INFORMATICA, 2008, Vol. 19, No. 3, 421–446

An Algebraic Framework for Schema Matching

Zhi ZHANG, Pengfei SHI
Institute of Image Processing and Pattern Recognition, Shanghai Jiaotong University
SAP Labs China, Shanghai 200030, China
e-mail: uphenix@hotmail.com

Haoyang CHE, Jun GU
Institute of Software, The Chinese Academy of Sciences
Beijing 100080, China
e-mail: chehy@hotmail.com

Received: 13 March 2006; accepted: 18 February 2008

Abstract. It is well known that a formal framework for the schema matching problem (SMP) is important because it facilitates the building of algorithm model and the evaluation of algorithms. An algebraic framework for schema matching is developed in this paper. First, based on universal algebra, we propose a meta-meta structure for schema, which is named multi-labeled schema. This definition has a distinctive feature: it is able to formally describe any particular style of schemas, and transforms a schema and other available information into a finite structure over specific signature. Later, we introduce a formal definition of schema matching that is called multivalent matching. Then, we formulize SMP as a schema homomorphism problem, and prove that SMP is equivalent to finding a semantic homomorphism from one schema to another. These results lead to the main contribution of this paper: an algebraic framework for SMP. This framework builds the algorithm model for SMP. Thirdly, we show a classification of schema matching based on the algebraic framework. Finally, we discuss the relations between matching cardinality and subclasses of schema homomorphism.

Keywords: schema matching, schema homomorphism, schema isomorphism, graph homomorphism, labeled graph matching, computation complexity.

1. Introduction

The schema matching problem plays a key role in various database applications, such as data integration, e-business, data warehousing, XML message mapping, metadata/model management, semantic query processing, and peer-to-peer data management (Rahm and Bernstein, 2001). Specifically, with the advance of Semantic web, for achieving semantic interoperability, ontology (schema) matching has come a very important problem that has to be solved effectively (Doan *et al.*, 2003). The goal of schema matching is to find the semantic correspondences between elements of two schemas.

For SMP, a major challenge is in properly addressing semantics: the semantic representations associated with the schemas. There are many practical techniques for representing the semantics of schemas, such as semantic data models (Hull and King, 1987),

description logics (Artale *et al.*, 2003), and corpora or dictionaries (Madhavan *et al.*, 2004), etc. A second major challenge is in developing such a framework that is applicable to a variety of data models, such as the relational, object-oriented, and XML models. The framework facilitates the building of algorithm model and the evaluation of algorithms.

To cope with SMP, there are many methods have been proposed. Many matching methods are based on machine learning techniques. Doan *et al.* (2003) developed a LSD system that uses Bayesian learners to match a pair of schemas, producing a $1:1$ atomic-level mapping. This approach is primarily instance-oriented; Li and Clifton (2000) presented a prototype SEMINT that uses neural network to obtain schema matching; Berlin and Motro (2001) devised Automatch system for database schema matching, which used machine learning techniques, based primarily on Bayesian learning, to achieve automate schema matching; Madhavan *et al.* (2004) implemented a CUPID system, which is a hybrid matcher based on both element- and structure-level matching. In this prototype, the authors proposed similar-based heuristics algorithms and tree matching algorithm to achieve semi-automatic schema matching; Bouquet *et al.* (2003) viewed each semantic schema as a context, and proposed an algorithm for automatically discovering the relations across contexts, which is named CTXMATCH; Furthermore, based on CTX-MATCH system, Shvaiko (2006) developed the S-Match algorithm to solve semantic matching between schemas. Do and Rahm (2002) devised the COMA schema matching system, which combines multiple matchers in a flexible way. Melnik (2004) carried out a generic model-management tool – RONDO. The author used directed labeled graph as internal schema model and proposed a graph-based algorithm – similarity flooding for schema matching. Two substantial reviews of SMP and prototypes have been given by Rahm and Bernstein (2001) and Do *et al.* (2002). However, these practical matching approaches have not utilized the theoretical foundation and most of them have relied on ad-hoc approaches.

The theoretical foundation and the formal framework for related problems of SMP have been developed. Paolini and Pelagatti (1977) analyzed mappings between external views of a database and conceptual views of the database itself, where database was represented by many-sorted algebra and mappings were treated as homomorphisms. They demonstrated how update operations on databases can be treated as mappings; Hull (1997) presented a tutorial for describing fundamental problems that are raised by semantic heterogeneity and surveyed theoretical frameworks that can provide solutions for them. Most work focused on schema integration in federated databases. For schema equivalence, Miller *et al.* (1994) proposed that two schemas could be compared by *information capacity*. They presented two concepts of equivalence: absolute equivalence and internal equivalence. Based on a graph model (*Schema Intension Graph*), they proposed that the isomorphic schemas are equivalent, and discussed the structural transformation of schemas; Alagić and Bernstein (2001) developed a categorical model theory for generic schema management, and investigated the category-theoretic approach to semantics in model management. By using the category to represent the schema, schema morphisms are defined as mappings of schema signatures that preserve the integrity constraints. Schema transformations within a particular category of schemas are viewed as

morphisms of that category. Then, they built the formal frameworks for schema integration and transformation. In fact, these formal frameworks are not applicable to schema matching, specifically cannot characterize many-to-many matching. Since little attention has been given to the theoretical foundation of SMP, in this paper, we address to develop a formal framework for SMP.

Schema matching is inherently heuristic. In general, matching two schemas requires information that is not present in the schemas, such as relational schemas, object-oriented schemas, and XML schemas. To achieving schema matching, we need more semantics of schemas. Therefore, it is a difficult problem to develop a formal framework for SMP. In this paper, we propose a formally *matching-oriented* definition of schema at first, which is a *meta-meta model* for schema and called *multi-labeled schema*. This definition has a distinctive feature that is based on the universal algebra, which can formally describe any particular style of schemas and transform them into the finite structures over specific signatures, and the signatures include some useful information for matching.

By *multi-labeled schema*, we introduce two important definitions: one is the concept of *individual matching*, the other is *schema matching*. We define schema matching as *multivalent matching*, which means that one object of a schema may be matched with a set of objects of another schema, i.e., this definition of schema matching can characterize many-to-many matching.

Based on the foundational definitions discussed above, we propose the algebraic framework for SMP. This framework is called *schema homomorphism*. We demonstrate that SMP is equivalent to finding a semantic homomorphism from one schema to another, which preserves the semantics between two schemas. Using this framework, we formulize SMP as a schema homomorphism problem. Moreover, homomorphism is a useful model for a wide variety of combinatorial problems dealing with mappings and assignments (Hell, 2003). This algorithmic model can guide practitioner to design the effective algorithms for SMP and evaluate the algorithms.

Based on this framework, we show a classification of schema matching. In addition, there are various subclasses of homomorphism, and then we discuss the relations between matching cardinality and subclasses of schema homomorphism. Finally, we use label graph model as the internal schema model, which is the instantiation of *multi-labeled schema*. We discuss the algorithmic complexity of SHOM and its subclasses. Most of them are NP-complete in the general case and polynomial when the underlying graph of schema is a tree.

The rest of this paper is organized as follows. Section 2 proposes a formal matching-oriented definition of schema, which is called *multi-labeled schema*, and based on universal algebra. Section 3 focuses on individual matching, and introduces the formal definition of schema matching: multivalent matching. Further, Section 4 proposes the definition of schema homomorphism. We prove that SMP is equivalent to finding a semantic homomorphism from one schema to another. Section 5 presents a new taxonomy for SMP under the algebraic framework. Section 6 investigates match cardinality and the variants of homomorphism. We present the relations between match cardinality and schema homomorphisms. Section 7 addresses the algorithmic complexity of SMP. Section 8 summarizes the contributions and suggests future research directions.

2. Schema

To build the algebraic framework, we present a formal *meta-meta structure (model)* to describe the various schemas at first. As we know it, there are many kinds of schemas, such as relational model, object-oriented model, ER model, conceptual graph, DTD, XML schema, and UML, etc., which are also called *meta-models*. A meta-model can be thought of as a model that describes the structure of another model. A *meta-meta model* is a representation language in which models and meta-models are represented. For example, the UML specification uses an object-oriented meta-meta model called MOF. All models and meta-models can be viewed as instances of the meta-meta model (Melnik, 2004).

Schemas are *finite structures* over the specific signatures. The matching objects and their properties are able to assemble in the structure. By the basic definition of structure (Dubhashi, 1995; Federa and Vardi, 1998; Federa *et al.*, 2004), we define a *meta-meta structure* for schema, which is based on universal algebra, and called *multi-labeled schema*. Here, a signature σ is a collection of individual, label, relation and function symbols. A *schema* or *structure* of the signature σ can be denoted by a 4-tuples $\mathcal{S} = (I^{\mathcal{S}}, Lab^{\mathcal{S}}, F^{\mathcal{S}}, R^{\mathcal{S}})$.

DEFINITION 1 (schema). A schema $\mathcal{S}$ is a finite structure over a signature σ, consists of individual set $I^{\mathcal{S}}$, label collection $Lab^{\mathcal{S}}$, function set $F^{\mathcal{S}}$, relation set $R^{\mathcal{S}}$, written a 4-tuples $\mathcal{S} = (I^{\mathcal{S}}, Lab^{\mathcal{S}}, F^{\mathcal{S}}, R^{\mathcal{S}})$, where,

1. σ is a finite collection that is composed of individual symbols, label symbols, function symbols, and relation symbols, where, each function symbol f or relation symbol R, respectively comes associated with an arity, $ar(f)$ and $ar(R)$, which are non-negative integers.
2. $I^{\mathcal{S}} = \{s_1, s_2, \ldots, s_n\}$ is a finite nonempty set that includes individuals, which denote the prepared-matching objects. Each of them is uniquely identified by an object identifier (OID).
3. $Lab^{\mathcal{S}} = \{Lab_1^{\mathcal{S}}, Lab_2^{\mathcal{S}}, \ldots, Lab_i^{\mathcal{S}}\}$ is a finite constant collection that includes the label sets for individuals. The labels are the strings for describing the properties of individuals.[1]
4. $F^{\mathcal{S}} = \{f_1^{\mathcal{S}}, f_2^{\mathcal{S}}, \ldots f_j^{\mathcal{S}}\}$ is a finite set that includes the labeling functions, which are partial function. The domain of each function is the individual set $I^{\mathcal{S}}$, accordingly, the codomain is the label collection $Lab^{\mathcal{S}}$. [2]

[1] REMARK 1. Each individual or relation can be associated with a set of labels that describe its properties, such as names, concept, and attributes, etc. In terms of different schema models and applications, the label collection may consist of a name set, concept set, attribute set, and the combination of these labels. These sets are different kinds of labels for representing the properties of individuals. For example, $name^{\mathcal{S}} = \{n_1^{\mathcal{S}}, n_2^{\mathcal{S}}, \ldots, n_m^{\mathcal{S}}\} = Lab_1^{\mathcal{S}}$ denotes a name label set, which includes names for individuals.

