

## 2.1 Schema Mappings: Second-Order tgds (SOTgds)

Based on previous considerations and weak points in the schema-mapping developments presented in the introduction (Sect. 1.4.2), a different semantics for composition that is valid for every class of queries has been proposed in [5]. The suggested semantics does not carry along a class of queries as a parameter, and the authors have shown that the set of formulae defining a composition is unique up to logical equivalence.

On the negative side, however, they demonstrated that the composition of a finite set of tgds with a finite set of *full* tgds (a tgd is full if no existentially quantified variables occur in the tgd) may not be definable by any set (finite or infinite) of tgds. Moreover, the composition of a finite set of tgds with a finite set of tgds is not definable in least-fixed point logics.

Based on these negative results, a class of existential second-order formulae with function symbols (introduced as a result of Skolemization of the existentially quantified variables) and equalities, the so-called second-order tgds (SOTgds), has been introduced in [5] as follows:

**Definition 6** [5] Let  $\mathcal{A}$  be a source schema and  $\mathcal{B}$  a target schema. A second-order tuple-generating dependency (SOTgd) is a formula of the form:

$$\exists \mathbf{f}((\forall \mathbf{x}_1(\phi_1 \Rightarrow \psi_1)) \wedge \cdots \wedge (\forall \mathbf{x}_n(\phi_n \Rightarrow \psi_n))),$$

where

1. Each member of the tuple  $\mathbf{f}$  is a functional symbol;
2. Each  $\phi_i$  is a conjunction of:
  - Atomic formulae of the form  $r_A(y_1, \dots, y_k)$ , where  $r_A \in S_A$  is a  $k$ -ary relational symbol of schema  $\mathcal{A}$  and  $y_1, \dots, y_k$  are variables in  $\mathbf{x}_i$ , not necessarily distinct;
  - The formulae with conjunction and negation connectives and with built-in predicate's atoms of the form  $t \odot t'$ ,  $\odot \in \{=, <, >, \dots\}$ , where  $t$  and  $t'$  are the terms based on  $\mathbf{x}_i$ ,  $\mathbf{f}$  and constants.

3. Each  $\psi_i$  is a conjunction of atomic formulae  $r_B(t_1, \dots, t_m)$  where  $r_B \in S_B$  is an  $m$ -ary relational symbol of schema  $\mathcal{B}$  and  $t_1, \dots, t_m$  are terms based on  $\mathbf{x}_i, \mathbf{f}$  and constants.
4. Each variable in  $\mathbf{x}_i$  appears in some atomic formula of  $\phi_i$ .

Notice that each constant  $\bar{a}$  in an atom on the left-hand side of implications must be substituted by a new fresh variable  $y_i$  and by adding a conjunct  $(y_i = \bar{a})$  on 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 SOTgd tautology  $r_{\emptyset} \Rightarrow r_{\emptyset}$ . The forth condition is a “safety” condition, analogous to that made for the (first-order) tgds. It is easy to see that every tgd is equivalent to one SOTgd without equalities. For example, let  $\sigma$  be the tgd  $\forall x_1 \dots \forall x_m (\phi_A(x_1, \dots, x_m) \Rightarrow \exists y_1 \dots \exists y_n \psi_B(x_1, \dots, x_m, y_1, \dots, y_n))$ . It is logically equivalent to the following SOTgd without equalities, which is obtained by Skolemizing existential quantifiers in  $\sigma$ :

$$\exists f_1 \dots \exists f_n (\forall x_1 \dots \forall x_m (\phi_A(x_1, \dots, x_m) \Rightarrow \psi_B(x_1, \dots, x_m, f_1(x_1, \dots, x_m), \dots, f_n(x_1, \dots, x_m))))).$$

Given a finite set  $S$  of tgds of an inter-schema mapping in Definition 3, we can find a *single* SOTgd that is equivalent to  $S$  by taking, for each tgd  $\sigma$  in  $S$ , a conjunct of the SOTgd to capture  $\sigma$  as described above (we use disjoint sets of function symbols in each conjunct, as before). Based on these observations, we can define the algorithm for transformation of a given set of tgds into an single SOTgd:

**Transformation algorithm** *TgdsToSOTgd*( $S$ )

**Input.** A set  $S$  of tgds from a schema  $\mathcal{A}$  into a schema  $\mathcal{B}$ ,

$$S = \{ \forall \mathbf{x}_1 (\phi_{A,1}(\mathbf{x}_1) \Rightarrow \exists \mathbf{z}_1 \psi_{B,1}(\mathbf{y}_1, \mathbf{z}_1)), \dots, \forall \mathbf{x}_n (\phi_{A,n}(\mathbf{x}_n) \Rightarrow \mathbf{z}_n \psi_{B,n}(\mathbf{y}_n, \mathbf{z}_n)) \},$$

where  $\mathbf{y}_i \subseteq \mathbf{x}_i$  and possibly  $\mathbf{z}_i$  the empty set, for  $1 \leq i \leq n$ .

**Output.** A SOTgd

- (Create the set of implications from tgds)  
Initialize  $S_{AB}$  to be the empty set.  
Put each of the  $n$  implications of tgds in  $S$ ,  $\phi_{A,i}(\mathbf{x}_i) \Rightarrow \exists \mathbf{z}_i \psi_{B,i}(\mathbf{y}_i, \mathbf{z}_i)$  in  $S_{AB}$ .
- (Eliminate the existential quantifiers by Skolemization)  
Repeat the following as far as possible:  
For each implication  $\chi$  in  $S_{AB}$  of the form  $\phi_{A,i}(\mathbf{x}) \Rightarrow \exists \mathbf{z} \psi_{B,i}(\mathbf{y}, \mathbf{z})$  with  $\mathbf{z} = \langle z_1, \dots, z_k \rangle$ ,  $k \geq 1$ , eliminate this  $\chi$  from  $S_{AB}$  and add the following implication in  $S_{AB}$  (by introducing the set of new functional symbols  $f_{i,j}$ , for  $1 \leq j \leq k$ ):

$$(\phi_{A,i}(\mathbf{x}) \wedge (z_1 \doteq f_{i,1}(\mathbf{x})) \wedge \dots \wedge (z_k \doteq f_{i,k}(\mathbf{x}))) \Rightarrow \psi_{B,i}(\mathbf{y}, \mathbf{z}).$$

- (Remove the variables originally quantified)  
For each implication  $\chi_i$  in  $S_{AB}$  constructed in the previous step, perform an elimination of the variables  $z_i$  from this implication as far as possible: Select an equality  $z_j \doteq f_{i,j}(\mathbf{x})$  that was generated in the previous step. Remove the

equality  $z_j \doteq f_{i,j}(\mathbf{x})$  from  $\chi_i$  and replace every remaining occurrence of  $z_j$  in  $\chi$  by  $f_{i,j}(\mathbf{x})$ .

- (*Construct* SOTgd)

If  $S_{AB} = \{\chi_1, \dots, \chi_n\}$  is the set where  $\chi_1, \dots, \chi_n$  are all the implications from the previous step, then SOTgd is the formula  $\exists \mathbf{f}(\forall \mathbf{x}_1 \chi_1 \wedge \dots \wedge \forall \mathbf{x}_n \chi_n)$ ,  $\mathbf{f}$  is the tuple of all function symbols that appear in any of the implications in  $S_{AB}$ , and the variables in  $\mathbf{x}_i$  are all the variables found in the implication  $\chi_i$ , for  $1 \leq i \leq n$ .

**Return** the SOTgd  $\exists \mathbf{f}(\forall \mathbf{x}_1 \chi_1 \wedge \dots \wedge \forall \mathbf{x}_n \chi_n)$ .

Every SOTgd is equivalent to an SOTgd in a *normal form*, where the right-hand sides (i.e., the formulae  $\psi_i$ ) are atomic formulae rather than conjunctions of atomic formulae. For example,  $\exists f \forall x (r(x) \Rightarrow (r_1(x, f(x)) \wedge r_2(f(x), x)))$  is logically equivalent to  $\exists f (\forall x (r(x) \Rightarrow r_1(x, f(x))) \wedge \forall x (r(x) \Rightarrow r_2(f(x), x)))$ . This is unlike the situation for (first-order) tgds (with existential quantifiers on the right-hand sides of implications), where we would lose expressive power if we required that the right-hand sides consist only of atomic formulae and not conjunctions of atomic formulae.

It was shown that the composition of SOTgds is also definable by an SOTgd (with algorithm given in [5]), and that the computing of certain answers of conjunctive queries in data exchange settings specified by SOTgds can be done in *polynomial* time in the size of source database instance. The class of SOTgds is obtained from (first-order) tgds by Skolemizing the existential first-order quantifiers into existential quantified functional symbols. This process gives rise to a class of existential second-order formulae with no equalities, but having equalities is not sufficient to expressively define the composition of finite sets of (first-order) tgds.

In the study of data exchange [5], one usually assumes an open-world assumption (OWA) semantics, making it possible to extend instances of target schema. The closed-world assumption (CWA) semantics was considered in [6] as an alternative that only moves ‘as much data as needed’ from the source instance into the target instance to satisfy constraints of a inter-schema mapping. It extends instances with nulls as well (incomplete information), and their language of CQ-SkSTDs slightly extends the syntax of SOTgds and is closed under composition. It was shown [6] that the Skolemized constraints are closed under composition not only under OWA but also under the CWA. If only conjunctive queries are used in mappings then, under both OWA and CWA, the composition problem is generally NP-complete (a model-checking for SOTgds, i.e., determining whether a given instance over the source and target schema satisfies a SOTgd can be NP-complete, in contrast with the first-order case, where such model-checking can be done in polynomial time). It is valid for composition of inter-schema mappings that mix open and closed attributes in mapping tgds.

There is subtle issue about the choice of universe in the semantics of SOTgds. In [5], the universe has been taken to be a countably infinite set of elements that included *active domain*. We recall that for a given instance  $(A, B) \in \text{Inst}(\mathcal{M}_{AB})$  of an inter-schema mapping  $\mathcal{M}_{AB} : \mathcal{A} \rightarrow \mathcal{B}$  the active domain consists of those values that appear in  $A$  and  $B$ , so that the second-order variables (here functional symbols

that can be semantically represented by a binary relation of the function-graph) are interpreted over relations on the active domain.

Since SOTgds have existentially quantified function symbols, one needs sufficiently many elements in the universe in order to interpret these function symbols by values that are not contained in the source database instance as well. In [5], it was shown that as long as we take the universe to be finite but sufficiently large, the semantics of SOTgds remains unchanged from an infinite universe semantics. The most natural choice for the universe, of an instance  $(A, C)$  of a composition  $\mathcal{M}_{AC} : \mathcal{A} \rightarrow \mathcal{C}$  (obtained from mappings  $\mathcal{M}_{AB} : \mathcal{A} \rightarrow \mathcal{B}$  and  $\mathcal{M}_{BC} : \mathcal{B} \rightarrow \mathcal{C}$  with a “intermediate” database  $\mathcal{B}$ ), seems to be the active domain. But for the semantics of composition of mappings in [5] the authors did not use this simple choice when considering the necessity in the composition of instances  $(A, B)$  and  $(B, C)$  to take on values in the missing middle instance  $B$  for quantified functions in SOTgds.

The approach for what is the *proper* semantics of a given SOTgd mapping between  $\mathcal{A}$  and  $\mathcal{B}$  in our setting is more general than in the OWA data-exchange setting used in [5]: in our case, we consider that a more adequate semantics of a mapping from  $\mathcal{A}$  into  $\mathcal{B}$  has to be a *strict mapping* semantics, which considers the part of information that is mapped (only) from  $\mathcal{A}$  into  $\mathcal{C}$ , without the information taken from another intermediate databases (as  $\mathcal{B}$  in this case), and we need to derive it formally from the SOTgd.

In fact, in the former approach to the semantics of composition of mappings, the authors are using a non-well-defined “extended” semantics for mapping instances  $(\mathcal{U}, A, C)$  instead of  $(A, C)$ , where  $\mathcal{U}$  is an unspecified but “sufficiently large” universe, “fixed and understandable from the context” [5], that includes the active domain  $A$  and  $C$ .

*Remark* In what follows, we introduce the characteristic functions for predicates (i.e., relations) during the composition of mappings and hence assume that the set of the classic truth logical values  $\mathbf{2} = \{0, 1\}$  is a subset of the universe  $\mathcal{U} = \mathbf{dom} \cup SK$ .

With the syntax choice for SOTgds in [5], the authors encapsulate the necessary information contained in the intermediate database  $\mathcal{B}$  into the semantics of the quantified functional symbols. In the interpreted functions of the composed mapping SOTgd, the information is mixed from both database instances  $A$  and  $B$ . Consequently, we will not consider two mappings between a given source and target databases equal if they are logically equivalent (as in data-exchange setting presented in [5]), but only if the *strict semantics* of the SOTgds of these two mappings are equal. Let us clarify these different approaches to the semantics of inter-schema mappings:

*Example 2* Let us consider the following four schemas  $\mathcal{A}, \mathcal{B}, \mathcal{C}$  and  $\mathcal{D}$  (an extension of Example 2.3 in [5]). Schema  $\mathcal{A}$  consists of a single binary relational symbol *Takes* that associates student names with courses they take, i.e.,  $S_A = \{\text{Takes}(x_n, x_c)\}$ ,  $\Sigma_A = \emptyset$ . Schema  $\mathcal{B}$  consists of one binary relational symbol

Takes1 that is intended to provide a copy of Takes, and of an additional binary relational symbol Student that associates each student name with a student id, i.e.,

$$S_B = \{\text{Takes1}(x_n, x_c), \text{Student}(x_n, y)\}, \quad \Sigma_B = \emptyset.$$

Schema  $\mathcal{D}$  consists of one binary relational symbol Enrolment that associates student ids with the courses the student takes, and of an additional binary relational symbol Teaching that associates professor names with the courses, i.e.,  $S_D = \{\text{Enrolment}(y, x_c), \text{Teaching}(x_p, x_c)\}$ ,  $\Sigma_D = \emptyset$ . Schema  $\mathcal{C}$  consists of one ternary relational symbol Learning that associates student name with courses he/she takes and professor names, and of an additional unary relational symbol Professor with professor names, i.e.,  $S_C = \{\text{Learning}(x_n, x_c, x_p), \text{Professor}(x_p)\}$ , and with integrity constraints

$$\begin{aligned} \Sigma_C = \{ & \forall x_p (\text{Professor}(x_p) \Rightarrow \exists x_n \exists x_c \text{Learning}(x_n, x_c, x_p)), \\ & \forall x_n \forall x_c \forall y \forall z ((\text{Learning}(x_n, x_c, z_1) \wedge \text{Learning}(x_n, x_c, z_2)) \\ & \Rightarrow z_1 = z_2) \}, \end{aligned}$$

where the second constraint defines the tuple  $(x_n, x_c)$  as a key of the relation Learning.

Let us consider now the following schema mappings  $\mathcal{M}_{AB} : \mathcal{A} \rightarrow \mathcal{B}$ ,  $\mathcal{M}_{BD} : \mathcal{B} \rightarrow \mathcal{D}$ ,  $\mathcal{M}_{AC} : \mathcal{A} \rightarrow \mathcal{C}$  and  $\mathcal{M}_{CD} : \mathcal{C} \rightarrow \mathcal{D}$ , where

$$\begin{aligned} \mathcal{M}_{AB} = \{ & \forall x_n \forall x_c (\text{Takes}(x_n, x_c) \Rightarrow \text{Takes1}(x_n, x_c)), \\ & \forall x_n \exists y \forall x_c (\text{Takes}(x_n, x_c) \Rightarrow \text{Student}(x_n, y)) \}, \end{aligned}$$

that is, by Skolemization of  $\exists y$ ,

$$\begin{aligned} \mathcal{M}_{AB} = \{ & \forall x_n \forall x_c (\text{Takes}(x_n, x_c) \Rightarrow \text{Takes1}(x_n, x_c)), \\ & \forall x_n \forall x_c (\text{Takes}(x_n, x_c) \Rightarrow \text{Student}(x_n, f_1(x_n))) \}, \\ \mathcal{M}_{BD} = \{ & \forall x_n \forall x_c \forall y ((\text{Takes1}(x_n, x_c) \wedge \text{Student}(x_n, y)) \\ & \Rightarrow \text{Enrolment}(y, x_c)) \}, \\ \mathcal{M}_{AC} = \{ & \forall x_n \forall x_c \exists x_p (\text{Takes}(x_n, x_c) \Rightarrow \text{Learning}(x_n, x_c, x_p)) \}, \end{aligned}$$

that is, by Skolemization of  $\exists x_p$ ,

$$\begin{aligned} \mathcal{M}_{AC} = \{ & \forall x_n \forall x_c (\text{Takes}(x_n, x_c) \Rightarrow \text{Learning}(x_n, x_c, f_2(x_n, x_c))) \}, \\ \mathcal{M}_{CD} = \{ & \forall x_n \forall x_c \forall x_p (\text{Learning}(x_n, x_c, x_p) \Rightarrow \text{Teaching}(x_p, x_c)) \}. \end{aligned}$$

Then the composition  $\mathcal{M}_{AD} : \mathcal{A} \rightarrow \mathcal{D}$ , obtained by the composition (with the algorithm in [5]) of  $\mathcal{M}_{AB}$  and  $\mathcal{M}_{BD}$ , is equal to the SOTgd:

- (i)  $\exists f_1 (\forall x_n \forall x_c (\text{Takes}(x_n, x_c) \Rightarrow \text{Enrolment}(f_1(x_n), x_c)))$ .

Analogously, the composition  $\mathcal{M}'_{AD} : \mathcal{A} \rightarrow \mathcal{D}$ , obtained by the composition of  $\mathcal{M}_{AC}$  and  $\mathcal{M}_{CD}$ , can be equivalently represented by the SOTgd:

(ii)  $\exists f_2 (\forall x_n \forall x_c (\text{Takes}(x_n, x_c) \Rightarrow \text{Teaching}(f_2(x_n, x_c), x_c)))$ .

Clearly, by using the semantics of inter-schema mappings defined in [5], we obtain that these two composed mappings  $\mathcal{M}_{AD}$  and  $\mathcal{M}'_{AD}$  are different.

As we mentioned, both SOTgds (i) and (ii) are not strict mappings from  $\mathcal{A}$  into  $\mathcal{D}$ . Since  $f_1$  in (i) encapsulates the data from  $\mathcal{B}$  as well, it substantially remains a mapping from  $\mathcal{A}$  and  $\mathcal{B}$  into  $\mathcal{D}$ . Analogously,  $f_2$  in SOTgd in (ii) encapsulates the data from  $\mathcal{C}$  as well, so it substantially remains a mapping from  $\mathcal{A}$  and  $\mathcal{C}$  into  $\mathcal{D}$ . The differences from the intermediate databases  $\mathcal{B}$  and  $\mathcal{C}$  explain why these two SOTgds are different (i.e., are not logically equivalent).

In spite of that, the strict information that is transferred (only) from  $\mathcal{A}$  into  $\mathcal{D}$  is equal for SOTgds (i) and (ii) and corresponds for each instance  $A$  of the schema  $\mathcal{A}$  to the projection of its relation  $\text{Takes}(x_n, x_c)$  over the attribute  $x_c$  (see the complete proof in Examples 6 and 15). Consequently, in our semantics for composed mappings, these two composed mappings have to be equal (differently from the data exchange framework presented in [5]) and hence we need a new formal definition for this strict inter-schema mapping composition.

Consequently, the main difference between the data-exchange framework and our more general approach is the following: In a data-exchange framework, the mapping from a source to a target database determines, for a given source database instance, the target database instance, so that two equal mappings “produce” two equal solutions for the target database. From this point of view, two equal inter-schema mappings have to be logically equivalent [5].

In our more general framework, each database in a schema mapping system is relatively independent and can be consistently modified locally, by requiring only that after this local modification of this database all other instances of the correlated database schemas in this mapping system have to be (minimally) updated in order to obtain a new instance of this mapping system where all inter-schema mappings are satisfied again (as explained in Chap. 7 dedicated to operational semantics for database mappings).

Consequently, we do not require the strong determination of the target instance by a given mapping, but only that this target instance *satisfies* the mappings as well, and hence this general framework satisfies the OWA. Hence in given composed mappings from a source schema to a given target schema, in order to verify the equality of two different mappings between these two databases, we are not interested in contributions of the intermediate databases to the target database, but only in the strict information contribution from a given source database-instance into the target database-instance. Such an information contribution of the source database can be only filtered (thus decremented) by intermediate databases and not incremented. This strict contribution can be represented only at the instance-level semantics of a composed mapping for a given instance of the source schema and hence the *equality of two mappings* in our framework can be considered only at this *instance-level*. What is common for both semantic approaches to composition of

schema mappings is that they cannot be obtained by the first-order logic formulae. From the logical point of view, we can consider the representation of inter-schema mappings and their compositions by SOTgds also in our approach, but with different meaning assigning to them.

In our case, the two equal compositions are not necessarily logically equivalent formulae (which requires the consideration of all intermediate databases involved in this composition). We will use SOTgds to define a mathematical framework from which we will be able to determine the strict meaning of composed mappings at an instance-level, as a quantity of information (set of views) specified by mapping to be transmitted from the active domain of the given source instance into the target instance.

Moreover, in order to pass to a categorical logic and its functorial semantics, we will use an algebraic representation based on the theory of operads both at schema (e.e., database sketches) and at instance database levels. We will show that this algebraic representation based on operads is semantically equivalent to the representation of the logical mappings based on SOTgds. Consequently, we will develop the algorithms for the transformation of SOTgds into the mapping operads and the algorithms for determination of the strict semantics at an instance database level. We will show that the egds can be represented as particular SOTgds and mapping operads so that the whole set of integrity constraints over a given schema can be equivalently represented by a schema mapping as well. The algorithm for composition of SOTgds given in [5] for the data-exchange setting has to be generalized for this more general semantics of schema mappings that we intend to use.

---

## 2.2 Transformation of Schema Integrity Constraints into SOTgds

Codd initially defined two sets of constraints but, in his second version of the relational model, he came up with the following integrity constraints:

- *Entity integrity*. The entity integrity constraint states that no primary key value can be null. This is because the primary key value is used to identify individual tuples in a relation. Having null value for the primary key (PK) implies that we cannot identify some tuples. This also specifies that there may not be any duplicate entries in the primary key column key row.
- *Referential Integrity*. The referential integrity constraint is specified between two relations and is used to maintain the consistency among tuples in the two relations. Informally, the referential integrity constraint states that a tuple in one relation that refers to another relation must refer to an existing tuple in that relation. It is a rule that maintains consistency among the rows of the two relations.
- *Domain Integrity*. The domain integrity states that every element from a relation should respect the type and restrictions of its corresponding attribute. A type can have variable length which needs to be respected. Restrictions could be the range of values that the element can have, the default value if none is provided, and if the element can be NULL.

- **User Defined Integrity.** (For example,  $Age \geq 18 \wedge Age \leq 60$ .) A business rule is a statement that defines or constrains some aspect of the business. It is intended to assert business structure or to control or influence the behavior of the business. A typical example of the entity integrity is the primary-key integrity constraint for relational database relations, and it can be expressed in the FOL by egds. A typical example of the referential integrity is the foreign-key integrity constraint (FK) between relations in a give database, an it can be expressed in the FOL by tgds.

The domain integrity for a given attribute  $a \in \mathbf{att}$  is defined by  $dom(a) \subset \mathcal{U}$  in the introduction (Sect. 1.4).

Each user-defined integrity of a given relation  $r$  with attribute-variables in  $\mathbf{x}$ , expressed by a formula  $\psi(\mathbf{x})$  by using the build-in predicates  $\doteq, \neq, <, \dots$  (Extensions of FOL, Sect. 1.3.1), can be defined as a tgd  $r(\mathbf{x}) \Rightarrow \psi(\mathbf{x})$ .

Consequently, we need only to consider two kinds of transformations, from the egds into a SOTgd and from the tgds into a SOTgd.

Such transformations of the integrity constraints into an SOTgd have to define a unique schema mapping  $\mathcal{M} : \mathcal{A} \rightarrow \mathcal{A}_\top$  between a schema database  $\mathcal{A}$  into a distinct target schema  $\mathcal{A}_\top$ , in a way such that a mapping has no significant compositions with another “real” inter-schema mappings of a given database mapping system.

Moreover, these obtained SOTgd (in  $\mathcal{M}$ ) from the integrity constraints of a given schema, have to be only “logical”, that is, with no transfer of information from the source database  $\mathcal{A}$  into the target database  $\mathcal{A}_\top$  (differently from the inter-schema mappings in Definition 3).

As we will see, it will be provided by the fact that the right-hand side of each implication in the obtained SOTgd (in  $\mathcal{M}$ ) will have the false ground atom  $r_\top(\bar{0}, \bar{1})$ , where the relational symbol  $r_\top$  (introduced in Sect. 1.3 as a binary built-in predicate for the FOL identity  $\doteq$ ) is the unique relational symbol of the schema  $\mathcal{A}_\top = (\{r_\top\}, \emptyset)$ .

### 2.2.1 Transformation of Tuple-Generating Constraints into SOTgds

A normalized (with a simple atom on the right-hand side of implication) integrity constraint (tgd)  $\forall \mathbf{x}(\phi_A(\mathbf{x}) \Rightarrow r(\mathbf{t})) \in \Sigma_A$  of a given schema database  $\mathcal{A} = (S_A, \Sigma_A)$  (in Definition 2) has on the right-hand side of implication the relational symbol  $r \in \mathcal{A}$  of this database schema. Thus, in order to satisfy the previously considered requirements for a “logical” representation of such a tgd, we have to transform such an implication by keeping in mind the following considerations:

Each normalized tgd  $\forall \mathbf{x}(\phi_A(\mathbf{x}) \Rightarrow r(\mathbf{t}))$  is satisfied if  $(\phi_A(\mathbf{x}) \wedge \neg r(\mathbf{t}))$  is a *falsity* (false for every assignment of the values to variables in  $\mathbf{x}$ ). Consequently, we will represent this tgd by the formula  $\forall \mathbf{x}((\phi_A(\mathbf{x}) \wedge \neg r(\mathbf{t})) \Rightarrow r_\top(\bar{0}, \bar{1}))$ , with the built-in identity relational symbol  $r_\top$  used for the FOL identity  $\doteq$ , as follows:

**Lemma 2** *Any normalized tgd constraint of a schema*

$$\mathcal{A} = (S_A, \Sigma_A), \forall \mathbf{x}(\phi_A(\mathbf{x}) \Rightarrow r(\mathbf{t})) \in \Sigma_A^{\text{egd}} \subseteq \Sigma_A,$$



where  $\mathbf{t}$  is a tuple of terms with variables in  $\mathbf{x}$  and  $r \in S_A$ , is logically equivalent to the FOL sentence  $\forall \mathbf{x}((\phi_A(\mathbf{x}) \wedge \neg r(\mathbf{t})) \Rightarrow r_{\top}(\bar{0}, \bar{1}))$ .

*Proof* For any assignment  $g$ , the formula  $r_{\top}(\bar{0}, \bar{1})$  cannot be satisfied because  $\langle g(\bar{0}), g(\bar{1}) \rangle = \langle 0, 1 \rangle \notin R_{\top} = I_T(r_{\top})$ , so that  $\phi_A(\mathbf{x}) \Rightarrow r(\mathbf{t})$  is logically equivalent to the formula  $\phi_A(\mathbf{x}) \Rightarrow (r(\mathbf{t}) \vee r_{\top}(\bar{0}, \bar{1}))$ , i.e., to the formula  $\neg \phi_A(\mathbf{x}) \vee r(\mathbf{t}) \vee r_{\top}(\bar{0}, \bar{1})$ , or equivalently, to the formula  $\neg(\phi_A(\mathbf{x}) \wedge \neg r(\mathbf{t})) \vee r_{\top}(\bar{0}, \bar{1})$ , and hence to the formula  $(\phi_A(\mathbf{x}) \wedge \neg r(\mathbf{t})) \Rightarrow r_{\top}(\bar{0}, \bar{1})$ .  $\square$

Based on these considerations, we can define the following algorithm for transformation of a given set of tgds into a single SOTgd:

**Transformation algorithm**  $TgdsToConSOTgd(\Sigma_A^{\text{tgd}})$

**Input.** A set  $\Sigma_A^{\text{egd}}$  of tgds of a given schema  $\mathcal{A}$ ,

$$\Sigma_A^{\text{tgd}} = \{ \forall \mathbf{x}_1 (\phi_{A,1}(\mathbf{x}_1) \Rightarrow \exists \mathbf{z}_1 \psi_{B,1}(\mathbf{y}_1, \mathbf{z}_1)), \dots, \forall \mathbf{x}_n (\phi_{A,n}(\mathbf{x}_n) \Rightarrow \mathbf{z}_n \psi_{B,n}(\mathbf{y}_n, \mathbf{z}_n)) \},$$

where  $\mathbf{y}_i \subseteq \mathbf{x}_i$  and possibly  $\mathbf{z}_i$  is the empty set, for  $1 \leq i \leq n$ .

**Output.** An SOTgd

1. (Create the set of implications from tgds)

Initialize  $S_{AB}$  to be the empty set.

Put each of the  $n$  implications of tgds in  $\Sigma_A^{\text{tgd}}$ ,  $\phi_{A,i}(\mathbf{x}_i) \Rightarrow \exists \mathbf{z}_i \psi_{B,i}(\mathbf{y}_i, \mathbf{z}_i)$ , in  $S_{AB}$ .

