then the set of *all possible answers* to union of conjunctive queries over this set of relations r_i, $i = 1, .., n$ (note, that each of these possible answers is also some view of A) defines the "information flux" of the mapping f. Thus, the "information flux" of the composition of two mappings $f \circ g$ is the set intersection of "information fluxes" of them, that is $\widetilde{f \circ g} = \widetilde{f} \bigcap \widetilde{g}$.
A mapping is a monomorphism ('injective') if $\widetilde{f} = TA$, and an epimorphism (surjective') if $\widetilde{f} = TB$; it is an isomorphism if it is monomorphic and epimorphic, that is, when $\widetilde{f} = TA = TB$. Any two mappings f,g are *equivalent* , $f \approx g$, if they contain the same "information flux" ($\widetilde{f} = \widetilde{g}$).
By duality of the category DB, for each mapping $f : A \longrightarrow B$ between two (extensional) databases there exists also its equivalent reverse mapping $f^{inv} : B \longrightarrow A$ in DB, that is, $f^{inv} \approx f$.
Each pair (A, h), where $h : A \longrightarrow TA$ is some database mapping from a database A into a database TA (for example, each single query over A with a resulting view in TA) is called $T - coalgebra$, and h is called "structural map". A homomorphism f between $T - coalgebras$ (A, h) and $B, k)$ is a mapping $f : A \longrightarrow B$ such that holds $k \circ T(f) = f \circ h$.

2.1 The (strong) behavioral equivalence for databases

We can characterize [13] each object in DB (a database instance) by its behavior according to a given set of observations (views) obtained by conjunctive queries. Indeed, if one object A is considered as a black-box, the object TA is just the set of all observations on A: so given two objects A and B we are able to define the relation of equivalence between them based on the notion of the bisimulation relation - if the observations (resulting views of queries) of A and B are always equal, independently of their particular internal structure, then they seam equivalent to the observer.
In fact, any database can be seen as a system with a number of internal states which can be observed by using query operators (i.e, programs without side-effects). Thus databases A and B are equivalent (bisimilar) if they have the same set of its observable internal state, i.e. when TA is equal to TB:

Definition 1 ([13]). *the relation of (strong) behavioral equivalence $'\approx'$ between objects (databases) in DB is given by $A \approx B$ iff $TA = TB$, and the equivalence relation for morphisms is given by $f \approx g$ iff $\widetilde{f} = \widetilde{g}$.*

This relation of behavior equivalence between objects $A \approx B$ corresponds to the notion of isomorphism in the category DB ,i.e., $A \simeq B$. Let us prove that the equivalence relations on objects and morphisms are based on the "inclusion" Partial Order (PO) relations, which define the DB as a 2-category:

Proposition 1 ([13]) *The subcategory $DB_I \subseteq DB$, with $Ob_{DB_I} = Ob_{DB}$ and with only monomorphic arrows, is a Partial Order category with PO relation of "inclusion" $A \preceq B$ defined by a monomorphism $f : A \hookrightarrow B$. The "inclusion" PO relations for objects and arrows are defined as follows:*

$$A \preceq B \text{ iff } TA \subseteq TB \quad , \quad f \preceq g \text{ iff } \widetilde{f} \preceq \widetilde{g} \text{ (i.e.,} \widetilde{f} \subseteq \widetilde{g})$$

they determine observation equivalences, i.e.,

$$A \simeq B \text{ (i.e., } A \approx B) \quad \text{iff} \quad A \preceq B \text{ and } B \preceq A$$

$$f \approx g \quad \text{iff} \quad f \preceq g \text{ and } g \preceq f$$

The power-view endofunctor $T : DB \longrightarrow DB$ is a 2-endofunctor and the closure operator for this PO relation: any object A such that $A = TA$ will be called "closed object".
DB is a 2-category where 1-cells are its ordinary morphisms, while 2-cells are the arrows between ordinary morphisms : for any two morphisms $f, g : A \longrightarrow B$, such that $f \preceq g$, a 2-cell arrow is the "inclusion" $\sqrt{\alpha} : f \xrightarrow{\preceq} g$.

Example for equivalent morphisms: for any view-map $q_{A_i} : A \longrightarrow TA$ holds $\widetilde{q_{A_i}} = T\partial_0(q_{A_i}) \bigcap T\partial_1(q_{A_i}) = T\partial_1(q_{A_i})$. Thus, the equivalence with an other view-map $q_{B_j} : B \longrightarrow TB$ is given by: $q_{A_i} \approx q_{B_j}$ iff $TA \ni \partial_1(q_{A_i}) = \partial_1(q_{B_j}) \in TB$, i.e., when they produce the same view.

2.2 Weak observational equivalence for databases

Some database instances can also have relations with tuples containing also *Skolem constants* (minimal Herbrand models for Global schema of some Data integration system [10, 3, 6]). In the following we consider a recursively enumerable set of all Skolem constants as marked (labelled) nulls $SK = \{\omega_0, \omega_1, ...\}$, disjoint from the domain set **dom** of all values for databases, and we introduce the unary predicate $Val(\_)$, such that $Val(t)$ is true iff $t \in$ **dom** (so, $Val(\omega_i)$ is false for any $\omega_i \in SK$).

