Saturation of the morphisms in the database category

Zoran Majkić
ISRST, Tallahassee, FL, USA
majk.1234@yahoo.com
http://zoranmajkic.webs.com/

Abstract. In this paper we present the problem of saturation of a given morphism in the database category $\mathbf{DB}$, which is the base category for the functorial semantics of the database schema mapping systems used in Data Integration theory. This phenomena appears in the case when we are using the Second-Order tuple-generating dependencies (SOtgd) with existentially quantified non-built-in functions, for the database schema mappings.

We provide the algorithm of the saturation for a given morphism, which represents a mapping between two relational databases, and show that the original morphism in $\mathbf{DB}$ can be equivalently substituted by its more powerful saturated version in any commutative diagram in $\mathbf{DB}$.

1 Introduction

Since the late 1960s, there has been considerable progress in understanding the algebraic semantics of logic and type theory, particularly because of the development of categorical analysis of most of the structures of interest to logicians. Although there have been other algebraic approaches to logic, none has been as far reaching in its aims and in its results as the categorical approach has been. From a fairly modest beginning, categorical logic has matured very nicely in the past four decades.

Categorical logic is a branch of category theory within mathematics, adjacent to mathematical logic but more notable for its connections to theoretical computer science [1]. In broad terms, categorical logic represents both syntax and semantics by a category, and an interpretation by a functor. The categorical framework provides a rich conceptual background for logical and type-theoretic constructions. The subject has been recognizable in these terms since around 1970.

The recent monograph [2], relevant to this paper, presents a categorical logic (denotational semantics) for database schema mapping based on views in a very general framework for database-integration/exchange and peer-to-peer. The base database category $\mathbf{DB}$ (instead of traditional $\mathbf{Set}$ category), with objects instance-databases and with morphisms (mappings which are not simple functions) between them, is used at an instance level as a proper semantic domain for a database mappings based on a set of complex query computations [2].

The higher logical schema level of mappings between databases, usually written in some high expressive logical language (ex. [3], GLAV (LAV and GAV), tuple generating dependency) can then be translated functorially into this base “computation” category.
The formal logical framework for the schema mappings is defined, based on the second-order tuple generating dependencies (SOtgds), with existentially quantified functional symbols. Each tgd is a material implication from the conjunctive formula (with relational symbols of a source schema, preceded with negation as well) into a particular relational symbol of the target schema. It was provided in [2] a number of algorithms which transform these logical formulae into the algebraic structure based on the theory of R-operads. The schema database integrity constraints are transformed in similar way so that both, the schema mappings and schema integrity-constraints, are formally represented by R-operads.

A database mapping system is represented as a graph where the nodes are the database schemas and the arrows are the schema mappings or the integrity-constraints for schemas. This representation is used to define the database mapping sketches (small categories), based on the fact that each schema has an identity arrow (mapping) and that the mapping-arrows satisfy the associative law for the composition of them.

Each Tarski’s interpretation of a logical formulae (SOtgds), used to specify the database mappings, results in the instance-database mappings composed of a set of particular functions between the source instance-database and the target instance-database. Thus, an interpretation of a database-mapping system may be formally represented as a functor from the sketch category (schema database graph) into a category where an object is an instance-database (i.e., a set of relational tables) and an arrow is a set of mapping functions. This paper is an extension of the denotational semantics for the database mappings presented in [2].

The plan of this paper is the following: In Section 2 we present the categorial logic and its functorial semantics used for the denotational semantics of the schema mappings between RDBs, based on the DB category [2]. Then, in Section 3 we provide the algorithm for the saturation of the morphisms in the category DB and we show that the saturated morphism is equal to the standard, functorially derived from a schema mapping, morphisms. Then we present two significant examples how we can use the saturation of the morphisms for the definition of 1:N relationships between RDB tables and for the parsing of RDBS into the intensional RDBs (IRDBs).

2 Functorial semantics for database mappings

A database schema is a pair \( \mathcal{A} = (S_{\mathcal{A}}, \Sigma_{\mathcal{A}}) \) where \( S_{\mathcal{A}} \) is a countable set of relational symbols (predicates in FOL) \( r \in \mathbb{R} \) with finite arity \( n = ar(r) \geq 1 \) (\( ar : \mathbb{R} \to \mathbb{N} \)).

A domain \( \mathcal{D} \) is a nonempty finite set of individual symbols. A relation symbol \( r \in \mathbb{R} \) represents the relational name and can be used as an atom \( r(x) \) of FOL with variables in \( x = \langle x_1, ..., x_{ar(r)} \rangle \) (taken from a given set of variables \( x_i \in \mathcal{V} \)) assigned to its columns, so that \( \Sigma_{\mathcal{A}} \) denotes a set of sentences (FOL formulae without free variables) called integrity constraints.

An instance-database of a nonempty schema \( \mathcal{A} \) is given by \( A = (\mathcal{A}, I_T) = \{ R = \|r\| = I_T(r) \mid r \in S_{\mathcal{A}} \} \) where \( I_T \) is a Tarski’s FOL interpretation which satisfies all integrity constraints in \( \Sigma_{\mathcal{A}} \) and maps a relational symbol \( r \in S_{\mathcal{A}} \) into an n-ary relation \( R = \|r\| \in A \). Thus, an instance-database \( A \) is a set of n-ary relations, managed by relational database systems. We denote by \( r_0 \) a nullary relational symbol corresponding
logically to a propositional symbol of a tautology, such that $\bot = ||r_0|| = \{<><>\}$ where $<>$ denotes the empty tuple. We assume that $r_0$ is part of any database schema $\mathcal{A}$.

If $A$ is an instance-database and $\phi$ is a sentence then we write $A \models \phi$ to mean that $A$ satisfies $\phi$. If $\Sigma$ is a set of sentences then we write $A \models \Sigma$ to mean that $A \models \phi$ for every sentence $\phi \in \Sigma$. Thus the set of all instances of $\mathcal{A}$ is defined by $\text{Inst}(\mathcal{A}) = \{A \mid A \models \Sigma_A\}$.

We consider a rule-based conjunctive query over a database schema $\mathcal{A}$ as an expression $q(x) \leftarrow r_1(u_1),...,r_n(u_n)$, with finite $n \geq 0$, $r_i$ are the relational symbols (at least one) in $\mathcal{A}$ or the built-in predicates (e.g., $\leq$, $\equiv$, etc.), $q$ is a relational symbol not in $\mathcal{A}$ and $u_i$ are free tuples (i.e., one may use either variables or constants). Recall that if $v = (v_1,...,v_m)$ then $r(v)$ is a shorthand for $r(v_1,...,v_m)$. Finally, each variable occurring in $x$ is a distinguished variable that must also occur at least once in $u_1,...,u_n$. Rule-based conjunctive queries (called rules) are composed of a subexpression $r_1(u_1),...,r_n(u_n)$ that is the body, and the head of this rule $q(x)$. The deduced head-facts of a conjunctive query $q(x)$ defined over an instance $A$ (for a given Tarski’s interpretation $I_T$ of schema $\mathcal{A}$) are equal to $||q(x_1,...,x_k)|| = \{<v_1,...,v_k> \in \mathcal{D}^k \mid A \models \exists y (r_1(u_1) \land \ldots \land r_n(u_n))\} = I_T^k(\exists y (r_1(u_1) \land \ldots \land r_n(u_n)))$, where the $y$ is a set of variables which are not in the head of query, and $I_T^k$ is the unique extension of $I_T$ to all formulae. Each conjunctive query corresponds to a “select-project-join” term $t(x)$ of SPRJU algebra obtained from the formula $\exists y (r_1(u_1) \land \ldots \land r_n(u_n))$.

We consider a finitary view as a union of a finite set $S$ of conjunctive queries with the same head $q(x)$ over a schema $\mathcal{A}$, and from the equivalent algebraic point of view, it is a “select-project-join-rename + union” (SPRJU) finite-length term $t(x)$ which corresponds to union of the terms of conjunctive queries in $S$. In what follows we will use the same notation for a FOL formula $q(x)$ and its equivalent algebraic SPRJU expression $t(x)$. A materialized view of an instance-database $A$ is an n-ary relation $R = \bigcup_{q(x) \in S} \{q(x)\}$. We denote the set of all finitary materialized views that can be obtained from an instance $A$ by $TA$.