2. (Eliminate the existential quantifiers by Skolemization)

Repeat the following as far as possible:

For each implication  $\chi$  in  $S_{AB}$  of the form  $\phi_{A,i}(\mathbf{x}) \Rightarrow \exists \mathbf{z} \psi_{B,i}(\mathbf{y}, \mathbf{z})$  with  $\mathbf{z} = \langle z_1, \dots, z_k \rangle$ ,  $k \geq 1$ , eliminate this  $\chi$  from  $S_{AB}$  and add the following implication in  $S_{AB}$  (by introducing the set of new functional symbols  $f_{i,j}$ , for  $1 \leq j \leq k$ ):

$$(\phi_{A,i}(\mathbf{x}) \wedge (z_1 \doteq f_{i,1}(\mathbf{x})) \wedge \dots \wedge (z_k \doteq f_{i,k}(\mathbf{x}))) \Rightarrow \psi_{B,i}(\mathbf{y}, \mathbf{z}).$$

3. (Remove the variables originally quantified)

For each implication  $\chi_i$  in  $S_{AB}$ , constructed in the previous step, perform the elimination of the variables  $z_i$  from this implication as far as possible: Select an equality  $z_j \doteq f_{i,j}(\mathbf{x})$  that was generated in the previous step. Remove the equality  $z_j \doteq f_{i,j}(\mathbf{x})$  from  $\chi_i$  and replace every remaining occurrence of  $z_j$  in  $\chi$  by  $f_{i,j}(\mathbf{x})$ .

4. (Normalize the tgds)

Each implication  $\chi$  in  $S_{AB}$  has the form  $\phi_{A,i}(\mathbf{x}_i) \Rightarrow \bigwedge_1^k r_j(\mathbf{t}_j)$  where every member of  $\mathbf{x}_i$  is a universally quantified variable, and each  $\mathbf{t}_j$ , for  $1 \leq j \leq k$ , is a sequence (a tuple) of terms over  $\mathbf{x}_i$ . We then replace each such implication  $\chi$  in  $S_{AB}$  with  $k$  implications  $\phi_{A,i}(\mathbf{x}_i) \Rightarrow r_1(\mathbf{t}_1), \dots, \phi_{A,i}(\mathbf{x}_i) \Rightarrow r_k(\mathbf{t}_k)$ .

5. (Convert the tgds)

Remove each implication  $\chi$  in  $S_{AB}$  of the form  $\phi_{A,i}(\mathbf{x}_i) \Rightarrow r_j(\mathbf{t}_j)$  from  $S_{AB}$  and add the implication  $(\phi_{A,i}(\mathbf{x}_i) \wedge \neg r_j(\mathbf{t}_j)) \Rightarrow r_{\top}(\bar{0}, \bar{1})$  in  $S_{AB}$ .

### 6. (Construct SOTgd)

If  $S_{AB} = \{\chi_1, \dots, \chi_m\}$  is the set where  $\chi_1, \dots, \chi_m$  are all the implications from the previous step then SOTgd is the formula  $\exists \mathbf{f}(\forall \mathbf{x}_1 \chi_1 \wedge \dots \wedge \forall \mathbf{x}_m \chi_m)$ , where  $\mathbf{f}$  is the tuple of all function symbols that appear in any of the implications in  $S_{AB}$ , and the variables in  $\mathbf{x}_i$  are all the variables found in the implication  $\chi_i$ , for  $1 \leq i \leq m$ .

**Return** the SOTgd  $\exists \mathbf{f}(\forall \mathbf{x}_1 \chi_1 \wedge \dots \wedge \forall \mathbf{x}_m \chi_m)$ .

Let us consider the well known foreign-key integrity constraints:

*Example 3* Let us consider Example 2 with relation  $\text{Student}(x_n, y)$  where  $x_n$  is the primary-key of this relation (student's id) in the schema  $\mathcal{B}$  and the relation  $\text{Takes1}(x_n, x_c)$  where  $x_n$  is the foreign-key of this relation.

Thus, the foreign-key integrity constraint can be given by a single tgdc in  $\Sigma_B^{\text{tgdc}} = \{\forall x_n, x_c (\text{Takes1}(x_n, x_c) \Rightarrow \exists y \text{Student}(x_n, y))\}$ . Consequently, with this input  $\Sigma_B^{\text{tgdc}}$ , we have the following steps of the algorithm:

1. We obtain  $S_{AB} = \{\text{Takes1}(x_n, x_c) \Rightarrow \exists y \text{Student}(x_n, y)\}$ .
2. We obtain

$$S_{AB} = \{(\text{Takes1}(x_n, x_c) \wedge (y = f_1(x_n, x_c))) \Rightarrow \text{Student}(x_n, y)\}.$$

3. We obtain  $S_{AB} = \{\text{Takes1}(x_n, x_c) \Rightarrow \text{Student}(x_n, f_1(x_n, x_c))\}$ .
4. Non applicable.
5. We obtain

$$S_{AB} = \{(\text{Takes1}(x_n, x_c) \wedge \neg \text{Student}(x_n, f_1(x_n, x_c))) \Rightarrow r_{\top}(\bar{0}, \bar{1})\}.$$

### 6. Return

$$\exists f_1(\forall x_n, x_c((\text{Takes1}(x_n, x_c) \wedge \neg \text{Student}(x_n, f_1(x_n, x_c))) \Rightarrow r_{\top}(\bar{0}, \bar{1}))).$$

Consequently, the output of this algorithm is the SOTgd

$$\begin{aligned} \text{TgdsToConSOTgd}(\{\forall x_n, x_c(\text{Takes1}(x_n, x_c) \Rightarrow \exists y \text{Student}(x_n, y))\}) = \\ \exists f_1(\forall x_n, x_c((\text{Takes1}(x_n, x_c) \wedge \neg \text{Student}(x_n, f_1(x_n, x_c))) \Rightarrow r_{\top}(\bar{0}, \bar{1}))). \end{aligned}$$

## 2.2.2 Transformation of Equality-Generating Constraints into SOTgds

An integrity constraint (egd)  $\forall \mathbf{x}(\phi_A(\mathbf{x}) \Rightarrow (\mathbf{y} \doteq \mathbf{z})) \in \Sigma_A$  of a given schema database  $\mathcal{A} = (S_A, \Sigma_A)$  (in Definition 2) has no relation from the target database on the right-hand side of the implication. It can be directly represented by SOTgds if we consider the equality ' $\doteq$ ' as a built-in binary predicate with relational symbol  $r_{\top}$  such that for any two given tuples  $\mathbf{y} = \langle y_1, \dots, y_k \rangle$  and  $\mathbf{z} = \langle z_1, \dots, z_k \rangle$  the formula  $\mathbf{y} \doteq \mathbf{z}$  is an abbreviation for the formula  $r_{\top}(y_1, z_1) \wedge \dots \wedge r_{\top}(y_k, z_k)$ .

However, in order to unify this presentation with  $\text{tgd}$ 's transformation in the previous section, we will have in mind the following considerations:

Each  $\text{egd}$  is logically equivalent to the sentence  $\forall \mathbf{x} \neg(\phi_A(\mathbf{x}) \wedge \neg(\mathbf{y} \doteq \mathbf{z}))$  which is satisfied if  $\phi_A(\mathbf{x}) \wedge \neg(\mathbf{y} \doteq \mathbf{z})$  is a *falsity* (i.e., false for every assignment of the values to variables in  $\mathbf{x}$ ).

Consequently, we will represent this  $\text{egd}$  by the formula

$$\forall \mathbf{x}((\phi_A(\mathbf{x}) \wedge (\mathbf{y} \neq \mathbf{z})) \Rightarrow r_{\top}(\bar{0}, \bar{1})),$$

where  $\mathbf{y} \neq \mathbf{z}$  is equivalent to  $\neg(\mathbf{y} \doteq \mathbf{z})$ , i.e., an abbreviation for the formula  $(y_1 \neq z_1) \vee \dots \vee (y_k \neq z_k)$ , as follows:

**Lemma 3** Any  $\text{egd} \forall \mathbf{x}(\phi_A(\mathbf{x}) \Rightarrow (\mathbf{y} \doteq \mathbf{z})) \in \Sigma_A^{\text{egd}} \subseteq \Sigma_A$  of a given schema database  $\mathcal{A} = (S_A, \Sigma_A)$ , where  $\mathbf{y} = \langle x_{j_1}, \dots, x_{j_k} \rangle \subseteq \mathbf{x}$  and  $\mathbf{z} = \langle x_{l_1}, \dots, x_{l_k} \rangle \subseteq \mathbf{x}$  such that  $j_i \neq l_i$  for  $1 \leq i \leq k$ , is logically equivalent to the FOL formula:

$$\forall \mathbf{x}((\phi_A(\mathbf{x}) \wedge (\mathbf{y} \neq \mathbf{z})) \Rightarrow r_{\top}(\bar{0}, \bar{1})),$$

where  $\mathbf{y} \neq \mathbf{z}$  is an abbreviation for the formula  $(x_{j_1} \neq x_{l_1}) \vee \dots \vee (x_{j_k} \neq x_{l_k})$ .

*Proof* For any assignment  $g$ , the atom  $r_{\top}(\bar{0}, \bar{1})$  cannot be satisfied because

$$\langle g(\bar{0}), g(\bar{1}) \rangle = \langle 0, 1 \rangle \notin R_{=} = I_T(r_{\top}),$$

so that  $\phi_A(\mathbf{x}) \Rightarrow (\mathbf{y} \doteq \mathbf{z})$  is logically equivalent to the formula  $\phi_A(\mathbf{x}) \Rightarrow ((\mathbf{y} \doteq \mathbf{z}) \vee r_{\top}(\bar{0}, \bar{1}))$ , i.e., to the formula  $\neg\phi_A(\mathbf{x}) \vee (\mathbf{y} \doteq \mathbf{z}) \vee r_{\top}(\bar{0}, \bar{1})$ , or equivalently, to the formula  $\neg(\phi_A(\mathbf{x}) \wedge (\mathbf{y} \neq \mathbf{z})) \vee r_{\top}(\bar{0}, \bar{1})$ , and hence to the formula  $(\phi_A(\mathbf{x}) \wedge (\mathbf{y} \neq \mathbf{z})) \Rightarrow r_{\top}(\bar{0}, \bar{1})$ .  $\square$

This formula does not transfer any information, differently from tgds, as it is expected because on the right side of the implication we have the ground atom without variables.

Based on these considerations, we can define the algorithm for transformation of a given set of egds into a single SOTgd:

**Transformation algorithm**  $\text{EgdsToSOTgd}(\Sigma_A^{\text{egd}})$

**Input.** A set of egds of a given schema  $\mathcal{A}$ ,

$$\Sigma_A^{\text{egd}} = \{\forall \mathbf{x}_1(\phi_{A,1}(\mathbf{x}_1) \Rightarrow (\mathbf{y}_1 \doteq \mathbf{z}_1)), \dots, \forall \mathbf{x}_n(\phi_{A,n}(\mathbf{x}_n) \Rightarrow (\mathbf{y}_n \doteq \mathbf{z}_n))\}.$$

**Output.** A SOTgd

1. (Create the set of implications from egds)

Initialize  $S_{AB}$  to be the empty set.

Put each of the  $n$  implications of tgds in  $\Sigma_A^{\text{egd}}$ ,  $\phi_{A,i}(\mathbf{x}_i) \Rightarrow (\mathbf{y}_i \doteq \mathbf{z}_i)$ , in  $S_{AB}$ .

2. (Convert the egds)

Remove each implication  $\chi$  in  $S_{AB}$  of the form  $\phi_{A,i}(\mathbf{x}_i) \Rightarrow (\mathbf{y}_i \doteq \mathbf{z}_i)$  from  $S_{AB}$  and add the implication  $(\phi_{A,i}(\mathbf{x}_i) \wedge (\mathbf{y}_i \neq \mathbf{z}_i)) \Rightarrow r_{\top}(\bar{0}, \bar{1})$  in  $S_{AB}$ .

### 3. (Construct SOTgd)

If  $S_{AB} = \{\chi_1, \dots, \chi_n\}$  is the set, where  $\chi_1, \dots, \chi_n$  are all the implications from the previous step, then SOTgd is the formula  $\forall \mathbf{x}_1 \chi_1 \wedge \dots \wedge \forall \mathbf{x}_n \chi_n$ , where the variables in  $\mathbf{x}_i$  are all the variables found in the implication  $\chi_i$ , for  $1 \leq i \leq n$ .

**Return** the SOTgd  $\forall \mathbf{x}_1 \chi_1 \wedge \dots \wedge \forall \mathbf{x}_n \chi_n$ .

*Example 4* Let us consider Example 2 with the relation  $\text{Student}(x_1, x_2)$  in the schema  $\mathcal{B}$  and introduce the primary-key integrity constraint for the attribute  $x_1$  (i.e., the student's id) of this relation  $\text{Student}$ , expressed by the following egd:

$$\forall x_1 \forall x_2 \forall x_3 ((\text{Student}(x_1, x_2) \wedge \text{Student}(x_1, x_3)) \Rightarrow (x_2 \doteq x_3)),$$

i.e., the egd  $\forall \mathbf{x}(\phi_A(\mathbf{x}) \Rightarrow (\mathbf{y} \doteq \mathbf{z}))$  with  $\mathbf{x} = (x_1, x_2, x_3)$ ,  $\mathbf{y} = x_2$ ,  $\mathbf{z} = x_3$  where  $\phi_A(\mathbf{x})$  is the conjunctive formula  $\text{Student}(x_1, x_2) \wedge \text{Student}(x_1, x_3)$ .

Then,

$$\begin{aligned} \text{EgdsToSOTgd}(\{\forall x_1 \forall x_2 \forall x_3 ((\text{Student}(x_1, x_2) \\ \wedge \text{Student}(x_1, x_3)) \Rightarrow (x_2 \doteq x_3))\}) \\ = (\forall x_1 \forall x_2 \forall x_3 ((\text{Student}(x_1, x_2) \\ \wedge \text{Student}(x_1, x_3) \wedge (x_2 \neq x_3)) \Rightarrow r_{\top}(\bar{0}, \bar{1})). \end{aligned}$$

## 2.3 New Algorithm for General Composition of SOTgds

First of all, we have to specify what is an *atomic* (or basic) schema mapping in our framework, so that we can use them in the process of composition of other (composed) mappings. We will use only these atomic mappings in order to define a database mapping graphs  $G = (V_G, E_G)$  where the vertices (or nodes) in  $V_G$  are the database schemas and the directed edges in  $E_G$  are the atomic mappings. The mappings obtained by the composition of two consecutive directed edges of this mapping graph are called *composed* mappings and hence, in what follows, we will define a new algorithm for this composition.

**Definition 7** An *atomic mapping* is:

1. An inter-schema mapping  $\mathcal{M}_{AB} = \{\Phi\} : \mathcal{A} \rightarrow \mathcal{B}$  where an SOTgd  $\Phi$  is obtained by the algorithm  $TgdsToSOTgd$  for a given set  $S$  of tgds from a source schema  $\mathcal{A}$  to the target database schema  $\mathcal{B}$ ;
2. An integrity-constraints mapping  $\mathcal{M}_{AA_{\top}} = \{\Phi \wedge \Psi\} : \mathcal{A} \rightarrow \mathcal{A}_{\top}$  where  $\Phi$  is obtained by the algorithm  $EgdsToSOTgd$  for the set of egds  $\Sigma_A^{\text{egd}}$  and  $\Psi$  is obtained by the algorithm  $TgdToConSOTgd$  for the set  $\Sigma_A^{\text{tgd}}$  of tgd-constraints of a schema  $\mathcal{A}$ .

We can use the atomic mappings in order to compose another composed mappings. The algorithm for composition of SOTgds

$$\exists f((\forall \mathbf{x}_1(\phi_{A,1} \Rightarrow \psi_{B,1})) \wedge \cdots \wedge (\forall \mathbf{x}_n(\phi_{A,n} \Rightarrow \psi_{B,n}))) \in \mathcal{M}_{AB} : \mathcal{A} \rightarrow \mathcal{B}$$

and

$$\exists g((\forall \mathbf{y}_1(\phi_{B,1} \Rightarrow \psi_{D,1})) \wedge \cdots \wedge (\forall \mathbf{y}_m(\phi_{B,m} \Rightarrow \psi_{D,m}))) \in \mathcal{M}_{BD} : \mathcal{B} \rightarrow \mathcal{D},$$

presented in [5] for the data-exchange settings, assumes that all relational symbols in  $\{\phi_{B,1}, \dots, \phi_{B,m}\}$  are contained in  $\{\psi_{B,1}, \dots, \psi_{B,n}\}$  as well, so that a composition of these two SOTgds does not introduce new functional symbols. Differently from this setting in [5], it often happens that in our general framework, where a target database  $\mathcal{B}$  is not completely determined by the mappings from the source database  $\mathcal{A}$ , the target database  $\mathcal{B}$  can have some relational symbols that are not used in the mappings  $\mathcal{M}_{AB} : \mathcal{A} \rightarrow \mathcal{B}$ .

Consequently, if these ‘independent’ (w.r.t. mapping  $\mathcal{M}_{AB}$ ) relational symbols in  $\mathcal{B}$  are used in a mapping  $\mathcal{M}_{BD} : \mathcal{B} \rightarrow \mathcal{D}$  then we will need to introduce new characteristic-functional symbols for them (by considering that they are the predicates in FOL) during the composition of the mappings  $\mathcal{M}_{AB}$  and  $\mathcal{M}_{BD}$  into a composed mapping  $\mathcal{M}_{AD} = \mathcal{M}_{BD} \circ \mathcal{M}_{AB} : \mathcal{A} \rightarrow \mathcal{D}$  because they are not the relational symbols in the source schema  $\mathcal{A}$ .

Hence, in our more general setting, we need a more general algorithm than that presented in [5]. Note that, differently from the data-exchange setting, the mapping  $\mathcal{M}_{AB}$  has SOTgds where each relational symbol in  $\psi_{B,i}$  is in  $\mathcal{B}$ , and the mapping  $\mathcal{M}_{BD}$  has SOTgds where each relational symbol in  $\psi_{D,i}$  is in  $\mathcal{D}$ .

Notice that this is, from a logical point of view, a conservative extension of the algorithm in [5] because we only substitute some atoms with logically equivalent equations composed by characteristic functions of the relational symbols of these substituted atoms. This point is important as we will see in the next subsection in order to demonstrate the associativity property of the composition of schema mappings.

In fact, the algorithm in [5] is extended by the new step 4, while steps 1 and 3 are identical (and step 2 is slightly modified) to those presented in [5], as follows:

**New algorithm** Compose  $(\mathcal{M}_{AB}, \mathcal{M}_{BD})$

**Input.** Two schema mappings  $\mathcal{M}_{AB} : \mathcal{A} \rightarrow \mathcal{B}$  and  $\mathcal{M}_{BD} : \mathcal{B} \rightarrow \mathcal{D}$  given by a set of SOTgds. We assumed by Definition 1 of FOL that a “sufficiently large” universe  $\mathcal{U}$  contains the set  $\mathbf{2} = \{0, 1\}$  of classic truth values for the constants  $\bar{0}$  and  $\bar{1}$ .

**Output.** A schema mapping  $\mathcal{M}_{AD} = \mathcal{M}_{BD} \circ \mathcal{M}_{AB} : \mathcal{A} \rightarrow \mathcal{D}$ , which is the composition of  $\mathcal{M}_{AB}$  and  $\mathcal{M}_{BD}$ , represented by an SOTgd.

1. (*Normalize the SOTgds in  $\mathcal{M}_{AB}$  and  $\mathcal{M}_{BD}$* )

Rename the functional symbols so that the functional symbols which appear in  $\mathcal{M}_{AB}$  are all distinct from the functional symbols which appear in  $\mathcal{M}_{BD}$ . For notational convenience, we shall refer to variables in  $\mathcal{M}_{AB}$  as  $x$ ’s, possibly with

subscripts, and the variables in  $\mathcal{M}_{BD}$  as  $y$ 's, possibly with subscripts. Initialize  $S_{AB}$  and  $S_{BD}$  to be the empty sets. Assume that the SOTgd in  $\mathcal{M}_{AB}$  is

$$\exists \mathbf{f}((\forall \mathbf{x}_1(\phi_{A,1} \Rightarrow \psi_{B,1})) \wedge \cdots \wedge (\forall \mathbf{x}_n(\phi_{A,n} \Rightarrow \psi_{B,n}))).$$

For  $1 \leq i \leq n$ , put the  $n$  implications  $\phi_{A,i} \Rightarrow \psi_{B,i}$  into  $S_{AB}$  if these implications are not ground formulas. We do likewise for  $\mathcal{M}_{BD}$  and  $S_{BD}$ . If  $S_{AB} = \emptyset$  or  $S_{BD} = \emptyset$  then set  $S_{BD} = \{r_\emptyset \Rightarrow r_\emptyset\}$  and go to step 5.

Each implication  $\chi$  in  $S_{AB}$  has the form  $\phi_{A,i}(\mathbf{x}_i) \Rightarrow \wedge_{1 \leq j \leq k} r_j(\mathbf{t}_j)$  where every member of  $\mathbf{x}_i$  is a universally quantified variable, and each  $\mathbf{t}_j$ , for  $1 \leq j \leq k$ , is a sequence (tuple) of terms over  $\mathbf{x}_i$ . We then replace each such implication  $\chi$  in  $S_{AB}$  with  $k$  implications  $\phi_{A,i}(\mathbf{x}_i) \Rightarrow r_1(\mathbf{t}_1), \dots, \phi_{A,i}(\mathbf{x}_i) \Rightarrow r_k(\mathbf{t}_k)$ .

In all implications in  $S_{BD}$  replace all equations  $(f_{r_i}(\mathbf{t}_i) = 1)$  with  $r_i \in \mathcal{A}$  on the left-hand sides by the atoms  $r_i(\mathbf{t}_i)$ .

2. (*Compose  $S_{AB}$  with  $S_{BD}$* )

Repeat the following as far as possible:

For each implication  $\chi$  in  $S_{BD}$  of the form  $\varphi \Rightarrow \psi_D$  where there is an atom  $r(\mathbf{y})$  in  $\varphi$ , we perform the following steps to (possibly) replace  $r(\mathbf{y})$  with atoms over  $\mathcal{A}$ . (The part of formula  $\varphi$  composed of built-in predicates  $\doteq, \neq, >, <, \dots$  is left unchanged). Let  $\phi_1 \Rightarrow r(\mathbf{t}_1), \dots, \phi_l \Rightarrow r(\mathbf{t}_l)$  be all the implications in  $S_{AB}$  whose right-hand side is an atom with the relational symbol  $r$ . For each such implication  $\phi_i \Rightarrow r(\mathbf{t}_i)$ , rename the variables in this implication so that they do not overlap with the variables in  $\chi$ . (In fact, every time we compose this implication, we take a fresh copy of the implication, with new variables.) Let  $\theta_i$  be the conjunction of the equalities between the variables in  $r(\mathbf{y})$  and the corresponding terms in  $r(\mathbf{t}_i)$ , position by position. For example, the conjunction of equalities, position by position, between  $r(y_1, y_2, y_3)$  and  $r(x_1, f_2(x_2), f_1(x_3))$  is  $(y_1 \doteq x_1) \wedge (y_2 \doteq f_2(x_2)) \wedge (y_3 \doteq f_1(x_3))$ . Observe that every equality that is generated has the form  $y \doteq t$  where  $y$  is a variable in  $\mathcal{M}_{BD}$  and  $t$  is a term based on variables in  $\mathcal{M}_{BD}$  and on functions in the tuple  $\mathbf{f}$ . Remove  $\chi$  from  $S_{BD}$  and add  $l$  implications to  $S_{BD}$  as follows: replace  $r(\mathbf{y})$  in  $\chi$  with  $\phi_i \wedge \theta_i$  and add the resulting implication to  $S_{BD}$ , for  $1 \leq i \leq l$ .

3. (*Remove the variables originally in  $\mathcal{M}_{BD}$* )

For each implication  $\chi$  constructed in the previous step, perform the elimination of the variables  $y_i$  from  $\mathcal{M}_{BD}$  as far as possible: Select an equality  $y_i \doteq t$  that was generated in the previous step (thus  $y_i$  is a variable in  $\mathcal{M}_{BD}$  and  $t$  is a term based on variables in  $\mathcal{M}_{AB}$  and on  $\mathbf{f}$ ). Remove the equality  $y_i \doteq t$  from  $\chi$  and replace every remaining occurrence of  $y_i$  in  $\chi$  by  $t$ .

4. (*Remove remained atoms with relational symbols in  $\mathcal{B}$  that are not in  $\mathcal{A}$* )

For the implications in  $S_{BD}$  repeat the following, in order to eliminate all relational symbols (on the left-hand side of implications) that are not in  $\mathcal{A}$ :

For each  $\chi$  in  $S_{BD}$  such that its left-hand side is equal to a conjunction  $\phi_1(\mathbf{x})$  (that contains only the relational symbols in  $\mathcal{A}$  and the equations with functional symbols) and  $\phi_2(\mathbf{x}', y_1, \dots, y_k)$  (with  $\mathbf{x}'$  a subset of  $\mathbf{x}$ ) that contains only the atoms with relational symbols that are not in  $\mathcal{A}$ , we modify  $\phi_2$  from the left-hand side

of  $\chi$  by replacing each atom  $r_i(\mathbf{x}_i, \mathbf{y}_i)$  in  $\phi_2$  (where  $\mathbf{x}_i \subseteq \mathbf{x}'$  and  $\mathbf{y}_i \subseteq \{y_1, \dots, y_k\}$ ) with the equation  $(f_{r_i}(\mathbf{x}_i, \mathbf{y}_i) \doteq \bar{1})$  by introducing its characteristic function  $f_{r_i}$  (i.e., for a given Tarski's interpretation  $I_T$  in Definition 1,  $I_T(f_{r_i})(\mathbf{c}) = 1$  if  $\mathbf{c}$  is a tuple of the interpreted relation  $I_T(r_i)$ ; 0 otherwise).

5. (*Construct  $\mathcal{M}_{AD}$* )

If  $S_{BD} = \{\chi_1, \dots, \chi_j\}$  is the set, where  $\chi_1, \dots, \chi_j$  are all the implications from the previous step, then  $\mathcal{M}_{AD}$  is a singleton set composed of an SOTgd element

$$\exists \mathbf{g}(\forall \mathbf{z}_1 \chi_1 \wedge \dots \wedge \forall \mathbf{z}_j \chi_j),$$

where  $\mathbf{g}$  is the tuple of all function symbols that appear in any of the implications in  $S_{BD}$ , and where the variables in  $\mathbf{z}_i$  are all the variables found in the implication  $\chi_i$ , for  $1 \leq i \leq j$ .

**Return**  $\mathcal{M}_{AD} = \{\exists \mathbf{g}(\forall \mathbf{z}_1 \chi_1 \wedge \dots \wedge \forall \mathbf{z}_j \chi_j)\} : \mathcal{A} \rightarrow \mathcal{D}$ .

The assumption that  $\mathbf{2} \subset \mathcal{U}$  is another generalization of the previous algorithm in [5]; it is necessary for the introduction of characteristic functions in the step 4 of this new algorithm.

Note that we also do not obtain the empty composition if there is no direct mapping from the relations in  $\mathcal{A}$  into  $\mathcal{D}$ . For example, let us consider the database

$$\begin{aligned} \mathcal{A} &= (S_A, \emptyset) \quad \text{with } S_A = \{\text{Takes}(x_n, x_c)\}, \\ \mathcal{B} &= (S_B, \emptyset) \quad \text{with } S_B = \{\text{Takes1}(x_n, x_c), \text{Professor}(x_p)\}, \\ \mathcal{D} &= (S_D, \emptyset) \quad \text{with } S_D = \{\text{Professor1}(x_p)\} \end{aligned}$$

and with mappings

$$\mathcal{M}_{AB} = \{\forall x_n \forall x_c (\text{Takes}(x_n, x_c) \Rightarrow \text{Takes1}(x_n, x_c))\} : \mathcal{A} \rightarrow \mathcal{B}$$

and

$$\mathcal{M}_{BD} = \{\forall x_p (\text{Professor}(x_p) \Rightarrow \text{Professor1}(x_p))\} : \mathcal{B} \rightarrow \mathcal{D}.$$

Then,

$$\mathcal{M}_{AD} = \mathcal{M}_{BD} \circ \mathcal{M}_{AB}$$

$$= \{\exists f_{\text{Professor}} (\forall x_p ((f_{\text{Professor}}(x_p) = \bar{1}) \Rightarrow \text{Professor1}(x_p)))\} : \mathcal{A} \rightarrow \mathcal{D}.$$

*Example 5* Let us consider Example 2.3 in [5] (also discussed in our Example 2) for the composition of SOTgds  $\mathcal{M}'_{AD} = \mathcal{M}_{BD} \circ \mathcal{M}_{AB}$ , where

$$\begin{aligned} \mathcal{M}_{AB} &= \{\forall x_n \forall x_c (\text{Takes}(x_n, x_c) \Rightarrow \text{Takes1}(x_n, x_c)), \\ &\quad \forall x_n \exists y \forall x_c (\text{Takes}(x_n, x_c) \Rightarrow \text{Student}(x_n, y))\}, \end{aligned}$$

or expressed equivalently by SOTgd

$$\mathcal{M}_{AB} = \{ \exists f_1 (\forall x_n \forall x_c (\text{Takes}(x_n, x_c) \Rightarrow \text{Takes1}(x_n, x_c)) \\ \wedge \forall x_n \forall x_c (\text{Takes}(x_n, x_c) \Rightarrow \text{Student}(x_n, f_1(x_n)))) \},$$

and

$$\mathcal{M}_{BD} = \{ \forall x_n \forall x_c \forall y ((\text{Takes1}(x_n, x_c) \\ \wedge \text{Student}(x_n, y)) \Rightarrow \text{Enrolment}(y, x_c)) \}.$$

1. We rename the variables as required by the algorithm and obtain

$$\mathcal{M}_{AB} = \{ \exists f_1 (\forall x_1 \forall x_2 (\text{Takes}(x_1, x_2) \Rightarrow \text{Takes1}(x_1, x_2)) \\ \wedge \forall x_1 \forall x_2 (\text{Takes}(x_1, x_2) \Rightarrow \text{Student}(x_1, f_1(x_1)))) \} \\ \mathcal{M}_{BD} = \{ \forall y_1 \forall y_2 \forall y_3 ((\text{Takes1}(y_1, y_2) \\ \wedge \text{Student}(y_1, y_3)) \Rightarrow \text{Enrolment}(y_3, y_2)) \}.$$

Then we obtain

$$S_{AB} = \{ \text{Takes}(x_1, x_2) \Rightarrow \text{Takes1}(x_1, x_2), \\ \text{Takes}(x_1, x_2) \Rightarrow \text{Student}(x_1, f_1(x_1)) \}, \\ S_{BD} = \{ (\text{Takes1}(y_1, y_2) \wedge \text{Student}(y_1, y_3)) \Rightarrow \text{Enrolment}(y_3, y_2) \}.$$

2. We obtain

$$S_{BD} = \{ ((\text{Takes}(x_1, x_2) \wedge (y_1 \doteq x_1) \wedge (y_2 \doteq x_2)) \\ \wedge (\text{Takes}(x_1, x_2) \wedge (y_1 \doteq x_1) \wedge (y_3 \doteq f_1(x_1)))) \\ \Rightarrow \text{Enrolment}(y_3, y_2) \}.$$

3. We obtain  $S_{BD} = \{ \text{Takes}(x_1, x_2) \Rightarrow \text{Enrolment}(f_1(x_1), x_2) \}$ .

4. Non applicable.

5. **Return**

$$\mathcal{M}'_{AD} = \{ \exists f_1 (\forall x_1 \forall x_2 (\text{Takes}(x_1, x_2) \Rightarrow \text{Enrolment}(f_1(x_1), x_2))) \}.$$

Let us explain the difference of this more general algorithm w.r.t. the original algorithm presented in [5]:

*Example 6* Let us consider the composition of SOTgds  $\mathcal{M}'_{AD} = \mathcal{M}_{BD} \circ \mathcal{M}_{AB}$  where  $\mathcal{M}_{AB}$  in Example 5 above is reduced to

$$\mathcal{M}_{AB} = \{ \forall x_n \forall x_c (\text{Takes}(x_n, x_c) \Rightarrow \text{Takes1}(x_n, x_c)) \}$$

while  $\mathcal{M}_{BD}$  remains unchanged.