[2] REMARK 2. For different kinds of labels, the labeling functions are different accordingly. For example, if $f_1^{\mathcal{S}}$: $I^{\mathcal{S}} \rightarrow concept^{\mathcal{S}}$ is a labeling function of concept, $c_i^{\mathcal{S}} \in concept^{\mathcal{S}}$, $f_1^{\mathcal{S}}(s_i) = c_i^{\mathcal{S}}$ means that s_i is mapped to $c_i^{\mathcal{S}}$, where, $\mathrm{dom}(f_1^{\mathcal{S}}) \subseteq I^{\mathcal{S}}, \mathrm{codom}(f_2^{\mathcal{S}}) \subseteq concept^{\mathcal{S}}$. If $f_2^{\mathcal{S}}$: $I^{\mathcal{S}} \times I^{\mathcal{S}} \rightarrow Lab_R^{\mathcal{S}}$ is a labeling function of binary relation, $f_2^{\mathcal{S}}(R(s_i, s_j)) = l_{ij}^{\mathcal{S}}$ denotes that the label $l_{ij}^{\mathcal{S}}$ is assigned to the relation $R(s_i, s_j)$, where, $\mathrm{dom}(f_2^{\mathcal{S}}) \subseteq I^{\mathcal{S}} \times I^{\mathcal{S}}, \mathrm{codom}(f_2^{\mathcal{S}}) \subseteq Lab_R^{\mathcal{S}}, Lab_R^{\mathcal{S}} \subseteq Lab^{\mathcal{S}}, l_{ij}^{\mathcal{S}} \in Lab_R^{\mathcal{S}}$.

5. $R^{\mathcal{S}} = \{R_1, R_2, \ldots, R_k\}$ is a finite nonempty set that includes the relations between individuals. If R is a b-ary relation, then $R \subseteq (I^{\mathcal{S}})^b$. In general, the relations are binary relations $R^{\mathcal{S}} \subseteq I^{\mathcal{S}} \times I^{\mathcal{S}}$. [3]
6. The size of schema $\mathcal{S}$ is the size of individuals and is denoted by $|I^{\mathcal{S}}|$.

Since an individual of schema can be labeled with several labels, the schema is called *multi-labeled schema*. Unlike conventional schema, the schema defined by Definition 1 includes some available information for matching, such as the concepts of individuals, which are developed by the external dictionary (WordNet). However, the conventional schema does not contain these concepts directly. Alternatively, in this paper, a schema can be regard as a semantic *corpus* in a broad sense.

In Example 1, we use an XML schema and a relational schema to illustrate the definition of schema (Definition 1).

EXAMPLE 1. In Fig. 1(a), there is an XML schema $\mathcal{S}$. By Definition 1, we present a *multi-labeled schema* representation of $\mathcal{S}$. The signature $\sigma_{\mathcal{S}}$ includes individual symbols, label symbols, function symbols, and relation symbols. The schema $\mathcal{S}$ is a finite structure over $\sigma_{\mathcal{S}}, \mathcal{S} = (I^{\mathcal{S}}, Lab^{\mathcal{S}}, F^{\mathcal{S}}, R^{\mathcal{S}})$.

```
<schema xmlns="…">
  <complexType name="Product">
    <element name="ProductID" type="xs:int"/>
    <element name="ProductName" type="xs:string"/>
    <element name="ProductType" type="xs:string"/>
  </complexType>
</schema>
```

a. An XML schema $\mathcal{S}$

```
CREATE TABLE PRODUCTS (
  PID int PRIMARY KEY,
  PName varchar)
```

b. A relational schema $\mathcal{T}$

Fig. 1. An XML Schema $\mathcal{S}$ and a relational schema $\mathcal{T}$.

a. Individual set: $I^{\mathcal{S}} = \{s_1, s_2, s_3, s_4\}, |I^{\mathcal{S}}| = 4$.
b. Label collection: $Lab^{\mathcal{S}} = \{N^{\mathcal{S}}, C^{\mathcal{S}}, T_1^{\mathcal{S}}, T_2^{\mathcal{S}}, L_R^{\mathcal{S}}\}$, where, $N^{\mathcal{S}}$ is the name set of individuals, $C^{\mathcal{S}}$ is the concept set of individuals, $T_1^{\mathcal{S}}$ is the type set of individuals, $T_2^{\mathcal{S}}$ is the data type set of individuals, and $L_R^{\mathcal{S}}$ is the label set of relations:
   $N^{\mathcal{S}} = \{$Product, ProductID, ProductName, ProductType$\}$,
   $C^{\mathcal{S}} = \{$(product#*n*#1), (ID#*n*#2), (name#*n*#1), (type#*n*#1)$\}$ [4]
   $T_1^{\mathcal{S}} = \{$complexType, element$\}$, $T_2^{\mathcal{S}} = \{$xs:int, xs:string$\}$, $L_R^{\mathcal{S}} = \{$include$\}$.
c. Function set: $F^{\mathcal{S}} = \{f_1, f_2, f_3, f_4, f_5\}$, where
   $f_1(s_1) = \text{Product}, f_1(s_2) = \text{ProductID},$
   $f_1(s_3) = \text{ProductName}, f_1(s_4) = \text{ProductType};$

[3] REMARK 3. If every $R_i \in R$ is a binary relation (i.e., $R \subseteq I^{\mathcal{S}} \times I^{\mathcal{S}}$), then schema $\mathcal{S}$ is transformed into a graph structure. Each individual is a vertex of graph, and each $R_i(s_i, s_j)$ is an edge that connects vertex s_i and s_j. If f_1 is a unary labeling function, we obtain vertex-labeled graph, and if f_2 is a binary labeling function for relations, then we obtain edge-labeled graph. One vertex and edge may have different kinds of labels in that the graph is termed multi-labeled graph.

[4] We use WordNet 2.0 to obtain the concepts of individuals. Here, they are all nouns.

$f_2(s_1) = \text{product}, f_2(s_2) = \text{ID}, f_2(s_3) = \text{name}, f_2(s_4) = \text{type};$
$f_3(s_1) = \text{complexType}, f_3(s_2) = \text{element}, f_3(s_3) = \text{element}, f_3(s_4) = \text{element};$
$f_4(s_2) = \text{xs} : \text{int}, f_4(s_3) = \text{xs} : \text{string}, f_4(s_4) = \text{xs} : \text{string}.$

d. Relation set $R^{\mathcal{S}} = \{(s_1, s_2), (s_1, s_3), (s_1, s_4)\}$,
$f_5((s_1, s_2)) = \text{include}, f_5((s_1, s_3)) = \text{include}, f_5((s_1, s_4)) = \text{include},$
where $f_5((s_1, s_2)) = \text{include}$ denotes that s_1 includes s_2.

Following the same procedure, we obtain a *multi-labeled schema* representation of the relational schema $\mathcal{T}$ (Fig. 1(b)), which is a finite structure of $\sigma_{\mathcal{T}}$, $\mathcal{T} = (I^{\mathcal{T}}, Lab^{\mathcal{T}}, F^{\mathcal{T}}, R^{\mathcal{T}})$.

a. Individual set: $I^{\mathcal{T}} = \{t_1, t_2, t_3\}, |I^{\mathcal{T}}| = 3$.
b. Label collection: $Lab^{\mathcal{T}} = \{N^{\mathcal{T}}, C^{\mathcal{T}}, T_1^{\mathcal{T}}, T_2^{\mathcal{T}}, L_R^{\mathcal{T}}\}$,
$N^{\mathcal{T}} = \{\text{PRODUCTS, PID, PName}\}$;
$C^{\mathcal{T}} = \{(\text{product\#}n\#1), (\text{ID\#}n\#2), (\text{name\#}n\#1)\}$;
$T_1^{\mathcal{T}} = \{\text{TABLE, field}\}; T_2^{\mathcal{T}} = \{\text{int, varchar}\}; L_R^{\mathcal{T}} = \{\text{include}\}$
c. Function set: $F^{\mathcal{T}} = \{g_1, g_2, g_3, g_4, g_5\}$, where
$g_1(t_1) = \text{PRODUCTS}, g_1(t_2) = \text{PID}, g_1(t_3) = \text{PName};$
$g_2(t_1) = \text{product}, g_2(t_2) = \text{ID}, g_2(t_3) = \text{name};$
$g_3(t_1) = \text{TABLE}, g_3(t_2) = \text{field}, g_3(t_3) = \text{field}; g_4(t_2) = \text{int}, g_4(t_3) = \text{varchar}$
d. Relation set: $R^{\mathcal{T}} = \{(t_1, t_2), (t_1, t_3)\}$
$g_5((t_1, t_2)) = \text{include}, g_5((t_1, t_3)) = \text{include}.$

3. Schema Matching

3.1. *Problem Description*

We describe SMP informally as follows:
INSTANCE: *Given two schemas* $\mathcal{S} = (I^{\mathcal{S}}, Lab^{\mathcal{S}}, F^{\mathcal{S}}, R^{\mathcal{S}})$ *and* $\mathcal{T} = (I^{\mathcal{T}}, Lab^{\mathcal{T}}, F^{\mathcal{T}}, R^{\mathcal{T}})$, $\mathcal{S}$ *is a source schema, and* $\mathcal{T}$ *is a target schema.*
QUESTION: *To find the semantic correspondences between individuals in* $I^{\mathcal{S}}$ *and* $I^{\mathcal{T}}$?

For convenience in later discussion, suppose that $\mathcal{S} = (I^{\mathcal{S}}, Lab^{\mathcal{S}}, F^{\mathcal{S}}, R^{\mathcal{S}})$ and $\mathcal{T} = (I^{\mathcal{T}}, Lab^{\mathcal{T}}, F^{\mathcal{T}}, R^{\mathcal{T}})$ are two schemas of the same signature σ. Let $\mathcal{S}$ be the source schema, and $\mathcal{T}$ be the target schema.

3.2. *Individual Matching*

For schema transformation, Hull (1986) and Miller *et al.* (1994) used *information capacity* to compare two schemas. A key question in the work on information capacity has been whether a given database schema is more, less, or equally expressive than another database schema, i.e., whether there exists a surjective or bijective function between $\mathcal{S}$ and $\mathcal{T}$ (Melnik, 2004). In contrast, schema matching is inherently heuristic. The schema matching approaches focus on obtaining the actual correspondences between $\mathcal{S}$ and $\mathcal{T}$.