We consider that a mapping between two database schemas $\mathcal{A} = (S_\mathcal{A}, \Sigma_\mathcal{A})$ and $\mathcal{B} = (S_\mathcal{B}, \Sigma_\mathcal{B})$ is expressed by an union of "conjunctive queries with the same head". Such mappings are called "view-based mappings", defined by a set of FOL sentences $\{\forall x_i (q_{\mathcal{A}_i}(x_i) \Rightarrow q_{\mathcal{B}_i}(y_i))\}$ with $y_i \subseteq x_i, 1 \leq i \leq n$, where $\Rightarrow$ is the logical implication between these conjunctive queries $q_{\mathcal{A}_i}(x_i)$ and $q_{\mathcal{B}_i}(x_i)$, over the databases $\mathcal{A}$ and $\mathcal{B}$, respectively. Schema mappings are often specified by the source-to-target tuple-generating dependencies (tgds), used to formalize a data exchange [4], and in the data integration scenarios under a name "GLAV assertions" [45]. A tgd is a logical sentence (FOL formula without free variables) which says that if some tuples satisfying certain equalities exist in the relation, then some other tuples (possibly with some unknown values) must also exist in another specified relation.

An equality-generating dependency (egd) is a logical sentence which says that if some tuples satisfying certain equalities exist in the relation, then some values in these tuples must be equal. Functional dependencies are egds of a special form, for example primary-key integrity constraints. Thus, egds are only used for the specification of integrity constraints of a single database schema, which define the set of possible models of this database. They are not used for inter-schema database mappings.
These two classes of dependencies together comprise the embedded implication dependencies (EID) \cite{6} which seem to include essentially all of the naturally-occurring constraints on relational databases (we recall that the bold symbols \(x, y, \ldots\) denote a nonempty list of variables):

**Definition 1.** We introduce the following two kinds of EIDs \cite{6}:

1. A tuple-generating dependency (tgd) \(\forall x(q_A(x) \Rightarrow q_B(x))\), where \(q_A(x)\) is an existentially quantified formula \(\exists y \phi_A(x, y)\) and \(q_B(x)\) is an existentially quantified formula \(\exists z \psi_A(x, z)\), and where the formulae \(\phi_A(x, y)\) and \(\psi_A(x, z)\) are conjunctions of atomic formulae (conjunctive queries) over the given database schemas. We assume the safety condition, that is, that every distinguished variable in \(x\) appears in \(q_A\).

We will consider also the class of weakly-full tgds for which query answering is decidable, i.e., when \(q_B(x)\) has no existentially quantified variables, and if each \(y_i \in y\) appears at most once in \(\phi_A(x, y)\).

2. An equality-generating dependency (egd) \(\forall x(q_A(x) \Rightarrow (y \equiv z))\), where \(q_A(x)\) is a conjunction of atomic formulae over a given database schema, and \(y = <y_1, \ldots, y_k>, z = <z_1, \ldots, z_k>\) are among the variables in \(x\), and \(y \equiv z\) is a shorthand for the formula \((y_1 \equiv z_1) \land \ldots \land (y_k \equiv z_k)\) with the built-in binary identity predicate \(\equiv\) of the FOL.

Note that a tgd \(\forall x(\exists y \phi_A(x, y) \Rightarrow \exists z \psi_A(x, z))\) is logically equivalent to the formula \(\forall x \forall y(\phi_A(x, y) \Rightarrow \exists z \psi_A(x, z))\), i.e., to \(\forall x_1(\phi_A(x_1) \Rightarrow \exists z \psi_A(x_1, z))\) with the set of distinguished variables \(x \subseteq x_1\).

We use for the integrity constraints \(\Sigma_A\) of a database schema \(A\) both tgds and egds, while for the inter-schema mappings, between a schema \(A = (S_A, \Sigma_A)\) and a schema \(B = (S_B, \Sigma_B)\), only the tgds \(\forall x(q_A(x) \Rightarrow q_B(x))\). So called second-order tgds (SO tgds), has been introduced in \cite{7} as follows:

**Definition 2.** \cite{7} Let \(A\) be a source schema and \(B\) a target schema. A second-order tuple-generating dependency (SO tgd) is a formula of the form:

\[\exists f((\forall x_1(\phi_1 \Rightarrow \psi_1)) \land \ldots \land (\forall x_n(\phi_n \Rightarrow \psi_n))),\]

where

1. Each member of the tuple \(f\) is a functional symbol.
2. Each \(\phi_i\) is a conjunction of:
   - atomic formulae of the form \(r_A(y_1, \ldots, y_k)\), where \(r_A \in S_A\) is a k-ary relational symbol of schema \(A\) and \(y_1, \ldots, y_k\) are variables in \(x_i\), not necessarily distinct;
   - the formulae with conjunction and negation connectives and with built-in predicate’s atoms of the form \(t \odot t’\), \(\odot \in \{=, <, >, \ldots\}\), where \(t\) and \(t’\) are the terms based on \(x_i, f\) and constants.
3. Each \(\psi_i\) is a conjunction of atomic formulae \(r_B(t_1, \ldots, t_m)\) where \(r_B \in S_B\) is an m-ary relational symbol of schema \(B\) and \(t_1, \ldots, t_m\) are terms based on \(x_i, f\) and constants.
4. Each variable in \(x_i\) appears in some atomic formula of \(\phi_i\).

Notice that each constant \(\overline{a}\) in an atom on the left-hand side of implications must be substituted by new fresh variable \(y_i\) and by adding a conjunct \((y_i = \overline{a})\) in the left-hand
side of this implication, so that such atoms will have only the variables (condition 2 above). For the empty set of tgds, we will use the SOtgds tautology \( r_0 \Rightarrow r_0 \). The forth condition is a “safety” condition, analogous to that made for (first-order) tgds. It is easy to see that every tgd is equivalent to one SOtgds without equalities. For example, let \( \sigma \) be the tgd \( \forall x_1... \forall x_m(\phi_A(x_1,...,x_m)) \Rightarrow \exists y_1... \exists y_n \psi_B(x_1, ..., x_m, y_1, ..., y_n) \).

It is logically equivalent to the following SOtgds without equalities, which is obtained by Skolemizing existential quantifiers in \( \sigma \):

\[
\exists f_1... \exists f_n (\forall x_1... \forall x_m (\phi_A(x_1,...,x_m) \Rightarrow \psi_B(x_1, ..., x_m, f_1(x_1, ..., x_m), ..., f_n(x_1, ..., x_m))))
\]

Given a finite set \( S \) of tgds of an inter-schema mapping, we can find a single SOtgds that is equivalent to \( S \) by taking, for each tgd \( \sigma \) in \( S \), a conjunct of the SOtgds to capture \( \sigma \) as described above (we use disjoint sets of function symbols in each conjunct, as before).

The simultaneous inductive definition of the set \( TX \) of terms is as follows:

1. All variables \( X \subseteq V \) and constants are terms.
2. If \( t_1, ..., t_k \) are terms and \( f_i \) a k-ary functional symbol then \( f_i(t_1, ..., t_k) \) is a term.

An assignment \( g : V \rightarrow D \) for variables in \( V \) is applied only to free variables in terms and formulae. Such an assignment \( g \in D^V \) can be recursively uniquely extended into the assignment \( g^* : TX \rightarrow D \), where \( TX \) denotes the set of all terms with variables in \( X \subseteq V \), by:

1. \( g^*(t_k) = g(x) \in D \) if the term \( t_k \) is a variable \( x \in V \).
2. \( g^*(t_k) = c \in D \) if the term \( t_k \) is a constant (nullary functional symbol) \( \tau \), with \( g^*(\top) = 1 \) for the truth-constant \( \top \).
3. \( g^*(f_i(t_1,...,t_k)) = I_T(f_i)(g^*(t_1),...,g^*(t_k)) \in D \), where \( I_T(f_i) \) is a function obtained by Tarski’s interpretation of the functional symbol \( f_i \).

We denote by \( t_k/g \) (or \( \phi/g \)) the ground term (or formula) without free variables, obtained by assignment \( g \) from a term \( t_k \) (or a formula \( \phi \)), and by \( \phi[x/t_k] \) the formula obtained by uniformly replacing \( x \) by a term \( t_k \) in \( \phi \).

In what follows we use the algorithm \text{MakeOperads} in \[2\] in order to transform logical schema mappings \( M_{AB} = \{ \Phi \} : A \rightarrow B \) given by the SOtgds \( \Phi \) in Definition\[2\] into the algebraic operads \( M_{AB} = \text{MakeOperad}(M_{AB}) = \{ q_{A,i} \cdot r, ..., q_{A,1} \cdot r \cdot 1 \} : A \rightarrow B \). The basic idea of the operad’s operations \( v_i \in O(r', r_B) \) and \( q_{A,i} \in O(v_1, ..., v_k, r') \), where \( r_i, 1 \leq i \leq k \) are relational symbols of the source schema \( A = (S_A, \Sigma_A) \) and \( r_B \) is a relational symbol of the target schema \( B \), and \( r' \) has the same type as \( r_B \), is to formalize algebraically a mapping from the set of source relations \( r_i \) into a target relation \( r_B \).