1. We rename the variables as required by the algorithm and obtain

$$\begin{aligned}\mathcal{M}_{AB} &= \{\forall x_1 \forall x_2 (\text{Takes}(x_1, x_2) \Rightarrow \text{Takes1}(x_1, x_2))\}, \\ \mathcal{M}_{BD} &= \{\forall y_1 \forall y_2 \forall y_3 ((\text{Takes1}(y_1, y_2) \wedge \text{Student}(y_1, y_3)) \\ &\quad \Rightarrow \text{Enrolment}(y_3, y_2))\}.\end{aligned}$$

Hence,

$$\begin{aligned}S_{AB} &= \{\text{Takes}(x_1, x_2) \Rightarrow \text{Takes1}(x_1, x_2)\}, \\ S_{BD} &= \{(\text{Takes1}(y_1, y_2) \wedge \text{Student}(y_1, y_3)) \Rightarrow \text{Enrolment}(y_3, y_2)\}.\end{aligned}$$

2. We obtain

$$\begin{aligned}S_{BD} &= \{((\text{Takes}(x_1, x_2) \wedge (y_1 \doteq x_1) \\ &\quad \wedge (y_2 \doteq x_2)) \wedge \text{Student}(y_1, y_3)) \Rightarrow \text{Enrolment}(y_3, y_2)\}.\end{aligned}$$

3. We obtain

$$S_{BD} = \{(\text{Takes}(x_1, x_2) \wedge \text{Student}(x_1, y_3)) \Rightarrow \text{Enrolment}(y_3, x_2)\}.$$

4. The left-hand side of the implication in  $S_{BD}$  contains the atom  $\text{Student}(x_1, y_3)$  where  $\text{Student}$  is a relational symbol in  $\mathcal{B}$ , and hence we replace it by the equation  $(h_{\text{Student}}(x_1, y_3) \doteq \bar{1})$ . Thus we obtain

$$S_{BD} = \{(\text{Takes}(x_1, x_2) \wedge (h_{\text{Student}}(x_1, y_3) \doteq \bar{1})) \Rightarrow \text{Enrolment}(y_3, x_2)\}.$$

### 5. Return

$$\begin{aligned}M'_{AD} &= \{\exists h_{\text{Student}} (\forall x_1 \forall x_2 \forall y_3 ((\text{Takes}(x_1, x_2) \wedge (h_{\text{Student}}(x_1, y_3) \doteq \bar{1})) \\ &\quad \Rightarrow \text{Enrolment}(y_3, x_2)))\}.\end{aligned}$$

Note that in the two cases, for

$$\begin{aligned}\mathcal{M}_{AB} &= \{\exists f_1 (\forall x_n \forall x_c (\text{Takes}(x_n, x_c) \Rightarrow \text{Takes1}(x_n, x_c)) \\ &\quad \wedge \forall x_n \forall x_c (\text{Takes}(x_n, x_c) \Rightarrow \text{Student}(x_n, f_1(x_n))))\}\end{aligned}$$

and for  $\mathcal{M}_{AB} = \{\forall x_n \forall x_c (\text{Takes}(x_n, x_c) \Rightarrow \text{Takes1}(x_n, x_c))\}$ , we obtained different SOTgds of the resulting composition. In fact, in the second case, we did not express the fact that the second argument in  $\text{Student}$  depends only on its first argument, as in the first case. So, if we wanted to obtain equivalent functions for both  $f_1$  and  $h_{\text{Student}}$ , in the second case we would need to introduce the key integrity constraint for the relation  $\text{Student}$  in the schema  $\mathcal{B}$  given by the egd (from Example 4)

$$\forall z_1 \forall z_2 \forall z_3 ((\text{Student}(z_1, z_2) \wedge \text{Student}(z_1, z_3)) \Rightarrow (z_2 \doteq z_3))$$

so that for any instance  $\mathcal{D}$  that satisfies also the mapping  $\mathcal{M}'_{AD}$ ,  $h_{Student}(x_1, y_3) \doteq 1$  is equivalent to  $y_3 \doteq f_1(x_1)$ .

### 2.3.1 Categorical Properties for the Schema Mappings

Now we will show that this new algorithm for composition of schema mappings expressed by SOTgds can be used as the composition of morphisms in a category where the objects are database schemas and morphisms (the arrows in such a category) are the mappings. We recall that, as it was presented in [5], the space of instances  $Inst(\mathcal{M}_{AB})$  of a mapping  $\mathcal{M}_{AB} : \mathcal{A} \rightarrow \mathcal{B}$  is a binary relation between instance-databases of  $\mathcal{A}$  and target instance-databases of  $\mathcal{B}$ . Consequently, the schema mapping  $\mathcal{M}_{AC}$  is a composition of two schema mappings  $\mathcal{M}_{AB}$  and  $\mathcal{M}_{BC}$ , denoted by  $\mathcal{M}_{BC} \circ \mathcal{M}_{AB}$ , if the space of instances of  $\mathcal{M}_{AC}$  is the set-theoretic composition of the spaces of instances of  $\mathcal{M}_{AB}$  and  $\mathcal{M}_{BC}$ . The advantage of this approach is that the set of formulae defining a composition  $\mathcal{M}_{AC}$  of  $\mathcal{M}_{AB}$  and  $\mathcal{M}_{BC}$  is *unique* up to logical equivalence. Consequently,

$$(Comp) \quad \mathcal{M}_{AC} = \mathcal{M}_{BC} \circ \mathcal{M}_{AB} \quad \text{if} \quad Inst(\mathcal{M}_{AC}) = Inst(\mathcal{M}_{AB}) * Inst(\mathcal{M}_{BC}),$$

where  $*$  denotes the composition of binary relations.

**Lemma 4** *The composition of two schema mappings  $\mathcal{M}_{AB}$  and  $\mathcal{M}_{BC}$ , presented by the new algorithm as  $\mathcal{M}_{BC} \circ \mathcal{M}_{AB} \triangleq Compose(\mathcal{M}_{AB}, \mathcal{M}_{BC})$ , is associative.*

*Proof* From the fact that the new algorithm is a conservative extension of the algorithm in [5] (see the proof of Proposition 4.1 in [5] for more details) such that in the new step 4 we only replace the remaining atoms (that are removed in this step) by their characteristic functions, the property (Comp) above is still valid. Consequently,  $\mathcal{M}_{AD} = \mathcal{M}_{CD} \circ (\mathcal{M}_{BC} \circ \mathcal{M}_{AB}) = \mathcal{M}_{CD} \circ \mathcal{M}_{AC}$  if

$$\begin{aligned} Inst(\mathcal{M}_{AD}) &= Inst(\mathcal{M}_{AC}) * Inst(\mathcal{M}_{CD}) \\ &= (Inst(\mathcal{M}_{AB}) * Inst(\mathcal{M}_{BC})) * Inst(\mathcal{M}_{CD}) \quad (\text{by (Comp)}) \\ &= Inst(\mathcal{M}_{AB}) * Inst(\mathcal{M}_{BC}) * Inst(\mathcal{M}_{CD}). \quad (\text{by associativity of } *) \end{aligned}$$

Analogously,  $\mathcal{M}'_{AD} = (\mathcal{M}_{CD} \circ \mathcal{M}_{BC}) \circ \mathcal{M}_{AB} = \mathcal{M}_{BD} \circ \mathcal{M}_{AB}$  if

$$\begin{aligned} Inst(\mathcal{M}'_{AD}) &= Inst(\mathcal{M}_{AB}) * Inst(\mathcal{M}_{BD}) \\ &= Inst(\mathcal{M}_{AB}) * (Inst(\mathcal{M}_{BC}) * Inst(\mathcal{M}_{CD})) \quad (\text{by (Comp)}) \\ &= Inst(\mathcal{M}_{AB}) * Inst(\mathcal{M}_{BC}) * Inst(\mathcal{M}_{CD}). \quad (\text{by associativity of } *) \end{aligned}$$

Consequently,  $Inst(\mathcal{M}'_{AD}) = Inst(\mathcal{M}_{AD})$ , that is,  $\mathcal{M}'_{AD} = \mathcal{M}_{AD}$ , i.e.,

$$(\mathcal{M}_{CD} \circ \mathcal{M}_{BC}) \circ \mathcal{M}_{AB} = \mathcal{M}_{CD} \circ (\mathcal{M}_{BC} \circ \mathcal{M}_{AB}),$$

or in other words, the operation of composition  $\circ$  for the new algorithm *Compose* is associative.  $\square$

Let us show that we are able to define the identity mappings for any schema  $\mathcal{A}$ , such that composition with another arrow is equal to this arrow:

**Lemma 5** *For a schema  $\mathcal{A} = (S_A, \Sigma_A)$  we define its identity mapping as*

$$\mathcal{M}_{AA} = Id_A = \left\{ \bigwedge \{ \forall \mathbf{x}_i (r_i(\mathbf{x}_i) \Rightarrow r_i(\mathbf{x}_i)) \mid r_i \in S_A \} \right\} : \mathcal{A} \rightarrow \mathcal{A}.$$

*Consequently, for any two mappings  $\mathcal{M}_{BA} : \mathcal{B} \rightarrow \mathcal{A}$  and  $\mathcal{M}_{AB} : \mathcal{A} \rightarrow \mathcal{B}$ ,*

$$\mathcal{M}_{AB} \circ Id_A = Compose(\mathcal{M}_{AB}, Id_A) = \mathcal{M}_{AB}$$

*and  $Id_A \circ \mathcal{M}_{BA} = Compose(Id_A, \mathcal{M}_{BA}) = \mathcal{M}_{BA}$ .*

*Proof* Let us verify that  $Compose(\mathcal{M}_{AB}, Id_A) = \mathcal{M}_{AB}$ . In step 2, each  $r_i(\mathbf{y}_i)$  on the left-hand side of the implication in  $S_{BD}$  is replaced by the conjunction  $r_i(\mathbf{x}_i) \wedge (\mathbf{y}_i \doteq \mathbf{x}_i)$ . Consequently, in step 3, we replace all  $\mathbf{y}$ 's on the right-hand side of implications with  $\mathbf{x}$ 's and eliminate the equations  $\mathbf{y}_i \doteq \mathbf{x}_i$  from the left-hand side of these implications, so that we obtain exactly the SOTgd of the mapping  $\mathcal{M}_{AB}$ .

Let us verify that  $Compose(Id_A, \mathcal{M}_{BA}) = \mathcal{M}_{BA}$ . In step 2, each  $r_i(\mathbf{y}_i)$  on the left-hand side of the implication in  $S_{BD}$  such that there is an implication  $\phi_i(\mathbf{y}_i) \Rightarrow r_i(\mathbf{t}_i)$  in  $S_{AB}$  is replaced by the conjunction  $\phi_i(\mathbf{y}_i) \wedge (\mathbf{y}_i \doteq \mathbf{t}_i)$ . Consequently, in step 3, we replace all  $\mathbf{y}$ 's on the right-hand side of implications with terms in  $\mathbf{t}_i$  and we eliminate the equations  $\mathbf{y}_i \doteq \mathbf{t}_i$  from the left-hand side of these implications, so that we obtain exactly the SOTgd of the mapping  $\mathcal{M}_{BA}$ .  $\square$

Based on these two lemmas, we have the following important corollary:

**Corollary 2** *The database schemas and the composition of their schema mappings can be represented by a sketch category.*

*Proof* It is a direct result of Lemma 4 which demonstrates the associativity for composition of the mappings (i.e., morphisms of a sketch category) and Lemma 5 that demonstrates that for each schema (i.e., object of a sketch category) there is an identity mapping.  $\square$

Note that the identity SOTgds are not tuple-generating, that is, they do not insert new tuples in the target database. From the fact that each implication is of the form  $r_i(\mathbf{x}_i) \Rightarrow r_i(\mathbf{x}_i)$  and hence if for a tuple of values  $\mathbf{c}$ ,  $r_i(\mathbf{c})$  is true, then  $\mathbf{c}$  is already a tuple in the relation  $r_i$ . Thus, this implication is true without inserting a new tuple in the relation  $r_i$ .

From the logical point of view, the SOTgd of an identity mapping is a tautology and, consequently, its conjunction with another SOTgd is equivalent logically to this SOTgd. Consequently, based on this general algorithm, we can represent the

logic-syntax of (composed) schema mappings by SOTgds. However, we need a new semantics for the mappings in order to define the (strict) equality between two mappings, different from the logical equivalence [5] used in data-exchange settings. The data-exchange setting is only a particular case of our more general setting. Thus, in our setting two logically equivalent schema-mappings are always equal at instance-level as well. However, we can have equal mappings (at instance level) which are not logically equivalent at schema level as it was presented by Example 2. Thus, in what follows, we will investigate this equality of mappings at the instance-level between instance-databases, and the first step is to pass from logic to algebras and hence to functorial semantics of database mappings.

## 2.4 Logic versus Algebra: Categorification by Operads

We consider that logical syntax for composition of schema mappings, represented by SOTgds introduced in [5], is well suited for consideration of the logical syntax and semantics of schema mappings, and we explained how SOTgds hide the relations of ‘intermediate’ databases by substituting them with existentially quantified characteristic functions of such relations.

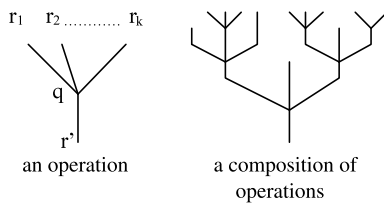
Another way, more closed to the functorial categorical representation, is to assume an algebraic point of view for the representation of atomic and composed mappings. It is possible because each conjunctive formula (a view) used on the left-hand side of the implication forms of tgds can be equivalently represented by a term of SPJRU-relational algebra. Any elementary (or atomic) mapping may be represented by a set of these algebraic terms, while the composition of atomic mappings may be represented algebraically by the term-trees, or formally, by using the algebraic theory of operads.

In what follows, we will use this algebraic point of view based on the operads, which is a more convenient than a standard Lindenbaum-like translation of the second-order tgds logic into the algebraic categorical setting.

In what follows, we will work with the typed operads, first developed for homotopy theory [1, 3, 7], having a set  $\mathbb{R}$  of types (each finitary relation symbol is a type), or “R-operads” for short. The basic idea of an R-operad  $O$  is that, given types  $r_1, \dots, r_k, r \in \mathbb{R}$ , there is a set  $O(r_1, \dots, r_k, r)$  of abstract  $k$ -ary “operations” with inputs of type  $r_1, \dots, r_k$  and output of type  $r$ . In short, an operad describes a family of composable operations with multiple inputs and one output, satisfying several intuitive properties like associativity of composition and permutability of the inputs. The main difference from the *universal algebra* approach where  $R$  would be a carrier set and the operad’s operations would compose the signature of such an algebra, here we emphasize just the “operations” by treating them as a carrier set so that the symbol of their composition ‘ $\cdot$ ’ is just an operator of the R-operads signature.

Categorification is a process of finding category-theoretic analog of set-theoretic concepts by introducing ‘ $n$ -categories’, i.e., algebraic structures having objects, morphisms between objects, 2-morphisms between morphisms, and so on up to  $n$ -morphisms. In [2], the existence of a close relation between  $n$ -categories and the

**Fig. 2.1** Operations of an R-operad



homotopy theory is demonstrated, based on theory of operads. As we will see in our case of **DB** category used for denotational semantics of database mappings, this **DB** category is an  $n$ -category as well, but with a particular property that each arrow can be equivalently represented by an object (i.e., a categorical symmetry) and hence the higher-order arrows are represented just by ordinary 1-arrows in **DB**. Basically, the work presented in this book is a categorification of the logic theory of the schema database mappings.

*Remark* In our case, the input types are the relational symbols of the source database of a given schema mapping (represented logically by an SOTgd) while the output type is a relational symbol of the target database. Each single abstract “operation” is derived from a logical implication, whose left-hand side is a conjunctive query over the source database schema, into a particular relation of the target database schema. Thus, it represents an SPJRU term of the relational algebra corresponding to the left-hand side (a conjunctive query, possibly with negations (in the case of integrity constraints), over the source schema) of this implication, labeled by a node  $q$  on the left tree (an operation) in Fig. 2.1. In what follows, we will provide an algorithm *MakeOperads* that transforms a given SOTgd (of a given mapping from a source to a target schema database) into the set of such abstract operations with input types in the source schema and output types in the target schema.

We can visualize such an operation as a tree with only one node (a relational symbol of the target schema). In an operad, we can obtain new operations from old ones by composing them; it can be visualized in terms of trees in Fig. 2.1. We can obtain the new operators from old ones by permuting arguments, and there is a unary “identity” operation of each type. Finally, we insist on a few plausible axioms: the identity operations act as identities for composition, permuting arguments is compatible with composition, and *composition is associative*. Thus, formally, we have the following:

**Definition 8** For any set  $\mathbb{R}$  of relational symbols with finite arity (the predicate letters in Definition 1) an R-operad  $\mathcal{O}$  consists of:

1. A set  $\mathcal{O}(r_1, \dots, r_k, r)$  for any  $r_1, \dots, r_k, r \in \mathbb{R}$  and an element  $1_r \in \mathcal{O}(r, r)$  for any  $r \in \mathbb{R}$ . We denote the empty type by  $r_\emptyset$  (the truth propositional letter in FOL, Definition 1), with  $ar(r_\emptyset) = 0$  and  $1_{r_\emptyset} \in \mathcal{O}(r_\emptyset, r_\emptyset)$ .
2. An element  $f \cdot (g_1, \dots, g_k) \in \mathcal{O}(r_{11}, \dots, r_{1i_1}, \dots, r_{k1}, \dots, r_{ki_k}, r)$  for any  $f \in \mathcal{O}(r_1, \dots, r_k, r)$  and any  $g_1 \in \mathcal{O}(r_{11}, \dots, r_{1i_1}, r_1), \dots, g_k \in \mathcal{O}(r_{k1}, \dots, r_{ki_k}, r_k)$ .

3. A map  $\sigma : O(r_1, \dots, r_k, r) \rightarrow O(r_{\sigma(1)}, \dots, r_{\sigma(k)}, r)$ ,  $f \mapsto f\sigma$  for any permutation  $\sigma \in \mathbb{R}_k$  such that:
- (a) (associativity) Whenever both sides make sense,  $f \cdot (g_1 \cdot (h_{11}, \dots, h_{1i_1}), \dots, g_k \cdot (h_{k1}, \dots, h_{ki_k})) = (f \cdot (g_1, \dots, g_k)) \cdot (h_{11}, \dots, h_{1i_1}, \dots, h_{k1}, \dots, h_{ki_k})$ ;
  - (b) For any  $f \in O(r_1, \dots, r_k, r)$ ,  $f = 1_r \cdot f = f \cdot (1_{r_1}, \dots, 1_{r_k})$ ;
  - (c) For any  $f \in O(r_1, \dots, r_k, r)$  and  $\sigma, \sigma_1 \in \mathbb{R}_k$ ,  $f(\sigma\sigma_1) = (f\sigma)\sigma_1$ ;
  - (d) For any  $f \in O(r_1, \dots, r_k, r)$ ,  $\sigma \in \mathbb{R}_k$  and  $g_1 \in O(r_{11}, \dots, r_{1i_1}, r_1), \dots, g_k \in O(r_{k1}, \dots, r_{ki_k}, r_k)$ ,  $(f\sigma) \cdot (g_{\sigma(1)}, \dots, g_{\sigma(k)}) = (f \cdot (g_1, \dots, g_k))\rho(\sigma)$  where  $\rho : \mathbb{R}_k \rightarrow \mathbb{R}_{i_1+\dots+i_k}$  is the obvious homomorphism;
  - (e) For any  $f \in O(r_1, \dots, r_k, r)$ ,  $g_1 \in O(r_{11}, \dots, r_{1i_1}, r_1), \dots, g_k \in O(r_{k1}, \dots, r_{ki_k}, r_k)$  and  $\sigma_1 \in \mathbb{R}_{i_1}, \dots, \sigma_k \in \mathbb{R}_{i_k}$ ,

$$(f \cdot (g_1\sigma_1, \dots, g_k\sigma_k)) = (f \cdot (g_1, \dots, g_k))\varrho_1(\sigma_1, \dots, \sigma_k),$$

where  $\varrho_1 : \mathbb{R}_{i_1} \times \dots \times \mathbb{R}_{i_k} \rightarrow \mathbb{R}_{i_1+\dots+i_k}$  is the obvious homomorphism.

Thus, an R-operad  $O$  can be considered as an algebra where the variables (a carrier set) are the typed operators  $f \in O(r_1, \dots, r_k, r)$  and the signature is composed of the set of basic operations used for composition of the elements of this carrier set:

1. The binary associative composition operation (a partial function) ‘ $\cdot$ ’ such that

$$\cdot(f, (g_1, \dots, g_k)) = f \cdot (g_1, \dots, g_k),$$

if  $f$  is a  $k$ -ary typed operator.

2. The left and right identity (partial) operations  $1_r \cdot \_$ ,  $\_ \cdot (1_{r_1}, \dots, 1_{r_1})$  such that for a  $k$ -ary typed operator  $f$ ,

$$1_r \cdot \_ (f) = 1_r \cdot f = f, \quad \_ \cdot (1_{r_1}, \dots, 1_{r_1})(f) = f \cdot (1_{r_1}, \dots, 1_{r_1}) = f.$$

3. The permutation (partial) operations,

$$\sigma : O(r_1, \dots, r_k, r) \rightarrow O(r_{\sigma(1)}, \dots, r_{\sigma(k)}, r),$$

such that for each  $f \in O(r_1, \dots, r_k, r)$ ,  $\sigma(f) = f\sigma$ .

Thus, all operations in the signature of this R-operads algebra are the *partial* functions (defined only for the subsets of elements of the carrier set of this algebra).

It is clear that in this formal syntax of R-operads algebra, the relational symbols are left out of its syntax; thus, the terms of this algebra do not contain the relational symbols. Consequently, it appears fundamentally different from the relational algebras (as, for example, the basic SPRJU Codd’s algebra and its extensions, examined in Chap. 5). However, in Chap. 5, where we will consider the extensions of the Codd’s relational algebra, we will take a different point of view of the R-algebras of operads by considering its carrier set (the variables) equal to the set  $\mathbb{R}$  of relational symbols (considered only as implicit “types” for operads). Hence, each  $f \in O(r_1, \dots, r_k, r)$  will be a composed operation of such a relational  $\Sigma_\alpha$  algebra,

and we will define its complete signature  $\Sigma_\alpha$ . Then we will demonstrate in Sect. 5.4, Theorem 9 and Corollary 19, that such a  $\Sigma_\alpha$  algebra (where  $f \in O(r_1, \dots, r_k, r)$  are considered not as a carrier-set but as the set of composed operations in this algebra) is computationally equivalent to the largest extension of the Codd's relational algebra (which contains all update operations for the relations).

We consider an R-algebra in Definition 8 in the way where not the relations but the operads  $f \in O(r_1, \dots, r_k, r)$  constitute its carrier-set and hence it is similar to the consideration of a category as a kind of algebra (see the introduction, Sect. 1.5.1) where the carrier set is composed of objects and arrows with the basic operation just the associative composition 'o' of the arrows, which is a partial function, as is the associative operation '.' for the operads. This analogy (both with a composition with the identity elements) is a natural justification for why we are using the interpretation of R-operads algebra in Definition 8 as a basis for the definition of the **DB** category for database-mappings.

Let us define the algorithm that transform an SOTgd into a set of abstract operads:

**Operads algorithm** *MakeOperads*( $\mathcal{M}_{AB}$ )

**Input.** A schema mapping  $\mathcal{M}_{AB} : \mathcal{A} \rightarrow \mathcal{B}$  given by an SOTgd

$$\exists \mathbf{f}((\forall \mathbf{x}_1(\phi_{A,1} \Rightarrow \psi_{B,1})) \wedge \dots \wedge (\forall \mathbf{x}_n(\phi_{A,n} \Rightarrow \psi_{B,n}))).$$

**Output.** Set of abstract operad's operations from  $\mathcal{A}$  into  $\mathcal{B}$

1. (*Normalize the SOTgd*)

Initialize  $S$  to be the empty set  $\emptyset$ . Let  $\mathcal{M}_{AB}$  be the singleton set

$$\{\exists \mathbf{f}((\forall \mathbf{x}_1(\phi_{A,1} \Rightarrow \psi_{B,1})) \wedge \dots \wedge (\forall \mathbf{x}_n(\phi_{A,n} \Rightarrow \psi_{B,n})))\}.$$

For  $1 \leq i \leq n$ , put the implication  $\phi_{A,i} \Rightarrow \psi_{B,i}$  into  $S$  if this implication is not a ground formula. If  $S$  is empty go to step 4.

Each implication  $\chi$  in  $S$  has the form  $\phi_{A,i}(\mathbf{x}_i) \Rightarrow \wedge_{1 \leq j \leq k} r_j(\mathbf{t}_j)$  where every member of  $\mathbf{x}_i$  is a universally quantified variable and each  $\mathbf{t}_j$ , for  $1 \leq j \leq k$ , is a sequence (tuple) of terms over  $\mathbf{x}_i$ . We then replace each such implication  $\chi$  in  $S$  with  $k$  implications  $\phi_{A,i}(\mathbf{x}_i) \Rightarrow r_1(\mathbf{t}_1), \dots, \phi_{A,i}(\mathbf{x}_i) \Rightarrow r_k(\mathbf{t}_k)$ .

2. (*Reinsert the hidden relations*)

In each implication  $\phi_{A,i}(\mathbf{x}_i) \Rightarrow r_k(\mathbf{t}_k)$  in  $S$  (obtained in the previous step), on the left-hand side of this implication replace each equation  $(f_r(\mathbf{y}_i) = \bar{1})$ , where  $\mathbf{y}_i \subseteq \mathbf{x}_i$  and  $f_r$  is the characteristic function of a hidden relational symbol  $r \notin \mathcal{A}$  (introduced by the algorithm *Compose*), by the atom  $r(\mathbf{y}_i)$ .

3. (*Transformation into operad's operations*)

Consider  $S = \{\chi_1, \dots, \chi_m\}$  with  $\chi_i$  being a view-based query mapping (implication from conjunctive query over  $\mathcal{A}$  into a relation of  $\mathcal{B}$ ),  $q_{Ai}(\mathbf{x}_i) \Rightarrow r'(\mathbf{t}_i)$ , where  $S_q = (r_{i_1,1}, \dots, r_{i_k,k})$  is the ordered list of relational symbols as they appear (from left to right) in the conjunctive query  $q_{Ai}(\mathbf{x}_i)$ . Then replace each such implication  $\chi_i$  in  $S$  with the operad's operations  $q_i \in O(r_{i_1,1}, \dots, r_{i_k,k}, r')$  where  $(r_{i_1,1}, \dots, r_{i_k,k}) = S_q$  and  $q_i$  is the expression obtained from  $q_{Ai}(\mathbf{x}_i)$  by replacing

each relational symbol  $r_{i_j,j}$  by the place symbol  $(\_ )_j$  (while replacing  $\neg r_{i_j,j}$  by the place symbol  $\neg(\_ )_j$ ) and by replacing  $r'$  by unlabeled  $(\_ )$ .

We will represent each operation  $q_i$  by a composition of two operations  $v_i \cdot q_{A,i}$ , where  $q_{A,i} \in O(r_{i_1,1}, \dots, r_{i_k,k}, r_q)$  is the same expression as  $q_i$  above where  $r_q$  is a relational symbol of the same type as  $r'$ , and  $v_i \in O(r_q, r')$  is the expression  $(\_ )_1(\mathbf{y}_i) \Rightarrow (\_ )(\mathbf{y}_i)$  where  $\mathbf{y}_i$  is the tuple of variables of the atoms  $r'(\mathbf{y}_i)$  and  $r_q(\mathbf{y}_i)$ .

In the simplest case when  $\chi_i$  is a tautology  $r(\mathbf{x}_i) \Rightarrow r(\mathbf{x}_i)$ ,  $\chi_i$  is replaced in  $S$  by  $q_i = q_{A,i} = v_i = 1_r \in O(r, r)$  (that is, by the identity mapping expression  $(\_ )_1(\mathbf{x}_i) \Rightarrow (\_ )(\mathbf{x}_i)$  if  $\mathbf{x}_i$  is not empty; and by the expression  $(\_ )_1 \Rightarrow (\_ )$  for the identity operation  $1_{r_\emptyset} \in O(r_\emptyset, r_\emptyset)$ , otherwise).

