

Mechanism and Algorithm for Indirect Schema Mapping Composition

Bo Wang

College of Information System and Management, National University of Defense Technology, Changsha, China
wbsteven@163.com

Bo Guo

College of Information System and Management, National University of Defense Technology, Changsha, China

Abstract—There are a large number of indirect schema mappings between peers in the network. To improve the efficiency of data exchange and queries, indirect mappings are needed to be composed. Direct mappings can be derived directly by the constraints defined between schemas, but not for indirect mappings' composition. Defined the combination operations of schema elements in indirect mappings, and gave the expression of indirect mappings. Analyzed the composition of indirect mappings, and proposed a strategy, named schema element back, to solve the problem of indirect mapping composition, and gave the indirect mapping composition generation algorithm based on such strategy. Experiments showed that indirect mapping composition can improve the efficiency of data exchange, and compared with other non-full mapping composition generation algorithms, and indirect mapping composition generated by our algorithm based on schema element back strategy can completely eliminate the infection of media schema with no reduction of the composition efficiency.

Index Terms—indirect mapping composition; combination operation; schema element back

I. INTRODUCTION

Many data management tasks, such as data translation, information integration, and database design require manipulation of database schemas and mappings between them^[1]. Schema mappings define the relationship between instances of two given schemas which can be divided into direct and indirect ones by the relationships of elements. Indirect mappings are widely existed, especially when schemas evolve, most mappings that used to be direct become indirect. In indirect mappings, elements between schemas are not directly associated, but related by some algebra operations.

Mapping composition refers to combining two mappings into a single one, which is useful for a variety of data management problems. Figure 1 shows a topology of a data sharing network, each node can be a data sources, an integrated data center or a logical mediator. When data are exchanged from node E to the integrated

schema G, instances of E should be translated to G by mapping sequences: $E \rightarrow C$, $C \rightarrow A$ and $A \rightarrow G$, and chaining mappings at run-time may be expensive because we may need to follow long and possibly redundant paths in the network. Note that different paths between a pair of nodes may yield different sets of query answers. To ensure the reliability of data exchange, we usually need to get all of the possible paths, and execute each possible data transplantation process according to the paths. Cost of performing such work will be quite large when there are a large amount of data sources. If certain nodes leave the network, then we may lose the mapping paths. Addressing these issues raises several static analysis questions regarding the network of mappings, and mapping composition lies at the core of them all. By pre-composing a select set of mapping chains in the network, we can directly execute the data exchange between the source schema and the target schema, which leads to significant run-time savings. Moreover, mapping composition arises in many practical settings like data integration, schema evolution and database designing:

In data integration, a query needs to be composed with a view definition. If the view definition is expressed using global-as-view (GAV), then this is an example of composing two functional mappings: a view definition that maps a database to a view, and a query that maps a view to a query result.

In schema evolution, a schema evolves to a new one, and the relationships between the two schemas may be described by a mapping. The original mappings on the old schema can be updated by mapping composition.

In a database design process, schema may evolve frequently via a sequence of incremental modifications. This produces a sequence of mappings between successive versions of the schema until the desired schema is reached, so a mapping from the original schema to the final one is needed, which can be obtained by composing the mappings between the successive versions of the schema. With this mapping, the designer can migrate data from the old schema to the new schema.

Manuscript received February 11, 2009; revised June 21, 2009; accepted July 21, 2009.

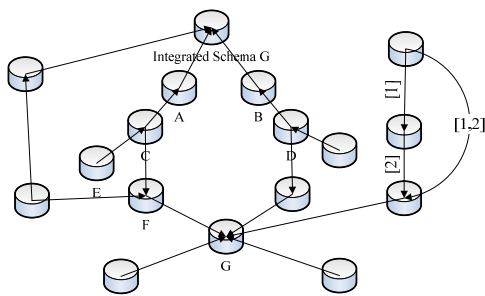


Figure 1. The topology of a data sharing network.

Another motivation for mapping composition comes from the framework of model management^[2]. One of the basic operators in model-management algebra is composition, and mappings are treated mostly as syntactic objects. Mapping management can be achieved by defining the composition and inverse operations.

Query composition is widely supported by most of the commercial data management tools, while the mapping composition is still rest on researches and experiments. If the mappings are functional, we can do the composition similarly as the query composition. However, there are a large number of non-functional mappings, most of which are indirect mappings, and we can not directly give the composition. Different from the query composition, indirect mappings can not be composed directly.

In this paper, we mainly discuss the composition problems of indirect mappings, and the rest of the paper is organized as follows. In section 2, we show the related works on mapping composition. In section 3 we introduce some concepts of mapping composition. In section 4 and 5, we firstly define the indirect mappings composition, and then propose an indirect mapping composition algorithm. Section 4 presents the experimental results. And the last part is the conclusion.

II. RELATED WORKS

Mapping composition is a challenging problem^[3] whose difficulty is quite sensitive to the expressiveness of the allowed mappings, and there are several researches on such problem^[4, 5, 6, 7, 8], including methods based on tuple generation constraints and schema transformation^[9, 10, 6], etc. Those mapping composition methods based on schema transformation define complicated schema transformation operations, which only support the computing of direct mappings, and not for those indirect ones with complex algebra operations between schema elements. Model management is a generic approach to solving problems of data programmability where precisely engineered mappings are required^[2]. A model management system supports the creation, compilation, reuse, evolution, and execution of mappings between schemas represented in a wide range of meta-models, where the composition operation is one of the ways to realize mapping reusing.