**Example 1:** Schema \( A = (S_A, \emptyset) \) consists of a unary relation \( \text{EmpAcme} \) that represents the employees of Acme, a unary relation \( \text{EmpAjax} \) that represents the employees of Ajax, and unary relation \( \text{Local} \) that represents employees that work in the local office of their company. Schema \( B = (S_B, \emptyset) \) consists of a unary relation \( \text{Emp} \) that represents all employees, a unary relation \( \text{Local1} \) that is intended to be a copy of \( \text{Local} \), and unary relation \( \text{Over65} \) that is intended to represent people over age 65.

Schema \( C = (S_C, \emptyset) \) consists of a binary relation \( \text{Office} \) that associates employees with office numbers and unary relation \( \text{CanRetire} \) that represents employees eligible for retirement. Consider now the following schema mappings:
\[ M_{AB} = \{ \forall x.e_{\text{EmpAcme}}(x) \Rightarrow \text{Emp}(x) \} \land \forall x.e_{\text{EmpAjax}}(x) \Rightarrow \text{Emp}(x) \} \land \exists p.(\text{Local}(x) \Rightarrow \text{Local}(x)) \}
and
\[ M_{BC} = \{ \exists f.\forall x.(\text{Emp}(x) \land \text{Local}(x)) \Rightarrow \text{Office}(x, f_1(x)) \} \land \forall x.e_{\text{Emp}(x) \land \text{Over65}(x)) \Rightarrow \text{CanRetire}(x)) \} \right \}.
\]

Then, by their composition, we obtain the composed mapping \(M_{AC} : A \rightarrow C\) equal to
\[ M_{AC} = \{ \exists f_1 \exists f_2 \exists f_{\text{Over65}}(\forall x.e_{\text{EmpAcme}}(x) \land \text{Local}(x)) \Rightarrow \text{Office}(x, f_1(x)) \} \land \forall x.e_{\text{EmpAjax}}(x) \land \text{Local}(x) \Rightarrow \text{Office}(x, f_2(x)) \} \land \forall x.e_{\text{Emp}(x) \land \text{Over65}(x)) \Rightarrow \text{CanRetire}(x)) \} \right \}.
\]

where \(f_{\text{Over65}}\) is the characteristic function of the relation (predicate) \text{Over65}\. Which is not part of schema \(A\). Then, by transformation into abstract operad’s operations, we obtain
\[ M_{AC} = \text{MakeOperads}(M_{AC}) = \{ q_1^A, q_2^A, q_3^A, q_4^A, 1, n\}, q_i^A = v_i q_{A,i}, \]
where:
1. The operations \(q_1^A \in O(\text{EmpAcme, Local, Office})\ and \(q_{A,1} \in O(\text{EmpAcme, Local, } A)\) correspond to the expression \((\exists_1(x, e) \land \exists_2(x, e)) \Rightarrow (\exists_1(x, e, f_1(x))) and \(v_1 \in O(r_1^A, \text{Office})\) to \(((\exists_1(x, e) \land \exists_2(x, e)) \Rightarrow (\exists_1(x, e, f_1(x))))\);
2. The operations \(q_2^A \in O(\text{EmpAjax, Local, Office})\ and \(q_{A,2} \in O(\text{EmpAjax, Local, } A)\) correspond to the expression \((\exists_1(x, e) \land \exists_2(x, e)) \Rightarrow (\exists_1(x, e, f_2(x))) and \(v_2 \in O(r_2^A, \text{Office})\) to \(((\exists_1(x, e) \land \exists_2(x, e)) \Rightarrow (\exists_1(x, e, f_2(x))))\);
3. The operations \(q_3^A \in O(\text{EmpAcme, Over65, CanRetire})\ and \(q_{A,3} \in O(\text{EmpAcme, Over65, } A)\) correspond to the expression \((\exists_1(x, e) \land \exists_2(x, e)) \Rightarrow (\exists_1(x, e)) and \(v_3 \in O(r_3^A, \text{CanRetire})\) to \(((\exists_1(x, e) \land \exists_2(x, e)) \Rightarrow (\exists_1(x, e))))\);
4. The operations \(q_4^A \in O(\text{EmpAjax, Over65, CanRetire})\ and \(q_{A,4} \in O(\text{EmpAjax, Over65, } A)\) correspond to the expression \((\exists_1(x, e) \land \exists_2(x, e)) \Rightarrow (\exists_1(x, e)) and \(v_4 \in O(r_4^A, \text{CanRetire})\) to \(((\exists_1(x, e) \land \exists_2(x, e)) \Rightarrow (\exists_1(x, e))))\).

These three arrows \(M_{AB} : A \rightarrow B, M_{BC} : B \rightarrow C\ and \(M_{AC} : A \rightarrow C\) compose a graph \(G\) of this database mapping system. From the fact that the operads can be composed, the composition of two schema mappings \(M_{AB}\) and \(M_{BC}\) can be translated into composition of operads which is associative, so that they can be represented by the sketch category \(\text{Sch}(G)\) derived from the graph \(G\) of the schema mappings.

Sketches are called graph-based logic and provide very clear and intuitive specification of computational data and activities. For any small sketch \(E\), the category of models \(\text{Mod}(E)\) is an accessible category by Lair’s theorem and reflexive subcategory of \(\text{Set}\) by Ehresmann-Kennison theorem. A generalization to base categories other than \(\text{Set}\) was proved by Freyd and Kelly (1972) [8]. The generalization to \(\text{DB}\) category is exhaustively provided in [2], so that the functorial semantics of a database mapping system expressed by a graph \(G\) is defined by a functor (R-algebra) \(\alpha^* : \text{Sch}(G) \rightarrow \text{DB}\.

The R-algebra \(\alpha\) is derived from a given Tarski’s interpretation \(I_T\) of the given database schema mapping graph \(G\) and represented by a sketch category \(\text{Sch}(G)\) (with arrows \(M_{AB} : A \rightarrow B\, as\ in\ Example\ 1\). R-algebra \(\alpha\) is equal to \(I_T\) for the relations of the data schemas, \(\alpha(r_1) = I_T(r_1)\ is\ a\ relational\ table\ of\ the\ instance\ database\ \(A = \alpha^*(A) = \{ \alpha(r_1) \mid r_1 \in S_A \}\) \(\alpha^*\ denotes\ the\ extension\ of\ \alpha\ to\ sets\), and \(\alpha(q_{A,i}) : \alpha(r_1) \times \ldots \times \alpha(r_k) \rightarrow \alpha(r')\ is\ a\ surjective\ function\ from\ the\ relations\ in\ the\ instance\ database\ \(A\)\ into\ its\ image\ (relation)\ \(\alpha(r')\), with\ a\ function\ \(\alpha(n_i) : \alpha(r') \rightarrow \alpha(r_B)\ into
the relation of the instance database $B = \alpha^*(B)$.

We have that for any R-algebra $\alpha$, $\alpha(r_b) = \bot = \{<>\}$ is the empty relation composed by only empty tuple $<>\in D_{-1}$, and $1_{r_b}$ is the identity operads operation of the empty relation $r_b$, so that $q_L = \alpha(1_{r_b}) = id_{\bot} : \bot \rightarrow \bot$ is the identity function.

**Example 2:** For the operads defined in Example 1, let a mapping-interpretation (an R-algebra) $\alpha$ be an extension of Tarski’s interpretation $I_T$ of the source schema $A = (S_A, \Sigma_A)$ that satisfies all constraints in $\Sigma_A$ and defines its database instance $A = \alpha^*(S_A) = \{\alpha(r_i) | r_i \in S_A\}$ and, analogously, an interpretation of $C$.

Let $\alpha$ satisfy the SOtgd of the mapping $M_{AC}$ by the Tarski’s interpretation for the functional symbols $f_i$, for $1 \leq i \leq 2$, in this SOtgd (denoted by $I_T(f_i)$).

Then we obtain the relations $\alpha(EmpAcme), \alpha(EmpAjax), \alpha(Local), \alpha(Office)$ and $\alpha(CanRetire)$. The interpretation of $f_{\text{Over65}}$ is the characteristic function of the relation $\alpha(\text{Over65})$ in the instance $B = \alpha^*(S_B)$ of the database $B = (S_B, \Sigma_B)$, so that $\alpha(\text{Over65})(a) = 1$ if $a > a \in \alpha(\text{Over65})$.