Definition 2 ([13]). *The weak power-view operator $T_w : Ob_{DB} \longrightarrow Ob_{DB}$ is defined as follows: for any database A in DB category holds:*
$T_w(A) \triangleq \{ v \mid v \in T(A) \text{ and } \forall_{1 \leq k \leq |v|} \forall(t \in \pi_k(v)) Val(t)\}$
where $|v|$ is the number of attributes of the view v, and π_k is k-th projection operator on relations. We define the partial order relation $\preceq_w$ for databases:
$A \preceq_w B$ *iff* $T_w(A) \subseteq T_w(B)$
and we define the weak observational equivalence relation $\approx_w$ for databases:
$A \approx_w B$ *iff* $A \preceq_w B$ *and* $B \preceq_w A$

The following properties holds for the weak partial order $\preceq_w$, w.r.t. the partial order $\preceq$ (we denote $'A \prec B'$ iff $A \preceq B$ and not $A \simeq B$):

Proposition 2 ([13]) *Let A and B be any two databases, then:*

1. $T_w(T_w(A)) = T(T_w(A)) = T_w(T(A)) = T_w(A) \subseteq T(A)$ *, thus each object $D = T_w(A)$ is a closed object (i.e., $D = T(D)$) such that $D \approx_w A$*
2. *T_w is a closure operator w.r.t. the "weak inclusion" relation $\preceq_w$*
3. $T_w(A) \simeq A$ *, if A is a database without Skolem constants*
   $T_w(A) \prec A$ *, otherwise*
4. *$A \prec B$ implies $A \preceq_w B$ and $A \simeq B$ implies $A \approx_w B$*

Note that from point 4, the partial order " $\preceq$ " is a more strong discriminator for database than the weak partial order " $\preceq_w$ ", i.e., we can have two non isomorphic objects $A \prec B$ which are weakly equivalent, $A \approx_w B$ (for example when $A = T_w(B)$ and B is a database with Skolem constants). Let us extend the notion of the type operator T into the notion of the endofunctor in DB category:

Theorem 1 ([13]) *There exists the weak power-view endofunctor*
$T_w = (T_w^0, T_w^1) : DB \longrightarrow DB$ *, such that*

1. *for any object A, the object component T_w^0 is equal to the type operator T_w.*
2. *for any morphism $f : A \longrightarrow B$, the arrow component T_w^1 is defined by*
   $T_w(f) \triangleq T_w^1(f) = inc_B^{inv} \circ T^1(f) \circ inc_A$
   where $inc_A : T_w(A) \hookrightarrow T(A)$ is a monomorphism (set inclusion) and $inc_B^{inv} : T(B) \twoheadrightarrow T_w(B)$ is an epimorphism (reversed monomorphism inc_B).
3. *Endofunctor T_w preserves properties of arrows, i.e., if a morphism f has a property P (monic, epic, isomorphic), then also $T_w(f)$ has the same property: let P_{mono}, P_{epi} and P_{iso} are monomorphic, epimorphic and isomorphic properties respectively, then the following formula is true* $\forall(f \in Mor_{DB})$
   $(P_{mono}(f) \equiv P_{mono}(T_w f) \wedge P_{epi}(f) \equiv P_{epi}(T_w f) \wedge P_{iso}(f) \equiv P_{iso}(T_w f)$
4. *There exist the natural transformations, $\xi : T_w \longrightarrow T$ (natural monomorphism), and $\xi^{-1} : T \longrightarrow T_w$ (natural epimorphism), such that for any object A, $\xi(A) = inc_A$ is a monomorphism and $\xi^{-1}(A) = inc_A^{inv}$ is an epimorphism such that $\xi(A) \approx \xi^{-1}(A)$.*

Like the monad (T, η, μ) and comonad (T, η^C, μ^C) of the endofunctor T, also for the weak endofunctor T_w we can define such structures:

Proposition 3 ([13]) *The weak power-view endofunctor* $T_w = (T_w^0, T_w^1) : DB \longrightarrow DB$ *defines the monad* (T_w, η_w, μ_w) *and the comonad* (T_w, η_w^C, μ_w^C) *in* DB*, such that* $\eta_w = \xi^{-1} \bullet \eta : I_{DB} \longrightarrow T_w$ *is a natural epimorphism and* $\eta_w^C = \eta^C \bullet \xi : T_w \longrightarrow I_{DB}$ *is a natural monomorphisms (* $'\bullet'$ *is a vertical composition for natural transformations), while* $\mu_w : T_w T_w \longrightarrow T_w$ *and* $\mu_w^C : T_w \longrightarrow T_w T_w$ *are equal to the natural identity transformation* $id_{T_w} : T_w \longrightarrow T_w$ *(because* $T_w = T_w T_w$ *).*