Researches on indirect mappings mostly consider the definitions and basic operations of non-direct element-to-

element mappings between the source and target schemas^[11] without any further discussions about composition. Madhavan and Haleby^[5] showed that the composition of two given mappings expressed as GLAV formulas may not be expressible in a finite set of first-order constraints. Fagin et al^[4] proved that the composition of certain kinds of first-order mappings may not be expressible in any first-order mappings, even by an infinite set of constraints, because the mapping language is not closed under composition. Nash et al^[8] showed that for certain classes of first-order languages, it is undecidable to determine whether there is a finite set of constraints in the same language that represents the composition of two given mappings. In [4], Fagin et al. demonstrated the second-order mapping language, namely second order source- to-target tuple-generation dependencies (denoted by SOTgd), is closed under composition, and they also present a composition algorithm for this language. The second-order languages uses existentially quantified function symbols, which essentially can be thought of as Skolem function. A tuple-generating dependency specifies an inclusion of two conjunctive queries, $Q_1 \subseteq Q_2$. It is called source-to-target when Q_1 refers only to symbols from the source schema and Q_2 refers only to symbols from the target schema. However, for implementation, such kind of languages is not supported by standard SQL-based database tools.

Yu and Popa extended the algorithm of Fagin's to handle nesting and apply it to some schema evolution scenarios, and they also discussed the optimizations of the composition result. Nash et al^[8] studied the composition of first-order constraints that are not necessarily source-to-target. They considered dependencies that can express key constraints and inclusions of conjunctive queries $Q_1 \subseteq Q_2$, where Q_1 and Q_2 may reference symbols from both the source and target schema, but the composition of constraints in this language is not closed and whether a composition result exists is undecidable. They also gave an algorithm that produces a composition.

Berstein et al^[3] explored the mapping composition problem for constraints that are not restricted to being source-to-target, which extended the work of Nash and Fagin. They applied a "left-compose" step which allows the algorithm to handle mappings on which the algorithm in [8] fails. They used algebra-based instead of logic-based language to express mappings, which can be directly supported by database tools. We call their method "best effort composition approach" (denoted by BECA). BECA tries to eliminate as many relation symbols from the middle schema as possible. Given schemas σ_1 , σ_2 and σ_3 , mappings computed by BECA have the form of $\sigma_1 \rightarrow \sigma'_2 \rightarrow \sigma_3$, where $\sigma'_2 \subset \sigma_2$. That means, in some cases it may be better to eliminate some symbols from σ_2 successfully, rather than insist on either

eliminating all them or failing. Since there may be some elements in the middle schema remain at the end of BECA, the result is not a perfect composition.

A main reason that σ'_2 is not empty after BECA, is that there may be indirect mappings. In this paper, we use a strategy namely *schema elements back* to solve the problem. Although this method expands $\sigma_1: \sigma_1 \rightarrow \sigma'_1$, it can completely eliminate σ_2 , and a composed mapping only depends on the source and target schema will be generated.

III. MAPPING COMPOSITION

Definition 1 Let R_1 and R_2 be two binary relations, the composition $R_1 \circ R_2$ of R_1 and R_2 is the binary relation: $R_1 \circ R_2 = \{(x, y) : (\exists z)((x, z) \in R_1 \wedge (z, y) \in R_2)\}$.

Let $M = \langle S, T, \Sigma \rangle$ be a schema mapping model, where S and T are the schemas with no relation symbols in common and Σ is a set of formulas of some logical formulas over $\langle S, T \rangle$. Then let $Inst(M)$ be a binary relation between instances over S and T . We define composition of two schema mappings M_{12} and M_{23} using the composition of the binary relations $Inst(M_{12})$ and $Inst(M_{23})$.

Definition 2^[4] Let $M_{12} = \langle S_1, S_2, \Sigma_{12} \rangle$ and $M_{23} = \langle S_2, S_3, \Sigma_{23} \rangle$ be two schema mappings such that the schemas S_1, S_2, S_3 have no relation symbol in common pairwise. A schema mapping $M = \langle S_1, S_3, \Sigma_{13} \rangle$ is a composition of M_{12} and M_{23} if $Inst(M) = Inst(M_{12}) \circ Inst(M_{23})$ which means that $Inst(M) = \{(I_1, I_3) | (\exists I_2)((I_1, I_2) \in Inst(M_{12}) \wedge (I_2, I_3) \in Inst(M_{23}))\}$, where I_i is the instance of S_i with $1 \leq i \leq 3$.

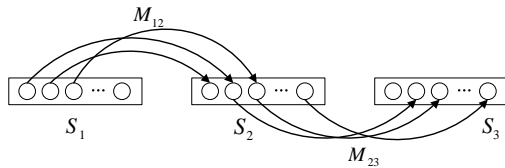


Figure 2. Composing direct mappings

Figure 2 shows an example of direct mapping composition, where S_2 is the middle schema. Data exchange between S_1 and S_3 is achieved by composing M_{12} and M_{23} . Fagin et al. have proved the existence inevitability of mapping composition, and Alan Nash^[7] gives the prerequisite for the existence of mapping composition, that is two mappings have a common schema, which is consistent with Fagin's result. Both of them proved the existence of mapping composition by the relationship between query and mappings, and the instance isomorphism.

Example 1: Consider the following schemas S_1, S_2 and S_3 . S_1 consists of a single binary

relation symbol *Maint*, which associates the name of maintenance person with the equipment he maintains. Schema S_2 consists of a similar binary relation symbol *Maint*₁, which is a copy of *Maint*, and of an additional binary relation symbol *MP*, that associates each person with an id. Schema S_3 consists of one binary relation symbol *Reg*, that associates person ids with the equipments the persons take. Consider the following schema mappings $M_{12} = \langle S_1, S_2, \Sigma_{12} \rangle$ and $M_{23} = \langle S_2, S_3, \Sigma_{23} \rangle$, where

$$\Sigma_{12} = \{\forall n \forall c (Maint(n, c) \rightarrow Maint_1(n, c)),$$

$$\forall n \forall c (Maint(n, c) \rightarrow \exists s MP(n, s))\},$$

$$\Sigma_{23} = \{\forall n \forall s \forall c (MP(n, s) \wedge Maint_1(n, c) \rightarrow Reg(s, c))\}.$$