A matching result of two schemas is a set of semantic correspondences between individuals of two schemas. For this reason, we introduce a definition of individual matching that based on the semantics of two individuals of schemas: if one or more labels of individual s in $\mathcal{S}$ are semantically related to corresponding labels of individual t in $\mathcal{T}$, or the relations of s and the relations of t are semantically equivalent, then we define that s and t are matched.

DEFINITION 2 (individual matching). If $\mathcal{S}$ is the source schema, $\mathcal{T}$ is the target schema, $s \in I^{\mathcal{S}}$, $t \in I^{\mathcal{T}}$, s and t are matched, such that:

1. There exists a function symbol f of arity a, $f^{\mathcal{S}}(s, s_1, \ldots, s_{a-1}) = l_i^{\mathcal{S}} \Rightarrow f^{\mathcal{T}}(t, t_1, \ldots, t_{a-1}) = l_j^{\mathcal{T}}$, which means that $l_i^{\mathcal{S}}$ is semantically associated with $l_j^{\mathcal{T}}$, written $l_i^{\mathcal{S}} \rightarrow l_j^{\mathcal{T}}$, where $f \in \sigma$, $f^{\mathcal{S}} \in F^{\mathcal{S}}$, $f^{\mathcal{T}} \in F^{\mathcal{T}}$, $s_1, \ldots, s_{a-1} \in I^{\mathcal{S}}$, $t_1, \ldots, t_{a-1} \in I^{\mathcal{T}}$, $l_i^{\mathcal{S}} \in Lab^{\mathcal{S}}$, $l_j^{\mathcal{T}} \in Lab^{\mathcal{T}}$. [5]
2. There exists a relation symbol R of arity b, $R^{\mathcal{S}}(s, s_1, \ldots, s_{b-1})$ holds $\Rightarrow$ $R^{\mathcal{T}}(t, t_1, \ldots, t_{b-1})$ holds, which denotes that the relation between s and $s_1, \ldots, s_{b-1}$ is equivalent to the relation between t and $t_1, \ldots, t_{b-1}$, where $R \in \sigma$, $R^{\mathcal{S}}(s, s_1, \ldots, s_{b-1}) \in R^{\mathcal{S}}$, $R^{\mathcal{T}}(t, t_1, \ldots, t_{b-1}) \in R^{\mathcal{T}}$, $s_1, \ldots, s_{b-1} \in I^{\mathcal{S}}$, $t_1, \ldots, t_{b-1} \in I^{\mathcal{T}}$.

The matched individuals s and t can be denoted by a binary relation $< s, t >$, which represents a semantic correspondence between s and t: $s \rightarrow t$.

If s and t only satisfy Condition 1, then the correspondence of s and t is called *label* matching.

If s and t only satisfy Condition 2, then the correspondence is termed *relation* matching; If s and t satisfy two conditions at the same time, the correspondence is called *structure* matching.

In addition, for the correspondence between s and t, the more semantic functions and relations can be matched (i.e., the more semantics can be preserved), the *stronger* matching between s and t is.

Specifically, if $\forall f \in \sigma, f^{\mathcal{S}}(s, s_1, \ldots, s_{a-1}) = l_i^{\mathcal{S}} \Rightarrow f^{\mathcal{T}}(t, t_1, \ldots, t_{a-1}) = l_j^{\mathcal{T}}$, and if $\forall R \in \sigma, R^{\mathcal{S}}(s, s_1, \ldots, s_{b-1}) holds \Rightarrow R^{\mathcal{T}}(t, t_1, \ldots, t_{b-1}) holds$, then the correspondence of s and t is the *strongest* matching.

If s and t dissatisfy two conditions in Definition 2, then s cannot be matched with t. If there does not exist an individual t in $\mathcal{T}$ that can be matched with s, then s is matched to ε, written $s \rightarrow \varepsilon$, where, ε stands for a void individual. In Formula 1, we show the individual matching result, where, $s \in I^{\mathcal{S}}$, $t \in I^{\mathcal{T}}$.

$$\textit{individual matching} \begin{cases} s \rightarrow t & \text{if } s \text{ and } t \text{ satisfy the conditions in Definition 2,} \\ s \rightarrow \varepsilon & \text{if } s \text{ and } \forall t \in I^{\mathcal{T}} \text{ dissatisfy the conditions in Definition 2,} \end{cases} \tag{1}$$

$$\text{i.e.,} \quad \forall s \in I^{\mathcal{S}}, \ \exists t \in I^{\mathcal{T}} \cup \varepsilon, \ s \rightarrow t, \ \text{and} \ \forall t \in I^{\mathcal{T}}, \ \exists s \in I^{\mathcal{S}} \cup \varepsilon, \ s \rightarrow t. \tag{2}$$

[5] If f is a unary function, $f^{\mathcal{S}}(s) = l^{\mathcal{S}} \Rightarrow f^{\mathcal{T}}(t) = l^{\mathcal{T}}$ indicates that $l^{\mathcal{S}}$ and $l^{\mathcal{T}}$ are equivalent, $l^{\mathcal{S}} \rightarrow l^{\mathcal{T}}$, where, $l^{\mathcal{S}}$ and $l^{\mathcal{T}}$ stands for the semantic label of individual s and t. To measure the semantic similarity of two labels or relations, there are many methods have been proposed (Bunke, 2000; Mugnier, 2000; Zhang *et al.*, 2006).

Formula 1 and 2 imply that one individual of $\mathcal{S}$ may be associated with a possibly empty set of individuals of $\mathcal{T}$.

3.3. *Schema Matching*

For measuring the similarity of labeled graph, Champin and Solnon (2003) and Sorlin and Solnon (2005) proposed multivalent mapping, which is used to characterize the many-to-many matching between the vertices of two labeled graphs. Analogously, we introduce *multivalent matching* for SMP, i.e., an individual of source schema may be associated with a set of individuals of target schema. The matching results are called *multivalent correspondences*.

DEFINITION 3 (multivalent correspondence). If $\mathcal{S}$ is the source schema, $\mathcal{T}$ is the target schema, the matching result of two schemas is a set $m \subseteq I^{\mathcal{S}} \times I^{\mathcal{T}}$ that contains every matched couple $< s, t > \in I^{\mathcal{S}} \times I^{\mathcal{T}}$.

Multivalent correspondences are binary relationships that establish many-to-many correspondences between the individuals of two schemas.

DEFINITION 4 (partially match). If $\mathcal{S}$ is the source schema, $\mathcal{T}$ is the target schema, $I^{\mathcal{S}} = I_1^{\mathcal{S}} \cup I_2^{\mathcal{S}}, \forall s \in I_1^{\mathcal{S}}, \exists\, t \in I^{\mathcal{T}}$ such that s and t are matched, and $\forall s \in I_2^{\mathcal{S}}$ hasn't a matched individual in $I^{\mathcal{T}}$, then we call schema $\mathcal{S}$ and $\mathcal{T}$ are partially matched, written $\mathcal{S} \rightarrow_{\mathcal{P}} \mathcal{T}$, where $I_1^{\mathcal{S}}$ and $I_2^{\mathcal{S}}$ are nonempty sets.

Roughly speaking, the schema matching remains with *partial* matching. To illustrate the definition of schema matching, we take two simple schemas in Fig. 1 for example.

EXAMPLE 2. By the results of Example 1, we find the semantic correspondences between $\mathcal{S}$ and $\mathcal{T}$.

Here, the signature $\sigma = \sigma_{\mathcal{S}\cup\mathcal{T}} = \sigma_{\mathcal{S}} \cup \sigma_{\mathcal{T}}$. In particular, the function symbols of $\mathcal{S}$ and $\mathcal{T}$ are identical: $\{f_1, f_2, f_3, f_4, f_5\}$, because $\mathcal{S}$ and $\mathcal{T}$ are two schemas of the same signature, i.e., $F^{\mathcal{S}} = \{f_1^{\mathcal{S}}, f_2^{\mathcal{S}}, f_3^{\mathcal{S}}, f_4^{\mathcal{S}}, f_5^{\mathcal{S}}\}$, $F^{\mathcal{T}} = \{f_1^{\mathcal{T}}, f_2^{\mathcal{T}}, f_3^{\mathcal{T}}, f_4^{\mathcal{T}}, f_5^{\mathcal{T}}\}$. The functions and the domains and codomains of them are shown in Table 1.

We will get:

$< s_2, t_2 >$, for
$f_1(s_2) = \text{ProductID} \Rightarrow f_1(t_2) = \text{PID}$; $f_2(s_2) = \text{ID} \Rightarrow f_2(t_2) = \text{ID}$;
$f_3(s_2) = \text{element} \Rightarrow f_3(t_2) = \text{field}$; $f_4(s_2) = \text{xs} : \text{int} \Rightarrow f_4(t_2) = \text{int}$;
$(s_1, s_2) \Rightarrow (t_1, t_2)$; $f_5((s_1, s_2)) = \text{include} \Rightarrow f_5((t_1, t_2)) = \text{include}$.

$< s_3, t_3 >$, for
$f_1(s_3) = \text{ProductType} \Rightarrow f_1(t_3) = \text{PName}$; $f_2(s_3) = \text{name} \Rightarrow f_2(t_3) = \text{name}$
$f_3(s_3) = \text{element} \Rightarrow f_3(t_3) = \text{field}$; $f_4(s_3) = \text{xs} : \text{string} \Rightarrow f_4(t_3) = \text{varchar}$;
$(s_1, s_3) \Rightarrow (t_1, t_3)$; $f_5((s_1, s_3)) = \text{include} \Rightarrow f_5((t_1, t_3)) = \text{include}$.

$< s_4, t_3 >$, for
$f_1(s_4) = \text{ProductName} \Rightarrow f_1(t_3) = \text{PName}$; $f_2(s_3) = \text{name} \Rightarrow f_2(t_3) = \text{name}$

Table 1
Label type and labeling function (*partial* function)

label type	*function*	*domain*	*codomain*
name	$f_1^{\mathcal{S}}(x)$	$I^{\mathcal{S}}$	$N^{\mathcal{S}}$
	$f_1^{\mathcal{T}}(x)$	$I^{\mathcal{T}}$	$N^{\mathcal{T}}$
concept	$f_2^{\mathcal{S}}(x)$	$I^{\mathcal{S}}$	$C^{\mathcal{S}}$
	$f_2^{\mathcal{T}}(x)$	$I^{\mathcal{T}}$	$C^{\mathcal{T}}$
type	$f_3^{\mathcal{S}}(x)$	$I^{\mathcal{S}}$	$T_1^{\mathcal{S}}$
	$f_3^{\mathcal{T}}(x)$	$I^{\mathcal{T}}$	$T_1^{\mathcal{T}}$
data type	$f_4^{\mathcal{S}}(x)$	$I^{\mathcal{S}}$	$T_2^{\mathcal{S}}$
	$f_4^{\mathcal{T}}(x)$	$I^{\mathcal{T}}$	$T_2^{\mathcal{T}}$
relation	$f_5^{\mathcal{S}}(x,y)$	$I^{\mathcal{S}} \times I^{\mathcal{S}}$	$L_R^{\mathcal{S}}$
	$f_5^{\mathcal{T}}(x,y)$	$I^{\mathcal{T}} \times I^{\mathcal{T}}$	$L_R^{\mathcal{T}}$

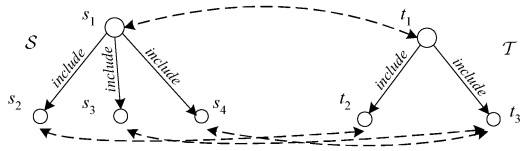

Fig. 2. The potential correspondences between $\mathcal{S}$ and $\mathcal{T}$.