Then this mapping interpretation $\alpha$ defines the following functions:

1. The function $\alpha(q_{A,1}) : \alpha(EmpAcme) \times \alpha(Local) \rightarrow \alpha(r_1')$, such that for any tuple $<a, b> \in \alpha(EmpAcme)$ and $b \in \alpha(Local)$,
   
   $\alpha(q_{A,1})(<a, b>) = \{<a, I_T(f_1(a))> \text{ if } a = b; <> \text{ otherwise.}
   
   \text{And for any } <a, b> \in \alpha(r_1'), \alpha(v_1)(<a, b>) = <a, b> \text{ if } a, b \in \alpha(Office); <> \text{ otherwise.}

2. The function $\alpha(q_{A,2}) : \alpha(EmpAjax) \times \alpha(Local) \rightarrow \alpha(r_2')$, such that for any tuple $<a, b> \in \alpha(EmpAjax)$ and $b \in \alpha(Local)$,
   
   $\alpha(q_{A,2})(<a, b>) = \{<a, I_T(f_2(a))> \text{ if } a = b; <> \text{ otherwise.}
   
   \text{And for any } <a, b> \in \alpha(r_2'), \alpha(v_2)(<a, b>) = <a, b> \text{ if } a, b \in \alpha(Office); <> \text{ otherwise.}

3. The function $\alpha(q_{A,3}) : \alpha(EmpAcme) \times \alpha(\text{Over65}) \rightarrow \alpha(r_3')$, such that for any tuple $<a, b> \in \alpha(EmpAcme)$ and $b \in \alpha(\text{Over65})$,
   
   $\alpha(q_{A,3})(<a, b>) = <a, b>, \text{ if } a = b; <> \text{ otherwise.}
   
   \text{And for any } <a> \in \alpha(r_3'), \alpha(v_3)(<a>) = <a> \text{ if } a \in \alpha(\text{CanRetire}); <> \text{ otherwise.}

4. The function $\alpha(q_{A,4}) : \alpha(EmpAjax) \times \alpha(\text{Over65}) \rightarrow \alpha(r_4')$, such that for any tuple $<a, b> \in \alpha(EmpAjax)$ and $b \in \alpha(\text{Over65})$,
   
   $\alpha(q_{A,4})(<a, b>) = <a, b>, \text{ if } a = b; <> \text{ otherwise.}
   
   \text{And for any } <a> \in \alpha(r_4'), \alpha(v_4)(<a>) = <a> \text{ if } a \in \alpha(\text{CanRetire}); <> \text{ otherwise.}

From the fact that the mapping-interpretation satisfies the schema mappings, based on Corollary 4 in Section 2.4.1 [2], all functions $\alpha(v_i)$, for $1 \leq i \leq 4$, are the injections.

Formal definition of an R-algebra $\alpha$ as a mapping-interpretation of a schema mapping $M_{AB} : A \rightarrow B$ is given in [2] (Section 2.4.1, Definition 11) as follows:

**Definition 3.** Let $\phi_{\alpha}(x) \Rightarrow r_B(t)$ be an implication $\chi$ in a normalized SOtgd $\exists(\Psi)$ (where $\Psi$ is a FOL formula) of the mapping $M_{AB}$, $t$ be a tuple of terms with variables in $x = <x_1, ..., x_m>$, and $q_t \in \text{MakeOperads}(M_{AB})$ be the operad’s operation of this implication obtained by MakeOperads algorithm, equal to the expression
(e ⇒ (φ)(t)) ∈ O(r₁,...,rₖ,r₉), where qᵢ = vᵢ · qₐᵢ with qₐᵢ ∈ O(r₁,...,rₖ,r₉) and vᵢ ∈ O(r₉,r₉) such that for a new relational symbol r₉, ar(r₉) = ar(r₉) ≥ 1.

Let S be an empty set and e([φ]_n/r₉) ≤ n ≤ k be the formula obtained from expression e where each place-symbol (φ)ᵢ is substituted by relational symbol rₙ for 1 ≤ n ≤ k.

Then do the following as far as it is possible: For each two relational symbols rₐ, rₙ in the formula e([φ]_n/r₉) ≤ n ≤ k such that jₐ-th free variable (which is not an argument of a functional symbol) in the atom rₐ(tₐ) is equal to nₙ-th free variable in the atom rₙ(tₙ), we insert the set {(jₐ,jₐ),(nₙ,nₙ)} as one element of S. At the end, S is the set of sets that contain the pairs of mutually equal free variables. An R-algebra α is a mapping-interpretation of M_AB : A → B if it is an extension of a Tarski's interpretation Iₚ, of all predicate and functional symbols in FOL formula Ψ, with Iₚ being its extension to all formulae), and if for each qᵢ ∈ MakeOperads(M_AB) it satisfies the following:

1. For each relational symbol rᵢ ≠ rₙ in A or B, α(rᵢ) = Iₚ(rᵢ).
2. We obtain a function f = α(qₐᵢ) : Rᵢ × ... × Rₖ → α(r₉),

where for each 1 ≤ i ≤ k, Rᵢ = D_{ar(rᵢ)}(α(rᵢ)) if the place symbol (φ)ᵢ ∈ qᵢ is preceded by negation operator ¬: α(rᵢ) otherwise, such that for every dᵢ ∈ Rᵢ:

f(<a₁,...,aₖ>) = gᵢ(tᵢ) = <gᵢ(t₁),...,gᵢ(t₉)>

if \( \bigwedge \{πₗₜ(a_j) = πₗₜ(aₗₜ) : (j₉,jₗₜ) ∈ S \} \) is true and the assignment g satisfies the formula e([φ]_n/r₉) ≤ n ≤ k: (empty tuple) otherwise,

where the assignment g : \{x₁,...,xₖ\} → D is defined by the tuple of values <g(x₁),...,g(xₖ) > = Cmp(S,<d₁,...,dₖ >), and its extension g* to all terms such that for any term fᵢ(t₁,...,tₙ):

\[ g^*(fᵢ(t₁,...,tₙ)) = \text{if } n > 1; Iₚ(f₁) \text{ otherwise.} \]

The algorithm Cmp (compacting the list of tuples by eliminating the duplicates defined in S) is defined as follows:

Input: a set S of joined (equal) variables defined above, and a list of tuples < d₁,...,dₖ >.

Initialize d to d_1. Repeat consecutively the following, for j = 2,...,k:

Let d_j by a tuple of values <v₁,...,v₉ >, then for i = 1,...,jₙ repeat consecutively the following:

d = d & vᵢ if there does not exist element {(j₉,jₗₜ),(n₉,nₗₜ)} in S such that jₗₜ ≤ j; d otherwise.

(The operation of concatenation ‘&’ appends the value vᵢ at the end of tuple d)

Output: The tuple Cmp(S,<d₁,...,dₖ >) = d.

3. α(r₉) is equal to the image of the function f in point 2 above.

4. The function h = α(vᵢ) : α(r₉) → α(r₉) such that for each b ∈ α(r₉),

h(b) = b if b ∈ α(r₉); empty tuple <> otherwise.

Note that the formulae φₐᵢ(x) and expression e([φ]_n/r₉) ≤ n ≤ k are logically equivalent, with the only difference that the atoms with characteristic functions fᵢ(t) in the first formula are substituted by the atoms r(t), based on the fact that the assignment g satisfies r(t) if g*(fᵢ(t)) = T for every assignment g(T) = 1, where T : D_{ar(r)} → \{0,1\} is the characteristic function of relation α(r) such that for each tuple c ∈ D_{ar(r)}, T(c) = 1 if c ∈ α(r); 0 otherwise.
Example 3: Let us show how we construct the set $S$ and the compacting of tuples given by Definition 3 above:

Let us consider an operad $q_i \in MakeOperads(M_{AB})$, obtained from a normalized implication $\phi_{AB}(x) \Rightarrow r_B(t)$ in $M_{AB}$, \((y = f_1(x,z)) \land r_1(x,y,z) \land r_2(v,x,w) \land (f_3(y,z,w',w) = T)\), so that $q_i$ is equal to the expression \((e \Rightarrow (\_))(t) \in O(r_1,r_2,r_3,r_B)\), where $x = < x, y, z, v, w, w' >$ (the ordering of variables in the atoms (with database relational symbols) from left to right), $t = < x, z, w, f_2(v, z) >$, and the expression $e$ equal to \((y = f_1(x,z)) \land (\_)(1)(t_1) \land (\_)(2)(t_2) \land (\_)(3)(t_3)\), with $t_1 = < x, y, z >$, $t_2 = < v, x, w >$ and $t_3 = < y, z, w', w >$. Consequently, we obtain,

\[ S = \{(1,1), (2,2), (2,1), (1,3), (3,1), (2,3), (3,2), (4,3)\}, \]

that are the positions of duplicates (or joined variables) of $x, y, z$ and $w$ respectively.