Give the instances of S_1, S_2, S_3 as follows:

$$I_1: Maint^1 = \{(A, Eq1), (A, Eq2)\}$$

$$I_2: Maint^2 = Maint^1, MP^2 = \{(A, 0001), (B, 0002)\}$$

$$I_3: Reg^3 = \{(0001, Eq1), (0001, Eq2)\}$$

According to Σ_{12} and Σ_{23} , we have $\langle I_1, I_2 \rangle \in Inst(M_{12})$ and $\langle I_2, I_3 \rangle \in Inst(M_{23})$. By definition 2, if there is a mapping M_{13} that satisfies $\langle I_1, I_3 \rangle \in Inst(M_{13})$, then M_{13} is the composition of M_{12} and M_{23} , so M_{13} can be expressed as $\forall n \forall c (Maint(n, c) \rightarrow \exists s Reg(s, c))$.

Theory 1 The mapping composing operation $Inst(M_{12}) \circ Inst(M_{23}) = Inst(M_{13})$ is closed under the homomorphism of instances.

Proof: Given $\langle I_1, I_3 \rangle \in Inst(M_{13})$, for the instances I'_1 and I'_3 , satisfying $I'_1 \cong I_1$ and $I'_3 \cong I_3$ (" \cong " denotes homomorphism), we just need to show that $\langle I'_1, I'_3 \rangle \in Inst(M_{13})$. Since $Inst(M_{12})$ and $Inst(M_{23})$ are closed under homomorphism, and $Inst(M_{12}) \circ Inst(M_{23}) = Inst(M_{13})$, there exists an instance I_2 that $\langle I_1, I_2 \rangle \in Inst(M_{12}) \wedge \langle I_2, I_3 \rangle \in Inst(M_{23})$. So there is an instance $I'_2 \cong I_2$ such that $\langle I'_1, I'_2 \rangle \in Inst(M_{12})$, and also $\langle I'_2, I'_3 \rangle \in Inst(M_{23})$, then $\langle I'_1, I'_3 \rangle \in Inst(M_{13})$. This was to be shown.

IV. INDIRECT MAPPING

A direct mappings can be given by a Source-to-Target tuple generating dependency (denoted by S2Tgd): $\forall x(\phi_s(x) \rightarrow \exists y \psi_T(x, y))$, where ϕ_s and ψ_T are respectively the formulas on S and T , and x, y are the set of variables. A full S2Tgd mapping can be expressed by $\forall x(\phi_s(x) \rightarrow \psi_T(x))$, and a mapping expressed by $\forall x(\phi_s(x) \rightarrow \wedge_{i=1}^k R_i(x_i))$ equals to the set of some full S2Tgds $\forall x(\phi_s(x) \rightarrow R_i(x_i))$, where $i = 1, \dots, k$, and $R_i(x)$ is the atomic formula.

A. Describe indirect mappings

In indirect mappings, the algebra operation results of the elements in both the source and target schema are related, as shown in figure 3.

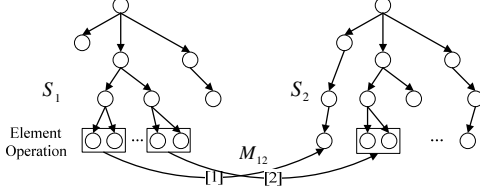


Figure 3. Indirect mappings

In element correspondence “[1]” (see Fig. 3), nodes in the pane of the source schema are mapped to an element in the target schema after some algebra operations. The algebra operations of schema elements includes some basic ones, like Union, Cartesian, Projection and Selection, and some compositions, like Intersection, Join, Nature Join and division etc.

Definition 3 *Indirect elements correspondence* (denoted by ICO_E) is the set of element correspondences that elements are mapped after some algebra operations. Let ϕ_S be the formulas of S , and S_i be the scheme of S , the algebra operations have six forms, as shown bellow, where 1~4 are the basic ones and 5~6 can be derived from 1~4:

Union Let $\phi_S(x) = \bigwedge_{k,i} (S_i(x) \vee \dots \vee S_k(x))$, $S_i(x) \in S$,

and it is the same for ψ_T .

ICO_E : $\forall e(S_1(e) \vee S_2(e) \vee \dots \vee S_n(e) \rightarrow T(e))$ (or $\forall e(S_1(e) \vee S_2(e) \vee \dots \vee S_n(e) \rightarrow \exists wT(e,w))$ if there is an existence quantifier).

Cartesian $\phi_S(x) = \bigwedge_i (S_i(x) \times \dots \times S_k(x))$, $S_m(x) \in S$

ICO_E : $\forall e(S_1(e) \times S_2(e) \times \dots \times S_n(e) \rightarrow T(e))$ (or $\forall e(S_1(e) \times S_2(e) \rightarrow \exists wT(e,w))$).

Projection $\phi_S(x) = \bigwedge_k (\pi_{k_1, \dots, k_m}(S_k(x)))$, $S_k(x) \in S$.

ICO_E : $\forall x(\pi_{m,n}(S_1(a_1, a_2, \dots, a_n)) \wedge \pi_{l,k}(S_2(b_1, b_2, \dots, b_n)) \rightarrow T(a_m, a_n, b_l, b_k, \dots))$ (or $\forall x(\pi_{m,n}(S(a_1, a_2, \dots, a_n)) \wedge \pi_{l,k}(S(b_1, b_2, \dots, b_n)) \rightarrow \exists yT(a_m, a_n, b_l, b_k, \dots, y_1, y_2, \dots, y_{n_2}))$)

), where x and y are the set of variables.