$f_3(s_4) = \text{element} \Rightarrow f_3(t_3) = \text{field}; f_4(s_4) = \text{xs:string} \Rightarrow f_4(t_3) = \text{varchar};$
$(s_1, s_4) \Rightarrow (t_1, t_3); f_5((s_1, s_4)) = \text{include} \Rightarrow f_5((t_1, t_3)) = \text{include}.$
$< s_1, t_1 >$, for
$f_1(s_1) = \text{Product} \Rightarrow f_1(t_1) = \text{PRODUCTS};$
$f_2(s_1) = \text{product} \Rightarrow f_2(t_1) = \text{product};$
$f_3(s_1) = \text{complexType} \Rightarrow f_3(t_1) = \text{TABLE};$
$(s_1, s_2) \Rightarrow (t_1, t_2); f_5((s_1, s_3)) = \text{include} \Rightarrow f_5((t_1, t_3)) = \text{include};$
$(s_1, s_3) \Rightarrow (t_1, t_3); f_5((s_1, s_3)) = \text{include} \Rightarrow f_5((t_1, t_3)) = \text{include};$
$(s_1, s_4) \Rightarrow (t_1, t_3); f_5((s_1, s_4)) = \text{include} \Rightarrow f_5((t_1, t_3)) = \text{include}.$

Hence, we have a matching result of $\mathcal{S}$ and $\mathcal{T}$, which is a set of matching couples: $m = \{< s_1, t_1 >, < s_2, t_2 >, < s_3, t_3 >, < s_4, t_3 >\}, m \subseteq I^{\mathcal{S}} \times I^{\mathcal{T}}$.[6]

[6] REMARK 4. Since the correspondences preserve all semantics of functions and relations, they belong to the strongest matching.

REMARK 5. If we only use the name label to compare two schemas, i.e., we only consider the semantics of function f_1, and then the matching method is called name-based and individual matcher. If we use the name and concept label together, then the matching method can be called combining matcher.

REMARK 6. In Example 2, functions are unary or binary function, and relations are binary relation. The two schemas can be transformed into multi-labeled graph model (Remark 3). Fig. 2 shows the potential correspondences between $I^{\mathcal{S}}$ and $I^{\mathcal{T}}$. For brevity, the labels of individual are not marked completely.

4. Schema Homomorphism

By the basic notion of *homomorphism* (HOM, Federa *et al.*, 2004; Hell, 2003), Definition 1, and Definition 2, we introduce the definition of *schema homomorphism*, which is an extension of homomorphism. By reason that schema homomorphism preserves the semantics of two schemas, which is also called *semantic homomorphism*.

4.1. *Schema Homomorphism (SHOM)*

A schema homomorphism from $\mathcal{S}$ to $\mathcal{T}$ is made up of the multivalent correspondences, which preserves the semantics between two schemas.

DEFINITION 5 (schema homomorphism). A schema homomorphism φ: $\mathcal{S} \rightarrow \mathcal{T}$ from the source schema $\mathcal{S}$ to the target schema $\mathcal{T}$ is a mapping φ: $I^{\mathcal{S}} \rightarrow I^{\mathcal{T}}$, which is a set of multivalent correspondences such that:

Condition 1. There exists a labeling function symbol f of arity n

$$f^{\mathcal{S}}(s_1, \ldots, s_n) = l_n^{\mathcal{S}} \Rightarrow f^{\mathcal{T}}(\varphi(s_1), \ldots, \varphi(s_n)) = \varphi(l_n^{\mathcal{T}})$$

for $s_1, \ldots, s_n \in I^{\mathcal{S}}$, $l_n^{\mathcal{S}} \in Lab^{\mathcal{S}}$.

Condition 2. There exists a semantic relation symbol R of arity m

$$R^{\mathcal{S}}(s_1, \ldots, s_m) \text{ holds} \Rightarrow R^{\mathcal{T}}(\varphi(s_1), \ldots, \varphi(s_m)) \text{ holds}$$

for $s_1, \ldots, s_m \in I^{\mathcal{S}}$.

If there exists a semantic homomorphism from $\mathcal{S}$ to $\mathcal{T}$, then we write $\mathcal{S} \rightarrow \mathcal{T}$. [7]

4.2. *Schema Matching in Homomorphism*

In this section, we demonstrate that SMP is equivalent to finding a semantic homomorphism from $\mathcal{S}$ to $\mathcal{T}$, which preserves the semantics between two schemas. We first show two lemmas.

[7] REMARK 7. Schema homomorphism is different from the common homomorphism. First, homomorphism is a map (function/morphism) between two structures. As a rule, term *map* or *mapping* is often a synonym for *function*, and HOM requires that one object of source structure is associated with at most one object in target structure. However, the mapping φ: $I^{\mathcal{S}} \rightarrow I^{\mathcal{T}}$ of SHOM does not denote a function, because one individual of schema $\mathcal{S}$ may be associated with a set of individuals of schema $\mathcal{T}$. The reason for the discrepancy is that schema matching allows *many-to-many* mapping between two schemas. Therefore, to formulize SMP, we propose SHOM that allows *many-to-many* mapping and does not limit to *one-to-one* and *many-to-one* mapping. In addition, to restrict the matching cardinality within one-to-one or many-to-one, the mapping φ: $I^{\mathcal{S}} \rightarrow I^{\mathcal{T}}$ is a function as usual.

REMARK 8. Second, a homomorphism is a map that preserves all the relevant structure of two algebraic structures (Federa *et al.*, 2004; Hell, 2003). For formulizing SMP, SHOM not merely preservers the structure, but also preserves the semantics of two schemas. In addition, schema homomorphism does not need to preserve all the semantic structure, and only requires that partial labeling functions and relations satisfy the two conditions in Definition 5. For instance, schema homomorphism is based on name and concept labels, provided that the name and concept labeling function satisfies the condition of Definition 5.

Lemma 1. *If $\mathcal{S}$ and $\mathcal{T}$ are matched, then there exists a semantic homomorphism from $\mathcal{S}$ to $\mathcal{T}$.*

Proof. By Definition 3, the result of schema matching consists of all the individual correspondences. Without loss of generality, suppose that s is matched to $t, s \in I^{\mathcal{S}}, t \in I^{\mathcal{T}}$:

i) $\exists f \in \sigma$, $f^{\mathcal{S}}(s, s_1, \ldots, s_{a-1}) = l_i^{\mathcal{S}} \Rightarrow f^{\mathcal{T}}(t, t_1, \ldots, t_{a-1}) = l_j^{\mathcal{T}}$, f is a semantic function symbol of *arity* a;

ii) $\exists R \in \sigma$, $R^{\mathcal{S}}(s, s_1, \ldots, s_{b-1})$ holds $\Rightarrow R^{\mathcal{T}}(t, t_1, \ldots, t_{b-1})$ holds, R is a semantic relation symbol of *arity* b. For every correspondence, let $\varphi(s) = t$, $\varphi(l_i^{\mathcal{S}}) = l_j^{\mathcal{T}}$, by Definition 5, Lemma 1 is completed.

To be more specific, we show an example to explain Lemma 1.

EXAMPLE 3. The individual correspondences in Fig. 2, i.e., $< s_1, t_1 >$, $< s_2, t_2 >$, $< s_3, t_3 >$, and $< s_4, t_3 >$, constitute matching result of two schemas, which is a semantic homomorphism from $\mathcal{S}$ to $\mathcal{T}$.

For $< s_2, t_2 >$, s_2 and t_2 are matched (Example 2): suppose we define $\varphi(s_2) = t_2$, $\varphi(\mathrm{ID}) = \mathrm{ID}$, $\varphi(\mathrm{ProductID}) = \mathrm{PID}$, $\varphi(\mathrm{element}) = \mathrm{field}, \varphi(\mathrm{xs:int}) = \mathrm{int}$, and $\varphi(\mathrm{include}) = \mathrm{include}$, we get:

$f_1(s_2) = \mathrm{ProductID} \Rightarrow f_1(\varphi(s_2)) = \varphi(\mathrm{ProductID})$, *name preserving*;

$f_2(s_2) = \mathrm{ID} \Rightarrow f_2(\varphi(s_2)) = \varphi(\mathrm{ID})$, *concept preserving*;

$f_3(s_2) = \mathrm{element} \Rightarrow f_3(\varphi(s_2)) = \varphi(\mathrm{element})$, *type preserving*;

$f_4(s_2) = \mathrm{xs:int} \Rightarrow f_4(\varphi(s_2)) = \varphi(\mathrm{xs:int})$, *data type preserving*;

$f_5((s_1, s_2)) = \mathrm{include} \Rightarrow f_5((\varphi(s_1),\ \varphi(s_2))) = \varphi(\mathrm{include})$, *relation label preserving*.

$(s_1, s_2) \Rightarrow (\varphi(s_1), \varphi(s_2))$, *relation (structure) preserving*.

We can see that the mapping $s_2 \rightarrow t_2$ satisfies two conditions of Definition 5. In much the same way that, let $\varphi(s_3) = t_3, \varphi(s_4) = t_3, \varphi(s_1) = t_1$, and so on, we can obtain a semantic homomorphism φ: $I^{\mathcal{S}} \rightarrow I^{\mathcal{T}}$. All the mappings satisfy two conditions in Definition 5.

Lemma 2. *If there exists a semantic homomorphism from $\mathcal{S}$ to $\mathcal{T}$, then $\mathcal{S}$ and $\mathcal{T}$ are matched.*

EXAMPLE 4. With reference to Example 1 and Fig. 2, there exists a semantic homomorphism from $\mathcal{S}$ to $\mathcal{T}$. The individual correspondences include: $\varphi(s_1) = t_1,\ \varphi(s_2) = t_2,\ \varphi(s_3) = t_3,\ \varphi(s_4) = t_4$. Without loss of generality, we explain the reason of matched couple $< s_2, t_2 >$.