Thus, for given tuples $d_1 = < a_1, a_2, a_3 > \in \alpha(r_1), d_2 = < b_1, b_2, b_3 > \in \alpha(r_2)$ and $d_3 = < c_1, c_2, c_3, c_4 > \in \alpha(r_3)$, the expression $\land \{\pi_jh(d_j) = \pi_nh(d_n)\} \{\{j, h, n\} \in \{1, 2, 3\}\}$ in $S$ is equal to $(\pi_1(d_1) = \pi_2(d_2)) \land (\pi_2(d_1) = \pi_1(d_3)) \land (\pi_3(d_2) = \pi_3(d_3))$, which is true when $a_1 = b_2, a_2 = c_1, a_3 = c_2$ and $b_4 = c_4$.

The compacting of these tuples is equal to $d = Comp(S, < d_1, d_2, d_3 >) = < a_1, a_2, a_3, b_1, b_3, c_1, c_2, c_3 >$, with the assignment to variables $[x/a_1], [y/a_2], [z/a_3], [v/b_1], [w/b_3]$ and $[w'/c_3]$.

That is, $d = \pi(x/a_1, y/a_2, z/a_3, v/b_1, w/b_3, w'/c_3)$ is obtained by this assignment $g$ to the tuple of variables $x$, so that the sentence $e(\{r_n\}_n \leq n \leq k)/g$ is well defined and equal to:

$\langle a_2 = I_T(f_1)(a_1, a_3) \land r_1(a_1, a_2, a_3) \land r_2(b_1, a_1, b_3) \land r_3(a_2, a_3, c_3, b_3) \rangle$, that is to say $\langle a_2 = I_T(f_1)(a_1, a_3) \land r_1(d_1) \land r_2(d_2) \land r_3(d_3) \rangle$, and if this formula is satisfied by such an assignment $g$, then $I_T(e(\{r_n\}_n \leq n \leq k)/g) = 1$, then

\[ f(\langle d_1, d_2, d_3 >) = g^*(t) = < g(x), g(z), g(w), g^*(f_2(v, z)) > \]

for a given Tarski’s interpretation $I_T$, where $I_T$ is the extension of $I_T$ to all FOL formulae.

If $M_{AB}$ is satisfied by the mapping-interpretation $\alpha$, this value of $f(\langle d_1, d_2, d_3 >)$ corresponds to the truth of the normalized implication in the SOtg of $M_{AB}, \phi_{AB}(x) \Rightarrow r_B(t)$ for the assignment $g$ derived by substitution $[x/d]$, when $\phi_{AB}(x)/g$ is true. Hence, $r_B(t)/g$ is equal to $r_B(< a_1, a_3, b_1, I_T(f_2)(b_1, a_3) >)$, i.e., to $r_B(f(< d_1, d_2, d_3 >))$ and has to be true as well (i.e. $I_T^*(r_B(f(< d_1, d_2, d_3 >))) = 1$ or, equivalently, $f(\langle d_1, d_2, d_3 >) \in \alpha(r_B) = I_T(r_B)$).

Consequently, if $M_{AB}$ is satisfied by a mapping-interpretation $\alpha$ and hence $\alpha(v_i)$ is an injection function with $\alpha(r_q) \subseteq \alpha(r_B)$ then $f(\langle d_1, d_2, d_3 >) \in \|r_B\|$, so that the function $f = \alpha(q_{AB})$ represents the transferring of the tuples in relations of the source instance databases into the target instance database $B = \alpha^*(B)$, according to the SOtg $\Phi$ of the mapping $M_{AB} = \{\Phi\} : A \rightarrow B$.

In this way, for a given $R$-algebra $\alpha$ which satisfies the conditions for the mapping-interpretations in Definition 3 we translate a logical representation of database mappings, based on SOtgds, into an algebraic representation based on relations of the instance databases and the functions obtained from mapping-operads.

\[ \square \]
It is easy to verify that for a query mapping $\phi_{A_i}(x) \Rightarrow r_B(t)$, a mapping-interpretation $\alpha$ is an R-algebra such that the relation $\alpha(r_q)$ is just equal to the image of the function $\alpha(q_{A_i})$. The mapping-interpretation of $v_i$ is the transfer of information of this computed query into the relation $\alpha(r_B)$ of the database $B$.

When $\alpha$ satisfies this query mapping $\phi_{A_i}(x) \Rightarrow r_B(t)$, then $\alpha(r_q) \subseteq \alpha(r_B)$ and, consequently, the function $\alpha(v_i)$ is an injection, i.e., the inclusion of $\alpha(r_q)$ into $\alpha(r_B)$.

Moreover, each R-algebra $\alpha$ of a given set of mapping-operads between a source schema $A$ and target schema $B$ determines a particular information flux from the source into the target schema.

**Definition 4. Information Flux**

Let $\alpha$ be a mapping-interpretation (an R-algebra in Definition 3) of a given set $M_{AB} = \{q_1, ..., q_n, 1_{r_q}\} = \text{MakeOperads}(M_{AB})$ of mapping-operads, obtained from an atomic mapping $M_{AB} : A \to B$, and $A = \alpha^*(S_A)$ be an instance of the schema $A = (S_A, \Sigma_A)$ that satisfies all constraints in $\Sigma_A$.

For each operation $q_i \in M_{AB}$, let $q_i = (e \Rightarrow (\ldots), t_i) \in O(r_{i,1}, ..., r_{i,k}, t_i)$, let $x_i$ be its tuple of variables which appear at least one time free (not as an argument of a function) in $t_i$ and appear as variables in the atoms of relational symbol of the schema $A$ in the formula $e([\ldots]_{j/r_{i,j}})_{1 \leq j \leq k}$. Then, we define

1. $\text{Var}(M_{AB}) = \bigcup_{1 \leq i \leq n} \{x_i \mid x_i \in x_i\}$.
2. $\Delta(\alpha, M_{AB}) = \{\pi_i(\text{im}(\alpha(q_i))) \mid q_i \in M_{AB}, \text{ and } x_i \text{ is not empty} \} \bigcup \perp^0, \text{ if } \text{Var}(M_{AB}) \neq \emptyset; \perp^0 \text{ otherwise.}$

We define the information flux by its kernel by

3. $\text{Flux}(\alpha, M_{AB}) = T(\Delta(\alpha, M_{AB}))$.

The flux of composition of $M_{AB}$ and $M_{BC}$ is defined by:

$\text{Flux}(\alpha, M_{BC} \circ M_{AB}) = \text{Flux}(\alpha, M_{AB}) \bigcap \text{Flux}(\alpha, M_{BC})$.

4. We say that an information flux is empty if it is equal to $\perp^0 = \{\perp\}$ (and hence it is not the empty set), analogously as for an empty instance-database.

The information flux of the SOtdg of the mapping $M_{AB}$ for the instance-level mapping $f = \alpha^*(M_{AB}) : A \to \alpha^*(B)$ composed of the set of functions $f = \alpha^*(M_{AB}) = \{\alpha(q_1), ..., \alpha(q_n), q_{\perp}\}$, is denoted by $f$. Notice that $\perp \in f$, and hence the information flux $f$ is an instance-database as well.

From this definition, each instance-mapping is a set of functions whose information flux is the intersection of the information fluxes of all atomic instance-mappings that compose this composed instance-mapping. These basic properties of the instance-mappings is used in order to define the database $DB$ category where the instance-mappings are the morphisms (i.e., the arrows) of this category, while the instance-databases (each instance-database is a set of relations of a schema also with the empty relation $\perp$) are its objects.

**Equality of morphisms:** The fundamental property in $DB$ is the following:

Any two arrows $f, g : A \to B$ where $A$ and $B$ are the instance databases (the simple sets of the relations) in $DB$ are equal if $f = g$, i.e., the have the same information fluxes.
3 Saturation of the morphisms in DB

Let $\phi_{AB}(x) \Rightarrow r_B(t)$, as in Definition 3, be an implication $\chi$ in a normalized SOtgd $\exists(\Psi)$ (where $\Psi$ is a FOL formula) of the mapping $M_{AB} : A \rightarrow B$ with the sketch’s arrow $M_{AB} = MakeOperads(M_{AB}) = \{q_1, ..., q_n, 1_{r_B}\}$, $t = \langle t_1, ..., t_{ar(r_B)} \rangle$ be a tuple of terms with variables in $x = < x_1, ..., x_m >$, and $q_i \in M_{AB}$ be the operad’s operation of this implication, equal to the expression $(e \Rightarrow (\ldots)(t)) \in O(r_1, ..., r_k, r_B)$, where $q_i = v_i.q_{A,i}$ with $q_{A,i} = (e \Rightarrow (\ldots)(t)) \in O(r_1, ..., r_k, r_q)$ and $v_i = ((\ldots)(y_1, ..., y_{ar(r_B)})) \in O(r_q, r_B)$ such that for a new relational symbol $r_q$, $ar(r_q) = ar(r_B) = 1$.