4. (*Construct operad's operations*)

Add the 'empty operation' operad  $1_{r_\emptyset} \in O(r_\emptyset, r_\emptyset)$  in  $S$ , represented by the mapping expression  $(\_ )_1 \Rightarrow (\_ )$ .

**Return** The set of abstract operad's operations denoted by an (mapping) arrow  $\mathbf{M}_{AB} = S : \mathcal{A} \rightarrow \mathcal{B}$ .

In what follows, we will use the term 'mapping-operad' from such a set of abstract operad operations. The reason why we insert the identity 'empty operad's operation'  $1_{r_\emptyset} \in O(r_\emptyset, r_\emptyset)$  in  $\mathbf{M}_{AB} = S$  is that, as we will see, we can have the Tarski's interpretations of database mappings where we do not transfer any information from the source to the target database (i.e., with the empty information flux of such a mapping). Consequently,  $1_{r_\emptyset}$  is always an element of the set of operad's operations obtained from a given SOTgd. As we will see in the algorithm of decomposition of SOTgds, we will have cases where the mappings are generated with SOTgd equal to the tautology  $r_\emptyset \Rightarrow r_\emptyset$ .

*Example 7* In the most trivial case when  $\mathcal{M}_{AB} = \{r_\emptyset \Rightarrow r_\emptyset\}$  (that is, when there is no effective mapping between the schemas  $\mathcal{A}$  and  $\mathcal{B}$ ) where  $r_\emptyset$  is a nullary predicate symbol (i.e., the *truth* propositional letter in FOL, Definition 1), so that SOTgd is a banal tautology  $r_\emptyset \Rightarrow r_\emptyset$  and, consequently,  $\mathcal{M}_{AB}$  is always satisfied, we obtain that  $\text{MakeOperads}(\mathcal{M}_{AB}) = \{1_{r_\emptyset}\}$ . The particular cases for such trivial mappings are the following:

$$\mathcal{M}_{AA_\emptyset} = \{r_\emptyset \Rightarrow r_\emptyset\} : \mathcal{A} \rightarrow \mathcal{A}_\emptyset,$$

$$\mathcal{M}_{A_\emptyset A} = \{r_\emptyset \Rightarrow r_\emptyset\} : \mathcal{A}_\emptyset \rightarrow \mathcal{A}, \text{ and}$$

$$Id_{A_\emptyset} = \{r_\emptyset \Rightarrow r_\emptyset\} : \mathcal{A}_\emptyset \rightarrow \mathcal{A}_\emptyset, \text{ the identity mapping,}$$

where  $\mathcal{A}_\emptyset = (\{r_\emptyset\}, \emptyset)$  is the empty database schema. It is easy to verify that

$$\text{MakeOperads}(\{r_\emptyset \Rightarrow r_\emptyset\}) = \{1_{r_\emptyset}\}.$$

Obviously, we cannot have the Tarski's interpretations for the empty schema  $\mathcal{A}_\emptyset$ , and we will need to define its interpretation in an ad hoc way in Sect. 2.4.1 dedicated to R-algebras for the operads.



There is an obvious inverse transformation of a given mapping-operad (a set of  $k$ -ary operations)  $\mathbf{M}_{AB} = \{q_1, \dots, q_n, 1_{r_\emptyset}\} : \mathcal{A} \rightarrow \mathcal{B}$ , into the SOTgd of this mapping.

**Inverse operads algorithm** *InverseOperads*( $\mathbf{M}_{AB}$ )

**Input.** A mapping-operad  $\mathbf{M}_{AB} = \{q_1, \dots, q_n, 1_{r_\emptyset}\} : \mathcal{A} \rightarrow \mathcal{B}$ .

**Output.** Mapping  $\mathcal{M}_{AB}$  from  $\mathcal{A}$  into  $\mathcal{B}$  represented by the SOTgd.

1. (*Transform operads into logical formulae*)

Initialize  $S$  to be the empty set  $\emptyset$ .

For each  $k$ -ary mapping operad's operation  $q_i \in O(r_{i,1}, \dots, r_{i,k}, r')$ , for  $1 \leq i \leq n$ , such that  $q_i \neq 1_{r_\emptyset}$ , add in  $S$  the logical sentence  $\forall \mathbf{x}_i \phi_i$  where  $\mathbf{x}_i$  is the tuple of all variables in the expression of the operad's operation  $q_i$  and  $\phi_i$  is the logical formula where each labeled place symbol  $(\_ )_m$ , for  $1 \leq m \leq k$ , is replaced by the relational symbol  $r_{i,k}$  and each unlabeled place symbol  $(\_ )$  is replaced by  $r' \in \mathcal{B}$ .

2. (*Elimination of relational symbols not in  $\mathcal{A}$* )

In each formula  $\forall \mathbf{x}_i \phi_i \in S$  where  $\phi$  is an implication  $\phi_{A,i}(\mathbf{x}_i) \Rightarrow r'(\mathbf{t}_i)$ , on the left-hand side of this implication replace each atom  $r(\mathbf{y}_i)$  with  $r \notin \mathcal{A}$ , where  $\mathbf{y}_i \subseteq \mathbf{x}_i$ , by the equation  $(f_r(\mathbf{y}_i) \doteq \bar{1})$ , where  $f_r$  is the characteristic function of a hidden relational symbol  $r \notin \mathcal{A}$ .

3. (*Construct SOTgd*)

The SOTgd is a formula  $\Phi$  equal to  $r_\emptyset \Rightarrow r_\emptyset$  if  $S$  is empty;  $\exists \mathbf{f}(\forall \mathbf{x}_1 \phi_1 \wedge \dots \wedge \forall \mathbf{x}_n \phi_n)$  otherwise, where  $\mathbf{f}$  is the tuple of all functional symbols in the formulae in  $S$ .

**Return** the logical mapping  $\mathcal{M}_{AB} = \{\Phi\} : \mathcal{A} \rightarrow \mathcal{B}$ .

Consequently, the algebraic formalism based on the mapping-operads is equivalent to the logical formalism based on SOTgds.

*Example 8* Let us consider the following example (corresponding to Example 4.3 in [5]) that explains in detail the transformation of the logic into R-algebras, that is, the transformation of logical formulae used to define a schema mapping into the set of abstract operad's operations and their R-algebras.

Schema  $\mathcal{A} = (S_A, \emptyset)$  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 a unary relation  $\text{Local}$  that represents employees that work in the local office of their company. Schema  $\mathcal{B} = (S_B, \emptyset)$  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 a unary relation  $\text{Over65}$  that is intended to represent people over age 65. Schema  $\mathcal{C} = (S_C, \emptyset)$  consists of a binary relation  $\text{Office}$  that associates employees with office numbers and a unary relation  $\text{CanRetire}$  that represents employees eligible for retirement. Consider now the following schema mappings:

$$\begin{aligned} \mathcal{M}_{AB} = \{ & \forall x_e (\text{EmpAcme}(x_e) \Rightarrow \text{Emp}(x_e)) \\ & \wedge \forall x_e (\text{EmpAjax}(x_e) \Rightarrow \text{Emp}(x_e)) \\ & \wedge \forall x_p (\text{Local}(x_p) \Rightarrow \text{Local1}(x_p)) \}, \end{aligned}$$

with  $\mathbf{M}_{AB} = \text{MakeOperads}(\mathcal{M}_{AB}) = \{q_1^A, q_2^A, q_3^A, 1_{r_\emptyset}\}$  where:

1. The operation  $q_1^A \in O(\text{EmpAcme}, \text{Emp})$  is the expression  $(\_)_1(x_e) \Rightarrow (\_)(x_e)$ ;
  2. The operation  $q_2^A \in O(\text{EmpAjax}, \text{Emp})$  is the expression  $(\_)_1(x_e) \Rightarrow (\_)(x_e)$ ;
  3. The operation  $q_3^A \in O(\text{Local}, \text{Local1})$  is the expression  $(\_)_1(x_p) \Rightarrow (\_)(x_p)$ ;
- and

$$\mathcal{M}_{BC} = \{\exists f_1 (\forall x_e ((\text{Emp}(x_e) \wedge \text{Local1}(x_e)) \Rightarrow \text{Office}(x_e, f_1(x_e)))) \\ \wedge \forall x_e ((\text{Emp}(x_e) \wedge \text{Over65}(x_e)) \Rightarrow \text{CanRetire}(x_e))\},$$

with  $\mathbf{M}_{BC} = \text{MakeOperads}(\mathcal{M}_{BC}) = \{q_1^B, q_2^B, 1_{r_\emptyset}\}$  where:

4. The operation  $q_1^B \in O(\text{Emp}, \text{Local1}, \text{Office})$  is the expression  $((\_)_1(x_e) \wedge (\_)_2(x_e)) \Rightarrow (\_)(x_e, f_1(x_e))$ ;
5. The operation  $q_2^B \in O(\text{Emp}, \text{Over65}, \text{CanRetire})$  is the expression  $((\_)_1(x_e) \wedge (\_)_2(x_e)) \Rightarrow (\_)(x_e)$ .

Then, by applying the new algorithm for composition, we obtain the composed mapping  $\mathcal{M}_{AC} = \text{Compose}(\mathcal{M}_{AB}, \mathcal{M}_{BC})$  equal to

$$\mathcal{M}_{AC} = \{\exists f_1 \exists f_2 \exists f_{\text{Over65}} ( \\ \forall x_e ((\text{EmpAcme}(x_e) \wedge \text{Local}(x_e)) \Rightarrow \text{Office}(x_e, f_1(x_e)) \\ \wedge \forall x_e ((\text{EmpAjax}(x_e) \wedge \text{Local}(x_e)) \Rightarrow \text{Office}(x_e, f_2(x_e))) \\ \wedge \forall x_e ((\text{EmpAcme}(x_e) \wedge (f_{\text{Over65}}(x_e) \doteq \bar{1})) \Rightarrow \text{CanRetire}(x_e)) \\ \wedge \forall x_e ((\text{EmpAjax}(x_e) \wedge (f_{\text{Over65}}(x_e) \doteq \bar{1})) \Rightarrow \text{CanRetire}(x_e))))\}.$$

Then, by a transformation into abstract operad's operations, we obtain

$$\mathbf{M}_{AC} = \text{MakeOperads}(\mathcal{M}_{AC}) = \{q_1, q_2, q_3, q_4, 1_{r_\emptyset}\}$$

where:

6. The operation  $q_1 \in O(\text{EmpAcme}, \text{Local}, \text{Office})$  is the expression

$$((\_)_1(x_e) \wedge (\_)_2(x_e)) \Rightarrow (\_)(x_e, f_1(x_e));$$

7. The operation  $q_2 \in O(\text{EmpAjax}, \text{Local}, \text{Office})$  is the expression

$$((\_)_1(x_e) \wedge (\_)_2(x_e)) \Rightarrow (\_)(x_e, f_2(x_e));$$

8. The operation  $q_3 \in O(\text{EmpAcme}, \text{Over65}, \text{CanRetire})$  is the expression

$$((\_)_1(x_e) \wedge (\_)_2(x_e)) \Rightarrow (\_)(x_e);$$

9. The operation  $q_4 \in O(\text{EmpAjax}, \text{Over65}, \text{CanRetire})$  is the expression

$$((\_)_1(x_e) \wedge (\_)_2(x_e)) \Rightarrow (\_)(x_e).$$

Note that in operad's operations  $q_3$  and  $q_4$ , we have the relational symbol *Over65* that is not in the source schema  $\mathcal{A}$  (it was a hidden relation in the *SOTgd*). Consequently, the operad's operations generally can have the relational symbols of the schemas that are different from the source and target schemas. Let us consider the case when no relational symbols of the source schema are used on the left-hand side of the *tgd*'s implications:

*Example 9* Let us consider the previous Example 8 where the mapping  $\mathcal{M}_{AB}$  is reduced to  $\mathcal{M}_{AB} = \{\forall x_p (\text{Local}(x_p) \Rightarrow \text{Local1}(x_p))\}$  with

$$\mathbf{M}_{AB} = \text{MakeOperads}(\mathcal{M}_{AB}) = \{q_1^A, 1_{r_\emptyset}\},$$

where the operation  $q_1^A \in O(\text{Local}, \text{Local1})$  is the expression  $(\_ )_1(x_e) \Rightarrow (\_ )_1(x_e)$ .

Then, by applying the new algorithm for composition, we obtain the composed mapping  $\mathcal{M}_{AC} = \text{Compose}(\mathcal{M}_{AB}, \mathcal{M}_{BC})$  equal to

$$\begin{aligned} \mathcal{M}_{AC} = \{ & \exists f_1 \exists f_{\text{Emp}} \exists f_{\text{Over65}} ( \\ & \forall x_e (((f_{\text{Emp}}(x_e) \doteq \bar{1}) \wedge \text{Local}(x_e)) \Rightarrow \text{Office}(x_e, f_1(x_e)) \\ & \wedge \forall x_e (((f_{\text{Emp}}(x_e) \doteq \bar{1}) \wedge (f_{\text{Over65}}(x_e) \doteq \bar{1})) \Rightarrow \text{CanRetire}(x_e))) \}. \end{aligned}$$

Note that in the second implication of the composed *SOTgd*, on the left-hand side of this implication, we have no any relational symbol from  $\mathcal{A}$ , so that in step 2 of the algorithm *MakeOperad* we will introduce the relational symbols *Emp* and *Over65* in this implication.

Then, by transformation into operads, we obtain

$$\mathbf{M}_{AC} = \text{MakeOperads}(\mathcal{M}_{AC}) = \{q_1, q_2, 1_{r_\emptyset}\}$$

where:

1. The operation  $q_1 \in O(\text{Emp}, \text{Local}, \text{Office})$  is the expression

$$((\_)_1(x_e) \wedge (\_)_2(x_e)) \Rightarrow (\_)(x_e, f_1(x_e));$$

2. The operation  $q_2 \in O(\text{Emp}, \text{Over65}, \text{CanRetire})$  is the expression

$$((\_)_1(x_e) \wedge (\_)_2(x_e)) \Rightarrow (\_)(x_e).$$

Note that the operation  $q_2$  means that we do not transfer any tuple from  $\mathcal{A}$  into  $\mathcal{C}$  (but only the tuples from  $\mathcal{B}$  into  $\mathcal{C}$ ). Consequently, the information flux transmitted from  $\mathcal{A}$  into  $\mathcal{C}$  by this operation has to be empty, as we will show in the next example.

Based on the equivalence of representation of schema mappings by *SOTgd* and by mapping-operads, we can introduce the composition of mapping operads analogously to the composition of schema mappings:

**Definition 9** For any given schema mapping  $\mathcal{M}_{AB} : \mathcal{A} \rightarrow \mathcal{B}$  and an *atomic* schema mapping  $\mathcal{M}_{BC} : \mathcal{B} \rightarrow \mathcal{C}$ , we can define the corresponding mapping-operads

$$\mathbf{M}_{AB} = \text{MakeOperads}(\mathcal{M}_{AB}) = \{q_1^A, \dots, q_n^A, 1_{r_\emptyset}\} : \mathcal{A} \rightarrow \mathcal{B},$$

$$\mathbf{M}_{BC} = \text{MakeOperads}(\mathcal{M}_{BC}) = \{q_1^B, \dots, q_m^B, 1_{r_\emptyset}\} : \mathcal{B} \rightarrow \mathcal{C},$$

and their composition

$$\mathbf{M}_{BC} \circ \mathbf{M}_{AB}$$

$$\begin{aligned} &\triangleq \{1_{r_\emptyset}\} \bigcup_{1 \leq i \leq m} \{q_i = q_i^B \cdot (q_{i1}^A, \dots, q_{in}^A) \mid q_i^B \in O(r_{B,i1}, \dots, r_{B,ij}, r_{C,i}) \in \mathbf{M}_{BC} \\ &\text{and } (q_{ik}^A \in O(r_{k1}, \dots, r_{kl}, r_{B,ik}) \text{ if } q_{ik}^A \in \mathbf{M}_{AB}; q_{ik}^A = 1_{r_{B,ik}} \text{ otherwise})_{1 \leq k \leq j}\}. \end{aligned}$$

Note that each abstract operation  $q_i$  in the composed operad mapping  $\mathbf{M}_{AC} = \mathbf{M}_{BC} \circ \mathbf{M}_{AB}$  is represented by the operation composition  $q_i^B \cdot (q_1^A, \dots, q_n^A)$  where  $q_i^B \in \mathbf{M}_{BC}$  and each  $q_j^A$ , for  $1 \leq j \leq n$ , is an operation in  $\mathbf{M}_{AB}$  or an identity operation for relations in  $\mathcal{B}$ . Let us show that the transformation of the SOTgd of a given mapping into operads is well defined and that the properties of the mapping-operads obtained by the algorithm *MakeOperad* satisfy the general properties of operads in Definition 8.

**Proposition 1** *The transformation by the algorithm MakeOperads of SOTgds of the schema mappings into the mapping-operads is well defined and satisfies the general properties of operads in Definition 8.*

*Proof* Let us show the following mapping-operads properties required by Definition 8:

1. There exist the identity mappings that are transformed into the identity operad's operations for each relational symbol (as required by point 3 in Definition 8).

In fact, for any relational symbol  $r$  we can define a database schema  $\mathcal{A} = (\{r\}, \emptyset)$  with only this relation and its identity mapping  $Id_A : \mathcal{A} \rightarrow \mathcal{A}$ , where  $Id = \forall \mathbf{x}(r(\mathbf{x}) \Rightarrow r(\mathbf{x}))$ . Then, the identity operad's operation  $1_r \in O(r, r)$  is defined by  $\text{MakeOperad}(Id_A) = \{(\_ )_1 \Rightarrow (\_ ), 1_{r_\emptyset}\} = \{1_r, 1_{r_\emptyset}\}$ .

2. Let us define the composition of mapping-operads ' $\cdot$ ' so that it satisfies the properties in point 2 of Definition 8, namely for any  $f \in O(r_1, \dots, r_k, r)$  and any  $g_1 \in O(r_{11}, \dots, r_{1i_1}, r_1), \dots, g_k \in O(r_{k1}, \dots, r_{ki_k}, r_k)$ , an element  $f \cdot (g_1, \dots, g_k) \in O(r_{11}, \dots, r_{1i_1}, \dots, r_{k1}, \dots, r_{ki_k}, r)$ .

Define the database schemas  $\mathcal{A} = (S_A, \emptyset)$  with  $S_A = \bigcup_{1 \leq j \leq k} \{r_{j1}, \dots, r_{ji_j}\}$ ,  $\mathcal{B} = (S_B, \emptyset)$  with  $S_B = \{r_1, \dots, r_k\}$ , and  $\mathcal{C} = (\{r\}, \emptyset)$ , with the mappings  $\mathbf{M}_{BC} = \{f, 1_{r_\emptyset}\} : \mathcal{B} \rightarrow \mathcal{C}$  and  $\mathbf{M}_{AB} = \{g_1, \dots, g_k, 1_{r_\emptyset}\} : \mathcal{A} \rightarrow \mathcal{B}$ .

Hence, the operad's operation composition ' $\cdot$ ', based on Definition 9, is defined by  $\{f \cdot (g_1, \dots, g_k), 1_{r_\emptyset}\} = \mathbf{M}_{BC} \circ \mathbf{M}_{AB}$ .

3. The permutation in a given operad's operation  $f \in O(r_1, \dots, r_k, r)$  of its relational symbols in  $\{r_1, \dots, r_k\}$  is possible on the left-hand side of implication of the expression  $e \Rightarrow (\_)$  because the atoms of these relational symbols are connected by the logical conjunction  $\wedge$  which is a commutative operation. Thus, all properties for the permutations in point 4 of Definition 8 are satisfied for the mapping-operads as well.  $\square$

**Corollary 3** *The database schemas and the composition of their operad-mappings can be represented by a sketch category.*

*Proof* It is a direct result of Corollary 2, the equality of these two representations, and from Proposition 1. From the fact that for each schema  $\mathcal{A} = (S_A, \Sigma_A)$  there is its identity mapping-operad  $\mathbf{Id}_A : \mathcal{A} \rightarrow \mathcal{A}$  such that  $\mathbf{Id}_A = \{1_r | r \in S_A\} \cup \{1_{r_\emptyset}\}$ , and that the composition of mapping-operads is associative, we conclude that the set of schemas (the objects) and the set of mapping-operads (the morphisms) can define a sketch category. The associativity of composition is a consequence of the associativity of the operads-composition ‘ $\cdot$ ’ defined in point 4(a) of Definition 8. Thus,  $\mathbf{M}_{CD} \circ (\mathbf{M}_{BC} \circ \mathbf{M}_{AB}) = (\mathbf{M}_{CD} \circ \mathbf{M}_{BC}) \circ \mathbf{M}_{AB}$ .  $\square$

### 2.4.1 R-Algebras, Tarski's Interpretations and Instance-Database Mappings

The theory of typed operads represents the syntax of the database mapping, translated from the schema-database logic into the algebraic framework. The semantic part of the operad's algebra theory, corresponding to the semantics of FOL based on Tarski's interpretations, is represented by the R-algebras which are particular interpretations of the operads.

Let us now define the R-algebra of a database mapping-operad based on homotopy theory [1, 3, 7], where its abstract operations are represented by actual functions:

**Definition 10** For a given universe of values  $\mathcal{U}$  and R-operad  $O$ , an R-algebra  $\alpha$  consists of (here ‘ $\setminus$ ’ denotes the set-difference):

1. A set  $\alpha(r)$  for any  $r \in \mathbb{R}$ , which is a set of tuples (relation), with the empty relation  $\alpha(r_\emptyset) = \perp = \{\langle \rangle\}$ , unary universe-relation  $\alpha(r_\infty) = \{\langle d \rangle \mid d \in \mathcal{U}\}$ , and binary relation  $\alpha(r_\top) = R_\equiv$  for the “equality” type  $r_\top$ . The  $\alpha^*$  is the extension of  $\alpha$  to a list of symbols  $\alpha^*({r_1, \dots, r_k}) \triangleq \{\alpha(r_1), \dots, \alpha(r_k), \alpha(r_\emptyset)\}$ .
2. A mapping function  $\alpha(q_i) : R_1 \times \dots \times R_k \longrightarrow \alpha(r)$  for any  $q_i \in O(r_1, \dots, r_k, r)$ , where for each  $1 \leq i \leq k$ ,  $R_i = \mathcal{U}^{ar(r_i) \setminus \alpha(r_i)}$  if the place symbol  $(\_)_i \in q_i$  is preceded by negation operator  $\neg$ ;  $\alpha(r_i)$  otherwise.

Consequently,  $\alpha^*({q_1, \dots, q_k}) \triangleq \{\alpha(q_1), \dots, \alpha(q_k)\}$ , and

- (a) For any  $r \in \mathbb{R}$ ,  $\alpha(1_r)$  acts as an identity function on relation  $\alpha(r)$ . The  $q_\perp = \alpha(1_{r_\emptyset}) : \perp \rightarrow \perp$  is the empty function (with  $q_\perp(\langle \rangle) = \langle \rangle$  for the empty tuple  $\langle \rangle$ ) with the empty graph).

(b) For the associative composition ‘ $\cdot$ ’,

$$\alpha(q \cdot (q_1, \dots, q_k)) = \alpha(q)(\alpha(q_1) \times \dots \times \alpha(q_k))$$

(c) For any  $q \in O(r_1, \dots, r_k, r)$  and a permutation  $\sigma \in \mathbb{R}_k$ ,  $\alpha(q\sigma) = \alpha(q)\sigma$ , where  $\sigma$  acts on the function  $\alpha(q)$  on the right by permuting its arguments.

3. We introduce two functions  $\partial_0$  and  $\partial_1$  such that for any  $\alpha(q)$ ,  $q \in O(r_1, \dots, r_k, r)$ ,  $\partial_0(q) = \{r_1, \dots, r_k\}$ ,  $\partial_0(\alpha(q)) = \alpha^*(\partial_0(q)) = \{\alpha(r_1), \dots, \alpha(r_k)\}$ ,  $\partial_1(q) = \{r\}$ , and  $\partial_1(\alpha(q)) = \alpha^*(\partial_1(q)) = \{\alpha(r)\}$ .

*Remark* In what follows, an instance database  $A = \alpha^*(S_A)$  of a schema  $\mathcal{A} = (S_A, \Sigma_A)$  will be denoted by  $\alpha^*(\mathcal{A})$  as well. The empty schema is denoted by  $\mathcal{A}_\emptyset = (\{r_\emptyset\}, \emptyset)$  so that for any  $\alpha$ , from point 1 of this definition,  $\perp^0 = \alpha^*(\mathcal{A}_\emptyset) = \alpha^*(S_{\mathcal{A}_\emptyset}) = \{\alpha(r_\emptyset)\} = \{\perp\}$  is the empty instance-database.

Note that this empty database is the zero object of the **DB** category defined in the next chapter, and this is the reason that in the computation of  $\alpha^*$  in point 1 of this definition we added  $\alpha(r_\emptyset)$  as well, so that we will have for each schema  $\mathcal{A} = (S_A, \Sigma_A)$  that  $\alpha(r_\emptyset) = \perp \in \alpha^*(S_A) = \alpha^*(\mathcal{A})$ .

Consequently, we can think of an operad as a simple sort of theory, used to define a schema mapping between databases, and its R-algebras as models of this theory used to define the mappings between instance-databases, where an R-algebra  $\alpha$  is considered as an interpretation of relational symbols of a given database schema. For the empty operation  $1_{r_\emptyset} \in \mathcal{M}_{AB}$  of a mapping  $\mathcal{M}_{AB}$ , we have, by Definition 10, that its interpretation is prefixed by the empty function  $q_\perp : \perp \rightarrow \perp$ , while for the rest of operad's operations in  $\mathcal{M}_{AB}$  we have to define their formal semantics. What we need is to specify the subsets of R-algebras that contain well defined *mapping-interpretations* for the operad's operations obtained from schema mappings by the *MakeOperads* algorithm. They are particular extensions of Tarski's interpretations of the database mappings (when we eliminate the existential quantifiers over functions from SOTgd's of a given schema database mapping system then we obtain the FOL with its Tarski's interpretations  $I_T$ , and their extensions  $I_T^*$  to all formulae introduced in Sect. 1.3) as follows:

**Definition 11** Let  $\phi_{Ai}(\mathbf{x}) \Rightarrow r_B(\mathbf{t})$  be an implication  $\chi$  in a normalized SOTgd  $\exists \mathbf{f}(\Psi)$  (where  $\Psi$  is an FOL formula) of the mapping  $\mathcal{M}_{AB}$ , let  $\mathbf{t}$  be a tuple of terms with variables in  $\mathbf{x} = \langle x_1, \dots, x_m \rangle$ , and  $q_i \in \text{MakeOperads}(\mathcal{M}_{AB})$  be the operad's operation of this implication obtained by *MakeOperads* algorithm, equal to the expression  $(e \Rightarrow (\_))(\mathbf{t}) \in O(r_1, \dots, r_k, r_B)$ , where  $q_i = v_i \cdot q_{A,i}$  with  $q_{A,i} \in O(r_1, \dots, r_k, r_q)$  and  $v_i \in O(r_q, r_B)$  such that for a new relational symbol  $r_q$ ,  $ar(r_q) = ar(r_B) \geq 1$ .

Let  $S$  be an empty set and  $e[(\_)_n/r_n]_{1 \leq n \leq k}$  be the formula obtained from expression  $e$  where each place-symbol  $(\_)_n$  is substituted by the relational symbol  $r_n$  for  $1 \leq n \leq k$ . Then do the following as far as possible: For each two relational symbols  $r_j, r_n$  in the formula  $e[(\_)_n/r_n]_{1 \leq n \leq k}$  such that the  $j_h$ th free variable (which

is not an argument of a functional symbol) in the atom  $r_j(\mathbf{t}_j)$  is equal to the  $n_h$ th free variable in the atom  $r_n(\mathbf{t}_n)$  (both atoms in  $e[(\_)_n/r_n]_{1 \leq n \leq k}$ ), we insert the set  $\{(j_h, j), (n_h, n)\}$  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  $\alpha$  is a *mapping-interpretation* of  $\mathcal{M}_{AB} : \mathcal{A} \rightarrow \mathcal{B}$  if it is an extension of a Tarski's interpretation  $I_T$  (in Definition 1, of all predicate and functional symbols in FOL formula  $\Psi$ , with  $I_T^*$  being its extension to all formulae), and if for each  $q_i \in \text{MakeOperads}(\mathcal{M}_{AB})$  it satisfies the following:

1. For each relational symbol  $r_i \neq r_\emptyset$  in  $\mathcal{A}$  or  $\mathcal{B}$ ,  $\alpha(r_i) = I_T(r_i)$ .
2. We obtain a function  $f = \alpha(q_{A,i}) : R_1 \times \cdots \times R_k \rightarrow \alpha(r_q)$ , where for each  $1 \leq i \leq k$ ,  $R_i = \mathcal{U}^{ar(r_i)} \setminus \alpha(r_i)$  if the place symbol  $(\_)_i \in q_i$  is preceded by negation operator  $\neg$ ;  $\alpha(r_i)$  otherwise, such that for every  $\mathbf{d}_i \in R_i$ :

$$f(\langle \mathbf{d}_1, \dots, \mathbf{d}_k \rangle) = g^*(\mathbf{t}) = \langle g^*(t_1), \dots, g^*(t_{ar(r_B)}) \rangle$$

if  $\bigwedge \{\pi_{j_h}(\mathbf{d}_j) = \pi_{n_h}(\mathbf{d}_n) \mid \{(j_h, j), (n_h, n)\} \in S\}$  is true and the assignment  $g$  satisfies the formula  $e[(\_)_n/r_n]_{1 \leq n \leq k}$ ;  $\langle \rangle$  (empty tuple) otherwise, where the assignment  $g : \{x_1, \dots, x_m\} \rightarrow \mathcal{U}$  is defined by the tuple of values  $\langle g(x_1), \dots, g(x_m) \rangle = \text{Cmp}(S, \langle \mathbf{d}_1, \dots, \mathbf{d}_k \rangle)$ , and its extension  $g^*$  to all terms is such that for any term  $f_i(t_1, \dots, t_n)$ :

$$g^*(f_i(t_1, \dots, t_n)) = I_T(f_i)(g^*(t_1), \dots, g^*(t_n)) \text{ if } n \geq 1; \quad I_T(f_i) \text{ otherwise.}$$

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  $\langle \mathbf{d}_1, \dots, \mathbf{d}_k \rangle$ .