3 GLAV Framework for Data Integration Systems

In this section we give a specific framework for data integration, where the mapping is GLAV, and integrity constraints on the global schema are a restricted class of tuple generating dependencies [1]. The characteristics of the components of a data integration system in our approach [2] are as follows:

- The *global schema*, enriched with the *new unary predicate* $Val(\_)$ such that $Val(c)$ is true if $c \in$ **dom**, is expressed in the relational model with Σ_T equal to a set of *weakly-full TGDs (WFTGDs*. A *weakly-full tuple-generating dependency (WFTGD)* is a logic formula of the form $\forall \mathbf{x}\,(\exists \mathbf{y}\ \phi_G(\mathbf{x}, \mathbf{y}) \implies \exists \mathbf{z}\ (\psi_G(\mathbf{x}, \mathbf{z}))$, where the right-hand side has no existentially quantified variables, i.e., each $y_i \in \mathbf{y}$ appears at most once in the left side.
- The *mapping* $\mathcal{M}$ is defined following the GLAV approach: each dependency in $\mathcal{M}$ is a *tuple-generating dependency (TGD)* of the form $\forall \mathbf{x}\,(\exists \mathbf{y}\ q_{\mathcal{S}}(\mathbf{x}, \mathbf{y}) \implies \exists \mathbf{z}\ q_{\mathcal{G}}(\mathbf{x}, \mathbf{z}))$, where the formula $q_S(\mathbf{x}, \mathbf{y})$ is a conjunction of atomic formulae over $\mathcal{S}$ and $q_G(\mathbf{x}, \mathbf{z})$ is a conjunction of atomic formulas over $\mathcal{G}$.
  We compute the relation Val for all constants in **dom**. The various relations obtained by the mapping $\mathcal{M}$ over the source database $\mathcal{D}$ define what we call the *retrieved global database* $ret(\mathcal{I}, \mathcal{D})$.

Example 1. Consider a data integration system $\mathcal{I} = \langle \mathcal{G}, \mathcal{S}, \mathcal{M} \rangle$, with $\mathcal{G} = \langle \mathcal{G}_T, \Sigma_T \rangle$. The schema $\mathcal{G}_T$ is constituted by the relations $R_1/2$ and $R_2/2$, the source schema by relations $S_1/2, S_2/1$. The set of TGDs Σ_T contains the single TGD $\theta :\ R_1(X, Y) \Longrightarrow R_1(Y, W), R_2(Y, X)$. The mapping $\mathcal{M}$ consists of the assertions $S_1(X, c) \Longrightarrow R_1(X, Y), R_2(Y, Z)$ and $S_2(X) \Longrightarrow R_2(X, Y)$. ■

In our case, *with integrity constraints* and with *sound mapping*, the semantics of the data integration system $\mathcal{I}$ is specified in terms of a set of *legal* global databases, namely, those databases (they exits iff $\mathcal{I}$ *is consistent* w.r.t. $\mathcal{D}$, i.e., iff $ret(\mathcal{I}, \mathcal{D})$ does not violate any constraint in $\mathcal{G}$) that are supersets of the *retrieved* global database $ret(\mathcal{I}, \mathcal{D})$. In [2], given the retrieved global database $ret(\mathcal{I}, \mathcal{D})$, we may construct inductively the canonical database $can(\mathcal{I}, \mathcal{D})$ by starting from $ret(\mathcal{I}, \mathcal{D})$ by applying repeatedly the following *chase rule*:

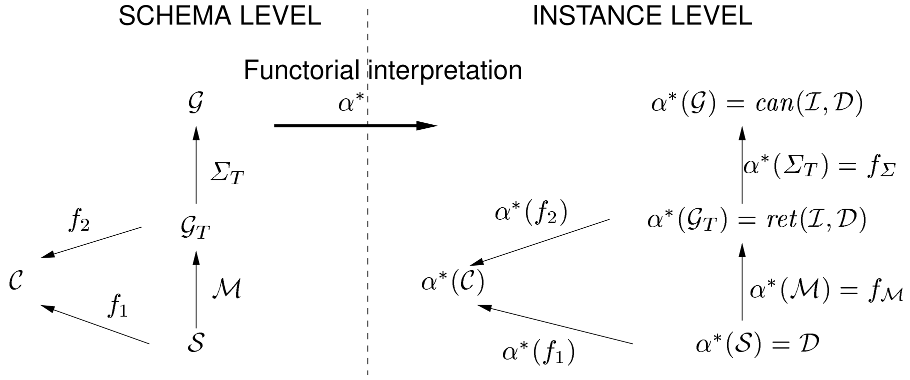

Fig. 1. Functorial translation

TGD CHASE RULE. Consider a retrived global database $ret(\mathcal{I},\mathcal{D})$ and a TGD θ of the form $\chi(\mathbf{x},\mathbf{y}) \rightarrow \psi(\mathbf{x},\mathbf{z})$. The TGD θ is *applicable* to $ret(\mathcal{I},\mathcal{D})$ if there is a substitution σ that sends the atoms of $\chi(\mathbf{x},\mathbf{y})$ to tuples of $ret(\mathcal{I},\mathcal{D})$, and there is no generalisation of σ that sends the atoms of $\psi(\mathbf{x},\mathbf{z})$ to tuples of $ret(\mathcal{I},\mathcal{D})$. In this case: *(i)* we define a substitution σ' such that $\sigma'(x_i) = \sigma(x_i)$ for each x_i in $\mathbf{x}$, and $\sigma'(z_j) = \zeta_j$ for each z_j in $\mathbf{z}$, where ζ_j is a fresh constant of $\mathbf{dom}$, not already introduced in the construction and not appearing in $ret(\mathcal{I},\mathcal{D})$; *(ii)* we add to $ret(\mathcal{I},\mathcal{D})$ the facts of $\sigma'(\varphi_\mathcal{G}(\mathbf{x},\mathbf{z}))$ that are not already in $ret(\mathcal{I},\mathcal{D})$.
Note that in the case of WFTGDs, the canonical database may be infinite.

Example 2. Consider Example 1, and let $\mathcal{B}$ be a retrieved global database constituted by a single fact $R_1(a,b)$. Let us construct $can(\mathcal{I},\mathcal{D})$: at the first step we add the facts $R_1(b,z_1), R_2(b,a)$; at the second step the facts $R_1(z_1,z_2), R_2(z_1,b)$; note that the construction process is infinite. ■

Based on the results [2], $can(\mathcal{I},\mathcal{D})$ is the right abstraction for answering queries posed to the data integration system. Note that terms involving Skolem functions are never part of the *certain answers*, so the lifted queries q use the Val(_) predicate in order to exclude from the answers the tuples with a Skolem constants in $can(\mathcal{I},\mathcal{D})$.

Thus at the logical level, this GLAV data integration system, can be represented by the graph composed by two arrows (Figure 1) , $\mathcal{M} : \mathcal{S} \longrightarrow \mathcal{G}_T$ and $\Sigma_T : \mathcal{G}_T \longrightarrow \mathcal{G}$ ($Sch(\mathcal{I})$ denotes the category derived by this graph).
Let us consider the most general case of GLAV mapping:

Definition 3 ([13]). *For a general GLAV data integration/exchange system* $\mathcal{I} = \langle \mathcal{G},\mathcal{S},\mathcal{M}\rangle$ *, when each TGD maps some **view** of an database into some **view** of other database , we define the following two **schema mappings**,* $f_1 : \mathcal{S} \longrightarrow \mathcal{C}$ *,* $f_2 : \mathcal{G}_T \longrightarrow \mathcal{C}$ *, where* $\mathcal{C}$ *is a new logical schema composed by a new predicate symbol* $r_i(\mathbf{x})$ *for a formulae* $q_\mathcal{G}(\mathbf{x},\mathbf{z})$*, for every i-th TGD* $\forall \mathbf{x}\,(\exists \mathbf{y}\; q_\mathcal{S}(\mathbf{x},\mathbf{y}) \implies \exists \mathbf{z}\; \mathbf{q}_\mathcal{G}(\mathbf{x},\mathbf{z}))$ *in* $\mathcal{M}$ *:*

$$f_1 \triangleq \bigcup_{\partial_0(q_i)=R_{1i}\ \&\ \partial_1(q_i)=\partial_0(v_i)\ \&\ \partial_1(v_i)\ =\ r_i} v_i \cdot q_i \ : \mathcal{S} \longrightarrow \mathcal{C}$$

$$f_2 \triangleq \bigcup_{\partial_0(q_{G_i})=R_{2i} \ \& \ \partial_1(q_{G_i}) = r_i} q_{G_i} : \mathcal{G}_T \longrightarrow \mathcal{C}$$

(R_{1i}, R_{2i} are, respectively, the set of predicate symbols used in the query $q_{\mathcal{S}}(\mathbf{x}, \mathbf{y})$ and the set of predicate letters used in the query $q_{\mathcal{G}}(\mathbf{x}, \mathbf{z})$)

Thus, we can define the functorial translation of this GLAV Data integration system into denotational semantics DB category:

Definition 4. *The **functorial interpretation** of this logical scheme into denotational semantic domain DB, $\alpha^* : Sch(\mathcal{I}) \longrightarrow DB$, is defined by two corresponding arrows (Fig. 1) $f_M : D \longrightarrow ret(\mathcal{I}, \mathcal{D})$, $f_\Sigma : ret(\mathcal{I}, \mathcal{D}) \longrightarrow can(\mathcal{I}, \mathcal{D})$, where $\alpha^*(\mathcal{S}) = \mathcal{D}$ is the extension of the source database $\mathcal{D}$, $\alpha^*(\mathcal{G}_T) = ret(\mathcal{I}, \mathcal{D})$ is the retrieved global database, $\alpha^*(\mathcal{G}) = \alpha^*(\mathcal{G}_T, \Sigma_T) = can(\mathcal{I}, \mathcal{D})$ is the universal (canonical) instance of the global schema with the integrity constraints, and*
$\alpha^*(f_2) = f_G \triangleq \bigcup\{ \alpha(q_{G_i}) \mid$ for every i-th TGD $\forall \mathbf{x}\ (\exists \mathbf{y}\ q_{\mathcal{S}}(\mathbf{x}, \mathbf{y}) \implies \exists \mathbf{z}\ \mathbf{q}_{\mathcal{G}}(\mathbf{x}, \mathbf{z})) \in \mathcal{M},\ \partial_{\mathbf{1}}(\alpha(\mathbf{q}_{\mathbf{G_i}})) = \alpha(\mathbf{r_i}),\ \partial_{\mathbf{0}}(\alpha(\mathbf{q}_{\mathbf{G_i}})) = \alpha^*(\mathbf{R_{2i}})\}$, where $\alpha(r_i) = \pi_X(q^G_{\mathcal{G}}(\mathbf{x}, \mathbf{y}))$ is the projection on x of the view $q^G_{\mathcal{G}}(\mathbf{x}, \mathbf{y})$, obtained from the query $q_{\mathcal{G}}(\mathbf{x}, \mathbf{y})$ over $ret(\mathcal{I}, \mathcal{D})$, and for predicate symbols in $\partial_0(q_{G_i}) = R_{2i}$, $\alpha^*(R_{2i})$ is the set of their extensions (relations) in $ret(\mathcal{I}, \mathcal{D})$.
$\alpha^*(f_1) = f_D \triangleq \bigcup\{ \alpha(v_i) \circ \alpha(q_i) \mid$ for every i-th TGD $\forall \mathbf{x}\ (\exists \mathbf{y}\ q_{\mathcal{S}}(\mathbf{x}, \mathbf{y}) \implies \exists \mathbf{z}\ \mathbf{q}_{\mathcal{G}}(\mathbf{x}, \mathbf{z})) \in \mathcal{M},\ \partial_{\mathbf{0}}(\alpha(\mathbf{q_i})) = \alpha^*(\mathbf{R_{1i}}),\ \partial_{\mathbf{1}}(\alpha(\mathbf{q_i})) = \pi_{\mathbf{X}}(\mathbf{q}^{\mathbf{D}}_{\mathcal{S}}(\mathbf{x}, \mathbf{z}))$, and $\alpha(v_i) : \pi_X(q^D_{\mathcal{S}}(\mathbf{x}, \mathbf{z})) \longrightarrow \pi_{\mathbf{X}}(\mathbf{q}^{\mathbf{G}}_{\mathcal{G}}(\mathbf{x}, \mathbf{y}))$ is an inclusion function $\}$, where $q^D_{\mathcal{S}}(\mathbf{x}, \mathbf{z})$ is the view obtained from the query $q_{\mathcal{S}}(\mathbf{x}, \mathbf{z})$ over the source extension $\mathcal{D}$. Thus,

- *$\alpha^*(\mathcal{C}) = C = \bigcup\{\alpha(r_i)\}$ is the extension of the new logical schema $\mathcal{C}$.*
- *$f_M = f^{inv}_G \circ f_D$.*
- *$f_\Sigma \triangleq \bigcup \{v_k \circ q_{ret_k} \mid where\ \ q_{ret_k} : ret(\mathcal{I}, \mathcal{D}) \longrightarrow T(ret(\mathcal{I}, \mathcal{D}))$ such that $\partial_0(q_{ret_k}) = \partial_1(q_{ret_k}) \in ret(\mathcal{I}, \mathcal{D})$, and $v_k : \partial_1(q_{ret_k}) \longrightarrow can(\mathcal{I}, \mathcal{D})$ an inclusion function $\}$.*

4 Least fixpoint for the canonical solution

Intuitively, the procedure to produce a canonical database for the global schema can be described as follows: start with an instance $< I, 0 >$ that consists of I, instance of the source schema, and of the empty instance 0 for the target (global schema); then chase $< I, 0 >$ by applying all the dependencies in Σ_{st} (a finite set of source-to-target dependencies) and Σ_t (a finite set of target integrity dependencies) for as long as they are applicable. This process may fail (if an attempt to identify two domain constants is made in order to define a homomorphism between two consecutive target instances) or it may never terminate. Let J_i and J_{i+1} denote two consecutive target instances of this process ($J_0 = 0$), then we introduce the function $C_h : \Theta \longrightarrow \Theta$, where Θ is the set of all pairs $< I, J >$ where I is some source instance and J some of the generated by I target instances, such that: $< I, J_{i+1} >= C_h(< I, J_i >) \supseteq < I, J_i >$. Such function is a monotone. Let define the sets $S_i = T_w(\pi_2(< I, J_i >)) = T_w(J_i)$ and the operator $\Psi : \Theta_w \longrightarrow \Theta_w$, where $\Theta_w = \{ T_w(\pi_2(S)) \mid S \in \Theta\}$, such that $\Psi(T_w(\pi_2(< I, J_i >))) = T_w(\pi_2(C_h(< I, J_i >)))$, i.e.,$\Psi T_w \pi_2 = T_w \pi_2 C_h : \Theta \longrightarrow \Theta_w$ and with the least fixpoint S, $S = \Psi(S)$.