It is important to underline that each term $t_i$ is a simple variable which appear in the tuple $x$ (left side of the implication) or the term $f_i(z)$ where the variables in the tuple $z$ is a subset of the variables in $x$.

For a given mapping-interpretation $\alpha$ such that $A = \alpha^*(A)$ and $B = \alpha^*(B)$ are two models of the schemas $A$ and $B$ respectively, and $\alpha$ satisfies the schema mapping $M_{AB} = \{\exists(\Psi)\}$, with the tuple of existentially quantified Skolem functions $f$, the process of saturation of the morphism $h = \alpha^*(M_{AB}) = \{\alpha(q_1), ..., \alpha(q_n), q_L\}$ is relevant only for the operads operations $q_i$ which have at least one functional symbol of $f$ in on the right side of implication, as follows:

**Saturation algorithm $Sat(\alpha^*(M_{AB}))$**

**Input:** A mapping arrow $M_{AB} = \{q_1, ..., q_n, 1_{r_B}\} : A \rightarrow B$, and a mapping-interpretation $\alpha$ such that $A = \alpha^*(A)$ and $B = \alpha^*(B)$ are two models of the schemas $A$ and $B$ respectively, and $\alpha$ obtained of a given Tarski’s interpretation $I_T$, satisfies the schema mapping $M_{AB} = \{\exists(\Psi)\}$, with the tuple of existentially quantified Skolem functions $f$.

**Output:** Saturated morphism from $A$ into $B$ in DB category.

1. Let $h = \alpha^*(M_{AB})$. Then initialize $Sat(h) = h, i = 0$.
2. $i = i + 1$. If $i > N$ go to 8.
3. Let the mapping component $q_i \in M_{AB}$ be the expression $(e \Rightarrow (\ldots)(t)) \in O(r_1, ..., r_k, r_B)$, where $q_i = v_i.q_{A,i}$ with $q_{A,i} = (e \Rightarrow (\ldots)(t)) \in O(r_1, ..., r_k, r_q)$ and $v_i = ((\ldots)(y_1, ..., y_{ar(r_B)})) \in O(r_q, r_B)$. Define the set $F$ of all functional symbols in the tuple of terms $t$. If $F$ is empty then go to 2.
4. (Fix the function of $q_i$ for given $\alpha$) Let $f = \alpha(q_{A,i}) : R_1 \times ... \times R_k \rightarrow r_q$ be the function of this mapping-interpretation provided in Definition 3 where $\alpha(r_q) \subseteq \|r_B\|$ is image of this function with relation $\|r_B\| = \alpha(r_B) \in B, x = < x_1, ..., x_m >$ be the tuple of all variables in the left-side expression $e$ of the operad’s operation $q_i$, and $S$ be the set of sets that contain the pairs of mutually equal free variables in the formula $e[J_n/J_n]_{1 \leq n \leq k}$ obtained from $q_i$ (in Definition 3).

Set $R_L = R_1 \times ... \times R_k$.
5. (Expansion of $q_{A,i}$) If $R_L$ is empty then go to 2.

Take a tuple $(d_1, ..., d_k) \in R_L \subseteq R_1 \times ... \times R_k$ and delete it from $R_L$. Then define the assignment $g : \{x_1, ..., x_m\} \rightarrow D$ such that $(g(x_1), ..., g(x_m)) = Cmp(S, (d_1, ..., d_k))$ (from Definition 3).

If $f((d_1, ..., d_k)) = g^*(t) = (g(t_1), ..., g(t_{ar(r_B)}) \neq <>$ then go to 6.

Go to 5.
Let the mapping component $R_B \in B$. Then we define the relation:
$$R = \{ \text{SELECT } \ast \text{ FROM } \|R_B\| \text{ WHERE } \bigwedge_{j \in Z} (nr_{R_B}(j) = g(t_j)) \forall \{g^*(t)\}. $$

7. If $R$ is empty relation then go to 5.

Take from $R$ a tuple $b = \langle b_1, \ldots, b_{ar(r_B)} \rangle$ and delete it from $R$. We define a new Tarski’s interpretation $I'_T$, different from $I_T$ only for the functional symbols $f_i \in F$ of the j-th term $t_j = f_i(x_{j1}, \ldots, x_{jp}) \in t$, as follows:

1. $I'_T(f_i)(g(x_{j1}), \ldots, g(x_{jp})) = b_j \neq g(t_j) = I_T(f_i)(g(x_{j1}), \ldots, g(x_{jp}));$

2. For all assignments $g_1 \neq g$ we have that
$$I'_T(f_i)(g_1(x_{j1}), \ldots, g_1(x_{jp})) = I_T(f_i)(g_1(x_{j1}), \ldots, g_1(x_{jp}));$$

so that for the $R$-algebra $\alpha'$ derived from the Tarski’s interpretation $I'_T$, we obtain the new function $f_b = \alpha'(q_{A,i})$ which satisfies $f_b(\langle d_1, \ldots, d_k \rangle) = b$.

Insert the function $f_b : R_1 \times \ldots \times R_k \rightarrow \|r_B\| \text{ in Sat}(h)$ and go to 7.

8. Return the saturated morphism $\text{Sat}(\alpha^*(\mathcal{M}_{AB})) : A \rightarrow B$.

Notice that for a mapping sketch’s arrow $\mathcal{M}_{AB} = \{ q_1, \ldots, q_N, 1_r \} : A \rightarrow B$, and a mapping-interpretation $\alpha$ (such that $A = \alpha^*(A)$ and $B = \alpha^*(B)$ are two models of the schemas $A$ and $B$, respectively, and $\alpha$, obtained of a given Tarski’s interpretation $I_T$, satisfies the schema mapping $\mathcal{M}_{AB}$), we obtain the DB morphism $h = \alpha^*(\mathcal{M}_{AB}) = \{ \alpha(q_1), \ldots, \alpha(q_N), q_1 \} : A \rightarrow B$ with the property that each k-ary function $\alpha(q_i) : R_1 \times \ldots \times R_k \rightarrow \|r_B\|$ for its argument returns a single tuple (or empty tuple $<>$) of $\|r_B\|$.

Let $\text{dom}$ and $\text{cod}$ be the operators which, for each function, return the domain and codomain of this function, respectively, and $\mathcal{P}$ be the powerset operation. By the saturation of $h$ we obtain the morphism $\text{Sat}(h) : A \rightarrow B$ from which we are able to define the set $S_q = \{ h_i \in \text{Sat}(h) | \text{dom}(h_i) = \text{dom}(\alpha(q_i)) \text{ and } \text{cod}(h_i) = \text{cod}(\alpha(q_i)) \}$ and function $f_{q_i} = \bigcup S_q \triangleq \bigcup_{h_i \in S_q} \text{graph}(h_i)$, where
$$(\exists) \ \text{graph}(h_i) = \{(\langle d_1, \ldots, d_k \rangle), h_i(\langle d_1, \ldots, d_k \rangle) | \langle d_1, \ldots, d_k \rangle \in \text{dom}(h_i) \text{ and } h_i(\langle d_1, \ldots, d_k \rangle) \neq <>\}$$
is the non-empty-tuple graph of this function. Thus, in this way we obtain the p-function:
$$(\forall) \ \ f_{q_i} : \text{dom}(\alpha(q_i)) \rightarrow \mathcal{P}(\text{cod}(\alpha(q_i))),$$
for each operad’s operation $q_i \in \mathcal{M}_{AB}$ which has the functional symbols on its right side of implication in $q_i$.

We have the following property for these derived p-functions:

**Lemma 1.** Let the mapping component $q_i \in \mathcal{M}_{AB} : A \rightarrow B$ be the expression $(e \Rightarrow (\omega(t))) \in A(r_1, \ldots, r_k, r_B)$, with the tuple $x = < x_1, \ldots, x_m >$ of all variables in the left-side expression $e$, such that the set of functional symbols in the tuple of terms $t$ is not empty. Let $R$-algebra $\alpha$ be a model of this mapping $\mathcal{M}_{AB}$ with $R_i = \alpha(r_i) \in A = \alpha^*(A)$, $i = 1, \ldots, k$, and $\|r_B\| = \alpha(r_B) \in B = \alpha^*(B)$, and $(\alpha(q_i) : R_1 \times \ldots \times R_k \rightarrow \|r_B\| \in h = \alpha^*(\mathcal{M}_{AB}) : A \rightarrow B$.