Initialize  $\mathbf{d}$  to  $\mathbf{d}_1$ . Repeat consecutively the following, for  $j = 2, \dots, k$ :

Let  $\mathbf{d}_j$  by a tuple of values  $\langle v_1, \dots, v_{j_n} \rangle$ , then for  $i = 1, \dots, j_n$  repeat consecutively the following:

$\mathbf{d} = \mathbf{d} \& v_i$  if there does not exist an element  $\{(j_h, j), (n_h, n)\}$  in  $S$  such that  $j \leq n$ ;  $\mathbf{d}$ , otherwise.

(The operation of concatenation ' $\&$ ' appends the value  $v_i$  at the end of tuple  $\mathbf{d}$ )

**Output.** The tuple  $\text{Cmp}(S, \langle \mathbf{d}_1, \dots, \mathbf{d}_k \rangle) = \mathbf{d}$ .

3.  $\alpha(r_q)$  is equal to the image of the function  $f$  in point 2 above.
4. The function  $h = \alpha(v_i) : \alpha(r_q) \rightarrow \alpha(r_B)$  such that for each  $\mathbf{b} \in \alpha(r_q)$ ,  $h(\mathbf{b}) = \mathbf{b}$  if  $\mathbf{b} \in \alpha(r_B)$ ; empty tuple  $\langle \rangle$  otherwise.

Note that the formulae  $\phi_{Ai}(\mathbf{x})$  and expression  $e[(\_)_n/r_n]_{1 \leq n \leq k}$  are logically equivalent, with the only difference that the atoms with characteristic functions  $f_r(\mathbf{t}) \doteq \bar{1}$  in the first formula are substituted by the atoms  $r(\mathbf{t})$ , based on the fact that the assignment  $g$  satisfies  $r(\mathbf{t})$  iff  $g^*(f_r(\mathbf{t})) = \bar{f}_r(g^*(\mathbf{t})) = 1$  (and for every assignment  $g(\bar{1}) = 1$ ), where  $\bar{f}_r : \mathcal{U}^{ar(r)} \rightarrow \{0, 1\}$  is the characteristic function of a relation  $\alpha(r)$  such that for each tuple  $\mathbf{c} \in \mathcal{U}^{ar(r)}$ ,  $\bar{f}_r(\mathbf{c}) = 1$  if  $\mathbf{c} \in \alpha(r)$ ; 0 otherwise.

*Example 10* Let us show how we construct the set  $S$  and the compacting of tuples given by Definition 11 above:

Let us consider an operad  $q_i \in \text{MakeOperads}(\mathcal{M}_{AB})$ , obtained from a normalized implication  $\phi_{Ai}(\mathbf{x}) \Rightarrow r_B(\mathbf{t})$  in  $\mathcal{M}_{AB}$ ,  $((y \doteq f_1(x, z)) \wedge r_1(x, y, z) \wedge r_2(v, x, w) \wedge (f_{r_3}(y, z, w', w) \doteq \bar{1})) \Rightarrow r_B(x, z, w, f_2(v, z))$ , so that  $q_i$  is equal to the expression  $(e \Rightarrow (\_)(\mathbf{t})) \in \mathcal{O}(r_1, r_2, r_3, r_B)$ , where  $\mathbf{x} = \langle x, y, z, v, w, w' \rangle$  (the ordering of variables in the atoms (with database relational symbols) from left to right),  $\mathbf{t} = \langle x, z, w, f_2(v, z) \rangle$ , and the expression  $e$  equal to  $(y \doteq f_1(x, z)) \wedge (\_)_1(\mathbf{t}_1) \wedge (\_)_2(\mathbf{t}_2) \wedge (\_)_3(\mathbf{t}_3)$ , with  $\mathbf{t}_1 = \langle x, y, z \rangle$ ,  $\mathbf{t}_2 = \langle v, x, w \rangle$  and  $\mathbf{t}_3 = \langle y, z, w', w \rangle$ .

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  $\mathbf{d}_1 = \langle a_1, a_2, a_3 \rangle \in \alpha(r_1)$ ,  $\mathbf{d}_2 = \langle b_1, b_2, b_3 \rangle \in \alpha(r_2)$  and  $\mathbf{d}_3 = \langle c_1, c_2, c_3, c_4 \rangle \in \alpha(r_3)$ , the statement

$$\bigwedge \{ \pi_{j_h}(\mathbf{d}_j) = \pi_{n_h}(\mathbf{d}_n) \mid \{(j_h, j), (n_h, n)\} \in S \}$$

is equal to

$$(\pi_1(\mathbf{d}_1) = \pi_2(\mathbf{d}_2)) \wedge (\pi_2(\mathbf{d}_1) = \pi_1(\mathbf{d}_3)) \wedge (\pi_3(\mathbf{d}_1) = \pi_2(\mathbf{d}_3)) \wedge (\pi_3(\mathbf{d}_2) = \pi_4(\mathbf{d}_3)),$$

which is true when  $a_1 = b_2$ ,  $a_2 = c_1$ ,  $a_3 = c_2$ , and  $b_3 = c_4$ .

The compacting of these tuples is equal to

$$\mathbf{d} = \text{Cmp}(S, \langle \mathbf{d}_1, \mathbf{d}_2, \mathbf{d}_3 \rangle) = \langle a_1, a_2, a_3, b_1, b_3, c_3 \rangle,$$

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,  $\mathbf{d} = \mathbf{x}[x/a_1, y/a_2, z/a_3, v/b_1, w/b_3, w'/c_3]$  is obtained by this assignment  $g$  to the tuple of variables  $\mathbf{x}$ , so that the sentence  $e[(\_)_n/r_n]_{1 \leq n \leq k}/g$  is well defined and equal to

$$(a_2 = I_T(f_1)(a_1, a_3)) \wedge r_1(a_1, a_2, a_3) \wedge r_2(b_1, a_1, b_3) \wedge r_3(a_2, a_3, c_3, b_3),$$

that is, to

$$(a_2 = I_T(f_1)(a_1, a_3)) \wedge r_1(\mathbf{d}_1) \wedge r_2(\mathbf{d}_2) \wedge r_3(\mathbf{d}_3),$$

and if this formula is satisfied by such an assignment  $g$ , i.e.,  $I_T^*(e[(\_)_n/r_n]_{1 \leq n \leq k}/g) = 1$ , then

$$\begin{aligned} f(\langle \mathbf{d}_1, \mathbf{d}_2, \mathbf{d}_3 \rangle) &= g^*(\mathbf{t}) = \langle g(x), g(z), g(w), g^*(f_2(v, z)) \rangle \\ &= \langle a_1, a_3, b_3, I_T(f_2)(b_1, a_3) \rangle, \end{aligned}$$



for a given Tarski's interpretation  $I_T$ , where  $I_T^*$  is the extension of  $I_T$  to all FOL formulae, as defined in Sect. 1.3.

If  $\mathcal{M}_{AB}$  is satisfied by the mapping-interpretation  $\alpha$ , this value of  $f(\langle \mathbf{d}_1, \mathbf{d}_2, \mathbf{d}_3 \rangle)$  corresponds to the truth of the normalized implication in the SOTgd of  $\mathcal{M}_{AB}$ ,  $\phi_{Ai}(\mathbf{x}) \Rightarrow r_B(\mathbf{t})$  for the assignment  $g$  derived by substitution  $[\mathbf{x}/\mathbf{d}]$ , when  $\phi_{Ai}(\mathbf{x})/g$  is true. Hence,  $r_B(\mathbf{t})/g$  is equal to  $r_B(\langle a_1, a_3, b_3, I_T(f_2)(b_1, a_3) \rangle)$ , i.e., to  $r_B(f(\langle \mathbf{d}_1, \mathbf{d}_2, \mathbf{d}_3 \rangle))$  and has to be true as well (i.e.,  $I_T^*(r_B(f(\langle \mathbf{d}_1, \mathbf{d}_2, \mathbf{d}_3 \rangle))) = 1$  or, equivalently,  $f(\langle \mathbf{d}_1, \mathbf{d}_2, \mathbf{d}_3 \rangle) \in \alpha(r_B) = I_T(r_B)$ ).

Consequently, if  $\mathcal{M}_{AB}$  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  $f(\langle \mathbf{d}_1, \mathbf{d}_2, \mathbf{d}_3 \rangle) \in \|\alpha(r_B)\|_{\alpha^*(\mathcal{B})}$ , so that the function  $f = \alpha(q_{A,i})$  represents the transferring of the tuples in relations of the source instance databases into the target instance database  $B = \alpha^*(\mathcal{B})$ , according to the SOTgd  $\Phi$  of the mapping  $\mathcal{M}_{AB} = \{\Phi\} : \mathcal{A} \rightarrow \mathcal{B}$ .

In this way, for a given R-algebra  $\alpha$  which satisfies the conditions for the mapping-interpretations in Definition 11, 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.

It is easy to verify that for a *query mapping*  $\phi_{Ai}(\mathbf{x}) \Rightarrow r_B(\mathbf{t})$ , 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  $\mathcal{B}$ .

When  $\alpha$  satisfies this query mapping  $\phi_{Ai}(\mathbf{x}) \Rightarrow r_B(\mathbf{t})$ , then  $\alpha(r_q) \subseteq \alpha(r_B)$  (with proof in Proposition 4) and, consequently, the function  $\alpha(v_i)$  is an injection, i.e., the *inclusion* of  $\alpha(r_q)$  into  $\alpha(r_B)$ .

**Proposition 2** *Let  $\phi_{Ai}(\mathbf{x}) \Rightarrow r_B(\mathbf{t})$  be an implication in the normalized SOTgd  $\exists f\psi$  of a mapping  $\mathcal{M}_{AB} : \mathcal{A} \rightarrow \mathcal{B}$ , where  $\mathbf{t}$  is a tuple of terms with variables in  $\mathbf{x} = \langle x_1, \dots, x_m \rangle$ . Let  $q_i \in \text{MakeOperads}(\mathcal{M}_{AB})$  be the operad's operation of this implication obtained by MakeOperads algorithm, equal to the expression  $(e \Rightarrow \_)(\mathbf{t}) \in O(r_1, \dots, r_k, r_B)$ , and let  $S$  be the set of sets that contain the pairs of mutually equal (joined) free variables in  $q_i$  as specified in Definition 11.*

*Then, for a given Tarski's interpretation of all FOL formulae in  $\psi$  and, extended from it, a mapping-interpretation  $\alpha$  in Definition 11 (such that for each  $1 \leq i \leq k$ ,  $R_i = \mathcal{U}^{ar(r_i)} \setminus I_T(r_i)$  if the place symbol  $\_ )_i \in q_i$  is preceded by negation operator  $\neg$ ;  $I_T(r_i)$  otherwise), the following is true:*

*If for every tuple  $\langle \mathbf{d}_1, \dots, \mathbf{d}_k \rangle \in R_1 \times \dots \times R_k$  such that  $\bigwedge \{\pi_{j_h}(\mathbf{d}_j) = \pi_{n_h}(\mathbf{d}_n) \mid \{(j_h, j), (n_h, n)\} \in S\}$  is true, we have that  $\alpha(q_i)(\langle \mathbf{d}_1, \dots, \mathbf{d}_k \rangle) \in I_T(r_B)$ , then  $I_T^*(\forall \mathbf{x}(\phi_{Ai}(\mathbf{x}) \Rightarrow r_B(\mathbf{t}))) = 1$ , and vice versa.*

*Proof* We have to show that for every  $\langle \mathbf{d}_1, \dots, \mathbf{d}_k \rangle \in R_1 \times \dots \times R_k$  such that  $\bigwedge \{\pi_{j_h}(\mathbf{d}_j) = \pi_{n_h}(\mathbf{d}_n) \mid \{(j_h, j), (n_h, n)\} \in S\}$  is true, with  $\mathbf{d} = \text{Cmp}(S, \langle \mathbf{d}_1, \dots, \mathbf{d}_k \rangle)$  and the assignment  $g$  derived from the substitution  $[\mathbf{x}/\mathbf{d}]$ ,  $I_T^*(\phi_{Ai}(\mathbf{x})/g \Rightarrow r_B(\mathbf{t})/g) = 1$ .

If  $I_T^*(\phi_{Ai}(\mathbf{x})/g) = 0$  then it is satisfied. Thus, we have to consider only the following cases:

- (a)  $I_T^*(\phi_{Ai}(\mathbf{x})/g) = 1$ , that is, when  $I_T^*(e[(\_)_n/r_n]_{1 \leq n \leq k}/g) = 1$  (see the comments after Definition 11).

From (a), the statement  $\bigwedge \{\pi_{j_h}(\mathbf{d}_j) = \pi_{n_h}(\mathbf{d}_n) \mid \{(j_h, j), (n_h, n)\} \in S\}$  is true and, by Definition 11,  $g^*(\mathbf{t}) = \alpha(q_i)(\langle \mathbf{d}_1, \dots, \mathbf{d}_k \rangle) \in I_T(r_B)$  (from assumption of this proposition), that is, equivalently;

- (b)  $I_T^*(r_B(g^*(\mathbf{t}))) = I_T^*(r_B(\mathbf{t})/g) = 1$ .

Thus, from (a) and (b), we obtain  $I_T^*((\phi_{Ai}(\mathbf{x}) \Rightarrow r_B(\mathbf{t}))/g) = 1$ , and hence from the fact that it holds for every  $\langle \mathbf{d}_1, \dots, \mathbf{d}_k \rangle \in R_1 \times \dots \times R_k$ , we obtain by generalization that  $I_T^*(\forall \mathbf{x}(\phi_{Ai}(\mathbf{x}) \Rightarrow r_B(\mathbf{t}))) = 1$ .

Vice versa, if  $I_T^*(\forall \mathbf{x}(\phi_{Ai}(\mathbf{x}) \Rightarrow r_B(\mathbf{t}))) = 1$ , then for each  $\langle \mathbf{d}_1, \dots, \mathbf{d}_k \rangle \in R_1 \times \dots \times R_k$  such that  $\bigwedge \{\pi_{j_h}(\mathbf{d}_j) = \pi_{n_h}(\mathbf{d}_n) \mid \{(j_h, j), (n_h, n)\} \in S\}$  is true, with  $\mathbf{d} = \text{Cmp}(S, \langle \mathbf{d}_1, \dots, \mathbf{d}_k \rangle)$  and the assignment  $g$  derived from the substitution  $[\mathbf{x}/\mathbf{d}]$ ,  $I_T^*((\phi_{Ai}(\mathbf{x}) \Rightarrow r_B(\mathbf{t}))/g) = 1$ . Hence, if  $I_T^*(\phi_{Ai}(\mathbf{x})/g) = 1$ , that is, if  $I_T^*(e[(\_)_n/r_n]_{1 \leq n \leq k}/g) = 1$ , then  $I_T^*(r_B(\mathbf{t})/g) = I_T^*(r_B(g^*(\mathbf{t}))) = 1$ , i.e.,  $g^*(\mathbf{t}) \in I_T(r_B)$ . Moreover, from Definition 11,  $g^*(\mathbf{t}) = \alpha(q_i)(\langle \mathbf{d}_1, \dots, \mathbf{d}_k \rangle)$  and hence  $\alpha(q_i)(\langle \mathbf{d}_1, \dots, \mathbf{d}_k \rangle) \in I_T(r_B)$ .  $\square$

Let us explain the mapping-interpretations by the following example:

*Example 11* Let us consider Example 5.2 in [5] for the composition of SOTgds  $\mathcal{M}_{AD} = \mathcal{M}_{BD} \circ \mathcal{M}_{AB}$ , where  $\mathcal{A} = (S_A, \emptyset)$  and  $S_A = \{\text{Emp}(x_e)\}$  with a single unary relational symbol of employees,  $\mathcal{B} = (S_B, \emptyset)$  and  $S_B = \{\text{Emp1}(x_e), \text{Mgr}(x_e, x_m)\}$  with  $\text{Emp1}$  intended as a copy of  $\text{Emp}$  and  $\text{Mgr}$  a binary relational symbol that associates each employee with a manager, and  $\mathcal{D} = (S_D, \emptyset)$  and  $S_D = \{\text{SelfMgr}(x_e)\}$  with a single unary relational symbol that is intended to store employees who are their own manager, and with mappings

$$\mathcal{M}_{AB} = \{\forall x_e(\text{Emp}(x_e) \Rightarrow \text{Emp1}(x_e)), \forall x_e(\text{Emp}(x_e) \Rightarrow \exists x_m \text{Mgr}(x_e, x_m))\},$$

or equivalently, by the following SOTgd:

$$\begin{aligned} \mathcal{M}_{AB} &= \{\exists f_1(\forall x_e(\text{Emp}(x_e) \Rightarrow \text{Emp1}(x_e)) \wedge \forall x_e(\text{Emp}(x_e) \Rightarrow \text{Mgr}(x_e, f_1(x_e))))\}, \\ \mathcal{M}_{BD} &= \{\forall x_e(\text{Mgr}(x_e, x_e) \Rightarrow \text{SelfMgr}(x_e))\}. \end{aligned}$$

Consequently, from the new algorithm for composition we obtain

$$\mathcal{M}_{AD} = \{\exists f_1(\forall x_e((\text{Emp}(x_e) \wedge (x_e \doteq f_1(x_e))) \Rightarrow \text{SelfMgr}(x_e)))\}.$$

Therefore,

$$\mathbf{M}_{AB} = \text{MakeOperads}(\mathcal{M}_{AB}) = \{q_1, 1_{r_\emptyset}\}$$

where  $q_1 \in O(\text{Emp}, \text{SelfMgr})$  is an abstract operation represented by the expression  $((\_)_1(x_e) \wedge (x_e \doteq f_1(x_e))) \Rightarrow (\_)_1(x_e)$  and by a composition  $q_1 = v_1 \cdot q_{A,1}$  where  $q_{A,1} \in O(\text{Emp}, r_q)$ ,  $v_1 \in O(r_q, \text{SelfMgr})$ , with a new relational symbol  $r_q$  for the relation obtained by the query  $\text{Emp}(x_e) \wedge (x_e \doteq f_1(x_e))$ .

Then, for a mapping-interpretation (an  $R$ -algebra)  $\alpha$  such that it is a model (i.e., satisfies all the constraints in  $\Sigma_A$ ) of the source schema  $\mathcal{A} = (S_A, \Sigma_A)$  and defines its database instance  $A = \alpha^*(S_A) = \{\alpha(r_i) \mid r_i \in S_A\}$  and, analogously, a model of  $\mathcal{B}$  with  $B = \alpha^*(S_B)$  such that it satisfies also the schema mapping  $\mathcal{M}_{AB}$ . Consequently,  $(A, B) \in \text{Inst}(\mathcal{M}_{AB})$  and  $\alpha$  satisfies the SOTgd of the mapping  $\mathcal{M}_{AB}$  by fixing Tarski's interpretation for the functional symbol  $f_1$  in this SOTgd (denoted by  $I_T(f_1)$ ). We obtain the function  $\alpha(q_1) = \alpha(v_1)(\alpha(q_{A,1})) : \alpha(\text{Emp}) \rightarrow \alpha(\text{SelfMGr})$ , such that for any tuple  $\langle a \rangle \in \alpha(\text{Emp})$ :

$$\alpha(q_{A,1})(\langle a \rangle) = \langle a \rangle \quad \text{if } a = I_T(f_1)(a); \quad \langle \rangle \quad \text{otherwise.}$$

Relation  $\alpha(r_q)$  is equal to the image of  $\alpha(q_{A,1})$ , so that for any  $\langle a \rangle \in \alpha(r_q)$  we have

$$\alpha(v_1)(\langle a \rangle) = \langle a \rangle \quad \text{if } \langle a \rangle \in \alpha(\text{SelfMGr}); \quad \langle \rangle \quad \text{otherwise.}$$

Hence, the function  $\alpha(v_1)$  is an injection for the inclusion  $\alpha(r_q) \subseteq \alpha(\text{SelfMGr})$ .

We consider that every relation has the empty tuple as well; an empty relation in this case is considered as a relation that has only the empty tuple.

*Remark* Note that in the case when  $(B, D) \in \text{Inst}(\mathcal{M}_{BD})$  as well (i.e., when the schema mapping  $\mathcal{M}_{BD}$  is satisfied by  $B$  and  $D$ ), the resulting mapping  $\mathcal{M}_{AD}$  is also satisfied and, as a consequence, the function  $\alpha(v_1)$  is an *inclusion*. If  $\mathcal{M}_{AD}$  is *not* satisfied then  $\alpha(v_1)$  is not an inclusion. Consequently, the function  $\alpha(v_1)$  distinguishes when the mapping is satisfied or not, while the function  $\alpha(q_{A,1})$  represents the computation of the query  $\text{Emp}(x_e) \wedge (x_e \doteq f_1(x_e))$  for the instance-database  $A$  (so that  $\alpha(r_q) = \|\text{Emp}(x_e) \wedge (x_e \doteq f_1(x_e))\|_A$  is the image of the function  $\alpha(q_{A,1})$ ) that corresponds to the left-hand side of the implication of the operad's operation  $q_1$ .

The example above introduced the important properties of mapping-interpretations ( $R$ -algebras) and the way of recognizing when they are *models* of the schema mappings (that is, when they satisfy the schema mappings). Thus, we can formalize this property of mapping-interpretations by the following corollary:

**Corollary 4** *Let  $\mathcal{M}_{AB} : \mathcal{A} \rightarrow \mathcal{B}$  be a schema mapping. Then, for a given  $R$ -algebra  $\alpha$  that is a mapping-interpretation, the function  $\alpha(v_i)$  of each operad's operation  $q_i = v_i \cdot q_{A,i} \in \mathbf{M}_{AB} = \text{MakeOperads}(\mathcal{M}_{AB})$  is an injection iff the mapping  $\mathcal{M}_{AB}$  is satisfied by the instances  $A = \alpha^*(\mathcal{A})$  and  $B = \alpha^*(\mathcal{B})$  (i.e., when  $(\alpha^*(\mathcal{A}), \alpha^*(\mathcal{B})) \in \text{Inst}(\mathcal{M}_{AB})$ ). That is,  $\mathcal{M}_{AB}$  is satisfied iff for each  $q_i = v_i \cdot q_{A,i} \in O(r_1, \dots, r_k, r_B)$  in  $\mathbf{M}_{AB}$ , with  $q_{A,i} \in O(r_1, \dots, r_k, r_q)$ ,  $v_i \in O(r_q, r_B)$ , it holds that the image of the function  $f = \alpha(q_{A,i}) : R_1 \times \dots \times R_k \rightarrow \alpha(r_q)$  (where for each  $1 \leq i \leq k$ ,  $R_i = \mathcal{U}^{ar(r_i)} \setminus \alpha(r_i)$  if the place symbol  $(\_)_i \in q_i$  is preceded by negation operator  $\neg$ ;  $\alpha(r_i)$  otherwise) is a subset of  $\alpha(r_B)$ , i.e.,  $\text{im}(f) \subseteq \alpha(r_B)$ .*

*Proof* Let  $\phi_{Ai}(\mathbf{x}) \Rightarrow r_B(\mathbf{t})$  be an implication  $\chi$  in the normalized SOTgd of the mapping  $\mathcal{M}_{AB}$  and  $\mathbf{t}$  a tuple of terms with variables in  $\mathbf{x} = \langle x_1, \dots, x_m \rangle$ . Then, based on Definition 11 for the mapping-interpretation  $\alpha$ , we have:

If for a tuple  $\mathbf{d} = \langle d_1, \dots, d_m \rangle$  of values (in a given universe  $\mathcal{U}$ , which defines an assignment  $g : \{x_1, \dots, x_m\} \rightarrow \mathcal{U}$  such that  $g(x_j) = d_j$ ,  $1 \leq j \leq m$ ), the sentence  $\phi_{Ai}(\mathbf{x})/g$  is true then  $g^*(\mathbf{t})$  is a tuple in the relation  $\alpha(r_B) \in B$  as it follows from Proposition 2.

Consequently, for the operad's operation  $q_i \in O(r_1, \dots, r_k, r_B)$  obtained from the implication  $\chi$ , where  $q_i = v_i \cdot q_{A,i}$  with  $q_{A,i} \in O(r_1, \dots, r_k, r_q)$  and  $v_i \in O(r_q, r_B)$ , the function  $f = \alpha(q_{A,i}) : R_1 \times \dots \times R_k \rightarrow \alpha(r_q)$  is well defined for each  $\mathbf{d}_i \in R_i$  with  $\mathbf{d} = \text{Cmp}(S, \langle \mathbf{d}_1, \dots, \mathbf{d}_k \rangle)$  and, from Definition 11,

$$f(\langle \mathbf{d}_1, \dots, \mathbf{d}_k \rangle) = g^*(\mathbf{t}) \in \alpha(r_q),$$

with  $\alpha(v_i)(g^*(\mathbf{t})) = g^*(\mathbf{t}) \in \alpha(r_B)$ . Otherwise, if  $\phi_{Ai}(\mathbf{x})/g$  is false then  $f(\langle \mathbf{d}_1, \dots, \mathbf{d}_k \rangle) = \langle \rangle$ , the empty tuple in the relation  $\alpha(r_q)$ , with  $\alpha(v_i)(\langle \rangle) = \langle \rangle \in \alpha(r_B)$ .

Consequently, the function  $\alpha(v_i) : \alpha(r_q) \rightarrow \alpha(r_B)$  is an injection.  $\square$

Let us consider the case when an operad's operation is obtained from the integrity constraints of a schema:

*Example 12* Let  $(\phi_{Ai}(\mathbf{x}) \wedge (\mathbf{y} \neq \mathbf{z})) \Rightarrow r_{\top}(\bar{0}, \bar{1})$  be an implication  $\chi$  in the SOTgd obtained from the algorithm *EgdsToSOTgd* of a given egd  $(\phi_{Ai}(\mathbf{x}) \Rightarrow (\mathbf{y} \doteq \mathbf{z})) \in \Sigma_A$  of a schema  $\mathcal{A} = (S_A, \Sigma_A)$ , with  $\{r_1, \dots, r_k\} \subseteq S_A$  the set of all relational symbols of  $\mathcal{A}$  that appear in the conjunctive formula  $\phi_{Ai}(\mathbf{x})$  and  $\mathbf{y} = \langle x_{j_1}, \dots, x_{j_m} \rangle \subseteq \mathbf{x}$ ,  $\mathbf{z} = \langle x_{l_1}, \dots, x_{l_m} \rangle \subseteq \mathbf{x}$ , with  $j_i \neq l_i$  for  $1 \leq i \leq m$ .

We recall that the formula  $(\mathbf{y} \neq \mathbf{z})$  is an abbreviation for the disjunctive formula  $(x_{j_1} \neq x_{l_1}) \vee \dots \vee (x_{j_m} \neq x_{l_m})$ .

Then, by the algorithm *MakeOperads*, from this implication  $\chi$  we obtain the operad's operation  $q_{A,i} \in O(r_1, \dots, r_k, r_{q_i})$ ,  $v_i \in O(r_{q_i}, r_{\top})$  such that  $q_{A,i}$  is the expression obtained from the implication  $(\phi_{Ai}(\mathbf{x}) \wedge (\mathbf{y} \neq \mathbf{z})) \Rightarrow r_{q_i}(\bar{0}, \bar{1})$ , that is, the expression  $(e \wedge (\mathbf{y} \neq \mathbf{z})) \Rightarrow (\_) (\bar{0}, \bar{1})$  where  $e$  is the expression on the left-hand side of the implication, obtained from the formula  $\phi_{Ai}(\mathbf{x})$ , where each relational symbol  $r_m$  is replaced by a place symbol  $(\_)_m$ , for  $1 \leq m \leq k$ . Thus:

1. If the integrity condition, given by the egd  $\forall \mathbf{x}(\phi_{Ai}(\mathbf{x}) \Rightarrow (\mathbf{y} \doteq \mathbf{z}))$ , is satisfied then  $\phi_{Ai}(\mathbf{x}) \wedge (\mathbf{y} \neq \mathbf{z})$  is false for each assignment  $g$  to variables in  $\mathbf{x}$ . Thus, from Definition 11 for the mapping-interpretation  $\alpha$ , for each  $\langle \mathbf{d}_1, \dots, \mathbf{d}_k \rangle \in \alpha(r_1) \times \dots \times \alpha(r_k)$ ,  $\alpha(q_{A,i})(\langle \mathbf{d}_1, \dots, \mathbf{d}_k \rangle) = \langle \rangle$  so that  $\alpha(q_{A,i})$  is a constant function and  $\alpha(r_{q_i}) = \{\langle \rangle\}$  and hence  $\alpha(v_i)(\langle \rangle) = \langle \rangle \in \alpha(r_{\top}) = R_{\perp}$ . Consequently, the function  $\alpha(v_i)$  is an injection.
2. If this integrity constraint is *not* satisfied then there exists a tuple  $\mathbf{d}$  which defines an assignment  $g$  for the variables in  $\mathbf{x}$  with  $\mathbf{d}_y = g^*(\mathbf{y}) = \langle d_{j_1}, \dots, d_{j_m} \rangle$  and  $\mathbf{d}_z = g^*(\mathbf{z}) = \langle d_{l_1}, \dots, d_{l_m} \rangle$  such that  $\phi_{Ai}(\mathbf{x})/g \wedge (\mathbf{d}_y \neq \mathbf{d}_z)$  is true. That is, there exist at least one index  $1 \leq i \leq m$  such that  $d_{j_i} \neq d_{l_i}$  and for the operad's operation  $q_i = v_i \cdot q_{A,i}$  (the expression  $e \Rightarrow (\_) (\bar{0}, \bar{1})$ ), from Definition 11, for  $\langle \mathbf{d}_1, \dots, \mathbf{d}_k \rangle \in \alpha(r_1) \times \dots \times \alpha(r_k)$  such that  $\mathbf{d} = \text{Cmp}(S, \langle \mathbf{d}_1, \dots, \mathbf{d}_k \rangle)$  and  $\bigwedge \{\pi_{j_h}(\mathbf{d}_j) = \pi_{n_h}(\mathbf{d}_n) \mid (j_h, j), (n_h, n) \in S\}$  is true,  $\alpha(q_{A,i})(\langle \mathbf{d}_1, \dots, \mathbf{d}_k \rangle) = g^*(\langle \bar{0}, \bar{1} \rangle) = \langle g(\bar{0}), g(\bar{1}) \rangle = \langle 0, 1 \rangle \in \alpha(r_{q_i})$  and hence  $\alpha(v_i)(\langle 0, 1 \rangle) = \langle \rangle$  (because  $\langle 0, 1 \rangle \notin \alpha(r_{\top}) = R_{\perp}$ ). Consequently, the function  $\alpha(v_i)$  is *not* an injection.

