Capturing Semantics by Object Relativity

William Wei Song
SYSLAB, Department of Computer and Systems Sciences
Royal Inst. of Technology and Stockholm University
and
Swedish Institute of Systems Development (SISU)
Electrum 212, 164 40, Kista, Sweden
E-mail: william@sisu.se

Abstract. To acquire the semantics of objects in schemata and hence to identify the objects is becoming a crucial research problem in schema integration. It is an effective and feasible way to make use of a group of schemata (schema family) to capture the meaning/concept of objects. In this paper we analyze the various relations between a given object and its related objects in the existing schemata and construct a characteristic set for the object. The relations include attributes, relationships and mapping constraints, contexts, and some specific inter-object relations like generalization, etc. The characteristic set so generated for an object can uniquely identify the object. We also attempt to apply the obtained characteristic set for the semantic conflict detection and resolution.

Keywords. view integration, schema integration, semantics, entity-relationship model, conceptual schema design, data dictionary

1 Introduction

Schemata are the products of an activity, called conceptual database design, which uses the semantic model to describe data in an abstract and understandable manner. Schema integration is the activity of integrating the schemata of existing or proposed databases into a global, unified schema [2]. Nowadays, the research on schema integration, has gained increasing attentions because of its significant effect on methods for conceptual information modelling. The aim of schema integration is either to create a global schema, which takes in all the possible user's views, and makes these views consistent; or to provide a common access or interface to local schemata in which the user's diverse requests (or views) are represented.

Usually, the process of schema integration consists of the following five iterating steps, canonization (pre-integration), comparison, conciliation, merging (integration), and restructuring. 1) In the canonizing step the intra-schema conflicts and inconsistencies are detected against some pre-defined criteria or requirements, such as "an entity type is not allowed to connect to an attribute by a relationship". 2) The comparison step performs a pairwise comparison of objects of the schemata to be integrated and finds possible object pairs which may be semantically similar. The object pair set so generated is called the semantic similarity relation with respect to some properties, such as synonym, equal key attributes and equal contexts. 3) In the conciliation step, a variety of user assisted techniques are used to resolve conflicts and mismatched objects. 4) The merging step generates an integrated schema from two component schemata. 5) In the last step, restructuring, the objective is to check the consistency of the integrated schema and build correspondences between the component schemata and the integrated schema [13].

One of the main tasks of schema integration is to detect semantic conflicts among objects from schemata or find out semantic similarities between objects. Due to the differences between the user and designer perceptions of the reality, semantic conflicts arise from time to time in the schema evolution and integration. These conflicts must be solved with specific methods, which are not considered in ordinary information system and database design methodologies.

It is apparent that a common problem both to the processes of schema design and integration is that of acquiring and applying semantic knowledge. In the process of schema design, the designer chooses some words to symbolize the concepts in the reality of interest. He also

chooses suitable words to reflect the associations between the concepts. He has still to choose some words to display the attributes of entities. Therefore, so generated schemata contain three basic components, entities, which represent the conceptual meaning of the objects in the real world, relationships, which describe the association between the entities, and attributes, which give the possible features of the entities. These schemata, which represent some portions of the reality of interest, are usually intersected and interrelated.
In the process of schema integration, the components of schemata are contrasted to and compared with each other. The aim of such contrasts and comparisons is to find the semantic similarity between schema objects for the subsequent object integration or to detect the semantic conflicts among schema objects for the schema semantic integrity. At present, these contrasts and comparisons are mainly fulfilled by human (such as DBA). He uses his experiences and knowledge (including both syntactic and semantic) to interpret the schemata to be integrated and further perform integration on schemata. In addition to the user's subjective understanding of the schemata to be integrated, he also uses the semantic relations which exists between the object types in the previous schemata and his previous integration assertions.