For $f_1(s_2) = \mathrm{ProductID} \Rightarrow f_1(\varphi(s_2)) = \varphi(\mathrm{ProductID})$, let $\varphi(\mathrm{ProductID}) = \mathrm{PID}$, we can obtain the correspondence based on the name label of s_2 and t_2;

for $f_2(s_2) = \mathrm{ID} \Rightarrow f_2(\varphi(s_2)) = \varphi(\mathrm{ID})$, let $\varphi(\mathrm{ID}) = \mathrm{ID}$, we see that the mapping preserve the conceptual semantics between s_2, and t_2, and we have the correspondence

based on the concept label;

$$(s_1, s_2) \Rightarrow (\varphi(s_1), \varphi(s_2)), f_5((s_1, s_2)) = \text{include}$$
$$\Rightarrow f_5((\varphi(s_1), \varphi(s_2))) = \varphi(\text{include}),$$

let $\varphi(\text{include}) = \text{include}$, then we can obtain the mapping by relation and relation label.

From Lemma 1 and 2, we have the theorem of SMP in homomorphism.

Theorem 1. *Two schemas $\mathcal{S}$ and $\mathcal{T}$ are matched iff there exists a semantic homomorphism from $\mathcal{S}$ to $\mathcal{T}$, $\mathcal{S} \to \mathcal{T}$.*

Proof. $\Leftarrow$ Lemma 2, if $\mathcal{S} \to \mathcal{T}$, then $\mathcal{S}$ and $\mathcal{T}$ are matched.
$\Rightarrow$ Lemma 1, if $\mathcal{S}$ and $\mathcal{T}$ are matched, then $\mathcal{S} \to \mathcal{T}$.

COROLLARY 1. $\mathcal{S}$ is a source schema, $\mathcal{T}$ is a target schema, $\mathcal{S}_{sub}$ is a subset of $\mathcal{S}$, $I^{\mathcal{S}}_{sub}$ denotes the individual set of $\mathcal{S}_{sub}$, and $I^{\mathcal{T}}$ denotes the individual set of $\mathcal{T}$. $\mathcal{S}_{sub}$ and $\mathcal{T}$ are matched iff there exists a semantic homomorphism from $\mathcal{S}_{sub}$ to $\mathcal{T}$, $\mathcal{S}_{sub} \to \mathcal{T}$, also written $\mathcal{S} \to_{\mathcal{P}} \mathcal{T}$.

4.3. *Algorithmic Model for SMP*

Now, we formulize SMP as the SHOM problem, and build the formal framework for schema matching.

We discuss the concept of strong matching in Section 3. For a schema homomorphism, if all the functions and relations satisfy two conditions of Definition 5, then this homomorphism is the strongest schema homomorphism that represents two schemas are close associated. Based on Theorem 1, we have developed the algorithm model for schema matching.

Algorithm model of SMP. *Given two schemas $\mathcal{S}$ and $\mathcal{T}$, the goal of matching algorithms is to find the strongest semantic homomorphism between $\mathcal{S}$ and $\mathcal{T}$.*

Because homomorphism is a useful model for a wide variety of combinatorial problems dealing with mappings and assignments (Hell, 2003), SMP is formulized as a combinatorial problem by SHOM framework. In virtue of the concept of strong SHOM, we can design the optimization algorithms to solve the schema matching: the goal of matching algorithms is to find the strongest schema homomorphism that preserves the semantics between two schemas. Therefore, under SHOM, the algorithm model for SMP is developed.

Algorithm 1. Algorithm for SMP
Input: Schemas $\mathcal{S}$ and $\mathcal{T}$
Output: The semantic correspondences between $\mathcal{S}$ and $\mathcal{T}$
Optimization Object: Obtain the strongest semantic homomorphism between $\mathcal{S}$ and $\mathcal{T}$

Using the labeled graph as the internal schema model, SMP can be transformed into a labeled graph matching problem. Based on graph homomorphism model, there are many

methods to solve this classic combinational problem (Bunke, 2000; Champin and Solnon, 2003; Hell, 2003; Melnik, 2004; Mugnier, 2000; Sorlin and Solnon, 2005; Zhang, *et al.*, 2005). In particular, by using the multi-labeled graph as the meta-model, Zhang *et al.* (2005, 2006) developed an objective function and proposed the hybrid search algorithms to achieve schema matching based on the multi-labeled graph matching. Algorithm 2 shows the basic idea of multi-labeled graph matching (Zhang *et al.*, 2005, 2006).

Algorithm 2. A search algorithm of SMP based on labeled graph matching
Input: Schemas $\mathcal{S}$ and $\mathcal{T}$
Object: Find a matching state that maximize the similarity of graph G_1 and G_2
Output: The semantic correspondences between $\mathcal{S}$ and $\mathcal{T}$

1. $G_1 =$ Graph($\mathcal{S}$); $G_2 =$ Graph ($\mathcal{T}$);
2. Iteratively search the matching space to find a matching state $\boldsymbol{m}$, such that similarity $(G_1, G_2)_{\boldsymbol{m}}$ is the maximum one among the matching states;
3. $\boldsymbol{m}$ is the matching result of G_1 and G_2.

The optimization object of SMP is to find one matching state that is the strongest semantic homomorphism between $\mathcal{S}$ and $\mathcal{T}$, in Algorithm 2, i.e., the matching state is the one such that the similarity of two graphs is the maximum one in all of the matching states.

5. Taxonomy of Schema Matching

5.1. *Hierarchy of SHOM and SMP*

In Subsection 3.2, we discuss the types of individual matching. Individual matching is divided into label matching, relation matching, and structure matching. By Definition 5, we study the classification of SHOM. SHOM requires that correspondences satisfy two conditions, i.e., Condition 1 and Condition 2. Condition 1 means that the labels of two matched individuals are semantically equivalent; Condition 2 represents that the relations of two mapped individuals are equivalent.

To be more specific, for example, suppose there is a schema homomorphism between two schemas, and suppose that the syntactic or semantic labeling function satisfies Condition 1, then we define that this schema homomorphism is a syntactic or semantic matching. If the relations of two schemas satisfies Condition 2, then the homomorphism is a relation-based matching. Furthermore, suppose that the homomorphic mapping simultaneously satisfies Condition 1 and Condition 2, then the mapping is a structural matching. Hence, based on two conditions of Definition 5, the matching methods can be divided

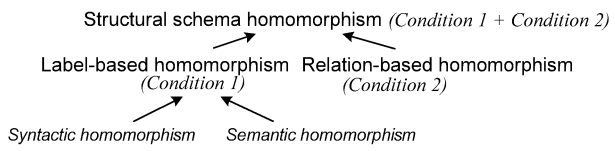

Fig. 3. Hierarchy of schema homomorphism based on two conditions.

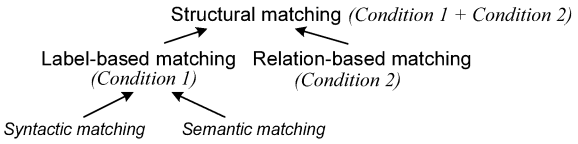

Fig. 4. Hierarchy of schema matching method.

into three methods: label-based, relation-based and structural-based method. We show the classification of schema matching in Fig. 4.

5.2. *Label-based Matching Methods*

Now, we discuss the classes of schema matching based on Condition 1 in detail, i.e., label-based matching in Fig. 4.

In Definition 1, the label collection includes all kinds of labels of individuals, such as name labels, concept labels, attribute labels, and type labels, etc. The label classification is convenient for practitioner to design and analyze the different matching approaches of SMP (Rahm and Bernstein, 2001). For different kinds of labels, the labeling functions are different accordingly, and the matching methods are different too.

For instance, if the matching approach is based on one kind of symbol, such as the name symbol or the attribute symbol, then the approach is an individual matcher that is name-based or attribute-based, and if the matching algorithm uses more than one kind of symbols, then the approach is a combining matcher. In Fig. 5, we present the matching methods based on the types of label symbols.

5.3. *Homomorphic Matching*

In Section 4, we prove that SMP is equivalent to finding the semantic homomorphism from the source schema to the target schema, therefore, we can call schema matching as *schema homomorphism*, also as *homomorphic matching*.

In the previous sections, we classify the matching methods by Definition 5 and the types of label. Here, we show the overall classification of SMP by SHOM framework. In particular, the matching method is called the labeled graph matching, provided that we use labeled graph model as internal schema model and consider two conditions together. We reclassify the schema matching approaches based on the classification in (Rahm and Bernstein, 2001). The classification of homomorphic matching is shown in Fig. 6.

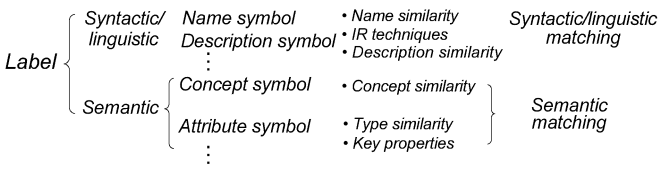

Fig. 5. Matching methods based on the different types of labels.

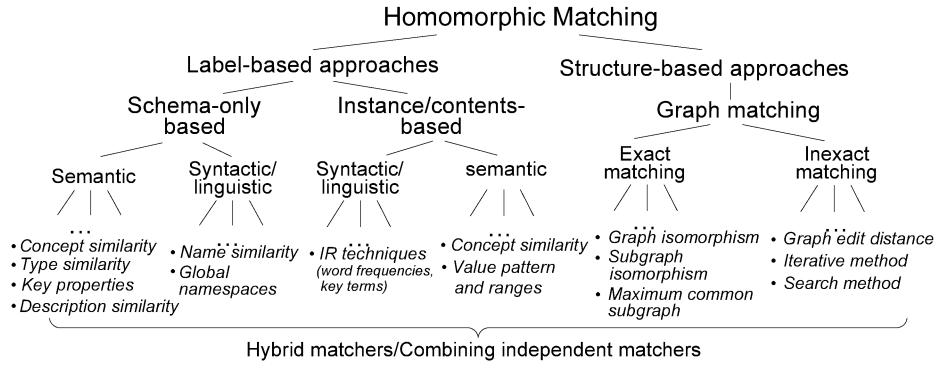

Fig. 6. Classification of schema matching approaches.

We illustrate the classification of SMP by means of the framework of schema homomorphism. The algebraic framework can characterize SMP elaborately.

6. Match Cardinality and Variants of Homomorphism

6.1. *Match Cardinality*

Rahm and Bernstein (2001) illustrated match cardinality by examples. Each element of the resulting mapping may match one or more elements of one schema to one or more elements of the other, yielding four cases: $1:1$, $1:n$, $n:1$, $n:m$.