Then, for the set $S_{q_i} = \{ h_i \in \text{Sat}(h) | \text{dom}(h_i) = \text{dom}(\alpha(q_i)) \text{ and } \text{cod}(h_i) = \text{cod}(\alpha(q_i)) \}$ we define the p-function:
$$(1) \ \ f_{q_i} = \bigcup S_{q_i} \triangleq \bigcup_{h_i \in S_{q_i}} \text{graph}(h_i) : R_1 \times \ldots \times R_k \rightarrow \mathcal{P}(\|r_B\|).$$
Let $Z$ be the set of indexes of the terms in $t = \langle t_1, \ldots, t_{\text{ar}(r_B)} \rangle$ which are simple variables and we denote by $nr_{r_B}(j)$ the name of the $j$-th column of the relation $r_B \in B$. Then, for each tuple $(d_1, \ldots, d_k) \in R_1 \times \ldots \times R_k$ with the assignment $g : \{x_1, \ldots, x_m\} \to D$ such that $\{g(x_1), \ldots, g(x_m)\} = Cmp(S, (d_1, \ldots, d_k))$ (from Definition 3), we obtain:

$$f_{q_i}(\langle d_1, \ldots, d_k \rangle) = \text{SELECT}(\ast) \text{FROM} \parallel r_B \parallel \text{WHERE} \bigwedge_{j \in Z}(nr_{r_B}(j) = g(t_j)),$$

and, if $\alpha(q_i)(\langle d_1, \ldots, d_k \rangle) = <>$ then $f_{q_i}(\langle d_1, \ldots, d_k \rangle) = \emptyset \in P(\parallel r_B \parallel)$.

**Proof:** From the step 6 and 7 of the algorithm for saturation, we have that for every tuple in $R = \text{SELECT}(\ast) \text{FROM} \parallel r_B \parallel \text{WHERE} \bigwedge_{j \in Z}(nr_{r_B}(j) = g(t_j))$, we have one function in the set $S_{q_i}$, and consequently the equation (2) is valid.

**Corollary 1** For every $R$-algebra $\alpha$ which is a model of a given schema mapping $M_{AB} : A \to B$, we have that $\alpha^*(M_{AB}) : \alpha^*(A) \to \alpha^*(B)$ and $\text{Sat}(\alpha^*(M_{AB})) : \alpha^*(A) \to \alpha^*(B)$ are two equal morphisms in the category $DB$. Consequently, the saturation of morphisms is an invariant process in $DB$, so that we can replace any non-saturated morphisms with its saturated version in any commutative diagram in $DB$.

**Proof:** From the fact that the introduction of the new functions changes only the terms with non-built-in functional symbols on the right sides of implications, so that they are not in $\text{Var}(M_{AB})$, and hence, from Definition 4 they do not change the information flux of the morphism $\alpha^*(M_{AB})$.

**Example 4:** Let us consider the following simple example with three relations in the database schema $A$:

1. **ZipLocations**($zipCode$, $city$, $state$) with primary key (PK)$zipCode$,
2. **Contacts**($contactID$, $firstName$, $lastName$, $street$, $zipCode$) with PK corresponding to $contactID$ and foreign key (FK) to $zipCode$, and
3. **PhoneNumbers**($contactID$, $phoneType$, $number$) with FK $contactID$. 

![Database Schema Diagram]

- **Contacts** with Candidate Key $zipCode$.
- **ZipLocations** with Candidate Key $zipCode$.
- **PhoneNumbers** with Candidate Key $contactID$.

The table represents the database schema with relationships and attributes.
such that for each contact we can store the name and forename of the contacted person, his address and phone numbers. Suppose that we want to know what hobbies each person on our contact list is interested in. It can be only done indirectly by introducing a database schema $B$ with a relation $\text{Hobbies}(\text{contactID}, \text{hobby})$ with FK $\text{contactID}$, and hence represented by the schema above.

Consequently, we define a schema mapping $M_{AB} : A \rightarrow B$ by the tgd $\forall x_1, x_2, x_3, x_4, x_5 \exists f_1(\forall x(\text{Contacts}(x_1, x_2, x_3, x_4, x_5) \Rightarrow \exists y \text{Hobbies}(x_1, y)))$, so that by Skolemization we obtain the SOtg $\Phi$ equal to the logic formula

$\exists f_1(\forall x(\text{Contacts}(x_1, x_2, x_3, x_4, x_5) \Rightarrow \text{Hobbies}(x_1, f_1(x_1))), \text{where} x = \langle x_1, x_2, x_3, x_4, x_5 \rangle$. Consequently, $M_{AB} = \text{MakeOperads}(\Phi) = \{ q_1, 1_{r_0} \}$ : $A \rightarrow B$, with $q_1 = \langle q_{A,1} \rangle \in O(\text{Contacts}, \text{Hobbies})$ with $q_{A,1} = (\bot(x) \Rightarrow (\bot(t)) \in O(\text{Contacts}, r_{f_1})$, where $t = \langle t_1, t_2 \rangle$ with the term $t_1$ equal to variable $x_1$ and term $t_2$ equal to $f_1(x_1)$, and $v_1 = (\bot(y_1, y_2) \Rightarrow (\bot(y_1, y_2)) \in O(r_{f_1}, \text{Hobbies})$.

Let us consider a model of this schema mapping $\alpha$, such that: $R_1 = \alpha(\text{Contacts})$ and $\|r_B\| = \|\text{Hobbies}\| = \alpha(\text{Hobbies})$, with

| contactID | firstName | lastName | street | zipCode |
|-----------|-----------|----------|--------|---------|
| 132       | Zoran     | Majkic   | Appia  | 0187    |

$$
\|r_B\| = \|\text{Hobbies}\| = \\
132 \quad \text{photography} \\
132 \quad \text{music} \\
132 \quad \text{art} \\
132 \quad \text{travel} \\
... \quad ... $$

so that for $d_1 = \langle 132, Zoran, Majkic, Appia, 0187 \rangle \in R_1$, we obtain the assignment $g : \{x_1, x_2, x_3, x_4, x_5\} \rightarrow D$ such that $g(x_1) = 132, g(x_2) = Zoran, g(x_3) = Majkic, g(x_4) = Appia$ and $g(x_5) = 0187$, and for $f = \alpha(q_{A,1}) : R_1 \rightarrow \alpha(r_{f_1})$ such that $f(d_1) = g^*(t) = \{g(x_1), g(f_1(x_1))\} = \langle 132, 132, art \rangle$, that is $I_f(f_1(132)) = \text{art}$.

Then in step 6 of the algorithm, we have that $Z = \{t_1\} = \{x_1\}$ with $nr_{\text{Hobbies}}(1) = \text{contactID}$ and

$$
R = (\text{SELECT } *) \text{ FROM } \|r_B\| \text{ WHERE } \bigwedge_{j \in Z}(nr_{R_{H}}(j) = g(j)) \setminus \{g^*(t)\} = (\text{SELECT } *) \text{ FROM } \|r_B\| \text{ WHERE } \text{contactID} = 132) \setminus \{g^*(t)\}
$$

Consequently, in step 7 of the algorithm will be introduced the three new functions
from \( R_1 \) into \( R_B \), \( f_b, b \in R \), into \( \text{Sat}(\alpha^*(M_{AB})) \) so that

- \( f_{132, photography}(d_1) = (132, photography) \), with \( I_f(f_1)(132) = \text{photography} \);
- \( f_{132, music}(d_1) = (132, music) \), with \( I_f(f_1)(132) = \text{music} \);
- \( f_{132, travel}(d_1) = (132, travel) \), with \( I_f(f_1)(132) = \text{travel} \).

Thus, for the derived function \( f_{q_1} = \bigcup S_{q_i} : \alpha(\text{Contacts}) \rightarrow \mathcal{P}(\alpha(\text{Hobbies})) \), we obtain that

\[
f_{a_1}(d_1) = f_{q_1}(132, Zoran, Majkic, Appia, 0187) =
\begin{array}{c|c}
\text{contactID} & \text{hobby} \\
\hline
132 & art \\
132 & photography \\
132 & music \\
132 & travel \\
\end{array}
\]

and hence, by using the second projection \( \pi_2 \), we obtain that \( (\pi_2 \circ f_{q_1})(d_1) = \{\text{photography, art, music, travel}\} \), that is, for each contact ID, the function \( \pi_2 \circ f_{q_1} \) returns the set of hobbies of this ID.

\( \square \)

In this way we are able to represent also the 1:N relationships between relational tables by the morphisms in DB category.