Proposition 4 *Let $< I, 0 >$ be an initial instance that consists of I, a finite instance of the source schema, and of the empty instance 0 for the target (global schema). Then there exists the least fixpoint $S)$ of the function $\Psi : \Theta_w \longrightarrow \Theta_w$, which is equal to $S = T_w \pi_2 C_h^n(< I, 0 >)$ for some finite n.*
*The closure operator T_w is algebraic, that is, given any infinite canonical database $can(\mathcal{I}, \mathcal{D})$, holds $T_w(can(\mathcal{I}, \mathcal{D})) = \bigcup\{T_w(X') \mid X' \subseteq_\omega can(\mathcal{I}, \mathcal{D})\}$, where $X' \subseteq_\omega can(\mathcal{I}, \mathcal{D})\}$ means that X' is some **finite** subset of $can(\mathcal{I}, \mathcal{D})$.*

Note that each infinite canonical database is weakly equivalent to its finite subset X' obtained as the least fixpoint of the operator Ψ, i.e., $can(\mathcal{I}, \mathcal{D})\} \approx_w X'$, where $X' = \Psi(X')$ is a finite subset of $can(\mathcal{I}, \mathcal{D})$.

5 Query rewriting coalgebra semantics

In the context of query-rewriting we consider only queries which resulting view belongs to the "information flux" of this mapping. Consequently, given any two queries, $q_{A_i} : A \longrightarrow TA$ and $q_{B_j} : B \longrightarrow TB$, they have to satisfy (w.r.t. query rewriting constraints) the condition $\partial_1(q_{A_i}) \in \widetilde{f}$ (the $\partial_1(q_{A_i})$ is just a resulting view of this query) and $\partial_1(q_{B_j}) \in \widetilde{f}$. So, the well-rewritten query over B, $q_{B_j} : B \longrightarrow TB$, such that it is equivalent to the original query, i.e., $q_{B_j} \approx q_{A_i}$, must satisfy the condition $\partial_1(q_{B_j}) = \partial_1(q_{A_i}) \in \widetilde{f}$.
The denotational semantics for a query-rewriting in a data integration/exchange environment is given by the following result.

Theorem 2 ([13]) *Each **database query** is a T-coalgebra; instead, each lifted query (for certain answers) is also a $T_w - coalgebra$. Any morphism between two T-coalgebras $f : (A, q_{A_i}) \longrightarrow (B, q_{B_j})$ defines the semantics for relevant query-rewriting, when $\partial_1(q_{A_i}) \in \widetilde{f}$ and $\partial_1(q_{B_j}) \in \widetilde{f}$; instead, when q_{A_i} and q_{B_j} are lifted queries, then f is also a morphism between $T_w - coalgebras$, $f : (A, q_{A_{wi}}) \longrightarrow (B, q_{B_{wj}})$, with $q_{A_{wi}} \approx q_{A_i}$ and $q_{B_{wj}} \approx q_{B_j}$.*

In, fact from the commutative diagram in Fig. 2 (where q_{A_i} and q_{B_j} are lifted queries) which corresponds to the homomorphism between these two query coalgebras, we obtain that, given a mapping f from a database A into a database B , given any original query $q_{A_i} \in \widetilde{f}$ over A, for the rewritten query q_{B_j} holds that $q_{B_j} = Tf \circ q_{A_i} \circ f^{inv}$; and viceversa, given any original query $q_{B_j} \in \widetilde{f}$ over B, for the rewritten query q_{A_i} holds that $q_{A_i} = Tf^{inv} \circ q_{B_j} \circ f$. Also, from commutative diagram which represents a homomorphism f of $T_w - coalgebras$, we obtain that $q_{B_j} = T_w f \circ q_{A_i} \circ f^{inv}$ and $q_{A_i} = T_w f^{inv} \circ q_{B_j} \circ f$.
The naive computation is impractical, because it requires to build the canonical database, which is in general infinite. In order to overcome the problem, the *query rewriting algorithm* [2] consists of two separate phases:
1. The algorithm transforms the original lifted query q into a new query $exp_{\mathcal{G}}(q)$ over the global schema, called the *expansion of q w.r.t. $\mathcal{G}$*, such that the answer to $exp_{\mathcal{G}}(q)$ over the retrieved global database is equal to the answer to q over the canonical database.
2. In order to avoid building the retrieved global database, that algorithm unfolds

$$
\begin{tikzcd}
 & TA \arrow[rr, "Tf"] & & TB \\
T_w A \arrow[ur, hook, "inc_A"] \arrow[rr, "T_w f"] & & T_w B \arrow[ur, hook, "inc_B"] & \\
A \arrow[u, "q_{A_{wi}}"] \arrow[uur, "q_{A_i}"] \arrow[rr, "f"] & & B \arrow[u, "q_{B_{wj}}"'] \arrow[uur, "q_{B_j}"'] \arrow[ll, bend left, "f^{inv}"] &
\end{tikzcd}
$$

Fig. 2. Query equivalence diagram in DB

$exp_{\mathcal{G}}(q)$ to a new query, called $unf_{\mathcal{M}}(exp_{\mathcal{G}}(q))$, over the source relations on the basis of $\mathcal{M}$, and than uses the unfolded query $unf_{\mathcal{M}}(exp_{\mathcal{G}}(q))$ to access the sources.