1.1 Problems to be addressed

One of the major problems of schema integration is the semantic knowledge acquisition from the existing schemata and the representation of such acquired knowledge in some way. It is difficult to precisely and exactly acquire the semantics of the objects in schemata, since it involves the correct interpretation of people's perceptions (views) of the reality, the correct representation of these perceptions (views) by using schemata, and the correct re-appearance of these perceptions after schema integration [12].
Key problems, addressed in this paper, are how to obtain and apply the semantic knowledge which exists in the previous schema design for the use of subsequent schema integration and semantic integrity checking. In schemata, entities are assigned certain conceptual meanings. Such meanings stipulate that an object is different from the other objects. Similarly, the conceptual meanings of relationships stipulate that what entities can be the domain and what entities can be the range of the relationships. Still as the like, the attributes of an entity may tell the distinction of this entity from the other entities. Besides, many schemata are modelled by an extended ER modelling tool. This extension will give some additional semantics. For instance, generalization offers a new association (supertype) between entities.
Consequently, the capture of the schema object semantics should focus on the existing schemata and make use of the semantic relationships among the schema objects for establishing the characteristic set of the objects. In the paper we analyze the various relations between a given object and its related objects in the existing schemata and construct a characteristic set for the object. The relations include attributes, relationships and mapping constraints, contexts, and some specific inter-object relations like generalization, etc. The characteristic set so generated for an object can uniquely identify the object. We also attempt to apply the obtained characteristic set for the semantic conflict detection and resolution. To our knowledge, little research work has been focused on this aspect of schema integration issues.

1.2 Related work

Since the mid eighties, the research direction in schema integration has been gradually changed. It can be featurized as 1) the entity relationship model or some extension of it is considered as the major schema design model [10, 13]; 2) the in-depth structure (the role of attributes, mapping constraints, etc.) of schemata is focused on [8, 9, 13, 15]; and 3) the semantics of schema objects is emphasized [13]. Recently, the research on data or semantic dictionary is emerging [1, 4, 11]. In our opinion, research on semantic of schema object types consists in the basis of the semantic dictionary while the development of semantic dictionary is undoubtedly the result of the semantic analysis on schemata. A comprehensive survey of early view integration research results can be found in [2].

An incisive analysis of schema integration semantics is made in [6], where some of the methods are analyzed for the database syntax and semantics which populate and can differ among the resources of an enterprise information system. These distinctions are caused by a number of factors, including the different perspectives of the schema designers, the different representations of the reality concepts by the users, etc. The author emphasizes that the database semantics should be taken into account at the early stages of the database design process.
It is stated in [3, 5] that the view integration is a semantic unification which takes into account the domains of atomic objects. Semantic relations are based on the name relation, domain relation, and structure relation of objects. The result of the semantic unification of two concepts is represented by a multiple information called similarity vector whose interpretation of the different components will vary from the semantic equivalence, the semantic similarity, to the semantic dissimilarity.
A deep analysis of the semantic similarity relations between schema objects is presented in [13]. Based on such semantic similarity relations, the user is supposed to have more decision information (suggestions) for the assertion makings during the schema integration. The similarity relation between two objects is obtained through the examination of their names, their attributes, their relationships, and their contexts. Such similarity relations are further categorized into 4 groups: weak semantic relation, compatible semantic relation, equivalence semantic relation, and mergeable semantic relation.
A methodology for data dictionary design is proposed in [1, 11]. The methodology is based on the concept of local area schema, a data description realized with a given model, at different levels of abstraction. Data abstraction is realized through the introduction of refinement, which is a mechanism allowing for modelling the same portion of reality in terms of several schemata at different levels of abstraction. Refinements are formally defined through the introduction of transformations from a source schema to target one. The methodology for data dictionary design performs a multilevel integration starting from the chain of refinements of local schemata and the sequence of transformations representing them, and generates an integrated schema.
Consequently, it is widely accepted that semantic relations in schemata play an important role in the schema integration. Even more important is the use of such semantics already-existing in the schemata. However, current research is restricted to obtaining such semantic knowledge from the current, isolated schemata to be integrated and to temporarily applying the semantic knowledge obtained only for the current integration work. To our knowledge, little attention has been paid to the capture of semantic knowledge existing in a group of schemata which are designed for a certain application domain and closely related to one another. We name such a group of schemata a schema family. In this paper, we attempt to take into account the semantic knowledge which exists in a schema family and to present an approach to expand the knowledge during the process of schema design and integration for an application domain.