1. Selection This operation considers the conditional mappings, which represents the user’s integrity. It has the form of

$\phi_S(x) = \bigwedge_i (\sigma_F(S_i(x))) = \bigwedge_i (S_i(x) \wedge F(t))$, $S_i(x) \in S$, $t \in Inst(S)$.

ICO_E : $\forall e((S(e) \wedge (Inst(e) \models \theta)) \rightarrow T(e))$.

2. Intersection refers to several schemas, which is a kind of selection operation essentially.

3. Join is the composition of **Cartesian and Selection**

Let $\otimes = \{\wedge, \vee, \sigma, \theta\}$ be the basic algebra operation symbol set (see 1~4 in definition 3), and indirect mappings can be expressed as $\phi'_S(x) = \bigotimes_k (\pi_{i_1, \dots, i_m}(S_k))$, $S_k(x) \in S$. Take the indirect

mapping shown in figure 4 as an example, M_{12} can be expressed as:

$$\forall x((\pi_{m_1}(R_1^{S_1}(x_1^1)) \otimes \pi_{n_1}(R_1^{S_1}(x_1^2))) \wedge (\pi_{m_2}(R_k^{S_1}(x_k^1)) \otimes \pi_{n_2}(R_k^{S_1}(x_k^2))) \rightarrow \exists y(R_1^{S_2}(y_1) \wedge \pi_{l_1} R_2^{S_2}(y_2))), \text{ and } M_{23} :$$

$$\forall y(R_1^{S_2}(y_1) \wedge (\pi_{l_1} R_2^{S_2}(y_1^1)) \otimes \pi_{p_1} R_2^{S_2}(y_1^2)) \rightarrow \exists z(R_3^{S_3}(z_1) \wedge R_3^{S_3}(z_2)) .$$

If $\psi'_T(x, y)$ has the same form with $\phi'_S(x)$, then the non-full indirect mapping can be expressed as: $\forall x(\phi'_S(x) \rightarrow \exists y\psi'_T(x,y))$.

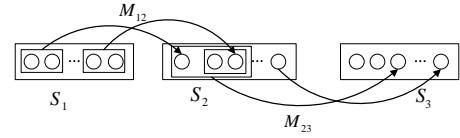


Figure 4. Indirect mappings between schemas.

B. Exchange data under indirect mappings

Now we give the concept of homomorphic instances under indirect mappings (concept of homomorphic direct mappings is given in [4], and we do not discuss it here). Consider the following example:

Example 2: Given schemas S_1, S_2 and S_3 . S_1 consists of a single binary relation symbol *Consistof*, which associates the name of equipments with the parts they have. S_2 consists of a similar binary relation symbol *Consistof₁*, which is a copy of *Consistof*, and of an additional binary relation symbol *MCode*, that associates each equipment name with an id. S_3 consists of a single binary relation symbol *MRegister*, which associates the code of equipments with the related parts. Consider the following mappings: $M_{12} = (S_1, S_2, \Sigma_{12})$ and $M_{23} = (S_2, S_3, \Sigma_{23})$, where

$$\Sigma_{12} = \{\forall e \forall p(Consistof(e, p) \rightarrow Consistof_1(e, p)), \forall e \forall p(Consistof(e, p) \rightarrow \exists c MCode(e, c))\},$$

$$\Sigma_{23} = \{\forall e \forall c \forall p(MCode(e, c) \wedge Consistof_1(e, p)) \rightarrow MRegister(c, p)\}$$

The second formula in Σ_{12} is a direct mapping that associate equipment name e in the source schema to the equipment name e in the target schema. According to **Theory 1**, let I, J be the instance of a mapping M , that means $\langle I, J \rangle \in Inst(M)$, and if there is an instance pair $\langle I', J' \rangle$ that $\langle I', J' \rangle \cong \langle I, J \rangle$, then $\langle I', J' \rangle \in Inst(M)$. For the converse, if $\langle I', J' \rangle$ and $\langle I, J \rangle$ are respectively the instance of M , then we have $\langle I', J' \rangle \cong \langle I, J \rangle$. The above conclusion is true for that M is the direct mapping. For the indirect mappings, the homomorphism between instances is different. For example, let S'_1 be a schema,

which is a copy of S_1 with a little differences that certain element in S_1 splits into several parts in S'_1 . For example, the equipment name in S'_1 contains two parts (surname+name), and the corresponding formula set is Σ'_{12} . Therefore, under the condition of homomorphism, there may be some non-one-to-one correspondences.

Let S , T be the source and target schema respectively, and R_i is a scheme of S . If there is an indirect mapping M from S to T , and let R'_i be the scheme constructed as follows: substituting each element combination that participates in the indirect mappings with a single element. We call R'_i a *generating scheme*. If we define mappings from R'_i to T with the form of $\forall x(R'_i(x) \rightarrow \psi_T(x))$, and mappings from R_i to T with the form of $\forall x(R_i(x) \xrightarrow{Indirect} \psi_T(x))$, then mappings from R'_i to $\psi_T(x)$ are direct, and those from $R_i(x)$ to $\psi_T(x)$ are indirect, as shown in figure 5:

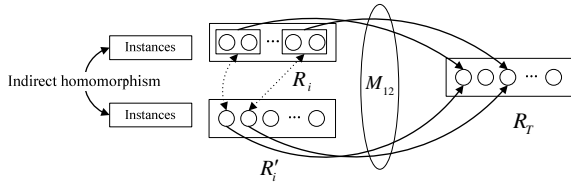


Figure 5. Instance homomorphism under indirect mappings

We define the homomorphism function h under indirect mappings as:

$$h(a_i) = \begin{cases} e & , e \in Inst(\pi_k(R_i)) \\ compound(e_1, \dots, e_n), e_i \in Inst(\pi_i(R_j)) \end{cases}$$

Then h is called a *general homomorphism function*, where *compound* is the algebra operation of elements.