It is important that the saturation can be done only for the non-built-in functional symbols. In fact we have only one prefixed interpretation of the built-in functional symbols, so that their interpretation is equal for every Tarski’s interpretation. Let us show one example with functional symbols that are built-in functions:

**Example 5**: Let us consider the IRDB with the parsing of the RDB instances into the vector relation \( r_V(x\text{-name, t\text{-index, a\text{-name, value}}} \), introduced in [9,10], where is demonstrated the following proposition:

- Let the IRDB be given by a Data Integration system \( \mathcal{I} = \langle A, S, M \rangle \) for a used-defined global schema \( A = (S_A, \Sigma_A) \) with \( S_A = \{r_1, \ldots, r_n\} \), the source schema \( S = \{r_V\}, \emptyset \) with the vector big data relation \( r_V \) and the set of mapping tgd’s \( M \) from the source schema into he relations of the global schema. Then a canonical model of \( \mathcal{I} \) is any model of the schema \( A^* = (S_A \cup \{r_V\}, \Sigma_A \cup M \cup M^{OP}) \), where \( M^{OP} \) is an opposite mapping tgd’s from \( A \) into \( r_V \) given by the following set of tgd’s:

\[
M^{OP} = \{ \forall x_1, \ldots, x_{ar(r_k)}((r_k(x_1, \ldots, x_{ar(r_k)}) \land x_i \text{NOT NULL}) \Rightarrow r_V(r_k, \text{Hash}(x_1, \ldots, x_{ar(r_k)}), nr_{r_k}(i), x_i)) \mid 1 \leq i \leq ar(r_k), r_k \in S_A \} : A \rightarrow S.
\]

Thus, \( M^{OP} = \text{MakeOperads}(M^{OP}) = \{ 1_{r_b} \cup \{ g_{k,i} | r_k \in S_A \text{ and } 1 \leq i \leq ar(r_k) \} : A \rightarrow S \) is a sketch’s mapping with \( g_{k,i} = ((_i(x_{k,1}, \ldots, x_{k,ar(r_k)}) \land x_{k,i} \text{NOT NULL}) \Rightarrow (_, (t_{k,i}) \in O(r_k, r_V)) \), where \( t_{k,i} = \{ t_1, \ldots, t_4 \} \) with the terms:

1. \( t_1 \) is the nullary built-in function, i.e., the fixed constant which does not depend on Tarski’s interpretations, equal to the relation table name \( r_k \);
2. \( t_2 = \text{Hash}(x_{k,1}, \ldots, x_{k,ar(r_k)}) \) where \( \text{Hash} \) is a built in-function equal for every Tarski’s interpretation;
3. \( t_3 \) is the nullary built-in function, i.e., the fixed constant which does not depend on
Tarski’s interpretations, equal to the i-th column name of the relational table \( r_k \);
4. \( t_4 \) is the variable \( x_{k,i} \).

Thus, no one of these three built-in functional symbols are obtained by elimination of the existentially quantified variables, so they are not the Skolem functions, and in the \( \text{SOtd} \) of \( M^{OP} \), the set of existentially quantified functional symbols \( f \) is empty, so that from the algorithm of saturation, for a given \( R \)-algebra \( \alpha \), such that \( A = \alpha^*(A) \) is the instance database of the schema \( A \) and \( \vec{A} = \alpha(r_V) \) is the obtained vector relation by parsing the database \( A \), we obtain that \( \text{Sat}(\alpha^*(M^{OP})) = \alpha^*(M^{OP}) \).

We recall that the operation of parsing, \( \text{PARSE} \), for a tuple \( d = \langle d_1, ..., d_{ar(r_k)} \rangle \) of the relation \( R_k = \| r_k \| = \alpha(r_k) \in A \), is defined by the mapping
\[
(r_k, d) \mapsto \{ \langle r_k, \text{Hash}(d), nr_{r_k}(i), d_i \rangle : \text{d,NOT NULL, } 1 \leq i \leq ar(r_k) \},
\]
so that \( \vec{A} = \bigcup_{r_k \in S_A, d \in \| r_k \|} \text{PARSE}(r_k, d) \).

Consequently, we obtain the function \( \alpha(q_{k,i}) : \alpha(r_k) \rightarrow \alpha(r_V) = \vec{A} \), such that for its image \( \text{im}(\alpha(q_{k,i})) \) we obtain that from the parsing \( \pi_d(\text{im}(\alpha(q_{k,i}))) = \pi_d(\alpha(r_k)) \).

If we make union of all functions in \( f^{OP} = \alpha^*(M^{OP}) \) with the same domain and codomain, for example, for the domain \( R_k = \alpha(r_k) \in A \), we obtain the p-function
\[
f_{r_k} = \bigcup_{1 \leq i \leq ar(r_k)} \alpha(q_{k,i}) : R_k \rightarrow \vec{A} \),
\]
such that for each tuple \( d = \langle d_1, ..., d_{ar(r_k)} \rangle \) of the relation \( R_k = \| r_k \| = \alpha(r_k) \in A \),
\[
f_{r_k}(d) = \text{PARSE}(r_k, d) = \text{Hash}(132, \text{Zoran, Majkic, Appia, 00187}),
\]

where \( \text{IND} = \text{Hash}(132, \text{Zoran, Majkic, Appia, 00187}) \).

Consequently, the parsing can be derived from the morphism in \( \text{DB} \) category,
\[
f^{OP} : A \rightarrow \{ \vec{A}, \bot \} = \alpha^*(S).
\]

\( \Box \)

4 Conclusion

It was demonstrated that a categorical logic (denotational semantics) for database schema mapping based on views is a very general framework for RDBs, the database-integration/exchange and peer-to-peer systems [2]. In this very general semantic framework was necessary to introduce the base database category \( \text{DB} \) (instead of traditional \( \text{Set} \) category), with objects instance-databases and with morphisms (mappings which are not simple functions) between them, at an instance level as a proper semantic domain for a database mappings based on a set of complex query computations.

The higher logical schema level of mappings between databases, usually written in some high expressible logical language (ex. [34], GLAV (LAV and GAV), tuple gener-
ating dependency) can then be translated functorially into this base "computation" category. Hence, the denotational semantics of database mappings is given by morphisms of the Kleisli category $\text{DB}_{T}$, based on the fundamental (from Universal algebra) monad (power-view endofunctor) $T$, which may be "internalized" in $\text{DB}$ category as "computations". Big Data integration framework presented in [2] considers the standard RDBs with Tarskian semantics of the FOL, where one defines what it takes for a sentence in a language to be true relative to a model.

In this paper we demonstrated that each morphisms in $\text{DB}$ can be equivalently substituted by its saturation-morphism, and we have shown that in this way by the morphisms in $\text{DB}$ we are able to represent also the 1:N relationships between the relational tables, but also to define the parsing of the RDBs into intensional RDBs with the vector relations containing the data and the metadata (IRDBs).

Moreover, the saturated morphisms are able to express the general mappings from any given tuple of some relational view (obtained by a given SQL statement) into the set of tuples of another relational tables, which generally can be used in intensional RDBS where we are using the intensional FOL with the extensionalization function for the intensional concepts. In a future work we will investigate these properties of saturated morphisms for more advanced features of the IRDBs as are the multivalued attributes (which can not be supported in the FOL and standard RDBs).

References

1. B.Jacobs, “Categorical Logic and Type Theory,” Studies in Logic and the Foundation of Mathematics, Vol. 141, Elsevier, Amsterdam, 1999.
2. Z. Majkić, “Big Data Integration Theory,” Springer-Verlag, Texts in Computer Science, New York Inc., pp. 516, 2014.
3. M.Lenzerini, “Data integration: a theoretical perspective,” 2002, pp. 233–246.
4. R.Fagin, P.G.Kolaitis, R.J.Miller, and L.Popâ, “DATA Exchange: Semantics and query answering,” 2003, pp. 207–224.
5. A.Calî, D.Calvanese, G.De Giacomo, and M.Lenzerini, “Data integration under integrity constraints,” 2002, pp. 262–279.
6. R.Fagin, “Horn clauses and database dependencies,” Journal of the ACM, vol. 29, pp. 952–985, 1982.
7. R.Fagin, P.G.Kolaitis, L.Popâ, and W.Tan, “Composing schema mappings: second-order dependencies to the rescue,” in ACM TODS, 30(4), pp. 994–1055, 2005.
8. P.J.Freyd and G.M.Kelly, “Categories of continuous functors,” 1, J. of Pure Appl. Algebra 2, pp. 169–191, 1972.
9. Z.Majkić, “Intensional RDB manifesto: a unifying NewSQL model for flexible Big Data,” arXiv: 1403.0017, 28 February, pp. 1–29, 2014.
10. Z.Majkić, “Intensional RDB for Big Data interoperability,” arXiv: 1403.6089, 24 March, pp. 1–30, 2014.