Figure 3 shows the basic idea of this approach (taken from [3]). In order to obtain the **certain answers** $q^{\mathcal{I},\mathcal{D}}$, the user lifted query q could in principle be evaluated (dashed arrow) over the (possibly infinite) canonical database $can(\mathcal{I},\mathcal{D})$, which is generated from the retrieved global database $ret(\mathcal{I},\mathcal{D})$. In turn, $ret(\mathcal{I},\mathcal{D})$ can be obtained from the source database $\mathcal{D}$ by evaluating the queries of the mapping. This query answering process instead expands the query according to the constraints in $\mathcal{G}$, unfolds it according to $\mathcal{M}$, and then evaluates it on the source database.

Let show how the symbolic diagram in Fig. 3 can be effectively represented by commutative diagrams in DB, correspondent to the homomorphisms between T-coalgebras representing equivalent queries over these three (extensional) databases: in DB category each query is represented by an arrow, and *can be composed* with arrows which semantically denote mappings and integrity constraints.

Theorem 3 *Let $\mathcal{I} = \langle \mathcal{G}, \mathcal{S}, \mathcal{M} \rangle$ be a data integration system , $\mathcal{D}$ a source database for $\mathcal{I}$, $ret(\mathcal{I},\mathcal{D})$ the retrieved global database for $\mathcal{I}$ w.r.t. $\mathcal{D}$, and $can(\mathcal{I},\mathcal{D})$ the universal (canonical) database for $\mathcal{I}$ w.r.t. $\mathcal{D}$.*
Then denotational semantics for query rewriting algorithms $exp_{\mathcal{G}}(q)$ and $unf_{\mathcal{M}}(q)$, for query expansion and query unfolding respectively, are given by two (partial) functions on T-coalgebras:

$$unf_{\mathcal{M}}(\_) \triangleq Tf_M^{inv} \circ \_ \circ f_M = T_w f_M^{inv} \circ \_ \circ f_M$$
$$exp_{\mathcal{G}}(\_) \triangleq Tf_{\Sigma}^{inv} \circ \_ \circ f_{\Sigma} = T_w f_{\Sigma}^{inv} \circ \_ \circ f_{\Sigma} \text{ and}$$
$$unf_{\mathcal{M}}(exp_{\mathcal{G}}(\_)) \triangleq T(f_{\Sigma} \circ f_M)^{inv} \circ \_ \circ (f_{\Sigma} \circ f_M) = T_w(f_{\Sigma} \circ f_M)^{inv} \circ \_ \circ (f_{\Sigma} \circ f_M)$$

where f_M and f_{Σ} are given by functorial translation of the mapping $\mathcal{M}$ and integrity constraints Σ_T, while $\circ$ is a composition of morphisms [13].