In Example 2, we obtain an $n:1$ matching. If we restrict the match cardinality is $1:1$, we get the matching result in Fig. 7, where, $\mathcal{S}$ and $\mathcal{T}$ are partial matched (Definition 4).

6.2. *Variants of Homomorphism*

There are various subclasses of homomorphism, such as *isomorphism*, *epimorphism*, and *monomorphism*. For SMP, we discuss some variants of SHOM, such as *schema isomorphism*, *schema epimorphism*, and *schema monomorphism*.

Schema epimorphism. *A schema epimorphism from $\mathcal{S}$ to $\mathcal{T}$ is a surjective mapping φ: $I^{\mathcal{S}} \rightarrow I^{\mathcal{T}}$, which satisfies two conditions of Definition* 5. *For epimorphism, $|I^{\mathcal{S}}| > |I^{\mathcal{T}}|$.*

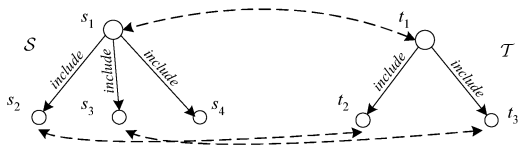

Fig. 7. The possible mappings ($n:1$) between $\mathcal{S}$ and $\mathcal{T}$.

Schema epimorphism is a many-to-one $(n:1)$ semantic mapping from $\mathcal{S}$ to $\mathcal{T}$. With reference to Fig. 2, we can see an example for schema epimorphism. Some schema matching approaches have $n:1$ matching cardinality (Rahm and Bernstein, 2001), such as SKAT (Mitra *et al.*, 1999) and CUPID (2004).

Schema monomorphism. *A schema monomorphism from $\mathcal{S}$ to $\mathcal{T}$ is an injective mapping φ: $I^{\mathcal{S}} \to I^{\mathcal{T}}$, which satisfies two conditions of Definition* 5. *For monomorphism*, $|I^{\mathcal{S}}| \leqslant |I^{\mathcal{T}}|$.

Schema monomorphism denotes a one-to-one and not onto semantic mapping from $I^{\mathcal{S}}$ to $I^{\mathcal{T}}$.

For schema transformation, Miller *et al.* (1994) used information capacity to compare two schemas. By using *SIG*, they proposed that an information preserving mapping between two schemas is a total, injective function φ: $\mathcal{S} \to \mathcal{T}$. Analogously, if there exists a semantic injection from $I^{\mathcal{S}}$ to $I^{\mathcal{T}}$, then $\mathcal{S}$ is subsumed by $\mathcal{T}$. we present the definition of schema subsumption, which denotes the semantic subsumption relation between two schemas.

DEFINITION 6 (schema subsumption). If there exist a schema monomorphism from $\mathcal{S}$ to $\mathcal{T}$, then $\mathcal{S}$ is semantically subsumed by $\mathcal{T}$, written $\mathcal{S} \prec \mathcal{T}$.

As shown in Fig. 5, there is an injective semantic mapping ϕ: $I^{\mathcal{T}} \to I^{\mathcal{S}}$. To be specific, the semantic correspondences include: $\phi(t_1) = s_1, \phi(t_2) = s_2$, and $\phi(t_3) = s_3$. Every individual of $\mathcal{T}$ has a semantically corresponding individual in $\mathcal{S}$. Thus, we have $\mathcal{T} \prec \mathcal{S}$.

Schema isomorphism. *A schema isomorphism from $\mathcal{S}$ to $\mathcal{T}$ is a bijective mapping φ: $I^{\mathcal{S}} \to I^{\mathcal{T}}$, such that*:

1. *There exists a labeling function symbol f of arity n*

$$f^{\mathcal{S}}(s_1, \ldots, s_n) = l_n^{\mathcal{S}} \Rightarrow f^{\mathcal{T}}(\varphi(s_1), \ldots, \varphi(s_n)) = \varphi(l_n^{\mathcal{T}}),$$
$$f^{\mathcal{T}}(t_1, \ldots, t_n) = l_n^{\mathcal{T}} \Rightarrow f^{\mathcal{S}}(\varphi^{-1}(t_1), \ldots, \varphi^{-1}(t_n)) = \varphi^{-1}(l_n^{\mathcal{S}}).$$

2. *There exists a semantic relation symbol R of arity m*

$$R^{\mathcal{S}}(s_1, \ldots, s_m)holds \Rightarrow R^{\mathcal{T}}(\varphi(s_1), \ldots, \varphi(s_m))holds$$
$$R^{\mathcal{T}}(t_1, \ldots, t_m)holds \Rightarrow R^{\mathcal{S}}(\varphi^{-1}(t_1), \ldots, \varphi^{-1}(t_m))holds$$

for $s_1, \ldots, s_m \in I^{\mathcal{S}}, t_1, \ldots, t_m \in I^{\mathcal{T}}, l_n^{\mathcal{S}} \in Lab^{\mathcal{S}}, l_n^{\mathcal{T}} \in Lab^{\mathcal{T}}$.
In isomorphism, $|I^{\mathcal{S}}| = |I^{\mathcal{T}}|$.

Fig. 7 shows a *bijective* mapping, which denotes a one-to-one and onto correspondence from $I^{\mathcal{S}}$ to $I^{\mathcal{T}}$.

Miller *et al.* (1994) proposed the isomorphic schemas are equivalent. In schema homomorphism framework, we introduce the concept of schema isomorphism that denotes $\mathcal{S}$ and $\mathcal{T}$ are equivalent.

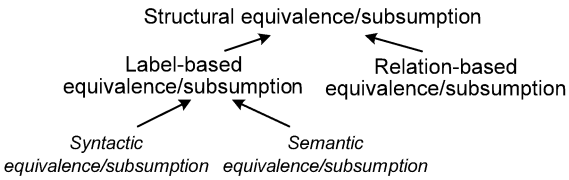

Fig. 8. Hierarchy of schema equivalence.

DEFINITION 7 (schema equivalence). Two schemas $\mathcal{S}$ and $\mathcal{T}$ are equivalent iff there exists a schema isomorphism from $\mathcal{S}$ to $\mathcal{T}$, i.e., $\mathcal{S} \succ \mathcal{T}$ and $\mathcal{S} \prec \mathcal{T}$, written $\mathcal{S} \cong \mathcal{T}$.

Similarly, suppose a mapping of two schemas is a schema monomorphism/isomorphism, and suppose that the syntactic or semantic labeling function satisfies Condition 1, then we define that two schemas are syntactic or semantic subsumption/equivalence. If the relations of two schemas satisfies Condition 2, then two schemas are relational subsumption/equivalence. Furthermore, suppose that the monomorphic/isomorphic mapping simultaneously satisfies Condition 1 and Condition 2, then two schemas are structural subsumption/equivalence.

In Example 5, we use an XML schema and a relational schema to illustrate the definition of schema equivalence.

EXAMPLE 5. In Fig. 9, there are two schemas. Fig. 9(a) shows an XML schema $\mathcal{S}$ that is similar to the XML schema in Fig. 1(a) and only delete individual s_4. Fig. 9(b) shows a relational schema $\mathcal{T}$ that is identical to the relational schema in Fig. 1(b). In Example 1, we show two schemas $\mathcal{S} = (I^{\mathcal{S}}, Lab^{\mathcal{S}}, F^{\mathcal{S}}, R^{\mathcal{S}})$ and $\mathcal{T} = (I^{\mathcal{T}}, Lab^{\mathcal{T}}, F^{\mathcal{T}}, R^{\mathcal{T}})$. Now, suppose that two schemas $\mathcal{S}_1$ and $\mathcal{T}_1$ are over the same signature, we present an isomorphic mapping between $\mathcal{S}_1$ and $\mathcal{T}_1$.

For $f_1(s_2) = \text{ProductID} \Leftrightarrow f_1(t_2) = \text{PID}$: name matching,
$f_2(s_2) = \text{ID} \Leftrightarrow f_2(t_2) = \text{ID}$: concept matching,
$f_3(s_2) = \text{element} \Leftrightarrow f_3(t_2) = \text{field}$: type matching,
$f_4(s_2) = \text{xs} : \text{int} \Leftrightarrow f_4(t_2) = \text{int}$: data type matching,
$(s_1, s_2) \Leftrightarrow (t_1, t_2)$: relation matching, and
$f_5((s_1, s_2)) = \text{include} \Leftrightarrow f_5((t_1, t_2)) = \text{include}$: relation label matching,
we say s_2 and t_2 are structural matching, $s_2 \simeq t_2$.

```
<schema xmlns="…">
  <complexType name="Product">
    <element name="ProductID" type="xs:int"/>
    <element name="ProductName" type="xs:string"/>
  </complexType>
</schema>
```

a. An XML schema $\mathcal{S}_1$

```
CREATE TABLE PRODUCTS (
  PID int PRIMARY KEY,
  PName varchar)
```

b. A relational schema $\mathcal{T}_1$

Fig. 9. An XML Schema $\mathcal{S}_1$ and a relational schema $\mathcal{T}_1$.

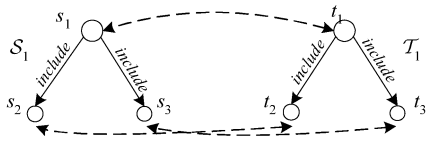

Fig. 10. An isomorphic mapping between $\mathcal{S}_1$ and $\mathcal{T}_1$.

$s_3 \simeq t_3$, for

$f_1(s_4) = \text{ProductType} \Leftrightarrow f_1(t_3) = \text{PName}; f_2(s_3) = \text{name} \Leftrightarrow f_2(t_3) = \text{name}$

$f_3(s_3) = \text{element} \Leftrightarrow f_3(t_3) = \text{field}; f_4(s_3) = \text{xs} : \text{string} \Rightarrow f_4(t_3) = \text{varchar};$

$(s_1, s_3) \Leftrightarrow (t_1, t_3); f_5((s_1, s_3)) = \text{include} \Leftrightarrow f_5((t_1, t_3)) = \text{include}.$

$s_1 \simeq t_1$, for

$f_1(s_1) = \text{Product} \Leftrightarrow f_1(t_1) = \text{PRODUCTS}; f_2(s_1) = \text{product} \Leftrightarrow f_2(t_1) = \text{product};$

$f_3(s_1) = \text{complexType} \Leftrightarrow f_3(t_1) = \text{TABLE};$

$(s_1, s_2) \Leftrightarrow (t_1, t_2); f_5((s_1, s_3)) = \text{include} \Leftrightarrow f_5((t_1, t_3)) = \text{include};$

$(s_1, s_3) \Leftrightarrow (t_1, t_3); f_5((s_1, s_3)) = \text{include} \Leftrightarrow f_5((t_1, t_3)) = \text{include};$

$(s_1, s_4) \Rightarrow (t_1, t_3); f_5((s_1, s_4)) = \text{include} \Leftrightarrow f_5((t_1, t_3)) = \text{include}.$

Then, we obtain an isomorphism from $\mathcal{S}_1$ to $\mathcal{T}_1$, which is a structural matching. In other words, $\mathcal{S}_1$ and $\mathcal{T}_1$ are structural equivalence. Using labeled graph, Fig. 10 shows an isomorphic mapping between $\mathcal{S}_1$ and $\mathcal{T}_1$. [8]

DEFINITION 8 (subschema isomorphism). If $\mathcal{S}$ is the source schema, $\mathcal{T}$ is the target schema, $\mathcal{S}_{sub} \subseteq \mathcal{S}$, $\mathcal{T}_{sub} \subseteq \mathcal{T}$, then $\mathcal{S}_{sub}$ and $\mathcal{T}_{sub}$ are equivalent iff there exists a schema isomorphism from $\mathcal{S}_{sub}$ to $\mathcal{T}_{sub}$, written $\mathcal{S}_{sub} \cong \mathcal{T}_{sub}$, called subschema isomorphism.