1.3 An overview of the paper

The acquisition of the semantics from the existing schemata is a key problem of correct understanding of the current schemata to be integrated. In this paper, we are going to propose an approach to schema semantics capture for schema integration, which makes use of various relations existing between different schema objects, such as attributes, relationships and mapping constraints, object contexts, generalization, etc., to extract the semantic knowledge (i.e., characteristic set) for objects. we also extend this concept to the semantic capture of objects from a group of schemata, we call it **schema family**. In the next section, we present some assumptions as the basis on which object semantics is expressed. We will discuss the relations among a concepts, its name (or word), and its physical occurrences. We define entity types, relationship types and attribute types in schemata as concepts which are represented by names. In the section 3, we analyze the functions of various relationships and contexts of a given entity type and investigate their contribution to the characteristic set of the given entity type, as well as the contribution from a schema family, and hence form some characteristic set expansions. The detection of possible semantic conflicts in terms of the entity characteristic set is observed in section 4, where three typical semantic conflicts are taken into account, i.e., synonym, homonym and cyclic generalization. We conclude the paper in the last section, the section 5, and discuss future work in this area.

2 Some Basic Concepts

2.1 Motivation

Two major steps of the database design process are: conceptual schema design and physical database design. The corresponding products are respectively conceptual schémata and databases which contain instances (the facts in the reality). For example, let us consider modelling Department and Employee, and the relationship works-at between them. Two possible schemata can be designed to model the case (see Fig.1), and the result database is shown in Table 1.

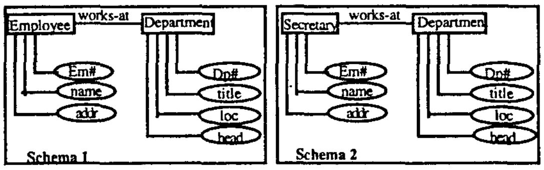


Fig.1 The schemata for employee, secretary and department.

Employee

Em#	name	addr
600102	John	addr1
600202	Mary	addr2

Department

Dp#	title	loc	head
1111	CS	loc1	John
2222	Ph	loc2	Mary

Table 1 An example of instances of the schema 1.

Comparing the two schemata, 1 and 2, one may possibly jump to the conclusion that the entity 'Secretary' could be identical to the entity 'Employee', which is hint by the facts that both the entities have the same attribute 'Em#', 'name' and 'addr', and the same context 'works-at-Department'. In other words, by attributes, relationships, and contexts of entities, we can capture the semantic of the entities of interest and identify them. Database design involves two aspects of matters: the concepts appearing in the database schema and the instances or the facts in the concrete database. Table 1 gives a part of instances of the proposed schema in Fig.1. The concepts reflect the understanding of the users or the designers to the reality and are used to describe the reality. The instances are the physical occurrences of the concepts in the real world. As a matter of fact, we have a third component, words (or names), which symbolize the concepts. In other words, an object in schema corresponds to three components: its name (word), a concept the word represents, and a set of instances the concept refers to. The relationships between them are illustrated in Fig.2. In the database schema design and integration, concept capture and representation are the main problems to be attacked. In the rest of this section we give some assumptions, based on which we can discuss the semantic properties of objects.

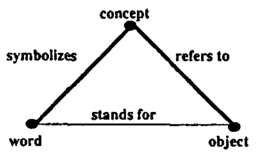


Fig.2 The meaning triangle

2.2 Words and concepts

Schemata are designed to describe a portion of the real world, its perceived concepts and the associations among the concepts. This portion of the real world is called the universe of discourse, denoted U. We assume that the universe of discourse U is monolithic, i.e., no association existing between U and outside U. All the elements contained in U are concrete objects[1] (in contrast to abstract, conceptual objects) for a certain application domain in the reality. We still assume that the schemata defined on U are homogeneous, i.e., designed for a certain particular area of interest and by the same modelling tool. To make such assumptions is to better investigate the contribution of schema family to the schema semantic capture. These concrete objects are grouped into classes. Each class corresponds to a concept in the concept set C. A class in U can be further divided into sub-classes, which also correspond to sub-concepts in C. A concept is symbolized by a word (or name) in the word set W. In the following, we will define these relations between them.