Definition 4 Let R and R' be two schemas, h is the *general homomorphism function* shown above. Let J, J' be the instances of R and R' respectively. If for each scheme symbol R_i in R , and each tuple $(a_1, \dots, a_n) \in R^J$, there is a tuple $(h(a_1), \dots, h(a_n)) \in R'^{J'}$, then R and R' are **homomorphic instances** under h , then the data exchange process under indirect mappings can be implemented by the homomorphism given in **Definition 4** comparing with the process under direct mappings (see[12]).

V. MECHANIC OF INDIRECT MAPPING COMPOSITION

A. Schema-elements-back

As shown in figure 4, although there is a common schema S_2 , we can not directly construct the mappings from S_1 to S_3 according to M_{12} and M_{23} , even can not give the mapping composition directly. Using direct mapping composition, elements in S_1 can only map to part of the certain elements of S_3 . During the data exchange procedure, a complete instance of S_3 may consist part of the instance from S_2 , so data exchange from S_1 to S_3 by

direct mapping composition $M_{12} \circ M_{23}$ may lose some information, that is because M_{12} and M_{23} contains some elements that is not in common, there is no such an instance I_2 of S_2 satisfying conditions in **Definition 2**.

Since the expressions of indirect mapping composition can not be derived by **Definition 2** directly, we deal with this problem based on general instance homomorphism, the main idea of which is to construct a schema that can be directly used for mapping composition, and make sure that the instances of the constructed schema is homomorphic with the instances of the original schema under the *general instance homomorphism function*. We propose a strategy named *schema-elements-back* by constructing some virtual elements in the source schema to deal with the composition of indirect mappings. Compared with BECA, *schema-elements-back* can eliminate the infection of the middle schema.

Definition 5 Let $M_{ST} = \langle S, T, \Sigma_{ST} \rangle$ be a model of indirect schema mapping, where Σ_{ST} is the formula set with the form of $\forall x(\phi'_S(x) \rightarrow \exists y \psi'_T(x, y))$, and mappings from S to T are sound^[13]. Let $Dom_M(S)$ be the domain of S , and $Dom(T)$ be the domain of T , and then we have $Dom_M(S) \subseteq Dom(T)$. Let y_{in} and x_{in} be the set of variables that participate in the indirect mappings in T and S respectively, and we construct the virtual elements in S by set $y_{in} - x_{in}$. As shown in figure 6, the newly constructed source schema is denoted by S' . We can construct direct mappings from T to S' on $Dom_M(Dom_M(S))$. Let $M_{S'T}$ be the mapping from S' to T , and $M_{S'T}^{-1}$ be the inverse of $M_{S'T}$, and then the process of virtual elements construction is called a *schema-elements-back* process.

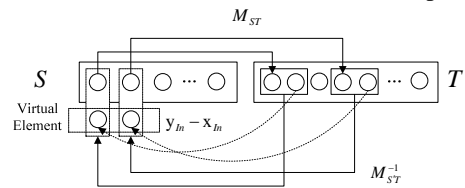


Figure 6. Inverse of indirect mappings based on virtual elements

Definition 6 Let $M_{12} = \langle S_1, S_2, \Sigma_{12} \rangle$ and $M_{23} = \langle S_2, S_3, \Sigma_{23} \rangle$ be two indirect schema mappings such that the schemas S_1, S_2, S_3 have no relation symbol in common pairwise. We construct a new schema S'_1 by adding virtual elements according to **Definition 5** with the mapping $M_{1'2}$ and its inverse $M_{1'2}^{-1}$. Let $M = \langle S'_1, S_3, \Sigma'_{13} \rangle$ be the newly constructed mapping model. If $Inst(M) = Inst(M_{1'2}^{-1}) \circ Inst(M_{23})$, that means $Inst(M) = \{ \langle I'_1, I_3 \rangle \mid (\exists I_2) \langle I'_1, I_2 \rangle \in Inst(M_{1'2}^{-1}) \wedge \langle I_2, I_3 \rangle \in Inst(M_{23}) \}$, then M is the composition of indirect mappings M_{12} and M_{23} .

B. Indirect mapping composition algorithm

When data are exchanged under the strategy of *schema-elements-back*, instance of some elements in the middle schema may be firstly transformed to the source, and there is a reverse data flow during the data exchanging process from the source to the target. Such a data flow can be implemented by the reverse mapping from the element in the middle schema to the newly added element in the source schema. Since the object of composing two schema mappings is to directly construct the mapping from the source to the target without considering the middle schema, *schema-elements-back* can insure that under the condition of indirect mapping, directly data exchange from the source to the target can be realized, which can improve the efficiency of data transformation. Now we show the process of indirect mapping composition.

Schema-elements-back is the process of adding new elements to the source schema according to the middle schema, in order to achieve a directly mapping from the source schema to the target schema under the condition that new elements are introduced. As shown in figure 7, the dashed lines represent the correspondences between the newly added schema elements and elements in the middle schema, from which we can see that when S_1 is translated to S'_1 , by **Theory 2**, mapping composition $M_{1'3}$ can be directly constructed from S'_1 to S_3 .

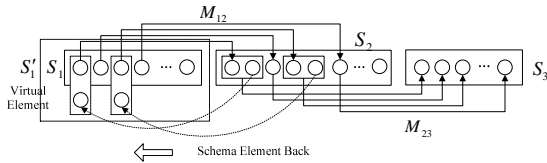


Figure 7. Process of *Schema-elements-back*

Now we give an *indirect mapping composition algorithm* (denoted by IMCA), without loss of generality, we consider the case shown in figure 7, and for the other cases we just need some small modifications.

Algorithm IMCA

Input: $M_{12} = (S_1, S_2, \Sigma_{12})$ and $M_{23} = (S_2, S_3, \Sigma_{23})$, where Σ_{12} and Σ_{23} are extended S2Tgds that describe indirect schema element correspondences.