Subschema isomorphism is a natural generalization of the concept of isomorphism. Since most schema matching tools are based on $1:1$ matching (Rahm and Bernstein, 2001), these matching methods fall into subschema isomorphism. If we restrict matching cardinality is $1:1$, then *maximum common subschema* between $\mathcal{S}$ and $\mathcal{T}$ is called the best matching result. It is means that the more individuals of two schemas are matched, the more accurate schema matching is.

Maximum common subschema. *A common subschema of two schemas $\mathcal{S}$ and $\mathcal{T}$ consists of a subschema $\mathcal{S}_{sub}$ of $\mathcal{S}$, and a subschema $\mathcal{T}_{sub}$ of $\mathcal{T}$ such that $\mathcal{S}_{sub}$ and $\mathcal{T}_{sub}$ are isomorphic. The maximum common subschema of two schemas is a common subschema that is not a proper subschema of another common subschema.*

In Table 2, we summarize the relations between matching cardinality and homomorphisms.

[8] REMARK 9. For schema homomorphism, if the mapping of two individuals preserves partial semantics, we can define two individuals are matched. However, for schema equivalence, we define that two individuals are semantically equivalent provided that the mapping preserves all the semantics between them. In other words, if the mapping between two schemas is the strongest isomorphism, then they are equivalent.

Table 2
Matching cardinality and homomorphisms

Matching cardinality		Homomorphisms
one-to-one(equivalent matched)	$1:1$	*schema isomorphism*
one-to-one(partially matched)	$1:1$	*subschema isomorphism*
one-to-one(best matched)	$1:1$	*maximum common subschema*
many-to-one	$n:1$	*schema epimorphism*
many-to-many	$n:m$	*schema homomorphism*

7. Computational Complexity of SMP

In this section, we address the algorithmic complexity of SMP. All SMPs studied here are clearly in NP.

For practical matching algorithm, we need an internal schema model, which is an instance of the meta-meta model. For the most part the labeling functions of schemas are either unary or binary, from Remark 3, the schema can be transformed into labeled graph (attribute graph) model. In practice, there are various labeled graph models, such as OIM, OEM, and DOM, etc. For later reference, suppose that two schemas are represented by labeled graphs.

For using labeled graph as internal schema model, SMP reduces to labeled graph matching, which are the classic combinational problem (Hell, 2003), and have been researched deeply in various areas, such as pattern recognition, image processing (Bunke, 2000; Champin and Solnon, 2003), and knowledge reasoning (Mugnier, 2000; Chein and Mugnier, 1992). By label graph model, we investigate the algorithmic complexity of SMP.

Theorem 2. *The schema matching problem is NP-complete.*

Proof. Two schemas are represented by labeled graphs. Suppose that we restrict the matching cardinality is $1:1$ or $n:1$. If the labels are all semantic-related, all the vertices are labeled the same notation, so do the edges, then SMP reduces to the *digraph matching/homomorphism*, given two digraphs $G=(V,E)$ and $G'=(V',E')$, is there a mapping from V to *V'* that satisfies: for all u,v of V, if (u,v) belongs to E then $(\varphi(u),\varphi(v))$ belongs to *E'*? This problem is NP-complete since it contains as a particular case of the NP-complete problem "clique" (Garey and Johnson, 1979, GT19; Chein and Mugnier, 1992).

From Theorem 2, we have the following corollary.

COROLLARY 2. The subschema matching problem is NP-complete.

Proof. By Definition 4, if we restrict the subschema is $\mathcal{S}$, the schema matching problem is a particular case of subschema matching. From Theorem 2, the subschema matching is NP-complete.

PROBLEM 1: Schema monomorphism (injective schema matching)
INSTANCE: *Given two schemas $\mathcal{S}$ and $\mathcal{T}$ in labeled graph model.*
QUESTION: *Is there a semantic monomorphism from $I^{\mathcal{S}}$ to $I^{\mathcal{T}}$?*

Theorem 3. *The schema monomorphism problem is NP-complete.*

Proof. It admits as a particular case of the classic NP-complete problem "isomorphic subgraph" (Garey and Johnson, 1979, GT48). The schema monomorphism problem asks for whether there exists a subschema $\mathcal{T}_{sub}$, $\mathcal{S}$ and $\mathcal{T}_{sub}$ is isomorphic.

PROBLEM 2: Subschema isomorphism
INSTANCE: *Given two schemas $\mathcal{S}$ and $\mathcal{T}$ in labeled graph model.*
QUESTION: *Is there a semantic isomorphism from $\mathcal{S}_{sub}$* to $\mathcal{T}_{sub}$?

Theorem 4. *The subschema matching problem is NP-complete.*

Proof. Two schemas are represented by labeled graphs. If we add the constraint that the vertices have the same label, and edges have not labels, then the problem becomes equivalent to the normal directed subgraph isomorphism, which is a NP-complete problem. Just it is a particular case of "isomorphic subgraph" (Garey and Johnson, 1979, GT48).

If $\mathcal{S}$ and $\mathcal{T}$ are two labeled trees, then subschema isomorphism can be solved in polynomial time (Chein and Mugnier, 1992; Garey and Johnson, 1979). If $|I^{\mathcal{S}}| = |I^{\mathcal{T}}|$, then the problem transforms into graph isomorphism, which is still an open problem.

PROBLEM 3: Schema isomorphism
INSTANCE: *Given two schemas $\mathcal{S}$ and $\mathcal{T}$ in labeled graph model.*
QUESTION: *Is there a semantic isomorphism from $\mathcal{S}$ to $\mathcal{T}$?*

If we add the constraint that the vertices have the same label, and the edges are unlabeled, then the problem becomes equivalent to the one of the existence of an isomorphism between any two digraphs. The graph isomorphism is NPI problem, if $\mathrm{P} \neq \mathrm{NP}$ (Garey and Johnson, 1979). One approach is to define it as being its own complexity class, *isomorphism-complete*. On the other hand, *if a partial order on labels is defined*, in (Chein and Mugnier, 1992), they prove the *labeled graphs isomorphism* is NP-complete. The particular case of LGI is polynomial equivalent to the NP-complete problem "isomorphic partial graph". Then, we obtain the following property.

Property. *If a partial order on labels is defined and the isomorphism has to be compatible with this order, then SISO becomes NP-complete. Otherwise, SISO problem is isomorphism-complete.*

Theorem 5. *Maximum common subschema problem is NP-complete.*

Proof. As in the proof of Theorem 2 and 4, two schemas are represented by labeled graphs, we add the constraint that the vertices have the same label, and edges have not

Table 3
Computational complexity of SMP and its subclasses

SHOMs / SMPs	Complexity Class
Homomorphic SMP	*NP-complete*
Isomorphic SMP	*Isomorphism-complete*/*NP-complete* (Chein and Mugnier, 1992)
Subschema isomorphism	*NP-complete* (Garey and Johnson, 1979)
Maximum common subschema	*NP-complete* (Garey and Johnson, 1979)
Monomorphic SMP	*NP-complete* (Garey and Johnson, 1979)
Epimorphic SMP	*NP-complete*
Stable marriage matching	P (Garey and Johnson, 1979)

labels, then the problem becomes equivalent to the normal maximum common subgraph, which is a NP-complete problem (Garey and Johnson, 1979, GT49).

If the underlying labeled graphs of two schemas are trees, the maximum common subschema problem can be solved in polynomial time. In addition, if we do not concern the relations of schemas, i.e., we define that two individuals are matched only need satisfy Condition 1 in Definition 2 and Definition 5, and do not concern Condition 2, then SMP is reduced to a famous problem – stable marriage problem, also called stable marriage matching, which can be solved in polynomial time (Garey and Johnson, 1979). The stable marriage matching between two schemas can obtain $1:1$ matching result.

Table 3 shows the basic conclusions of computational complexity of SMPs.

8. Conclusions and Future Work

A formal framework for SMP is very important because it facilitates the building of algorithm model and the evaluation of algorithms. Therefore, the main point of this paper is to develop an algebraic framework for generic schema matching.

In this paper, we have five contributions: First, since the schemas are the finite structures over the specific signatures, we propose the meta-meta model of schema, which called *generic matching-oriented* model. This definition has a distinctive feature – it is able to describe any particular style of schemas, and transforms a schema and other matching information into a finite structure over specific signature. The signature is a collection of individual, label, relation and function symbols. Second, we demonstrate that SMP is equivalent to finding the schema homomorphism between two schemas. Then, the algebraic framework for generic schema matching is developed, which is the main contribution of this paper. Thirdly, we show a new classification of schema matching in the context of SHOM (Definition 5) and individual matching (Definition 2). This algebraic framework is able to characterize SMP elegantly. Later, we discuss matching cardinality and some subclasses of homomorphism. We present the close relations between SMP and homomorphisms, and also present the important concept – schema equivalence in

algebraic framework: two isomorphic schemas are equivalent. Finally, using the labeled graph as internal schema model, we investigate the algorithmic complexity of SMP.

It is well known that homomorphism is a useful model for a wide variety of combinatorial problems dealing with mappings and assignments (Hell, 2003). In this framework, SMP is transformed into a semantic homomorphism problem, which can guide practitioner to design the effective algorithms for SMP and evaluate the algorithms. This is the main reason why we develop this formal framework. Because SMPs are NP-hard problem, we can use the approaches of combinatorial optimization, such as neural network, machine learning, local search, and other advanced optimization techniques to solve SMP. These optimization approaches are widely used to solve graph homomorphisms or graph matching problems (Bunke, 2000). For example, Champin and Solnon (2003) and Sorlin and Solnon (2005) designed a greedy and a tuba search algorithm to solve $n : m$ multi-labeled graph matching. Zhang *et al.* (2005, 2006) proposed a hybrid search method to implement schema matching based on the similarity of multi-labeled graph. In particular, for achieving large-scale schema matching, we have to design fast and effective algorithm. We can use local search, and tabu search, etc., *meta-heuristic* algorithms to find the more accurate matching result.