Let for a schema  $\mathcal{A} = (S_A, \Sigma_A)$ ,  $(\phi_{Ai}(\mathbf{x}) \wedge \neg r(\mathbf{t})) \Rightarrow r_{\top}(\bar{0}, \bar{1})$  be an implication  $\chi$  in the SOTgd obtained from the algorithm *EgdsToConSOTgd* of a given normalized implication  $(\phi_{Ai}(\mathbf{x}) \Rightarrow r(\mathbf{t}))$  of a given tgdt in  $\Sigma_A$ , with  $\{r_1, \dots, r_k\} \subseteq S_A$  the set of all relational symbols of  $\mathcal{A}$  that appear in the conjunctive formula  $\phi_{Ai}(\mathbf{x})$ .

Then, based on the algorithm *MakeOperads*, from this implication  $\chi$  we obtain the operad's operations  $q_{A,i} \in O(r_1, \dots, r_k, r, r_{q_i})$  and  $v_i \in O(r_{q_i}, r_{\top})$  such that  $q_{A,i}$  is the expression obtained from the implication  $(\phi_{Ai}(\mathbf{x}) \wedge \neg r(\mathbf{t})) \Rightarrow r_{q_i}(\bar{0}, \bar{1})$ , that is, the expression  $(e \wedge \neg(\_)_{k+1}(\mathbf{t})) \Rightarrow (\_)_{\top}(\bar{0}, \bar{1})$  (the expression  $e$  on the left-hand side of the implication is obtained from the formula  $\phi_{Ai}(\mathbf{x})$  where each relational symbol  $r_m$  is replaced by a place symbol  $(\_)_m$ , for  $1 \leq m \leq k$ ). Thus:

1. If the integrity condition given by the tgdt  $\forall \mathbf{x}(\phi_{Ai}(\mathbf{x}) \Rightarrow r(\mathbf{t}))$  is satisfied then  $\phi_{Ai}(\mathbf{x}) \wedge \neg r(\mathbf{t})$  is false for each assignment  $g$  to variables in  $\mathbf{x}$  and hence, from Definition 11 for the mapping-interpretation  $\alpha$ , for each  $\langle \mathbf{d}_1, \dots, \mathbf{d}_{k+1} \rangle \in \alpha(r_1) \times \dots \times \alpha(r_k) \times (\mathcal{U}^{ar(r)} \setminus \alpha(r))$ ,  $\alpha(q_{A,i})(\langle \mathbf{d}_1, \dots, \mathbf{d}_{k+1} \rangle) = \langle \rangle$ , so that  $\alpha(q_{A,i})$  is a constant functions and  $\alpha(r_q) = \{\langle \rangle\}$  and hence  $\alpha(v_i)(\langle \rangle) = \langle \rangle \in \alpha(r_{\top}) = R_{=}$ . Consequently, the function  $\alpha(v_i)$  is an injection.
2. If this integrity constraint is *not* satisfied then there exists a tuple  $\mathbf{d}$  which defines an assignment  $g$  for the variables in  $\mathbf{x}$  such that  $\phi_{Ai}(\mathbf{x})/g \wedge \neg r(\mathbf{t})$  is true and, for the operad's operation  $q_i = v_i \cdot q_{A,i}$  (i.e., the expression  $(e \wedge \neg(\_)_{k+1}(\mathbf{t})) \Rightarrow (\_)_{\top}(\bar{0}, \bar{1})$ ), from Definition 11, for  $\langle \mathbf{d}_1, \dots, \mathbf{d}_{k+1} \rangle \in \alpha(r_1) \times \dots \times \alpha(r_k) \times (\mathcal{U}^{ar(r)} \setminus \alpha(r))$  such that  $\mathbf{d} = \text{Cmp}(S, \langle \mathbf{d}_1, \dots, \mathbf{d}_{k+1} \rangle)$  and  $\bigwedge \{\pi_{j_h}(\mathbf{d}_j) = \pi_{n_h}(\mathbf{d}_n) \mid \langle (j_h, j), (n_h, n) \rangle \in S\}$  is true,

$$\alpha(q_{A,i})(\langle \mathbf{d}_1, \dots, \mathbf{d}_{k+1} \rangle) = g^*(\langle \bar{0}, \bar{1} \rangle) = \langle g(\bar{0}), g(\bar{1}) \rangle = \langle 0, 1 \rangle \in \alpha(r_q),$$

so that  $\alpha(v_i)(\langle 0, 1 \rangle) = \langle \rangle$  (because  $\langle 0, 1 \rangle \notin \alpha(r_{\top}) = R_{=}$ ). Consequently, the function  $\alpha(v_i)$  is *not* an injection.

Consequently, Corollary 4 can be applied to a schema mapping that represents the integrity constraints  $\Sigma_A = \Sigma_A^{\text{egd}} \cup \Sigma_A^{\text{tgd}}$  over a given schema  $\mathcal{A}$ , that is, to the mapping  $\top_{AA_{\top}} = \{\text{EgdsToSOTgd}(\Sigma_A^{\text{egd}}) \wedge \text{TgdsToConSOTgd}(\Sigma_A^{\text{tgd}})\} : \mathcal{A} \rightarrow \mathcal{A}_{\top}$ , from a schema  $\mathcal{A}$  into the auxiliary schema  $\mathcal{A}_{\top} = (\{r_{\top}\}, \emptyset)$ .

Let us now consider the examples of Corollary 4 for the schema mappings, based on the SOTgds obtained from the set of tgds. The first one is a continuation of Example 8.

*Example 13* For the operads defined in Example 8, let a mapping-interpretation (an R-algebra)  $\alpha$  be an extension of Tarski's interpretation  $I_T$  of the source schema  $\mathcal{A} = (S_A, \Sigma_A)$  that satisfies all constraints in  $\Sigma_A$  and defines its database instance  $A = \alpha^*(S_A) = \{\alpha(r_i) \mid r_i \in S_A\}$  and, analogously, an interpretation of  $\mathcal{C}$ .

Let  $\alpha$  satisfy the SOTgd of the mapping  $\mathcal{M}_{AC}$  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  $\alpha(\text{EmpAcme})$ ,  $\alpha(\text{EmpAjax})$ ,  $\alpha(\text{Local})$ ,  $\alpha(\text{Office})$ , and  $\alpha(\text{CanRetire})$ . 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  $\bar{f}_{\text{Over65}}(a) = 1$  if  $\langle a \rangle \in \alpha(\text{Over65})$ .

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

1. The function  $\alpha(q_{A,1}) : \alpha(\text{EmpAcme}) \times \alpha(\text{Local}) \rightarrow \alpha(r_{q_1})$  such that for any tuple  $\langle a \rangle \in \alpha(\text{EmpAcme})$  and  $\langle b \rangle \in \alpha(\text{Local})$ ,

$$\alpha(q_{A,1})(\langle a \rangle, \langle b \rangle) = \langle a, I_T(f_1(a)) \rangle \quad \text{if } a = b; \quad \langle \rangle \quad \text{otherwise.}$$

And for any  $\langle a, b \rangle \in \alpha(r_{q_1})$ ,  $\alpha(v_1)(\langle a, b \rangle) = \langle a, b \rangle$  if  $\langle a, b \rangle \in \alpha(\text{Office})$ ;  $\langle \rangle$  otherwise.

2. The function  $\alpha(q_{A,2}) : \alpha(\text{EmpAjax}) \times \alpha(\text{Local}) \rightarrow \alpha(r_{q_2})$  such that for any tuple  $\langle a \rangle \in \alpha(\text{EmpAjax})$  and  $\langle b \rangle \in \alpha(\text{Local})$ ,

$$\alpha(q_{A,2})(\langle a \rangle, \langle b \rangle) = \langle a, I_T(f_2(a)) \rangle \quad \text{if } a = b; \quad \langle \rangle \quad \text{otherwise.}$$

And for any  $\langle a, b \rangle \in \alpha(r_{q_2})$ ,  $\alpha(v_2)(\langle a, b \rangle) = \langle a, b \rangle$  if  $\langle a, b \rangle \in \alpha(\text{Office})$ ;  $\langle \rangle$  otherwise.

3. The function  $\alpha(q_{A,3}) : \alpha(\text{EmpAcme}) \times \alpha(\text{Over65}) \rightarrow \alpha(r_{q_3})$  such that for any tuple  $\langle a \rangle \in \alpha(\text{EmpAcme})$  and  $\langle b \rangle \in \alpha(\text{Over65})$ ,

$$\alpha(q_{A,3})(\langle a \rangle, \langle b \rangle) = \langle a \rangle \quad \text{if } a = b; \quad \langle \rangle \quad \text{otherwise.}$$

And for any  $\langle a \rangle \in \alpha(r_{q_3})$ ,  $\alpha(v_3)(\langle a \rangle) = \langle a \rangle$  if  $\langle a \rangle \in \alpha(\text{CanRetire})$ ;  $\langle \rangle$  otherwise.

4. The function  $\alpha(q_{A,4}) : \alpha(\text{EmpAjax}) \times \alpha(\text{Over65}) \rightarrow \alpha(r_{q_4})$  such that for any tuple  $\langle a \rangle \in \alpha(\text{EmpAjax})$  and  $\langle b \rangle \in \alpha(\text{Over65})$

$$\alpha(q_{A,4})(\langle a \rangle, \langle b \rangle) = \langle a \rangle \quad \text{if } a = b; \quad \langle \rangle \quad \text{otherwise.}$$

And for any  $\langle a \rangle \in \alpha(r_{q_4})$ ,  $\alpha(v_4)(\langle a \rangle) = \langle a \rangle$  if  $\langle a \rangle \in \alpha(\text{CanRetire})$ ;  $\langle \rangle$  otherwise.

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

The second example is a continuation of Example 9:

*Example 14* For the operads defined in Example 9, let a mapping-interpretation (an R-algebra)  $\alpha$  be an extension of Tarski's interpretation  $I_T$  of the source schema  $\mathcal{A} = (S_A, \Sigma_A)$  that satisfies all constraints in  $\Sigma_A$  and defines its database instance  $A = \alpha^*(S_A) = \{\alpha(r_i) \mid r_i \in S_A\}$  and, analogously, an interpretation of  $\mathcal{C}$ .

Let  $\alpha$  satisfy the SOTgd of the mapping  $\mathcal{M}_{AC}$  by Tarski's interpretation for the functional symbol  $f_1$  in this SOTgd (denoted by  $I_T(f_1)$ ).

Then we obtain the relations  $\alpha(\text{Local})$ ,  $\alpha(\text{Office})$  and  $\alpha(\text{CanRetire})$ . 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  $\bar{f}_{\text{Over65}}(a) = 1$  if  $\langle a \rangle \in \alpha(\text{Over65})$  and, analogously, the interpretation of  $f_{\text{Emp}}$  is the characteristic function of the relation  $\alpha(\text{Emp})$  in the instance  $B$  with  $\bar{f}_{\text{Emp}}(a) = 1$  if  $\langle a \rangle \in \alpha(\text{Emp})$ .

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

1. The function  $\alpha(q_{A,1}) : \alpha(\text{Local}) \times \alpha(\text{Emp}) \rightarrow \alpha(r_{q_1})$  such that for any tuple  $\langle a \rangle \in \alpha(\text{Local})$  and  $\langle b \rangle \in \alpha(\text{Emp})$ ,

$$\alpha(q_{A,1})(\langle a \rangle, \langle b \rangle) = \langle a, I_T(f_1(a)) \rangle \quad \text{if } a = b; \quad \langle \rangle \quad \text{otherwise.}$$

And for any  $\langle a, b \rangle \in \alpha(r_{q_1})$ ,  $\alpha(v_1)(\langle a, b \rangle) = \langle a, b \rangle$  if  $\langle a, b \rangle \in \alpha(\text{Office})$ ;  $\langle \rangle$  otherwise.

2. The function  $\alpha(q_{A,2}) : \alpha(\text{Emp}) \times \alpha(\text{Over65}) \rightarrow \alpha(r_{q_2})$  such that for any tuple  $\langle a \rangle \in \alpha(\text{Emp})$  and  $\langle b \rangle \in \alpha(\text{Over65})$

$$\alpha(q_{A,2})(\langle a \rangle, \langle b \rangle) = \langle a \rangle \quad \text{if } a = b; \quad \langle \rangle \quad \text{otherwise.}$$

And for any  $\langle a \rangle \in \alpha(r_{q_2})$ ,  $\alpha(v_2)(\langle a \rangle) = \langle a \rangle$  if  $\langle a \rangle \in \alpha(\text{CanRetire})$ ;  $\langle \rangle$  otherwise.

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

### 2.4.2 Query-Answering Abstract Data-Object Types and Operads

We consider the views as a universal property for the databases: they are the possible observations of the information contained in an instance-database, and we can use them in order to establish an equivalence relation between databases.

In the theory of *algebraic specifications*, an Abstract Data Type (ADT) is specified by a set of operations (constructors) that determine how the values of the carrier set are built up and by a set of formulae (in the simplest case, the equations) stating which values should be identified. In the standard initial algebra semantics, the defining equations impose a congruence on the initial algebra. Dually, a *coalgebraic specification* of a class of systems, i.e., Abstract Object Types (AOT), is characterized by a set of operations (destructors) that specify what can be *observed* out of a system-*state* (i.e., an element of the carrier) and how a state can be transformed to a successor-state.

We start by introducing the class of coalgebras for database query-answering systems for a given instance-database (a set of relations)  $A$ . They are presented in an algebraic style by providing a co-signature. In particular, the sorts include one single “hidden sort” corresponding to the carrier of the coalgebra and other “visible” sorts, for the inputs and outputs, that have a given fixed interpretation. Visible sorts will be interpreted as sets without any algebraic structure defined on them. For us, the coalgebraic terms built over destructors are interpreted as the basic *observations* that one can make on the states of a coalgebra. Input sorts are considered as a set  $\mathcal{L}_A$  of the finite unions of conjunctive finite-length queries  $q(\mathbf{x})$  for a given instance-database  $A$ , as specified in Sect. 1.4.1.

Based on the theory of database observations and its power-view operator  $T$ , defined in Sect. 1.4.1, the output sort of this database AOT is the set  $TA$  of all resulting views (i.e., resulting  $n$ -ary relations) obtained by computation of queries  $q(\mathbf{x}) \in \mathcal{L}_A$ . It is considered as the carrier of a coalgebra as well.



**Definition 12** AOT for a database query-answering system, for a given instance-database  $A$ , is a pair  $(S, \Sigma_{AOT})$  such that:

1. The carrier set  $S = (X_A, \mathcal{L}_A, \underline{\mathcal{U}})$  of the sorts where  $X_A$  is a hidden sort (a set of states of this database system),  $\mathcal{L}_A$  is an input sort (a set of the unions of conjunctive queries over  $A$ ), and  $\underline{\mathcal{U}}$  is the set of all finitary relations for a given universe  $\mathcal{U}$ .
  2. The signature  $\Sigma_{AOT} = \{Next, Out\}$  is a set of operations:
    - 2.1. A method  $Next : X_A \times \mathcal{L}_A \rightarrow X_A$  that corresponds to an execution of a next query  $q(\mathbf{x}) \in \mathcal{L}_A$  in a current state  $s \in X_A$  of a database  $A$  such that a database  $A$  passes to the next state; and
    - 2.2. An attribute  $Out : X_A \times \mathcal{L}_A \rightarrow \underline{\mathcal{U}}$  such that for each  $s \in X_A$ ,  $q(\mathbf{x}) \in \mathcal{L}_A$ ,  $Out(s, q(\mathbf{x}))$  is a relation computed by a query  $q(\mathbf{x})$ .
- The Data Object Type for a query-answering system is given by a coalgebra:
3.  $\langle \lambda Next, \lambda Out \rangle : X_A \rightarrow X_A^{\mathcal{L}_A} \times \underline{\mathcal{U}}^{\mathcal{L}_A}$  of the polynomial endofunctor  $(\_)^{\mathcal{L}_A} \times \underline{\mathcal{U}}^{\mathcal{L}_A} : \mathbf{Set} \rightarrow \mathbf{Set}$ , where  $\lambda$  is the currying operator for functions.

In an object-oriented terminology, the coalgebras are expressive enough in order to specify the parametric methods and the attributes for a database (conjunctive) query answering systems. In a transition system terminology, such coalgebras can model a deterministic, non-terminating, transition system with inputs and outputs. In [4], a complete equational calculus for such coalgebras of restricted class of polynomial functors has been defined.

Here we will consider only the database query-answering systems without the side effects. That is, the obtained results (views) *will not be materialized* as a new relation of this database  $A$  but only visualized. Thus, when a database answers to a query, it remains in the same initial state. Thus, the set  $X_A$  is a singleton  $\{A\}$  for a given database  $A$  and, consequently, it is isomorphic to the terminal object  $1$  in the **Set** category. As a consequence, from  $1^{\mathcal{L}_A} \simeq 1$ , we obtain that a method  $Next$  is just an identity function  $id : 1 \rightarrow 1$ . Thus, the only interesting part of this AOT is the attribute part  $Out : X_A \times \mathcal{L}_A \rightarrow \underline{\mathcal{U}}$ , with the fact that  $X_A \times \mathcal{L}_A = \{A\} \times \mathcal{L}_A \simeq \mathcal{L}_A$ .

Consequently, we obtain an attribute mapping  $Out : \mathcal{L}_A \rightarrow \underline{\mathcal{U}}$ , whose graph is equal to the query-evaluation surjective mapping  $ev_A : \mathcal{L}_A \twoheadrightarrow TA$ , introduced in Sect. 1.4.1.

This mapping  $ev_A : \mathcal{L}_A \twoheadrightarrow TA$  will be used as a semantic foundation for the database mappings.

**Corollary 5** *A canonical method for the construction of the power-view database  $TA$  can be obtained by an Abstract Data-Object Type  $(S, \Sigma_{AOT})$  for a query-answering system without side-effects as follows:*

$$TA \triangleq \{Out(q_i(\mathbf{x})) \mid q_i(\mathbf{x}) \in \mathcal{L}_A\}.$$

*Proof* In fact, from the reduction of  $Out$  to  $ev_A$  (for an AOT without side-effects), for a given database instance  $A$ ,  $\{Out(q_i(\mathbf{x})) \mid q_i(\mathbf{x}) \in \mathcal{L}_A\} = \{ev_A(q_i(\mathbf{x})) \mid q_i(\mathbf{x}) \in \mathcal{L}_A\} = TA$ , from the fact that (in Sect. 1.4.1)  $ev_A$  is a surjective function.  $\square$



This corollary is a direct proof that the power-view database operator  $T$ , introduced in Sect. 1.4.1, represents the *observational* point of view for the instance-databases.

In fact, a conjunctive query is an implication from the body of a query  $q_i(\mathbf{x}) \in \mathcal{L}_A$  with relational symbols in  $\{r_{i1}, \dots, r_{ik}\}$  (a subset of relational symbols in a schema  $\mathcal{A}$ ) into the head  $r_q(\mathbf{x})$ , that is, a formula  $q_i(\mathbf{x}) \Rightarrow r_q(\mathbf{x})$  that can be represented equivalently by an operad's operation  $q_{A,i} = O(r_{i1}, \dots, r_{ik}, r_q)$  of the mapping  $\mathcal{M} : \mathcal{A} \rightarrow T\mathcal{A}$  with an SOTgd  $\forall \mathbf{x}(q_i(\mathbf{x}) \Rightarrow r_q(\mathbf{x}))$ .

Thus, for a given R-algebra  $\alpha$  that is a mapping-interpretation of operads (Definitions 10 and 11) such that the instance  $A = \alpha^*(\mathcal{A})$  of the schema  $\mathcal{A}$  satisfies the SOTgd of the mapping  $\mathcal{M} = \{\forall \mathbf{x}(q_i(\mathbf{x}) \Rightarrow r_q(\mathbf{x}))\} : \mathcal{A} \rightarrow T\mathcal{A}$  above,  $Out(q_i(\mathbf{x})) = \|q_i(\mathbf{x})\|_A$  is the image of the function  $\alpha(q_{A,i}) : \alpha(r_{i1}) \times \dots \times \alpha(r_{ik}) \rightarrow \alpha(r_q)$  obtained from the operad's operation  $q_{A,i} = O(r_{i1}, \dots, r_{ik}, r_q)$ , that is,

$$Out(q_i(\mathbf{x})) = \{\alpha(q_{A,i})(\mathbf{d}_1, \dots, \mathbf{d}_m) \mid \mathbf{d}_j \in \alpha(r_{ij}), \text{ for } 1 \leq j \leq k\} = \alpha(r_q).$$

Consequently, the AOT without side-effects of the instance database  $A = \alpha^*(\mathcal{A})$  is the observational point of view for the basic database mappings and closely interrelated with mapping-operads.

Note that a single view-mapping at instance-database level can be defined as a T-coalgebra  $f = \alpha^*(MakeOperads(\mathcal{M})) = \{\alpha(q_{A,i}), q_\perp\} : A \rightarrow TA$  that, obviously, *is not* a function but a set of functions. These arguments will be analyzed in detail after the definition of the database category **DB**, and hence it will be demonstrated that  $f$  is a morphism in such a database category.

### 2.4.3 Strict Semantics of Schema Mappings: Information Fluxes

As it was explained previously, the SOTgds represent the logical language syntax for the mapping compositions and use the existentially quantified functional symbols whose interpretation introduces the data values not contained in the active data domain of the source schema.

In fact, these functional symbols, introduced in the new algorithm (defined in Sect. 2.3) for composition of mappings, are used in order to replace the data contained in 'intermediate' databases (the hidden databases between the source and target database schemas) in order to guarantee the logical equivalence of *composed* SOTgd and *the set* of SOTgds used in such a composition for the determination of the target database. This fact in the data-exchange setting guarantees that each query over a target database will give the same resulting view regardless of whether we are using the set of intermediate mappings from the source into this target database, or the single SOTgd of the resulting composition obtained from the algorithm in Sect. 2.3.

From this logical point of view, the composed SOTgd represents the *complete* information of all 'intermediate' databases used in this composition and includes the strict subset of information of the source database that is mapped into the target database.

In the categorial semantics of mappings and their composition, we are only interested in this strict subset of information that is mapped (only) *from the source database*.

Differently from the data-exchange settings, we are not interested in the ‘minimal’ instance  $B$  of the target schema  $\mathcal{B}$  such that, for a given instance (model)  $A$  of the source schema  $\mathcal{A}$ ,  $(A, B)$  is an instance of the SOTgd of the schema mapping  $\mathcal{M}_{AB}$ . In our more general framework, we do not intend to determine ‘exactly’, ‘canonically’ or ‘completely’ the instance of the target schema  $\mathcal{B}$  (for a fixed instance  $A$  of the source schema  $\mathcal{A}$ ). Such a setting is more general and the target database is *only partially* determined by the source database  $\mathcal{A}$ : the other part of this target database can be, for example, determined by another database  $\mathcal{C}$  (that is not any of the ‘intermediate’ databases between  $\mathcal{A}$  and  $\mathcal{B}$ ), or by the software programs which update the information contained in this target database  $\mathcal{B}$ .

In other words, the Data-exchange (and Data integration) settings are only special particular cases of this *general framework* of database mappings, where each database can be mapped from other sources, to map a part of its own information into other targets, and to be locally updated as well.

This last feature of the local update of a given database, in the database-mapping systems, will be considered in full detail in Chap. 7 dedicated to Operational semantics for database mappings. The process of an update of a database-mapping system begins with one local update of a given database and, after that, this update has to be propagated through the whole network of inter-mapped databases in order to guarantee that every (atomic) schema mapping in this network has to be satisfied at the end of this update-processing.

In our case, two equal compositions are not necessarily logically equivalent formulae, and their equality is considered only at the instance-database level.

Let us suppose that  $\mathcal{M}_{AB_1} : \mathcal{A} \rightarrow \mathcal{B}_1$  is an atomic mapping, based on the set of tgds that ‘transfer’, for a given instance  $A$  of  $\mathcal{A}$ , the set  $S$  of views from  $\mathcal{A}$  to  $\mathcal{B}_1$  (based on the left-hand side of implications in the tgds), and let  $\mathcal{M}_{B_1 B_n} : \mathcal{B}_1 \rightarrow \mathcal{B}_n$  be a (possibly non-atomic) mapping composed of a set of atomic mappings  $\mathcal{M}_{B_i B_{i+1}} : \mathcal{B}_i \rightarrow \mathcal{B}_{i+1}$ , for  $1 \leq i \leq n$ ,  $n \geq 2$ . Let  $S_i$  be the set of views of the  $i$ th atomic mapping for a given instance  $B_i$  of the schema  $\mathcal{B}_i$  (based on the left-hand side of implications in the tgds of this atomic mapping). This second (possibly composed) mapping  $\mathcal{M}_{B_1 B_n}$  filters the information contained in the set of all possible views  $TS$  that can be obtained from the set of relations in  $S$ . A view  $v \in TS$  will be propagated into the target database  $B_n$  if  $v \in TS_i$  for all intermediate atomic mappings, i.e., for  $1 \leq i \leq n$ .

Consequently, the *strict semantics* of a composed mapping  $\mathcal{M}_{B_1 B_n} \circ \mathcal{M}_{AB_1} : \mathcal{A} \rightarrow \mathcal{B}_n$  represents the subset of all views in  $TS$  that are transferred to the target database  $\mathcal{B}_n$ , that is, it is the intersection  $TS \cap TS_1 \cap \dots \cap TS_n$  and hence equal to  $TS \cap TS_R$  where  $TS_R = TS_1 \cap \dots \cap TS_n$  is the strict semantics of the composed mapping  $\mathcal{M}_{B_1 B_n}$ .

Based on these considerations, we are now able to define an abstraction of this transferred information from a given source to a target schema, called *information flux*.

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

**Definition 13** (INFORMATION FLUX) Let  $\alpha$  be a mapping-interpretation (an R-algebra in Definition 11) of a given set  $\mathbf{M}_{AB} = \{q_1, \dots, q_n, 1_{r_\emptyset}\} = \text{MakeOperads}(\mathcal{M}_{AB})$  of mapping-operads, obtained from an *atomic* mapping  $\mathcal{M}_{AB} : \mathcal{A} \rightarrow \mathcal{B}$ , and  $A = \alpha^*(S_A)$  be an instance of the schema  $\mathcal{A} = (S_A, \Sigma_A)$  that satisfies all constraints in  $\Sigma_A$ .

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

(i)  $\text{Var}(\mathbf{M}_{AB}) = \bigcup_{1 \leq i \leq n} \{\{x\} \mid x \in \mathbf{x}_i\}$ .

We define the kernel of the information flux of  $\mathbf{M}_{AB}$ , for a given mapping-interpretation  $\alpha$ , by (we denote the image of a function  $f$  by ' $\text{im}(f)$ ')  
(ii)  $\Delta(\alpha, \mathbf{M}_{AB}) = \{\pi_{\mathbf{x}_i}(\text{im}(\alpha(q_i))) \mid q_i \in \mathbf{M}_{AB}, \text{ and } \mathbf{x}_i \text{ is not empty}\} \cup \perp^0$ , if

$\text{Var}(\mathbf{M}_{AB}) \neq \emptyset$ ;  $\perp^0$  otherwise.

We define the information flux by its kernel by

(iii)  $\text{Flux}(\alpha, \mathbf{M}_{AB}) = T(\Delta(\alpha, \mathbf{M}_{AB}))$ .

The flux of composition of  $\mathbf{M}_{AB}$  and  $\mathbf{M}_{BC}$  is defined by

(iv)  $\text{Flux}(\alpha, \mathbf{M}_{BC} \circ \mathbf{M}_{AB}) = \text{Flux}(\alpha, \mathbf{M}_{AB}) \cap \text{Flux}(\alpha, \mathbf{M}_{BC})$ .

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.

We recall that the *kernels* of the information fluxes, defined in point (ii), will be used as the actions for the Labeled Transition Systems (LTS) in the operational semantics for the database mappings (in Sect. 7.3).

The information flux of the SOTgd of the mapping  $\mathcal{M}_{AB}$  for the instance-level mapping  $f = \alpha^*(\mathbf{M}_{AB}) : A \rightarrow \alpha^*(\mathcal{B})$  composed of the set of functions  $f = \alpha^*(\mathbf{M}_{AB}) = \{\alpha(q_1), \dots, \alpha(q_n), q_\perp\}$  is denoted by  $\tilde{f}$ . Notice that  $\perp \in \tilde{f}$ , and hence the information flux  $\tilde{f}$  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 will be used in order to define the database **DB** category where the instance-mappings will be 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  $\perp$ ) will be its objects.