Output: A schema mapping $M_{1'3} = (S'_1, S_3, \Sigma_{1'3})$, which is the composition of M_{12} and M_{23} .

Step1. Normalize Σ_{12} and Σ_{23} :

Rename the function and element symbols so that the symbols that appear in Σ_{12} are all distinct from those in Σ_{23} .

Step2. Construct virtual elements in S_1 :

If $M_{12}: \pi_{\theta_1}(R_{i_1}^{S_1}(x_{i_1})) \rightarrow \pi_{\theta_2}(R_{i_2}^{S_2}(x_{i_2}))$ and $M_{23}: \pi_{\theta_2'}(R_{i_2}^{S_2}(x_{i_2})) \rightarrow \pi_{\theta_3}(R_{i_3}^{S_3}(x_{i_3}))$ satisfying that $\theta_2' \supset \theta_2$.

Then For each i do projection operation $\pi_{\theta_2'-\theta_2} R_{i_2}^{S_2}(x_{i_2})$, and get an attribute set $E = \{E_1, \dots, E_l\}$, where E_i is the set of attributes generated by $\pi_{\theta_2'-\theta_2} R_{i_2}^{S_2}(x_{i_2})$.

Add each set E_i into S_1 and combine it with attributes computed by $\pi_{\theta_1}(R_{i_1}^{S_1}(x_{i_1}))$, which is called a schema elements back process. As shown in the left dashed box of figure 7, we get the new constructed schema S'_1 by combining elements in E to S_1 (virtual elements are from S_2). Then we get the mapping $M_{1'2}$ from S'_1 to S_2 with the formula set $\Sigma_{1'2}$. Since the mapping from S_2 to S'_1 is one-to-one, we get $M_{1'2}^{-1} = M_{21'}$, where $\Sigma_{21'}$ gives the data back rules from S_2 to S'_1 during the data exchange process from S'_1 to S_3 .

Step3. Construction of S_{12} and S_{23}

(3.1) Initialize S_{12} and S_{23} to each be the empty set.

Assume the formulas in $\Sigma_{1'2}$ is

$$(\forall x_1(\phi_1 \rightarrow \psi_1)) \wedge \dots \wedge (\forall x_n(\phi_n \rightarrow \psi_n))$$

(3.2) Put each of the n implications $\phi_i \rightarrow \psi_i$, for $1 \leq i \leq n$, into S_{12} . We do likewise for Σ_{23} and S_{23} . Each implication χ in S_{12} has the form $\phi(x) \rightarrow \bigotimes_j (\pi_{\theta_j}(R_j^{S_2}(x_j))) \otimes \dots \otimes \pi_{\theta_k}(R_k^{S_2}(x_k))$ where every member of \mathbf{x} is a universally quantified variable, and each x_j , for $1 \leq j \leq k$, is a sequences of terms on \mathbf{x} .

(3.3) Replace each such implication χ in S_{12} with k implications:

$$\phi(x) \rightarrow \pi_{\theta_1}(R_1^{S_2}(x_1)) \otimes \dots \otimes \pi_{\theta_k}(R_k^{S_2}(x_k)), \dots, \phi(x) \rightarrow \pi_{\theta_1}(R_1^{S_2}(x_1)) \otimes \dots \otimes \pi_{\theta_k}(R_k^{S_2}(x_k)).$$

Step4. Compose S_{12} and S_{23} :

Repeat the following until every schema symbol in the left-hand side of every formula in S_{23} is from S_1 .

For each implication χ in S_{23} of the form $\psi \rightarrow \gamma$ where there is an atom $R(y)$ (R is a schema symbol in S_2), perform the following steps to replace $R(y)$ with atoms in S'_1 .

(4.1) Let $\phi_1 \rightarrow R(t_1), \dots, \phi_p \rightarrow R(t_p)$ be all the formulas in S_{12} whose right-hand side has R in it. If no such implications exist in S_{12} , we remove χ from S_{23} . Otherwise, for each such formula $\phi_i \rightarrow R(t_i)$, rename the variables in this formula so that they do not overlap with the variables in χ .

(4.2) Remove χ from S_{23} and add p formulas to S_{23} as follows: replace $R(y)$ in χ with ϕ_i and add the resulting formula to S_{23} , for $1 \leq i \leq p$.

Step5. Construct $M_{1'3}$:

Let $S_{23} = (\chi_1, \dots, \chi_r)$ where χ_1, \dots, χ_r are all the formulas from step4, Let $\Sigma_{1'3}$ be $\forall z_1 \chi_1 \wedge \dots \wedge \forall z_r \chi_r$ where z_i is the set of variables found in χ_i , for $1 \leq i \leq r$.

Return $M_{13} = (S_1', S_3, \Sigma_{1'3})$.

A new source schema should be maintained by the composition algorithm, and such a schema may not be useful in other mapping compositions. However, if the scale of data to be exchanged is large, such new extended schema can improve the efficiency of data exchange.

VI. EXPERIMENTS

A. Complexity of ICMA

Let the elements that schema S_1 , S_2 and S_3 respectively contains be N_1 , N_2 and N_3 . In step1 of ICMA, the rename operation need to match each elements among S_1 , S_2 and S_3 , whose complexity is $O(N_1 \times N_2 \times N_3)$.

Let the correspondences in M_{12} be N_{12} and correspondences in M_{23} be N_{23} , where N_{12} and N_{23} are the numbers of atom elements. The number of virtual elements constructed in *schema-element-back* process is $N_{23} - N_{12}$, so the complexity for constructing virtual elements is $O(N_{23} - N_{12})$.