Definition 2.1 Let W and C be the word set and the concept set respectively. We define that symbolize is a mapping from W to C,

symbolize : W -> C,

so that **symbolize**(w) = c, $c \in C$ and $w \in W$.

When the mapping **symbolize** maps two different words to one concept, the two words are considered *synonymous*. When it maps one word to two different concepts, the word is considered *homonymous*.

Definition 2.2 Given that C is the concept set and U the universe of discourse, which contains the concrete objects, we define that **extension** is a mapping,

extension : C -> 2^U,

so that **extension**(c) = {o1, o2, ..., on}, $c \in C$ and $oi \in U$.

The concept of a word is also called the intension of the word. The *intension* of a word is that part of meaning that accords with general principles in semantic memory. The ***extension*** of a word is the set of all existing things (instances) to which the word applies [14]. In the other words, the intension of a word is a set of properties, a set of characteristics which uniquely identify a concept represented by the word. During the comparison step in schema integration process, one of the main tasks is to identify words. Because the difficulties of capturing the intension of words, this work is mainly carried out by the user. Our aim is to acquire the intension of words through investigating the interrelations of schema objects.

2.3 The characteristics of a word (or a concept)

There are three approaches discussed in [14] to capture the semantics (concept) of a word: 1) using the type definition, 2) using a prototype and 3) using schemata. *Type definition* of a word (concept), in general, is to give its genus and differentia. In a simpler way, we may explain it as that to define a word is to find out its super-concept (of course symbolized by a word) and one or several properties which differ the word to be defined from the words which represent the rest concepts within the super-concept. However, it is rather difficult to find a suitable super-concept and the properties which exactly identify the word to be defined[2]. *Prototype* approach is to recognize a word (concept) against an existing set of properties. When the word resemble all the properties, it is considered to be identified. However, the problem here is how to obtain the needed properties and according to what criteria to group the properties.*Concept schemata* are considered as a kind of means to acquire the properties of a word (or a concept). For each concept symbolized by a word, schemata describe the conventional, normally occurring, or default roles that it plays with respect to other concepts. Comparing to the type definition method, where for example, the definition of Employee will present the primary (identically, uniquely) defining characteristic, a schema would include the

[1]We use the 'concrete object' to indicate an individual instance (or fact) in the real world, e.g., Mr. Smith, in contrast to the 'abstract object' , e.g., person. We adopt the term object for the latter.

[2]In addition, normally the definitions are in natural languages, therefore difficult to deal with by computer.

neighbouring information about Employee such as having an employee number, earning salary, reporting to a manager, working in a department, and so forth, all of which together form the properties of Employee and may uniquely identify it.

Definition 2.3 (Characteristic function and characteristic set[3]**)** For each concept c in C, there is a function $\lambda: C \rightarrow 2^{\Sigma}$ (Σ is a set of all properties for all the concrete objects in U), that $\lambda(c) = \sigma$, σ can uniquely identify the concept c. We call λ the *characteristic function*, σ the *characteristic set*. This set is assumed to contains all the characteristics (or properties) of the concept c.

The intension of a word indicates the nature of the word and therefore indicates the common nature of all the objects in the word extension, i.e., the object type. Therefore, we may conclude that if two words have the same intension, their corresponding object types in the schemata can be integrated. In the section 3, we will discuss in detail how to form the characteristic set of a word from the schemata to be considered and the users' assertions given during the schema integration.