In the case of an atomic mapping, obtained from a set of egds of a given schema  $\mathcal{A}$ , we have the following particular property:

**Corollary 6** *Let  $\Phi$  be the SOTgd equal to  $EgdsToSOTgd(\Sigma_A^{\text{egd}})$  and  $\Psi$  be the SOTgd equal to  $TgdsToConSOTgd(\Sigma_A^{\text{tgd}})$  for a given schema  $\mathcal{A} = (S_A, \Sigma_A^{\text{egd}} \cup \Sigma_A^{\text{tgd}})$ . Then, for every  $R$ -algebra  $\alpha$ , the information flux  $\text{Flux}(\alpha, \text{MakeOperads}(\{\Phi \wedge \Psi\})) = \perp^0$  is empty.*

*Proof* From Definition 7, the integrity constraints are representable by an atomic mapping  $\top_{AA\top} = \{\Phi \wedge \Psi\} : \mathcal{A} \rightarrow \mathcal{A}_\top$ .

From the fact that each abstract operad-operation  $q_i$  in  $\text{MakeOperads}(\{\Phi\})$  and in  $\text{MakeOperads}(\{\Psi\})$  is of the form  $e \Rightarrow r_\top(\bar{0}, \bar{1})$ , we have no free variables on the right-hand side of these implications, so that  $\mathbf{x}_i$  in Definition 13 for the information flux is empty and  $\text{Var}(\mathbf{T}_{AA\top}) = \emptyset$ , so that, from Definition 13 of its kernel,  $\Delta(\alpha, \mathbf{T}_{AA\top}) = \perp^0$  and hence  $\text{Flux}(\alpha, \mathbf{T}_{AA\top}) = T \perp^0 = \perp^0$  is empty.  $\square$

Note that this corollary confirms that, for any database schema  $\mathcal{A} = (S_A, \Sigma_A)$  where  $\Sigma_A = \Sigma_A^{\text{egd}} \cup \Sigma_A^{\text{tgd}}$ , we can define the integrity-constraints mapping, as it was demonstrated by Example 12, by a schema mapping  $\top_{AA\top} = \{\Phi \wedge \Psi\} : \mathcal{A} \rightarrow \mathcal{A}_\top$  (where  $\Phi$  is the SOTgd equal to  $EgdsToSOTgd(\Sigma_A^{\text{egd}})$  and  $\Psi$  is the SOTgd equal to  $TgdsToConSOTgd(\Sigma_A^{\text{tgd}})$  and  $\mathcal{A}_\top = (\{r_\top\}, \emptyset)$ ) and, consequently, by equivalent operads-mapping  $\mathbf{T}_{AA\top} = \text{MakeOperads}(\{\Phi\}) \cup \text{MakeOperads}(\{\Psi\}) : \mathcal{A} \rightarrow \mathcal{A}_\top$  with the *empty* information flux.

We have no mapping from  $\mathcal{A}_\top$  into other schema mappings, so that this integrity-constraint mapping does not participate in any significant composition with other mappings in a given database mapping system. Moreover, from the fact that the information flux of composed mappings is equal to the intersection of the information fluxes of all atomic arrows which compose this mapping, such a composed mapping which contains an atomic integrity-constraint mapping will always have an empty flux. Consequently, the role of the integrity-constraint mappings is only “logical”, used to express the integrity constraints for schemas, and to verify if for a given mapping-interpretation  $\alpha$  in Definition 11 they are satisfied (as it was specified by Corollary 4). The extension of the database mapping systems with the integrity-constraints mappings will not modify its original semantic structure, but in this extended framework not only the inter-schema mappings but also the schema integrity constraints will be the “first objects” of big data integration theory and will be presented in a uniform elegant manner.

Let us consider the following example, in the case when the schema mappings are satisfied by a given mapping-interpretation  $\alpha$ :

**Example 15** Let us consider Example 2 for composition of the atomic mappings  $\mathcal{M}_{AB} : \mathcal{A} \rightarrow \mathcal{B}$ ,  $\mathcal{M}_{BD} : \mathcal{B} \rightarrow \mathcal{D}$ ,  $\mathcal{M}_{AC} : \mathcal{A} \rightarrow \mathcal{C}$  and  $\mathcal{M}_{CD} : \mathcal{C} \rightarrow \mathcal{D}$ , where

$$\begin{aligned} \mathcal{M}_{AB} &= \{\forall x_n \forall x_c (\text{Takes}(x_n, x_c) \Rightarrow \text{Takes1}(x_n, x_c)), \\ &\quad \forall x_n \forall x_c (\text{Takes}(x_n, x_c) \Rightarrow \text{Student}(x_n, f_1(x_n)))\}, \\ \mathcal{M}_{BD} &= \{\forall x_n \forall x_c \forall y ((\text{Takes1}(x_n, x_c) \end{aligned}$$

$$\begin{aligned}
& \wedge \text{Student}(x_n, y) \Rightarrow \text{Enrolment}(y, x_c) \}} \\
\mathcal{M}_{AC} &= \{\forall x_n \forall x_c (\text{Takes}(x_n, x_c) \Rightarrow \text{Learning}(x_n, x_c, f_2(x_n, x_c)))\}, \\
\mathcal{M}_{CD} &= \{\forall x_n \forall x_c \forall x_p (\text{Learning}(x_n, x_c, x_p) \Rightarrow \text{Teaching}(x_p, x_c))\}.
\end{aligned}$$

Hence, their composition is equal to the following mappings:

$$\begin{aligned}
\mathcal{M}_{AD} &= \mathcal{M}_{BD} \circ \mathcal{M}_{AB} \\
&= \{\exists f_1 (\forall x_1 \forall x_2 (\text{Takes}(x_1, x_2) \Rightarrow \text{Enrolment}(f_1(x_1), x_2)))\}, \\
\mathcal{M}'_{AD} &= \mathcal{M}_{CD} \circ \mathcal{M}_{AC} \\
&= \{\exists f_2 (\forall x_1 \forall x_2 (\text{Takes}(x_1, x_2) \Rightarrow \text{Teaching}(f_2(x_1, x_2), x_2)))\},
\end{aligned}$$

which are not logically equivalent.

However, we obtain from Definition 13, for  $A = \alpha^*(\mathcal{A})$ ,  $B = \alpha^*(\mathcal{B})$  and  $C = \alpha^*(\mathcal{C})$ :

$$\begin{aligned}
& \text{Flux}(\alpha, (\text{MakeOperads}(\mathcal{M}_{BD}) \circ \text{MakeOperads}(\mathcal{M}_{AB}))) \\
&= \text{Flux}(\alpha, \text{MakeOperads}(\mathcal{M}_{AB})) \cap \text{Flux}(\alpha, \text{MakeOperads}(\mathcal{M}_{BD})) \\
&= T(\|\text{Takes}(x_n, x_c)\|_A, \pi_{x_n} \|\text{Takes}(x_n, x_c)\|_A) \\
&\quad \cap T(\{\pi_{x_c, y} \|\text{Takes1}(x_n, x_c) \wedge \text{Student}(x_n, y)\|_B\}),
\end{aligned}$$

and

$$\begin{aligned}
& \text{Flux}(\alpha, (\text{MakeOperads}(\mathcal{M}_{CD}) \circ \text{MakeOperads}(\mathcal{M}_{AC}))) \\
&= \text{Flux}(\alpha, \text{MakeOperads}(\mathcal{M}_{AC})) \cap \text{Flux}(\alpha, \text{MakeOperads}(\mathcal{M}_{CD})) \\
&= T(\|\text{Takes}(x_n, x_c)\|_A) \cap T(\{\pi_{x_p, x_c} \|\text{Learning}(x_n, x_c, x_p)\|_C\}).
\end{aligned}$$

In the case when in the given universe  $\mathcal{U} = \mathbf{dom}$ , the domain  $\text{dom}(y)$  for the attribute  $y$  of the atom  $\text{Student}(x_n, y)$  is disjoint from  $\text{dom}(x_n)$  and from  $\text{dom}(x_c)$  of the atom  $\text{Takes}(x_n, x_c)$ , and if  $\text{dom}(x_p)$  for the attribute  $x_p$  of the atom  $\text{Learning}(x_n, x_c, x_p)$  is disjoint from  $\text{dom}(x_n)$  and from  $\text{dom}(x_c)$  of the atom  $\text{Takes}(x_n, x_c)$ , then it is easy to verify that

$$\begin{aligned}
& \text{Flux}(\alpha, (\text{MakeOperads}(\mathcal{M}_{BD}) \circ \text{MakeOperads}(\mathcal{M}_{AB}))) \\
&= \text{Flux}(\alpha, (\text{MakeOperads}(\mathcal{M}_{CD}) \circ \text{MakeOperads}(\mathcal{M}_{AC}))) \\
&= T(\|\text{Takes}(x_n, x_c)\|_A).
\end{aligned}$$

Consequently, in this case, the two composed mappings  $\mathcal{M}_{AD}$  and  $\mathcal{M}'_{AD}$  have the same information fluxes from  $\mathcal{A}$  into the target schema  $\mathcal{D}$ , so that, from the strict semantics point of view, they are equal instance-level mappings.

We have shown in Corollary 2 that the schema-level mappings have the categorical morphism's properties (an associative composition, with an identity mapping for each schema) and now we will show that the information fluxes of the schema mappings have the categorical morphism's properties at the instance-mapping level as well:

**Corollary 7** *The information fluxes of the schema mappings and the composition (i.e., the set intersection) of information fluxes satisfy categorical properties of the morphisms between instance-databases as objects of such a category.*

*Proof* For a given atomic schema mapping  $\mathcal{M}_{AB} : \mathcal{A} \rightarrow \mathcal{B}$  and an instance  $A = \alpha^*(\mathcal{A})$ , the information flux  $\tilde{f} = \text{Flux}(\alpha, \text{MakeOperads}(\mathcal{M}_{AB})) \subseteq TA$  may represent the instance-mapping from the instance  $A$  into an instance  $B = \alpha^*(\mathcal{B})$ .

Note that if  $(A, B) \in \text{Inst}(\mathcal{M}_{AB})$  (that is, when the instances  $A$  and  $B$  satisfy the schema mapping  $\mathcal{M}_{AB}$ ) then all information contained in this information flux  $\tilde{f}$  is transferred from  $A$  into  $B$ .

It is easy to verify that for an identity schema mapping (which is always satisfied)  $\text{Id}_A : \mathcal{A} \rightarrow \mathcal{A}$ , defined in Lemma 5,  $\text{Flux}(\alpha, \text{MakeOperads}(\text{Id}_A)) = TA$ , so that

$$\begin{aligned} & \text{Flux}(\alpha, \text{MakeOperads}(\text{Id}_A) \circ \text{MakeOperads}(\mathcal{M}_{AB})) \\ &= \text{Flux}(\alpha, \text{MakeOperads}(\text{Id}_A)) \cap \text{Flux}(\alpha, \text{MakeOperads}(\mathcal{M}_{AB})) \\ &= TA \cap \text{Flux}(\alpha, \text{MakeOperads}(\mathcal{M}_{AB})) \\ &= \text{Flux}(\alpha, \text{MakeOperads}(\mathcal{M}_{AB})), \end{aligned}$$

and hence the property of the categorial composition with identity morphisms (represented here by the information flux of an identity schema mapping) is satisfied.

The set intersection  $\cap$  is an associative operation so that, together with the identity property above, it satisfies the categorial properties for composition of morphisms.  $\square$

Based on Corollary 2 and Corollary 7, it is possible to define the categorial semantics for the schema mappings, by defining a functor from the sketch category of a given schema-mapping graph into an instance-level category where the objects are the database-instances and the morphisms are characterized by the information fluxes of the schema-mappings.

The formalization of the schema mappings by means of operads, as it will be demonstrated in next two chapters, is useful in order to be able to extend each R-algebra  $\alpha$  to a functor from the category sketch (obtained from a graph of (inter)schema mappings) into the base **DB** category, which represents the denotational semantics of this schema database mapping graph.

That is, each schema mapping  $\mathcal{M}_{AB}$  transformed into its mapping-operad  $\mathbf{M}_{AB} = \text{MakeOperads}(\mathcal{M}_{AB}) = \{q_1, \dots, q_n, 1_{r_0}\} : \mathcal{A} \rightarrow \mathcal{B}$  may be seen as a morphism of a sketch category, and hence it will be mapped by a functor (R-algebra)  $\alpha$ , which satisfies Definition 11 (i.e., such that it is a *mapping-interpretation*), into the

**DB** category morphism  $\alpha^*(\mathbf{M}_{AB}) = \{f_1, \dots, f_n, q_\perp\} : A \rightarrow B$ , where  $A = \alpha(S_A)$  is an instance-database of the schema  $\mathcal{A}$ ,  $B = \alpha(S_B)$  is an instance-database of the schema  $\mathcal{B}$ , and  $f_i = \alpha(q_i)$  are the functions obtained from the operads, with the domain equal to the Cartesian product of a subset of relations in the instance  $A$  and the codomain is a relation in the instance  $B$ .

The functors such that, for every operad's operation  $q_i = v_i \cdot q_{A,i}$  in each sketch's mapping, the function  $\alpha(v_i)$  is an inclusion (injection) will define a *model* of the schema database mapping system expressed by this sketch.

This is the principal idea for the *categorical semantics* of the schema database mapping systems and will be discussed with more details in next two chapters.

## 2.5 Algorithm for Decomposition of SOTgds

In this section, we will present an algorithm for decomposition of a given SOTgd into two SOTgds. This decomposition will be used for the demonstration of the extended symmetry properties of the database mappings at the instance-level (Sect. 3.2.3 in Chap. 3). Let us define this algorithm that transforms an SOTgd  $\Phi$  of a given schema mapping into two SOTgds  $\Phi_E$  and  $\Phi_M$  such that the composition of them is equivalent to the original SOTgd  $\Phi$ :

**Decomposition algorithm**  $DeCompose(\mathcal{M}_{AB})$

**Input.** A schema mapping  $\mathcal{M}_{AB} : \mathcal{A} \rightarrow \mathcal{B}$  given by a SOTgd  $\Phi$ .

$$\exists \mathbf{f}((\forall \mathbf{x}_1(\phi_{A,1} \Rightarrow \psi_{B,1})) \wedge \dots \wedge (\forall \mathbf{x}_n(\phi_{A,n} \Rightarrow \psi_{B,n}))).$$

**Output.** The pair  $(\Phi_E, \Phi_M)$  of SOTgds.

1. (*Normalize the SOTgd*)

Initialize  $S$ ,  $S_E$  and  $S_M$  to be the empty sets ( $\emptyset$ ). Let  $\mathcal{M}_{AB}$  be the singleton set

$$\{\exists \mathbf{f}((\forall \mathbf{x}_1(\phi_{A,1} \Rightarrow \psi_{B,1})) \wedge \dots \wedge (\forall \mathbf{x}_n(\phi_{A,n} \Rightarrow \psi_{B,n})))\}.$$

Put each of  $n$  implications  $\phi_{A,i} \Rightarrow \psi_{B,i}$ , for  $1 \leq i \leq n$ , into  $S$ .

Each implication  $\chi$  in  $S$  has the form  $\phi_{A,i}(\mathbf{x}_i) \Rightarrow \wedge_{1 \leq j \leq k} r_j(\mathbf{t}_j)$  where every variable in  $\mathbf{x}_i$  is universally quantified, and each  $\mathbf{t}_j$ , for  $1 \leq j \leq k$ , is a sequence (tuple) of terms with variables in  $\mathbf{x}_i$ . We then replace each such implication  $\chi$  in  $S$  with  $k$  implications:  $\phi_{A,i}(\mathbf{x}_i) \Rightarrow r_1(\mathbf{t}_1), \dots, \phi_{A,i}(\mathbf{x}_i) \Rightarrow r_k(\mathbf{t}_k)$ .

2. (*Transformation into two new SOTgds*)

Let  $S = \{\chi_1, \dots, \chi_m\}$  be the set of implications obtained in the previous step.

Then for each implication  $\chi_i$ , equal to the formula  $q_{Ai}(\mathbf{x}_i) \Rightarrow r_i(\mathbf{t}_i)$ , do as follows:

2.1. Let  $\mathbf{y}_i$  be the subset of variables  $x_l \in \mathbf{x}_i$  that appear in some atom in  $q_{Ai}$  (i.e., of a relational symbol in  $\mathcal{A}$ ) and in the atom  $r_i(\mathbf{t}_i)$  as a variable  $x_l \in \mathbf{t}_i$  (i.e., when  $r_i(\dots, x_l, \dots)$ ). If  $\mathbf{y}_i$  is not empty then we add the implication  $q_{Ai}(\mathbf{x}_i) \Rightarrow r_{q_i}(\mathbf{y}_i)$  in  $S_E$ , by introducing a new fresh symbol  $r_{q_i}$ ; otherwise we add the implication  $q_{Ai}(\mathbf{x}_i) \Rightarrow r_\emptyset$  if  $\mathbf{y}_i$  in  $S_E$ .

- 2.2. We add the implication  $(r_{q_i}(\mathbf{y}_i) \wedge \psi'_i(\mathbf{x}_i)) \Rightarrow r_i(\mathbf{t}_i)$  (or  $(r_{\emptyset} \wedge \psi'_i(\mathbf{x}_i)) \Rightarrow r_i(\mathbf{t}_i)$  if  $\mathbf{y}_i$  is empty) in  $S_M$ , where  $\psi'_i(\mathbf{x}_i)$  is the formula obtained from  $q_{Ai}(\mathbf{x}_i)$  by substituting each atom  $r(\mathbf{t})$  in it (where  $\mathbf{t}$  is a tuple of terms with variables in  $\mathbf{x}_i$ ) by a logically equivalent atom  $(f_r(\mathbf{t}) \doteq \bar{1})$ , where  $f_r \notin \mathbf{f}$  is a new fresh introduced symbol for the *characteristic function* of  $r$  (hence,  $\psi'_i(\mathbf{x}_i)$  and  $q_{Ai}(\mathbf{x}_i)$  are logically equivalent).
3. (Construct SOTgd  $\Phi_E$  and  $\Phi_M$ )
- 3.1. If  $S_M = \{\chi_1, \dots, \chi_m\}$  and  $\chi_1, \dots, \chi_m$  are the implications from the previous step then  $\Phi_M$  is a singleton set composed of an SOTgd

$$\exists \mathbf{h} (\forall \mathbf{x}_1 \chi_1 \wedge \dots \wedge \forall \mathbf{x}_m \chi_m)$$

where  $\mathbf{h} \supseteq \mathbf{f}$  is the tuple of all functional symbols that appear in any of the implications in  $S_M$  and the variables in  $\mathbf{x}_i$  are all the variables found in the formula  $\chi_i$ , for  $1 \leq i \leq m$ .

- 3.2. If  $S_E = \{\bar{\chi}_1, \dots, \bar{\chi}_m\}$  and  $\bar{\chi}_1, \dots, \bar{\chi}_m$  are the implications from the previous step then  $\Phi_E$  is a singleton set composed of an SOTgd

$$\exists \mathbf{g} (\forall \mathbf{z}_1 \chi_1 \wedge \dots \wedge \forall \mathbf{z}_m \chi_m)$$

where  $\mathbf{g} \subseteq \mathbf{f}$  is the tuple of all functional symbols that appear in any of the implications in  $S_E$  and the variables in  $\mathbf{z}_i$  are all the variables found in the formula  $\bar{\chi}_i$ , for  $1 \leq i \leq m$ .

**Return** the pair of SOTgds  $(\Phi_M, \Phi_E)$ .

It is easy to verify that for a mapping  $\mathcal{M}_{AB} = \{\Phi\} : \mathcal{A} \rightarrow \mathcal{B}$ , with this decomposition  $(\Phi_M, \Phi_E) = DeCompose(\mathcal{M}_{AB})$ , we obtain two mappings,  $\mathcal{M}_{AC} = \{\Phi_E\} : \mathcal{A} \rightarrow \mathcal{C}$  and  $\mathcal{M}_{CB} = \{\Phi_M\} : \mathcal{C} \rightarrow \mathcal{B}$ , where  $\mathcal{C} = (S_C, \emptyset)$  and  $S_C$  is the set of all new introduced relational symbols (that appear on the right-hand side of implications in  $\Phi_E$  and on the left-hand side of implications in  $\Phi_M$ ), such that  $\mathcal{M}_{AB} = \mathcal{M}_{CB} \circ \mathcal{M}_{AC} = Compose(\mathcal{M}_{CB}, \mathcal{M}_{AC})$  (from the fact that for each pair of implications  $\chi_i$  and  $\bar{\chi}_i$ , equal to  $q_{Ai}(\mathbf{x}_i) \Rightarrow r_{q_i}(\mathbf{y}_i)$  and  $(r_{q_i}(\mathbf{y}_i) \wedge \psi'_i(\mathbf{x}_i)) \Rightarrow r_i(\mathbf{t}_i)$ , respectively,  $\psi'_i(\mathbf{x}_i)$  is logically equivalent to  $q_{Ai}(\mathbf{x}_i)$ ).

*Example 16* Let us consider the following three cases:

1. The mapping  $\mathcal{M}'_{AD} = \{\Phi\} : \mathcal{A} \rightarrow \mathcal{D}$  in Example 5 with  $\Phi$  equal to SOTgd

$$\exists f_1 (\forall x_1 \forall x_2 (\text{Takes}(x_1, x_2) \Rightarrow \text{Enrolment}(f_1(x_1), x_2))).$$

Then  $\Phi_E$  is equal to  $\forall x_1 \forall x_2 (\text{Takes}(x_1, x_2) \Rightarrow r_{q_1}(x_2))$ , and  $\Phi_M$  is equal to

$$\begin{aligned} \exists f_1 \exists f_{\text{Takes}} (\forall x_1 \forall x_2 ((r_{q_1}(x_2) \\ \wedge (f_{\text{Takes}}(x_1, x_2) \doteq \bar{1})) \Rightarrow \text{Enrolment}(f_1(x_1), x_2))), \end{aligned}$$

where  $f_{\text{Takes}}$  is a new functional symbol introduced in step 2 of the algorithm *DeCompose* because the variable  $x_1$  on the right-hand side of the implication is not contained in the atom  $r_{q_1}(x_2)$  and, consequently, we replaced the original



atom  $\text{Takes}(x_1, x_2)$  with the equivalent to it equation  $(f_{\text{Takes}}(x_1, x_2) \doteq \bar{1})$ , because the relational symbol  $\text{Takes}$  is of schema  $\mathcal{A}$  and not of the new schema  $\mathcal{C} = (\{r_{q_1}\}, \emptyset)$  with a unary relational symbol  $r_{q_1}$ . Let us define these two mappings,  $\mathcal{M}_{AC} = \{\Phi_E\} : \mathcal{A} \rightarrow \mathcal{C}$  and  $\mathcal{M}_{CD} = \{\Phi_M\} : \mathcal{C} \rightarrow \mathcal{D}$ .

For any mapping-interpretation  $\alpha$  such that  $A = \alpha^*(\mathcal{A})$  and both schema mappings  $\mathcal{M}_{AC}$  and  $\mathcal{M}'_{AD}$  are satisfied, we have that

$$\text{Flux}(\alpha, \text{MakeOperads}(\mathcal{M}_{AC})) = \text{Flux}(\alpha, \text{MakeOperads}(\mathcal{M}'_{AD})).$$

If for such an interpretation (R-algebra)  $\alpha$ ,

$$\alpha(r_{q_1}) = \pi_{x_2}(\|\text{Takes}(x_1, x_2)\|_A) = \pi_{x_2}(\alpha(\text{Takes}))$$

then also

$$\text{Flux}(\alpha, \text{MakeOperads}(\mathcal{M}_{CD})) = \text{Flux}(\alpha, \text{MakeOperads}(\mathcal{M}'_{AD})).$$

2. The mapping  $\mathcal{M}'_{AD} = \{\Phi\} : \mathcal{A} \rightarrow \mathcal{D}$  in Example 5 with  $\Phi$  equal to SOTgd

$$\begin{aligned} & \exists f_{\text{Student}}(\forall x_1 \forall x_2 \forall y_3 ((\text{Takes}(x_1, x_2) \\ & \quad \wedge (f_{\text{Student}}(x_1, y_3) \doteq \bar{1})) \Rightarrow \text{Enrolment}(y_3, x_2))). \end{aligned}$$

Then  $\Phi_E$  is equal to

$$\begin{aligned} & \exists f_{\text{Student}}(\forall x_1 \forall x_2 \forall y_3 ((\text{Takes}(x_1, x_2) \\ & \quad \wedge (f_{\text{Student}}(x_1, y_3) \doteq \bar{1})) \Rightarrow r_{q_2}(y_3, x_2))) \end{aligned}$$

and  $\Phi_M$  is equal to

$$\begin{aligned} & \exists f_{\text{Student}} \exists f_{\text{Takes}}(\forall x_1 \forall x_2 \forall y_3 ((r_{q_2}(y_3, x_2)) \wedge (f_{\text{Takes}}(x_1, y_2) \doteq \bar{1}) \\ & \quad \wedge (f_{\text{Student}}(x_1, y_3) \doteq \bar{1})) \Rightarrow \text{Enrolment}(y_3, x_2)), \end{aligned}$$

with a new schema  $\mathcal{C} = (\{r_{q_2}\}, \emptyset)$  composed of a binary relational symbol  $r_{q_2}$ .

Let us define these two mappings,  $\mathcal{M}_{AC} = \{\Phi_E\} : \mathcal{A} \rightarrow \mathcal{C}$  and  $\mathcal{M}_{CD} = \{\Phi_M\} : \mathcal{C} \rightarrow \mathcal{D}$ .

For any mapping-interpretation  $\alpha$  such that  $A = \alpha^*(\mathcal{A})$  and both schema mappings  $\mathcal{M}_{AC}$  and  $\mathcal{M}'_{AD}$  are satisfied, we have that

$$\text{Flux}(\alpha, \text{MakeOperads}(\mathcal{M}_{AC})) = \text{Flux}(\alpha, \text{MakeOperads}(\mathcal{M}'_{AD})).$$

If for such an interpretation (R-algebra)  $\alpha$ ,

$$\alpha(r_{q_2}) = \pi_{x_2, y_3}(\|\text{Takes}(x_1, x_2) \wedge (f_{\text{Student}}(x_1, y_3) \doteq \bar{1})\|_A)$$

then also

$$\text{Flux}(\alpha, \text{MakeOperads}(\mathcal{M}_{CD})) = \text{Flux}(\alpha, \text{MakeOperads}(\mathcal{M}'_{AD})).$$

3. The mapping  $\mathcal{M}_{AC} = \{\Phi\} : \mathcal{A} \rightarrow \mathcal{C}$  in Example 9 with  $\Phi$  equal to SOTgd

$$\begin{aligned} & \exists f_1 \exists f_{Emp} \exists f_{Over65} ( \\ & \forall x_e (((f_{Emp}(x_e) \doteq \bar{1}) \wedge \text{Local}(x_e)) \Rightarrow \text{Office}(x_e, f_1(x_e)) \\ & \wedge \forall x_e (((f_{Emp}(x_e) \doteq \bar{1}) \wedge (f_{Over65}(x_e) \doteq \bar{1})) \Rightarrow \text{CanRetire}(x_e))))). \end{aligned}$$

Then  $\Phi_E$  is equal to

$$\begin{aligned} & \exists f_{Emp} \exists f_{Over65} ( \\ & \forall x_e (((f_{Emp}(x_e) \doteq \bar{1}) \wedge \text{Local}(x_e)) \Rightarrow r_{q_3}(x_e)) \\ & \wedge \forall x_e (((f_{Emp}(x_e) \doteq \bar{1}) \wedge (f_{Over65}(x_e) \doteq \bar{1})) \Rightarrow r_{q_4}(x_e))) \end{aligned}$$

and  $\Phi_M$  is equal to

$$\begin{aligned} & \exists f_1 \exists f_{Emp} \exists f_{Over65} \exists f_{Local} ( \\ & \forall x_e ((r_{q_3}(x_e) \wedge (f_{Emp}(x_e) \doteq \bar{1}) \wedge (f_{Local}(x_e) \doteq \bar{1})) \Rightarrow \text{Office}(x_e, f_1(x_e)) \\ & \wedge \forall x_e ((r_{q_4}(x_e) \wedge (f_{Emp}(x_e) \doteq \bar{1}) \wedge (f_{Over65}(x_e) \doteq \bar{1})) \Rightarrow \text{CanRetire}(x_e))))), \end{aligned}$$

with a new schema  $\mathcal{C}' = (\{r_{q_3}, r_{q_4}\}, \emptyset)$  composed of two unary relational symbols  $r_{q_3}$  and  $r_{q_4}$ .

Lets us define these two mappings,  $\mathcal{M}_{AC'} = \{\Phi_E\} : \mathcal{A} \rightarrow \mathcal{C}'$  and  $\mathcal{M}_{C'C} = \{\Phi_M\} : \mathcal{C}' \rightarrow \mathcal{C}$ .