The complexity for composing is decided by the atom implications that $\Sigma_{1'2}$ and Σ_{23} contains. Let N_C^{12} be the number of implications in $\Sigma_{1'2}$ with the form of $\phi_i \rightarrow \psi_i$ and n_i^{12} be the atom formula in ψ_i , and then the number of formulas in $\Sigma_{1'2}$ is $N_C^{12} \times n_i^{12}$. If the number of implications with the form of $\psi_i \rightarrow \gamma_i$ in Σ_{23} is N_C^{23} , where ψ_i is the set of atom formulas. The number of pairs of atom formulas that should be matched during the replacing process is $N_C^{12} \times n_i^{12} \times N_C^{23}$, so the composing complexity is $O(N_C^{12} \times n_i^{12} \times N_C^{23})$. The whole complexity for indirect mapping composition is $O(N_1 \times N_2 \times N_3) + O(N_{23} - N_{12}) + O(N_C^{12} \times n_i^{12} \times N_C^{23})$. The complexity of direct mapping composition process is $O(N_1 \times N_2 \times N_3) + O(N_C^{12} \times n_i^{12} \times N_C^{23})$, both are polynomial complexities.

B. Results and analysis

(1) Query efficiency under indirect mapping composition

Given the numbers of elements in schema S_1 , S_2 and S_3 , and the numbers of implications in $\Sigma_{1'2}$ and Σ_{23} . Given different scales of indirect mappings, now we compute the numbers of virtual elements constructed during the process of *schema-element-back*. We use the example in DBLP, chose 20 literature base by the sequence of letters, and pick up the meta-information of each literature. We construct the mappings manually as follows (take three of them for example):

Author \otimes Author $\otimes \dots \otimes$ Author \rightarrow AuthorSet

Country \otimes University $\otimes \dots \otimes$ City \rightarrow AuthorAddress

Country \otimes Street $\otimes \dots \otimes$ City \rightarrow ConferenceLocation

The relationship between the number of indirect element correspondences and the number virtual elements constructed are shown in table 1.

Table 1. Relationship between the number of virtual elements generated during the indirect mapping composition process and the number of indirect element correspondences.

$(S_1 \rightarrow S_2)$	5	10	15	20	25	30	35	40	45
$(S_2 \rightarrow S_3)$	5	11	15	22	23	30	36	45	45
virtual elements	2	5	4	7	5	10	11	9	15

Since virtual elements are introduced during the process of indirect mapping composition, figure 8 show a comparison for numbers related by the queries for non-mapping-composition and mapping-composition respectively.

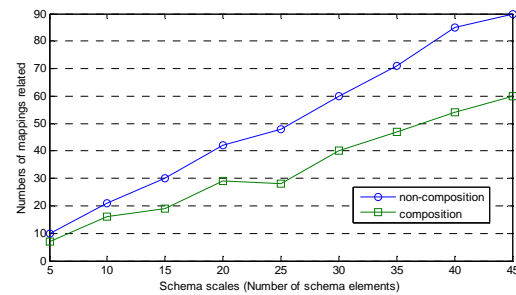


Figure 8. Comparison for mappings related under different condition.

As we can see that, although some virtual elements participate the querying and data exchanging, compared with the non-composition way, indirect mapping composition can reduce the number of mappings related by the query and improve the query efficiency.

Consider the indirect mappings shown in figure 4, we simulate the data exchange process from S_1 to S_3 , and compare the executing times between indirect mapping composition and non-composition under different scale of instances. We use Oracle 9i as the database, and the time for reading records in the oracle database is the increase function of record scale and attributes numbers.

In the case of non-composition, instances $\{I_1\}$ of S_1 are firstly transformed to instances of S_2 according to M_{12} , denoted as $\{I_2\}$, then $\{I_2\}$ are transformed to $\{I_3\}$ of S_3 . the read and write of data happens among three schemas.

In the case of composition, since there are indirect mappings from S_1 to S_3 , using the *schema element back* given in IMCA, instance for part attributes S_2 of are firstly transformed to S_2 , and we get the new instances $\{I_1'\}$, then instances $\{I_3\}$ of S_3 are generated according to M_{13} .

Now we give the executing time (milliseconds) for the data exchange process from S_1 to S_3 under different instance scale of S_1 . Figure 9 show the data exchange executing time comparison results:

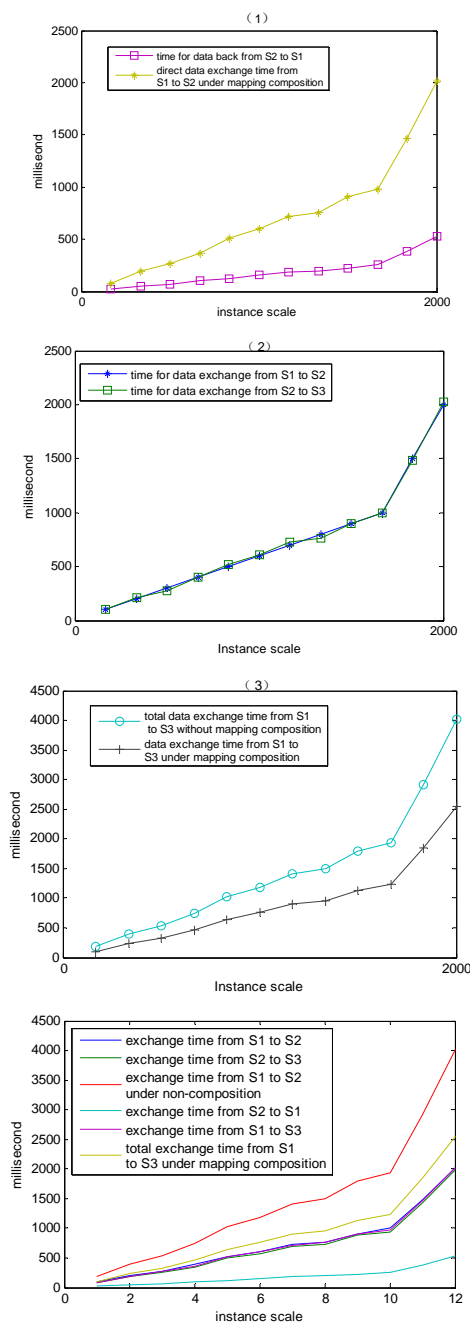


Figure 9. data exchange process executing time comparisons.