For SMP, most approaches are restricted in $1 : 1$ and $n : 1$ matching (Rahm and Bernstein, 2001). Few methods can obtain $n : m$ mappings, so that we will investigate the $n : m$ matching algorithms in the near future. The proposed algebraic framework is available for characterizing the many-to-many mappings. On the basis of this framework, for realizing the $n : m$ matching, we are going to use approximate matching techniques. For instance, if the matching algorithm uses the labeled graph as internal schema model, we can design the approximate matching algorithms that based on the basic graph homomorphism or isomorphism approaches (Champin and Solnon, 2003; Hell, 2003; Melnik, 2004; Mugnier, 2000; Sorlin and Solnon, 2005), to achieve $n : m$ matching. In particular, by use of the approximate matching algorithms, we may obtain the matching result more accurate and fast.

In addition, a homomorphism problem can be polynomially transformed into a constraint satisfaction problem (CSP) and $H-$coloring problem (Gottlob *et al.*, 2001; Hell, 2003; Mugnier, 2000). We can research SMP in CSP framework. Moreover, Frigioni *et al.* (2001) considered the dynamic version of some polynomially solvable CSPs, and presented solutions that are better than recomputing everything from scratch after each update. The dynamic or incremental algorithms of SMP will be investigated shortly.

Acknowledgments. This work was supported in part by the National 973 Information Technology and High-Performance Software Program of China under Grant No.G1998030408. This paper is the extended version of WAIM 2005. We would like to thank the anonymous reviewers for their careful work in evaluating an earlier version of this paper.

References

Alagic, S., and P.A. Bernstein (2001). A model theory for generic schema management. In *DBPL01*, *LNCS 2497*. Springer-Verlag. pp. 228–246.

Artale, A., E. Franconi and F. Mandreoli (2003). Description logics for modelling dynamic information. *Logics for Emerging Applications of Databases*. Springer.

Berlin, J., and A. Motro (2001). Autoplex: Automated discovery of content for virtual databases. In *CoopIS 2001*, *LNCS 2172*. pp. 108–122.

Bouquet, P., B. Magnini, L. Serafini and S. Zanobini (2003). A SAT-based algorithm for context matching. In *Modeling and Using Context*, *LNAI 2680*. pp. 66–79.

Bunke, H. (2000). Graph matching: Theoretical foundations, algorithms, and applications. In *Proc. Vision Interface 2000*. pp. 82–88.

Champin, P.A., and C. Solnon (2003). Measuring the similarity of labeled graphs. In *ICCBR 2003*, *LNAI 2689*. Springer-Verlag. pp. 80–95.

Chein, M., and M. Mugnier (1992). Conceptual graphs: fundamental notions. *Revue d'Intelligence Artificielle*, **6**(4), 365–406.

Do, H.H., and E. Rahm (2002). COMA – a system for flexible combination of schema matching approaches. *VLDB 2002*.

Do, H.H., S. Melnik and E. Rahm (2002). Comparison of schema matching evaluations. In *Web Databases and Web Services 2002*, *LNCS 2593*. pp. 221–237.

Doan, A., P. Domingos and A. Halevy (2003). Learning to match the schemas of data sources: a multistrategy approach. *Machine Learning*, **50**, 279–301.

Dubhashi, D.P. (1995). Complexity of logical theories. *BRICS LS-95-5*. Department of Computer Science, University of Aarhus.

Federa, T., and M.Y. Vardi (1998). The computational structure of monotone monadic SNP and constraint satisfaction: a study through datalog and group theory. *SIAM J. Comput*, **28**(1), 57–104.

Federa, T., F. Madelaineb and I.A. Stewartc (2004). Dichotomies for classes of homomorphism problems involving unary functions. *Theoretical Computer Science*, **314**, 1–43.

Frigioni, D., A. Marchetti-Spaccamela and U. Nanni (2001). Dynamic algorithms for classes of constraint satisfaction problems. *Theoretical Computer Science*, **259**(1–2), 287–305.

Garey, M., and D. Johnson (1979). *Computers and Intractability – a Guide to the Theory of NP-Completeness*. W.H. Freeman & Co. New York, NY, USA.

Gottlob, G., N. Leone and F. Scarcello (2001). Hypertree decompositions: A Survey. In *Mathematical Foundations of Computer Science 2001*, *LNCS 2136*, Springer-Verlag. pp. 37–57.

Hell, P. (2003). Algorithmic aspects of graph homomorphisms. In *Combinatorics 2003*. London Math. Society Lecture Note Series 307, Cambridge University Press. pp. 239–276.

Hull, R. (1986). Relative information capacity of simple relational database schemata. *SIAM Journal on Computing*, **15**(3), 856–886.

Hull, R. (1997). Managing semantic heterogeneity in databases: A theoretical perspective. In *PODS 1997*. pp. 51–61.

Hull, R., and R. King (1987). Semantic database modeling: survey, applications, and research issues. *ACM Computing Surveys*, **19**(3), 210–260.

Li, W., and C. Clifton (2000). SEMINT: a tool for identifying attribute correspondences in heterogeneous databases using neural network. *Data and Knowledge Engineering*, **33**(1).

Madhavan, J., P.A. Bernstein and E. Rahm (2004). Generic schema matching with cupid. In *VLDB 2001*.

Melnik, S. (2004). *Generic Model Management – Concepts and Algorithms*. LNCS 2967, Springer.

Miller, R.J., Y.E. Ioannidis and R. Ramakrishnan (1994). Schema equivalence in heterogeneous systems: Bridging theory and practice. *Information Systems*, **19**(1), 3–31.

Mitra, P., G. Wiederhold and J. Jannink (1999). Semi-automatic integration of knowledge sources. In *Proc. of Fusion '99*. Sunnyvale, USA.

Mugnier, M.L. (2000). Knowledge representation and reasonings based on graph homomorphism. In *Conceptual Structures: Logical, Linguistic, and Computational Issues*, *LNAI 1867*, Springer. pp. 172–192.

Paolini, P., and G. Pelagatti (1977). Formal definition of mappings in a data base. In *Proc. ACM SIGMOD Intl. Conf. on Management of Data*. pp. 40–46.

Rahm, E., and P.A. Bernstein (2001). A survey of approaches to automatic schema matching. *The VLDB Journal*, **10**, 334–350.

Shvaiko, P. (2006). Iterative schema-based semantic matching. University of Trento. *Technical Report DIT-04-020*.

Sorlin, S., and C. Solnon (2005). Reactive tabu search for measuring graph similarity. In *GbRPR 2005*. pp. 172–182.

Zhang, Z., H.Y. Che, P.F. Shi, Y. Sun and J. Gu (2005a). Multi-labeled graph matching – An algorithm model for schema matching. In *ASIAN 2005*, *LNCS 3818*.

Zhang, Z., H.Y. Che, P.F. Shi, Y. Sun and J. Gu (2005b). Multispace matching – A framework for schema matching. Technical report.

Zhang, Z., H.Y. Che, P.F. Shi, Y. Sun and J. Gu (2006). Formulation schema matching problem for combinatorial optimization problem. *International Journal of Interoperability in Business Information Systems*, **1**, 9–32.

WordNet. `http://wordnet.princeton.edu/`

Z. Zhang received the BS and MS degrees in mechanical engineering from Sichuan University in 1998 and Southwest Jiaotong University in 2002. In 2006, he received his PhD degree in pattern recognition and intelligent system with Institute of Image Processing and Pattern Recognition, Shanghai Jiaotong University, China. He is currently working at SAP Labs China. His research interests include metadata management, semantic interoperability, information/data integration, web services, and approximation algorithm.

P. Shi received the bachelor's and master's degrees in electrical engineering from Shanghai Jiao Tong University (SJTU), Shanghai, China, in 1962 and 1965, respectively. In 1980, he joined the Institute of Image Processing and Pattern Recognition (IPPR), SJTU. During the past 25 years, he worked in the area of image analysis, pattern recognition, and visualization. He has published more than 90 papers. He is currently the director of the Institute of IPPR at SJTU and a professor of pattern recognition and intelligent systems on the Faculty of Electronic and Information Engineering. He is a senior member of the IEEE.

H. Che received the BS and MS degrees in electrical engineering from Beijing Normal University in 1998 and 2001, respectively. In 2006, he received his PhD degree in computer science with Institute of Software, the Chinese Academy of Sciences, China. Now he is working for the Software Center, Bank of China. His research interests include trust management, software engineering, and P2P networks.

J. Gu received the BS degree in electrical engineering from the University of Science and Technology of China in 1982 and the PhD degree in computer science from the University of Utah in 1989. He has been the associate editor-in-chief of the IEEE Computer Society Press Editorial Board, an associate editor of the IEEE Transactions on Knowledge and Data Engineering, the IEEE Transactions on VLSI Systems, the Journal of Global Optimization, the Journal of Combinatorial Optimization, and the Journal of Computer Science and Technology, and is on the advisory board of International Book Series on Combinatorial Optimization. He was a chair of the 1995 National Academy of Sciences Information Technology Forum and was a chair of the 1996 National Science Foundation special event in celebration of 27 years of research on the satisfiability problem. He is a member of the ACM, the ISA, the ISTS, the INNS, a senior member of the IEEE, and a life member of the AAAI.

Algebrinė ontologijos atitikimo schema

Zhi ZHANG, Pengfei SHI, Haoyang CHE, Jun GU

Formali ontologijos atitikimo uždavinio schema yra svarbi, nes palengvina algoritminio modelio kūrimą ir algoritmų įvertinimą. Šiame straipsnyje išvystyta algebrinė ontologijos atitikimo schema. Pirmiausia, remiantis universalia algebra pasiūloma ontologijos meta-meta struktūra, kuri vadinama daugelio žymių ontologija. Vėliau pristatomas formalus ontologijos atitikimo apibrėžimas, vadinamas daugiareikšmiu atitikimu. Tada ontologijos atitikimo uždavinys formalizuojamas homomorfizmo uždaviniu ir įrodoma, kad nagrinėjamas uždavinys yra ekvivalentus semantinio homomorfizmo nustatymui. Tai leidžia sukurti algebrinę uždavinio schemą ir algoritminį modelį. Dar vėliau parodomas ontologijos atitikimo klasifikavimas, pagrįstas algebrine schema. Galiausia, aptariamas atitikimo matmenų ir homomorfizmo poklasių ryšys.