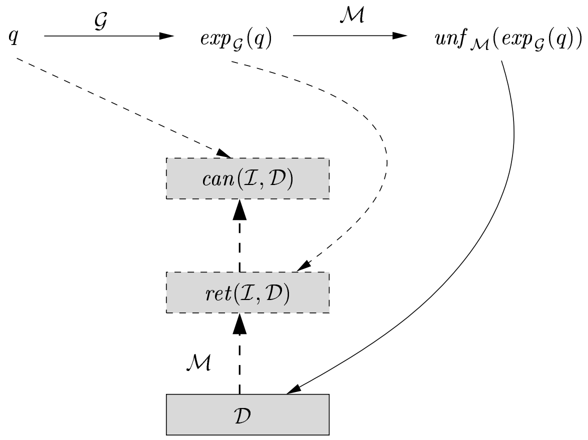

Fig. 3. Query answering process

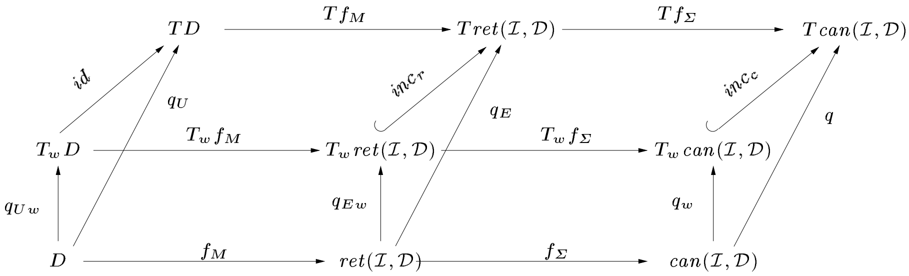

Fig. 4. T-coalgebra homomorphisms for a query answering process

Proof. Let denote by $q_E = exp_{\mathcal{G}}(q)$ and $q_U = unf_{\mathcal{M}}(exp_{\mathcal{G}}(q))$ the expanded and successively unfolded queries of the original lifted query q. Then by the query-rewriting theorem the diagrams, based on the composition of T-coalgebra homomorphisms $f_M : (\mathcal{D}, q_U) \longrightarrow (ret(\mathcal{I},\mathcal{D}), q_E)$ and $f_\Sigma : (ret(\mathcal{I},\mathcal{D}), q_E) \longrightarrow can(\mathcal{I},\mathcal{D})$, in Fig. 4 commute. It is easy to verify the first two facts. Then from composition of these two functions we obtain $unf_{\mathcal{M}}(exp_{\mathcal{G}}(\_)) = unf_{\mathcal{M}}(\_)exp_{\mathcal{G}}(\_) = Tf_M^{inv} \circ (exp_{\mathcal{G}}(\_)) \circ f_M = Tf_M^{inv} \circ (Tf_\Sigma^{inv} \circ \_ \circ f_\Sigma) \circ f_M = (Tf_M^{inv} \circ Tf_\Sigma^{inv}) \circ \_ \circ (f_\Sigma) \circ f_M) = T(f_\Sigma \circ f_M)^{inv} \circ \_ \circ (f_\Sigma \circ f_M)$ because of duality and functorial property of T. Analogously, for the weak power-view endofunctor T_w, is easy to verify that $q_{U_w} \approx q_U$, $q_{E_w} \approx q_E$ and $q_w \approx q$ are equivalent queries.

6 Conclusions

In this paper we introduced the fixpoint semantics for query answering in data integration systems with (possibly infinite) canonical models for global schemas. The main contributions of this paper can be summarized as follows:

1. A formal definition of the database information framework, given by [10] and spe-

cialised for GLAV mappings with tuple-generating dependencies [2], and a denotational semantics [13] for database mappings, with: the (strong) behavioral equivalence and the weak observational equivalence for databases.
2. A formal definition of the *finite* least fixpoint for (also infinite) canonical database models, used to obtain certain answers to the conjunctive queries: it is based on the chase and weak power-view operator.
3. We introduced the denotational semantics of the query rewriting algorithms, based on coalgebras of these two fundamental endofunctors of database category. Thus, the intuitive meta-database reasonings used in query rewriting algorithms, can be directly represented in the same semantic domain of the database category, as databases and their mappings, by simple commutative diagrams.
A further topic of future research is to generalize this approach also to other kind of data integration frameworks.

Acknowledgements This work was supported by the Project SEWASIE, funded by EU (IST-2001-34825). The author wishes to thank Maurizio Lenzerini and Andrea Calì for their support.

References

1. S. Abiteboul, R. Hull, and V. Vianu. *Foundations of Databases*. Addison Wesley Publ. Co., Reading, Massachussetts, 1995.
2. A. Calì. Reasoning in data integration systems: why lav and gav are siblings. In *Proc. of the 14th Int. Symp. on Methodologies for Intelligent Systems (ISMIS 2003)*, 2003. to appear.
3. A. Calì, D. Calvanese, G. De Giacomo, and M. Lenzerini. Data integration under integrity constraints. In *Proc. of the 14th Conf. on Advanced Information Systems Engineering (CAiSE 2002)*, pages 262–279, 2002.
4. A. Calì, D. Lembo, and R. Rosati. Query rewriting and answering under constraints in data integration systems. 2003. To appear.
5. E.Moggi. Notions of computation and monads. *Inf. and Comp.*, 93(1):55–92, 1991.
6. R. Fagin, P. G. Kolaitis, R. J. Miller, and L. Popa. Data exchange: Semantics and query answering. In *Proc. of the 9th Int. Conf. on Database Theory (ICDT 2003)*, pages 207–224, 2003.
7. M. Friedman, A. Levy, and T. Millstein. Navigational plans for data integration. In *Proc. of the 16th Nat. Conf. on Artificial Intelligence (AAAI'99)*, pages 67–73. AAAI Press/The MIT Press, 1999.
8. A. Y. Halevy. Answering queries using views: A survey. *Very Large Database J.*, 10(4):270–294, 2001.
9. D. S. Johnson and A. C. Klug. Testing containment of conjunctive queries under functional and inclusion dependencies. *J. of Computer and System Sciences*, 28(1):167–189, 1984.
10. M. Lenzerini. Data integration: A theoretical perspective. In *Proc. of the 21st ACM SIGACT SIGMOD SIGART Symp. on Principles of Database Systems (PODS 2002)*, pages 233–246, 2002.
11. S. Mac Lane. *Categories for the Working Mathematician*. Springer-Verlag, 1971.
12. Z. Majkić. Categories: symmetry, n-dimensional levels and applications. *PhD Thesis, University "La Sapienza", Roma*, 1998.
13. Z. Majkić. The category-theoretic semantics for database mappings. *Tecnical Report 1403, University 'La Sapienza', Roma*, 2003.
14. P.Cohn. *Universal algebra*. Harper and Row, 1965.