2.4 Schema and its components

As known to us, schemata are designed by some semantic models (for example, the ER model). Semantic models attempt to provide more powerful abstractions for the specification of database schemata. They allow the designers to think of data in ways that correlate more directly to how data arise in the real world and to model data at a higher, conceptual level [7]. ER model is one of such semantic models. The schemata created by this model support the representation of abstract sets of entities, relationships which establish some conceptual associations between entities, and attributes to describe the features of entities. In this paper, we apply an extended ER model, called ER+, which has been defined in [13]. However, for the aim of capturing semantics of the objects (i.e., entities, relationships, and attributes) we only focus on the parts of an object, which are dedicated to semantics, and rule out the other information attached to the object. Fundamentally, we consider the name and the characteristic set of an object.

Definition 2.4 (Object type) An object type O is a triple:

<name(O), λ(O), **extension**(O)>,

where

1) name(O) is the name of the object type O,

2) λ(O) is the characteristic set of O, which uniquely identifies O, and

3) **extension**(O) is all possible instances of O.

Definition 2.5 (Entity type) An entity type e is an object type. We assume that no entity type in a schema is isolated. That is, any entity type in a schema must be related to at least one other entity type by a relationship type.

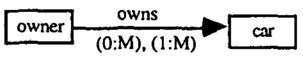
Definition 2.6 (Relationship type) A relationship type **r** is an object type, which, expressed by an arrow, connects an entity type, the domain of the relationship, with one or more entity types, the ranges of the relationship. Each relationship type has a *mapping constraint*, described by the predicate

rel_map(r, [(M1:N1), (M2:N2)]),

where **r** is the relationship type, (M_1:N_1) represents that the domain entity must participate in at least M_1 and at most N_1 instances of the relationship type **r**, and (M_2:N_2) indicates that the range entity must participate in at least M_2 and at most N_2 instances of the relationship.

For example, the mapping constraint of the relationship type 'owns' in the schema

[3]We use characteristic set and semantic set as synonym.



indicates that an owner may have zero up to many cars and a car must be owned by at least one owner.

Definition 2.7 (Attribute type) An attribute type A is a an object type. An attribute type must be associated to an entity type and an entity type must have at least one attribute. The entity type is called the domain of its attributes. Each attribute type has also a *mapping constraint*, denoted by a predicate as

attr_map(A, [(M1:N1), (M2:N2)]),

where A is the attribute, and the mapping constraint [(M1:N1), (M2:N2)] shares the same explanation as in the definition of relationship type.

Definition 2.8 (Key Attribute) Let **extension**(A) and **extension**(E) be the extensions of an attribute A and an entity type E respectively. If there is one-to-one mapping between **extension**(E) and **extension**(A), and the element number of **extension**(E) is the same as that of **extension**(A), it is said that A is the key attribute of E.

This definition implies that some attribute (i.e. key attribute) can uniquely determine an entity type.

Definition 2.9 (Schema) A schema is a meaningful conceptual structure (or graph), which has four components:

$$S = \langle S\_name, E', R', A' \rangle,$$

where S_name $\in$ W, $E' \subseteq$ **E**, $R' \subseteq$ **R**, and $A' \subseteq$ **A**.

Assumption The entity type set **E**, the relationship type set **R**, and the attribute type set **A** together partition the object type set and are pairwisely disjoint.

This assumption intentionally excludes the structural conflicts, such as a word is used for an entity type in one schema and an attribute type in another.

These definitions are graphically illustrated by the following meta-schema (Fig.3).

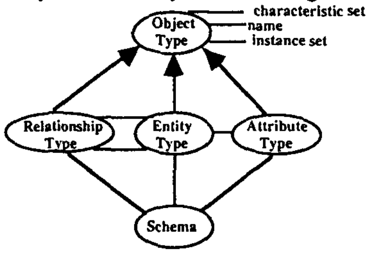


Fig.3 A meta-schema for the interrelations of schema components.

In addition to the general relationships described above, there are three relationships which are the elements of the above stated relationship types and play a special semantic role in the schema design and integration. They are generalization, coverage, and aggregation. Thus, it is necessary to define them separately.