(2) Comparison of BECA and ICMA

In BECA, symbols in the middle schema are eliminated by some algebra operations step by step using view unfolding, left composing and right composing. The main idea of BECA is to replace the symbols in the middle schema by the symbols in the source and target schemas as much as possible. Let the elements that participate in the process of elimination obey the equality distribution, and the numbers of virtual elements are decided by step 2 of IMCA, figure 10 shows the comparison between number of element symbols eliminated by BECA and number of virtual elements constructed in IMCA.

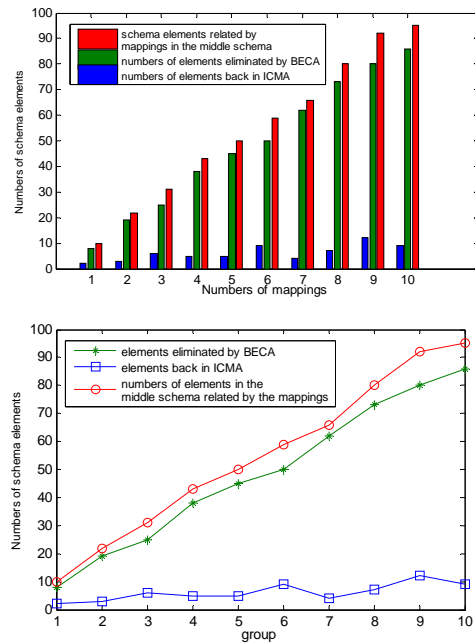


Figure 10. Comparison for affection of middle schema between BECA and ICMA

Seen from the results, the affections of the middle schema on both BECA and IMCA are almost the same. Besides, number of virtual elements in IMCA and number of elements in the middle schema that is not eliminated are equal. But using IMCA, we can completely eliminate the affection of middle schema, which can be seen as a real composition process.

VII. CONCLUSION

Mapping composition is an active aspect in the research of schema mapping management, the object of which is to reuse mappings as a model, and simplify the process of data exchange. Especially when the scale of peers in the network is large, composing mappings can improve the data exchange efficiency. In this paper, we give the definition of indirect mappings, and discuss the composability of indirect mappings. Then we propose an indirect mapping composition algorithm using the *schema-element-back* strategy. Experiment results show that although IMCA dose not improve the query performance, but the affection of middle schema can be eliminated.

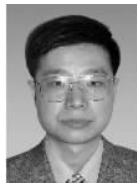
REFERENCES

- [1] A. Fuxman, M. A. Hernández, C. T. H. Ho, R. J. Miller, P. Papotti, L. Popa. Nested Mappings: Schema Mapping Reloaded. [C], VLDB 2006, Seoul Korea, 67-68,2006
- [2] P. A. Berstein, S. Melnik. Model Management 2.0: Manipulating Richer Mappings [C], SIGMOD, Beijing, China, 1-12,2007
- [3] P. A. Berstein, T. J. Green, S. Melnik. Implementing mapping composition [J]. VLDB Journal, 2008,17): 333-353

- [4] R. Fagin, P. G. Kolaitis, L. Popa. Composing Schema Mappings: Second-Order Dependencies to the Rescue [J]. ACM TODS, 2005, 30 (4): 994-1055
- [5] J. Madhavan, A. Y. Halevy. Composing Mappings Among Data Sources [C], VLDB 03, Berlin, Germany, 572-583,2003
- [6] P. McBrien, A. Poulouvasilis. Data Integration by Bi-Directional Schema Transformation Rules [C], ICDE03, Boston, 227-238,2003
- [7] A. Nash. Foundations of Information Integration, SAN DIEGO, UNIVERSITY OF CALIFORNIA, 2006
- [8] A. Nash, P. A. Bernstein, S. Melnik. Composition of Mappings Given by Embedded Dependencies [C], PODS 2005, Baltimore, Maryland, USA, 2005
- [9] M. Boyd, S. Kittivoravitkul, C. Lazanitis, P. McBrien, N. Rizopoulos. AutoMed: A BAV Data Integration System for Heterogeneous Data Sources [C], CAiSE04, Riga Latvia, 3084,82-97,2004
- [10] M. Boyd, P. McBrien. Comparing and Transforming Between Data Models via an Intermediate Hypergraph Data Model [J]. COMPUTER SCIENCE, 2005, 3730 (69-109
- [11] LiXu, D. W. Embley. A Composite Approach to Automating Direct and Indirect Schema Mappings [J]. information System, 2006, 31 (8): 697-132
- [12] L. Popa, Y. Velegrakis, R. J.Miller, M. A. Hernandez, R. Fagin. Translating Web Data [C], Proceedings of the 28th VLDB Conferences(VLDB'02) Hong Kong, China, 598-609,2002
- [13] M. Lenzerini. Data Integration: A Theoretical Perspective [C], PODS'02, 233-246,2002
- [14] Y. Velegrakis, R. J.Miller, L. Popa. Mapping Adaptation under Evolving Schemas [C], 29th VLDB Conference, Berlin, Germany, 2003
- [15] R. Fagin, P. G. Kolaitis, R. J.Miller, L. Popa. Data Exchange: Semantics and Query Answer. [J]. Theoretical Computer Science 2005, 336 (89-124



Wang Bo, born in Shenyang, China, in September 18th, 1980, got his Ph.D in National University of Defense Technology, Changsha, China in 2009. His current research interests include heterogeneous information integration and schema mapping.



Guo Bo, born in 1963. Professor and PH.D. supervisor. His main research interests include system management and integration and information management system.