For any mapping-interpretation  $\alpha$  such that  $A = \alpha^*(\mathcal{A})$  and both schema mappings  $\mathcal{M}_{AC}$  and  $\mathcal{M}_{AC'}$  are satisfied, we have that

$$\text{Flux}(\alpha, \text{MakeOperads}(\mathcal{M}_{AC'})) = \text{Flux}(\alpha, \text{MakeOperads}(\mathcal{M}_{AC})).$$

If for such an interpretation (R-algebra)  $\alpha$ ,

$$\alpha(r_{q_3}) = \pi_{x_e} (\| (f_{Emp}(x_e) \doteq \bar{1}) \wedge \text{Local}(x_e) \|_A) = \alpha(\text{Emp}) \cap \alpha(\text{Local})$$

and

$$\begin{aligned} \alpha(r_{q_4}) &= \pi_{x_e} (\| (f_{Emp}(x_e) \doteq \bar{1}) \wedge (f_{Over65}(x_e) \doteq \bar{1}) \|_A) \\ &= \alpha(\text{Emp}) \cap \alpha(\text{Over65}) \end{aligned}$$

then also

$$\text{Flux}(\alpha, \text{MakeOperads}(\mathcal{M}_{C'C})) = \text{Flux}(\alpha, \text{MakeOperads}(\mathcal{M}_{AC})).$$

Based on this schema-mapping decomposition, we can obtain the instance-mapping decomposition as well:

**Proposition 3** For each schema mapping  $\mathcal{M}_{AB} : \mathcal{A} \rightarrow \mathcal{B}$  with

$$(\Phi_E, \Phi_M) = \text{DeCompose}(\mathcal{M}_{AB}),$$

let us define the mapping-operads

$$\mathbf{M}_E = \text{MakeOperads}(\{\Phi_E\}) \quad \text{and} \quad \mathbf{M}_M = \text{MakeOperads}(\{\Phi_M\}).$$

Let  $S = \{\chi_1, \dots, \chi_m\}$  be the set of implications of the normalized SOTgd of  $\mathcal{M}_{AB}$ . If for a mapping-interpretation  $\alpha$  (with  $A = \alpha^*(\mathcal{A})$  and  $B = \alpha^*(\mathcal{B})$ ) such that for each implication  $\chi_i \in S$ ,  $q_{Ai}(\mathbf{x}_i) \Rightarrow r_i(\mathbf{t}_i)$ , and its derived corresponding implications  $q_{Ai}(\mathbf{x}_i) \Rightarrow r_{q_i}(\mathbf{y}_i)$  in  $S_E$  and  $(r_{q_i}(\mathbf{y}_i) \wedge \psi_i(\mathbf{x}_i)) \Rightarrow r_i(\mathbf{t}_i)$  in  $S_M$  (in the algorithm *Decompose*), we have  $\alpha(r_{q_i}) = \pi_{\mathbf{y}_i} \| q_{Ai}(\mathbf{x}_i) \|_A$ , then:

$$\text{Flux}(\alpha, \text{MakeOperads}(\mathcal{M}_{AB})) = \text{Flux}(\alpha, \mathbf{M}_E) \cap \text{Flux}(\alpha, \mathbf{M}_M).$$

For such a mapping-interpretation  $\alpha$  with the instance-mapping

$$f = \alpha^*(\text{MakeOperads}(\mathcal{M}_{AB})) : A \rightarrow B,$$

we denote the decomposition of  $f$  by two instance-mappings  $ep(f) = \alpha^*(\mathbf{M}_E) : A \rightarrow TC$  and  $in(f) = \alpha^*(\mathbf{M}_M) : TC \rightarrow B$ , with schema  $\mathcal{C} = (S_C, \emptyset)$  and  $S_C = \bigcup_{q'_i \in \mathbf{M}_E} \partial_1(q'_i) = \{r_{q_1}, \dots, r_{q_m}\}$  and  $C = \alpha^*(S_C)$ .

*Proof* Let  $\mathbf{M}_{AB} = \text{MakeOperads}(\mathcal{M}_{AB}) = \{q_1, \dots, q_m, 1_{r_\emptyset}\}$  with  $q_i = v_i \cdot q_{A,i} \in O(r_{i,1}, \dots, r_{i,k}, r_i^B)$ , for  $1 \leq i \leq m$ , so that  $\text{MakeOperads}(\Phi_E) = \{q'_1, \dots, q'_m, 1_{r_\emptyset}\}$ , with  $q'_i = v'_i \cdot q'_{A,i} \in O(r_{i,1}, \dots, r_{i,k}, r_{q_i})$ , for  $1 \leq i \leq m$ , and  $\text{MakeOperads}(\Phi_M) = \{q''_1, \dots, q''_m, 1_{r_\emptyset}\}$  with  $q''_i \in O(r_{q_i}, r_i^B)$ .

First of all, from the fact that  $\alpha(r_{q_i}) = \pi_{\mathbf{y}_i} \| q_{Ai}(\mathbf{x}_i) \|_A$ , for  $1 \leq i \leq m$ , it follows, from Definition 11 for a mapping-interpretation, that (a.1.) the functions  $\alpha(v_i)$  and  $\alpha(v'_i)$  are equal. Then, from Definition 13,

- (i)  $\text{Flux}(\alpha, \mathbf{M}_E) = \text{Flux}(\alpha, \mathbf{M}_{AB}) = T\{\pi_{\mathbf{y}_i} \| e[(\_)_j / r_{i,j}]_{1 \leq j \leq k} \|_A \mid q'_i = v'_i \cdot q'_{A,i} = (e \Rightarrow (\_) (\mathbf{y}_i)) \in \mathbf{M}_E, q'_i \in O(r_{i,1}, \dots, r_{i,k}, r_{q_i}) \text{ and } \mathbf{y}_i \neq \emptyset\} = T\{\pi_{\mathbf{y}_i} \| q_{A,i}(\mathbf{x}_i) \|_A \mid 1 \leq i \leq m \text{ and } \mathbf{y}_i \neq \emptyset\} = T\{\alpha(r_{q_i}) \mid 1 \leq i \leq m \text{ and } \mathbf{y}_i \neq \emptyset\} = TC$  (because  $\alpha(v'_i)$  is an identity function and hence an *injection* as well, for  $1 \leq i \leq m$  such that  $\mathbf{y}_i \neq \emptyset$ ).
- (ii)  $\text{Flux}(\alpha, \mathbf{M}_M) = T\{\pi_{\mathbf{y}_i} \| e[(\_)_1 / r_{q_i}] \|_C \mid q''_i = v''_i \cdot q''_{C,i} = (e \Rightarrow (\_) (\mathbf{t}_i)) \in \mathbf{M}_M, q''_i \in O(r_{q_i}, r_i^B) \text{ and } \mathbf{y}_i \neq \emptyset\} = T\{\pi_{\mathbf{y}_i} \| r_{q_i}(\mathbf{y}_i) \wedge \psi_i(\mathbf{x}_i) \|_C \mid 1 \leq i \leq m \text{ and } \mathbf{y}_i \neq \emptyset\} = T\{\pi_{\mathbf{y}_i} \| r_{q_i}(\mathbf{y}_i) \|_C \mid 1 \leq i \leq m \text{ and } \mathbf{y}_i \neq \emptyset\} =$   
(since  $\psi_i(\mathbf{d})$  is true when  $q_{A,i}(\mathbf{d})$  is true, and  $q_{A,i}(\mathbf{x}_i) \Rightarrow r_{q_i}(\mathbf{y}_i)$  is satisfied by  $\alpha$ )  
 $= T\{\alpha(r_{q_i}) \mid 1 \leq i \leq m \text{ and } \mathbf{y}_i \neq \emptyset\}$  if  $\alpha(v''_i)$  is an injection for all  $1 \leq i \leq m$  such that  $\mathbf{y}_i \neq \emptyset$ ;  $\perp^0$  otherwise,  
 $=$  (from (a.1.))  $= T\{\alpha(r_{q_i}) \mid 1 \leq i \leq m \text{ and } \mathbf{y}_i \neq \emptyset\}$  if  $\alpha(v_i)$  (equal to  $\alpha(v''_i)$ ) is an injection for all  $1 \leq i \leq m$  such that  $\mathbf{y}_i \neq \emptyset$ ;  $\perp^0$  otherwise,  
 $= T\{\pi_{\mathbf{y}_i} \| q_{Ai}(\mathbf{x}_i) \|_A \mid 1 \leq i \leq m \text{ and } \mathbf{y}_i \neq \emptyset\}$  if  $\alpha(v_i)$  is an injection for all

$1 \leq i \leq m$  such that  $\mathbf{y}_i \neq \emptyset$ ;  $\perp^0$  otherwise,  
 $= T\{\pi_{\mathbf{x}_i} \| e[(\_)_j / r_{i,j}]_{1 \leq j \leq k} \| \mathbf{q}_i \mid \mathbf{q}_i = v_i \cdot q_{A,i} = (e \Rightarrow (\_)_i(\mathbf{t}_i)) \in \mathbf{M}_{AB}, q_i \in$   
 $O(r_{i,1}, \dots, r_{i,k}, r_i^B) \text{ and } \mathbf{y}_i \neq \emptyset\}$ , if  $\alpha(v_i)$  is an injection for all  $1 \leq i \leq m$   
such that  $\mathbf{y}_i \neq \emptyset$ ;  $\perp^0$  otherwise,  
 $= Flux(\alpha, \mathbf{M}_{AB})$ .

Thus, from (i) and (ii),  $Flux(\alpha, \mathbf{M}_E) \cap Flux(\alpha, \mathbf{M}_M) = Flux(\alpha, \mathbf{M}_{AB})$ .

The instance mappings  $in(f)$  and  $ep(f)$  are well defined because  $C \subseteq TC$  (i.e., all relations in  $C$  are the relations in  $TC$  as well).  $\square$

In Example 16, we considered the decomposition of a mapping between two database schemas. Let us now consider the case of a decomposition of the integrity-constraint mapping for a given schema:

*Example 17* Let us consider a schema  $\mathcal{A} = (S_A, \Sigma_A)$  with the integrity constraints in  $\Sigma_A = \Sigma_A^{\text{egd}} \cup \Sigma_A^{\text{tgd}}$ , which can be represented (see Example 12) by a schema mapping  $\top_{AA\top} = \{\Phi \wedge \Psi\} : \mathcal{A} \rightarrow \mathcal{A}_\top$ , where  $\Phi$  is the SOTgd equal to  $EgdsToSOTgd(\Sigma_A^{\text{egd}})$  and  $\Psi$  is the SOTgd equal to  $TgdsToConSOTgd(\Sigma_A^{\text{tgd}})$  and  $\mathcal{A}_\top = (\{r_\top\}, \emptyset)$  is an auxiliary schema, and, consequently, by equivalent operads-mapping

$$\mathbf{T}_{AA\top} = MakeOperads(\{\Phi \wedge \Psi\}) : \mathcal{A} \rightarrow \mathcal{A}_\top$$

with the *empty* information flux.

That is, from Corollary 6, for each mapping-interpretation  $\alpha$ ,  $Flux(\alpha, \mathbf{T}_{AA\top}) = \perp^0$ , because each operad's operation  $q_i \in \mathbf{T}_{AA\top}$  has a form  $e \Rightarrow (\_) (\bar{0}, \bar{1})$  without free variables on the right-hand side of this implication and hence, from Definition 13,  $Var(\mathbf{T}_{AA\top}) = \emptyset$ .

Thus, from the fact that  $Var(\mathbf{T}_{AA\top}) = \emptyset$  and the Decomposition algorithm, we obtain that  $(\Phi_E, \Phi_M) = DeCompose(\top_{AA\top})$  where  $\Phi_E$  is composed of implications  $q_{Ai}(\mathbf{x}_i) \Rightarrow r_\emptyset$  and  $\Phi_M$  is composed of implications  $(r_\emptyset \wedge \psi_{Ai}(\mathbf{x}_i)) \Rightarrow r_\top(\bar{0}, \bar{1})$ .

Thus, from the fact that  $r_\emptyset$  is a tautology (truth propositional letter, Definitions 1 and 8), each implication in  $\Phi_E$ ,  $q_{Ai}(\mathbf{x}_i) \Rightarrow r_\emptyset$  is a tautology and hence  $\Phi_E$  is a tautology as well, so that we can substitute it by a trivial tautology  $r_\emptyset \Rightarrow r_\emptyset$ .

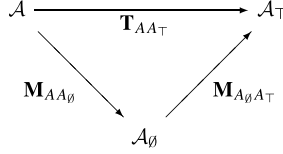
From the fact that  $r_\emptyset$  is a tautology, each implication  $(r_\emptyset \wedge \psi_{Ai}(\mathbf{x}_i)) \Rightarrow r_\top(\bar{0}, \bar{1})$  is equivalent to the implication  $\psi_{Ai}(\mathbf{x}_i) \Rightarrow r_\top(\bar{0}, \bar{1})$ , that is, to the implication  $q_{Ai}(\mathbf{x}_i) \Rightarrow r_\top(\bar{0}, \bar{1})$  (from the fact that  $q_{Ai}(\mathbf{x}_i)$  is logically equivalent to  $\psi_{Ai}(\mathbf{x}_i)$ ). Consequently,  $\Phi_M$  is equal to the normalized  $\Phi$ , so that SOTgd  $\Phi_M$  and the original SOTgd  $\Phi$  are logically equivalent.

Consequently,  $\Phi_E \wedge \Phi_M$  is logically equivalent to the original SOTgd  $\Phi$ .

Thus, we obtain a trivial mapping (see Example 7 for the trivial mappings with the empty database schema  $\mathcal{A}_\emptyset$ )

$$\mathcal{M}_{AA_\emptyset} = \{\Phi_E\} = \{r_\emptyset \Rightarrow r_\emptyset\} : \mathcal{A} \rightarrow \mathcal{A}_\emptyset \quad \text{and} \quad \mathcal{M}_{A_\emptyset A_\top} = \{\Phi_M\} : \mathcal{A}_\emptyset \rightarrow \mathcal{A}_\top.$$

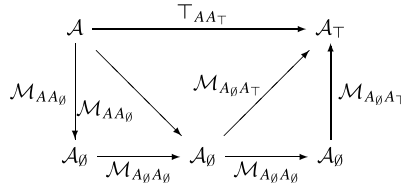
Consequently, the obtained mapping-operads are  $\mathbf{M}_{AA_\emptyset} = MakeOperads(\{\Phi_E\}) = \{1_{r_\emptyset}\} : \mathcal{A} \rightarrow \mathcal{A}_\emptyset$ , and  $\mathbf{M}_{A_\emptyset A_\top} = MakeOperads(\{\Phi_M\}) : \mathcal{A}_\emptyset \rightarrow \mathcal{A}_\top$ . This decomposition can be represented by the following commutative diagram



The commutativity  $\mathbf{T}_{\mathcal{A}\mathcal{A}_\top} = \mathbf{M}_{\mathcal{A}_\emptyset\mathcal{A}_\top} \circ \mathbf{M}_{\mathcal{A}\mathcal{A}_\emptyset}$  comes from the fact that  $\mathbf{T}_{\mathcal{A}\mathcal{A}_\top} = \mathcal{M}_{\mathcal{A}_\emptyset\mathcal{A}_\top} \circ \mathcal{M}_{\mathcal{A}\mathcal{A}_\emptyset} = \text{Compose}(\mathcal{M}_{\mathcal{A}_\emptyset\mathcal{A}_\top}, \mathcal{M}_{\mathcal{A}\mathcal{A}_\emptyset})$  and, in this case, from the fact that  $\Phi$  is logically equivalent to  $\Phi_M$  and to  $\Phi_E \wedge \Phi_M$ .

Note that for both obtained mappings  $\text{Var}(\mathbf{M}_{\mathcal{A}\mathcal{A}_\emptyset}) = \text{Var}(\mathbf{M}_{\mathcal{A}_\emptyset\mathcal{A}_\top}) = \emptyset$ , and hence they have also empty information fluxes for each mapping-interpretation  $\alpha$ . That is,  $\text{Flux}(\alpha, \mathbf{M}_{\mathcal{A}\mathcal{A}_\emptyset}) = \text{Flux}(\alpha, \mathbf{M}_{\mathcal{A}_\emptyset\mathcal{A}_\top}) = \perp^0$ .

The decomposition of any trivial mapping (in Example 7)  $\mathcal{M} = \{r_\emptyset \Rightarrow r_\emptyset\}$  will produce the pair of two trivial mappings as well, that is, we will not produce new mappings by their decomposition. If we apply the decomposition to the graph in the previous example, by using the original mappings instead of derived mapping-operators, we will obtain the following graph:



where  $\mathcal{M}_{\mathcal{A}_\emptyset\mathcal{A}_\emptyset} = \text{Id}_{\mathcal{A}_\emptyset} = \{r_\emptyset \Rightarrow r_\emptyset\} : \mathcal{A}_\emptyset \rightarrow \mathcal{A}_\emptyset$  is the identity mapping for the empty schema  $\mathcal{A}_\emptyset$  (see Example 7).

## 2.6 Database Schema Mapping Graphs

In this section, we will summarize all the results of this chapter by a formal translation of the logic theory, of a given database mapping system, into a graph's formalism which represents a categorification provided in the next two chapters, that is, a translation of the logic schema-mapping theory into a small sketch category with a functorial semantics for its models.

We recall that any tuple-generating *mapping* from a schema  $\mathcal{A}$  into a schema  $\mathcal{B}$  is represented by a set of tgds  $\forall \mathbf{x}(\phi_A(\mathbf{x}) \Rightarrow \exists \mathbf{z}(\psi_B(\mathbf{y}, \mathbf{z})))$  where  $\mathbf{y} \subseteq \mathbf{x}$ , by taking out the universal quantification  $\forall \mathbf{x}$  from the head of this tgd logical sentence, that is, by the *view mapping*  $q_A(\mathbf{y}) \Rightarrow q_B(\mathbf{y})$  where  $q_A(\mathbf{y})$  (equivalent to  $\exists \mathbf{y}_1 \phi_A(\mathbf{y}, \mathbf{y}_1)$  with  $\mathbf{x}$  equal to the tuple of all variables in  $\mathbf{y}$  and  $\mathbf{y}_1$ ) is a conjunctive query over the schema  $\mathcal{A}$  and  $q_B(\mathbf{y})$  (equivalent to  $\exists \mathbf{z} \psi_B(\mathbf{y}, \mathbf{z})$ ) is a conjunctive query over the schema  $\mathcal{B}$ . These view-based mappings are interpreted as follows:

- When a sentence  $q_A(\mathbf{d})$  is true for a tuple of values  $\mathbf{d}$  then  $q_B(\mathbf{d})$  has to be a true sentence as well, so that the information  $\mathbf{d}$  of an instance-database  $A$  (of the schema  $\mathcal{A}$ ) is “transferred” by this mapping into the instance-database  $B$  (of the schema  $\mathcal{B}$ ).

We have demonstrated that a set of tgds of a given inter-schema mapping can be equivalently represented by a single SOTgd (by the algorithm *TgdsToSOTgd*). A relational database schema  $\mathcal{A}$  is generally specified by a pair  $(S_A, \Sigma_A)$  where  $S_A$  is a set of  $n$ -ary relational symbols,  $\Sigma_A = \Sigma_A^{\text{tgd}} \cup \Sigma_A^{\text{egd}}$  with the set of the database integrity constraints  $\Sigma_A^{\text{egd}}$  expressed by *equality-generating dependencies* (egds) and the set of the *tuple-generating dependencies* (tgds)  $\Sigma_A^{\text{tgd}}$  in Definition 2.

Any integrity constraint (egd)  $\forall \mathbf{x}(\phi_A(\mathbf{x}) \Rightarrow (\mathbf{y} \doteq \mathbf{z}))$  of a given schema database  $\mathcal{A}$ , with  $\mathbf{y} = \langle y_1, \dots, y_k \rangle \subseteq \mathbf{x}$  and  $\mathbf{z} = \langle z_1, \dots, z_k \rangle \subseteq \mathbf{x}$ , will be represented (Lemma 3) by the new mapping  $\forall \mathbf{x}((\phi_A(\mathbf{x}) \wedge (\mathbf{y} \neq \mathbf{z})) \Rightarrow r_{\top}(\bar{0}, \bar{1}))$  (from Lemma 3), where  $r_{\top}$  is the built-in binary relational symbol for equality of FOL. The interpretation of this mapping is the same as for the standard inter-schema mappings:

When  $\phi_A(\mathbf{x}) \wedge (\mathbf{y} \neq \mathbf{z})$  is true for a tuple of values  $\mathbf{d}$ , from the fact that the ground atom  $r_{\top}(\bar{0}, \bar{1})$  is a false, it holds that the mapping  $(\phi_A(\mathbf{x}) \wedge (\mathbf{y} \neq \mathbf{z})) \Rightarrow r_{\top}(\bar{0}, \bar{1})$  is not satisfied. It is easy to see that if the instance-database  $A$  is a *model* of the schema  $\mathcal{A}$  (i.e., when the integrity constraints expressed by the corresponding egds are satisfied) then  $q_A(\mathbf{x}) \wedge (\mathbf{y} \neq \mathbf{z})$  cannot be satisfied. The “transferred” information flux by this mapping (from  $A$  into the built-in relational symbol  $r_{\top}$ ) is always empty (i.e., equal to the empty database  $\perp^0 = \{\perp\}$ ).

Analogously, any normalized tgd of an integrity constraint  $\forall \mathbf{x}(\phi_A(\mathbf{x}) \Rightarrow r(\mathbf{t}))$  where  $\mathbf{t}$  is a tuple of terms with variables in  $\mathbf{x}$  and  $r$  is a relational symbol of schema  $\mathcal{A}$ , can be equivalently represented (Lemma 2) by the formula  $\forall \mathbf{x}((\phi_A(\mathbf{x}) \wedge \neg r(\mathbf{t})) \Rightarrow r_{\top}(\bar{0}, \bar{1}))$ . It is easy to see that if the instance-database  $A$  is a *model* of the schema  $\mathcal{A}$  (i.e., when the integrity constraints expressed by the corresponding tgds are satisfied) then  $q_A(\mathbf{x}) \wedge r(\mathbf{t})$  cannot be satisfied. The “transferred” information flux by this mapping (from  $A$  into the built-in relational symbol  $r_{\top}$ ) is always empty (i.e., equal to the empty database  $\perp^0 = \{\perp\}$ ).

It is consistent with the representation of the integrity constraints (both egds and tgds) by the inter-schema mappings because the sentences as integrity-constraints do not transfer any data from source to target database and hence their information flux has to be empty. Note that in the case of ordinary query mappings, the minimal information flux is  $\{\perp\} = \perp^0$  as well. We have demonstrated that a set of tgds in  $\Sigma_A^{\text{tgd}}$  can be equivalently represented by a single SOTgd (by the algorithm *TgdsToConsOTgd* in Sect. 2.2.1) and that a set of egds in  $\Sigma_A^{\text{egd}}$  can be equivalently represented by a single SOTgd (by the algorithm *EgdsToSOTgd* in Sect. 2.2.2).

Moreover, in order to translate this particular second-order logic (based on SOTgds sentences) into the categorical setting, we explained how we can translate the SOTgds into the set of abstract operad’s operations that specify a mapping between database schemas and hence to use the functorial semantics for the database mappings based on R-algebras for operads (provided in the previous Sect. 2.4). Based on this translation into operads, we have seen that each operad’s operation  $q_i \in O(r_1, \dots, r_m, r)$  obtained from an *atomic* mapping (Definition 7) is an algebraic specification for an implication conjunct in a normalized SOTgd.

Based on these considerations and Example 12 for the integrity constraints, we will formally define a graph used to specify a database schema-mapping system:

**Definition 14** For a given set  $S$  of non-empty database schemas, we define the graph  $G = (V_G, E_G)$  with  $S \subseteq V_G$  as follows:

1. For each database schema  $\mathcal{A} = (S_A, \Sigma_A)$  where  $\Sigma_A = \Sigma_A^{\text{tgd}} \cup \Sigma_A^{\text{egd}}$  is not empty, we define the mapping edge  $\top_{AA\top} = \{\Phi\} : \mathcal{A} \rightarrow \mathcal{A}_\top$  in  $E_G$  where the database schema  $\mathcal{A}_\top = (\{r_\top\}, \emptyset)$  is defined as a new distinct vertex in  $V_G$  and  $\Phi$  is defined as follows:
  - If  $\Sigma_A^{\text{egd}}$  is an empty set of egds then we define  $\Phi$  to be

$$TgdsToConSOtgd(\Sigma_A^{\text{tgd}});$$

- If  $\Sigma_A^{\text{tgd}}$  is an empty set of tgds then we define  $\Phi$  to be  $EgdsToSOtgd(\Sigma_A^{\text{egd}})$ ;
- Otherwise we define  $\Phi$  to be

$$EgdsToSOtgd(\Sigma_A^{\text{egd}}) \wedge TgdsToConSOtgd(\Sigma_A^{\text{tgd}}).$$

2. For each inter-schema mapping defined by a non-empty set  $S$  of tgds, between two database schemas  $\mathcal{A} = (S_A, \Sigma_A)$  and  $\mathcal{B} = (S_B, \Sigma_B)$ , we define the edge  $\mathcal{M}_{AB} = \{\Phi\} : \mathcal{A} \rightarrow \mathcal{B}$  in  $E_G$  where  $\Phi$  is the SOtgd obtained from this set of tgds by  $TgdsToSOtgd(S)$ .

All defined edges in  $E_G$  of this graph  $G$  will be called *atomic* mappings.

Note that a mapping graph is a necessary step in order to define a small sketch category, as it follows from Corollary 2, and to make the full embedding of the database mapping system in the Categorical logic, based on functors (i.e., R-algebras) from such a sketch category into the denotational (instance database's) category. In fact, every directed graph can be transformed into a category as follows: the objects are the vertices of the graph and the arrows are paths in the graph.

Thus, the first next step is to examine if for this denotational database category we can use the standard **Set** category or, instead, we need a more appropriate one.

---

## 2.7 Review Questions

1. Can you explain the semantic difference between the ordinary tgds and *full* tgds? Why do we need a subset of the Second Order Logic? Can we deal with incomplete knowledge with the 2-valued logics and without the Second Order Logic?
2. Can we also deal with the inconsistent information by this kind of SOtgds? In which way are we able to resolve the problems with the mutually-inconsistent data in the standard GLAV semantics of an integration of a number of the source databases into a unique global schema?
3. The many-valued logics are used for the paraconsistent logics as well. What is gained by using the many-valued logics in order to deal with incomplete and inconsistent information, and what are the drawbacks? Which is a minimal many-valued logic able to also deal with incomplete and inconsistent data? It

is well known that the logic negation as an antitonic truth operator, and hence cannot be used in order to obtain the solutions for the schema mappings by using standard fix-point semantics. How we can overcome such a problem by the many-valued logics where we also use incomplete (unknown) information?

4. Are the provided algorithms for transformation of logic formulae (tgds) into the algebraic (operad's based) expressions able to deal with the logic negation operators as well? Is there any example presented in this chapter where we need negation logic operators, and, if so, why?
5. Why it is important to express the integrity constraints over the database schemas as a kind of the schema mappings? What is main difference between such integrity-mappings and standard inter-schema mappings based on the tgds? Are the integrity-constraint mappings important for the composition of the mappings? If not then explain why and which "trick" was used in order to make an effective separation of them and of inter-schema mappings. Do we have other technical possibilities to obtain an analogous result and compare it with the method proposed here?
6. Why are we using the typed operads as an effective algebraic language in order to provide the "algebraization" of the SOTgds logics used for the schema mappings? What is the difference of such an operad's algebra and the standard algebraic semantics presented in the introduction? What are the main operations in the operad's algebra? Can you reformulate the operad's algebra by the ordinary semantics of algebras and their operators, in the way as it was done with the category theory in Sect. 1.5.1?
7. What is the main reason for a definition of the semantics of the schema mappings in Definition 11, based on the Tarski's FOL semantics? Does this definition extend the FOL semantics into the Second Order Semantics for the tgds? Why did we introduce the negation operators in this method, if we did not use the logic negation in the tgds? Can we use this semantics of the mappings for the tgds with the queries that are using the logic negation operator as well? Try to elaborate such an extension and show some simple example.
8. An Abstract Data-Object is a basic concept for the observational point of view where the resulting relation (a view), obtained for a given query, is seen as an observation of the information contained in a given database instance. What is the relationship between the power-view database  $TA$  obtained as the (also infinite) set of observations on a given instance database  $A$ , with the syntax of the used query-language? If we are using the SQL for the relational databases then we obtain  $A \subseteq TA$ . Is this power-view operator  $T$  a monotonic operator, and, if it is, what is the reason for it? What do we obtain if we apply the operator  $T$  to the instance database  $TA$ ?
9. What is intuitively the strict semantics of the schema mappings? Do we need it in a special case of the Data exchange setting, and if not, is it the main generalization of the Data exchange setting? Is the information flux a database as well, and can it contain the data which are not provided by the source database of a composed mapping with a number of the intermediate databases between the source and target database? If an atomic-data in an RDB database  $A$  is any



- simple value contained in one of its relations, does it belong to the database  $TA$  as a single-attribute single-tuple relation?
10. Why do we provide the definition of equality of two database mappings at the instance level only, and how does this fact generalize the Data Exchange framework and the equivalence of the logic formulae used in SOTgds? Why can the category semantics presented in the introduction be satisfied by the information flux for a given instance mapping between two instance databases? Is the empty information flux between two database instances (of a source and target schema) equivalent to the fact that there is no mapping between them? How can it be explained in the case of an integrity-constraint mapping for the database schemas?
  11. What is the relationship between the categorical symmetry applied to the database mapping systems and the algorithm for decomposition of SOTgds? Is the decomposition of the mappings with the empty information flux meaningful? Is the decomposition of the integrity-constraints mappings meaningful? Why is it useful to represent the database mapping systems by the graphs, from the user point of view and from the semantic (denotational) point of view? Why is the compositional property of the schema mappings so important, and which relationship does it have with the compositional properties of the information fluxes at the instance level of the database mappings? Is it a necessary condition in order to obtain a denotational semantics of the database mappings based on categories?

---

## References

1. J.F. Adams, *Infinite Loop Spaces* (Princeton U. Press, Princeton, 1978)
2. J.C. Baez, J. Dofan, Categorification, in *Workshop on Higher Category Theory and Physics*, March 28–30, ed. by E. Getzler, M. Kapranov (Northwestern University, Evanston, 1997)
3. J.M. Boardman, R.M. Vogt, *Homotopy Invariant Structures on Topological Spaces*. Lecture Notes in Mathematics, vol. 347 (Springer, Berlin, 1973)
4. A. Corradini, A complete calculus for equational deduction in coalgebraic specification. Report SEN-R9723, National Research Institute for Mathematics and Computer Science, Amsterdam (1997)
5. R. Fagin, P.G. Kolaitis, L. Popa, W. Tan, Composing schema mappings: second-order dependencies to the rescue. *ACM Trans. Database Syst.* **30**(4), 994–1055 (2005)
6. L. Libkin, C. Sirangelo, Data exchange and schema mappings in open and closed worlds, in *Proc. of PODS'08*, Vancouver, Canada (2008)
7. J.P. May, *Simplicial Objects in Algebraic Topology* (Van Nostrand, Princeton, 1968)

Big Data Integration Theory  
Theory and Methods of Database Mappings,  
Programming Languages, and Semantics

Majkić, Z.

2014, XX, 516 p. 170 illus., Hardcover

ISBN: 978-3-319-04155-1