Definition 2.10 (Generalization) Let c1 and c2 be two concepts. The concept c1 is a generalization of c2, denoted <c1, c2> $\in$ **Gen** if **extension**(c2) $\subset$ **extension**(c1). **Gen** is a binary relation[4].

In terms of this definition, we can directly jump to the theorem as follows:

[4]The generalization relation sometimes is represented as **is-a**(c2, c1), which indicates c1 a generalization of c2. The reverse relation of **Gen** is specialization, denoted **Spec**.

Theorem 1 *The generalization relation Gen is non-reflexive and transitive.* That is, suppose that c1, c2, and c3 are three concepts in C and $<c1, c2> \in$ **Gen**, and $<c2, c3> \in$ **Gen**, then $<c1, c3> \in$ **Gen**.

Definition 2.11 (Coverage) Suppose that c1, c2, ..., cn, are n concepts, and **extension**(ci) are the extensions of ci, i=1, 2, ..., n, the relation coverage can be defined as a binary relation,

Cov: C -> 2^C, C is the concept set, so that

$<c1, \{c2, \ldots, cn\}> \in$ **Cov** if

extension(c1) = **extension**(c2)$\cup$ **extension**(c3)$\cup$ ...$\cup$**extension**(cn),

and ***extension****(ci)* $\cap$ **extension**(cj) = $\emptyset$, i, j = 2, ..., n, $i \neq j$.

3 The capture of characteristic set

The existing schemata provide us with a list of entity types, a list of relationship types which associate the entity types, and a list of attributes which describe the entity types. The integration activities provide us with the assertions which are formalized into a so called knowledge base. All these inter-object type relations provide us the semantic interpretation of the object types in schemata. In other words, we can build the characteristic set of an object type and then expand the characteristic set with the help of such inter-object type relations. Although the characteristic set so obtained may not contain all the features and may not be sufficient to identify an object type, they will lead to at least a correct and exact interpretation of the object type. The integration activities could be built on such solid basis -- a relatively full, correct, and exact interpretation of the meaning of the object type. In this section, we propose a set of methods for the obtainment and expansion of the characteristic set of a given entity type. Let us first consider to capture the characteristic set of an object type within **one** schema. Without loss of the generality, we consider just entity types in the schema. Suppose that e is an entity type in schema S. We denote the characteristic set of e $\lambda_S(e)$, which are obtained only from the schema S.
Then, a broader context, schema family, is taken into account for capturing the characteristic set of an entity type. Suppose that an entity type e occurs in a group of schemata **S1, S2, ..., Sn** (schema family **F**). We denote the characteristic set of e $\lambda_F(e)$, which are obtained only from the schema family **F**.

3.1 Capture $\lambda_S(e)$ from attributes

The attributes of an entity type play an important role in understanding the meaning of the entity. Any pair of object types whose identifying attributes can be integrated can themselves be integrated [9]. This statement claims that the attributes of an entity can in some way give the intension of the entity. In one place, an attribute (say, key attribute) or a group of attributes can determine its entity, for example, security number vs. person while in the other place, a group of attributes can partly determine an entity. That is, the group of attributes may at least constitute a part of the intension of the entity. For example, the attribute 'name' gives an intension of a concept. All the objects which have 'name' will be the extension of the concept.

Definition 3.1 (Key semantic expansion) Let $\lambda_S(e)$ be the characteristic set of entity type e in a schema S. If **k** is the key attribute of e, then $\mathbf{k} \in \lambda_S(e)$.

Definition 3.2 (Attribute semantic expansion) Let $\lambda_S(e)$ be the characteristic set of entity type e in a schema S. If a set of attributes {a1, a2, ..., an} of e can uniquely determine e, then $\{a1, a2, \ldots, an\} \in \lambda_S(e)$.

In fact, the definition 3.2 indicates that a set of attribute types may play a role of key which can uniquely identify the entity type they are attached to. For instance (see Fig.4), the entity type 'Book' can be certainly uniquely identified by its key attribute 'B#' (book number), so B# is an element of λ_S('Book'). Assuming that B# is missing from the schema S (This happens more often than not during the schema design), the other three attributes, 'authors', 'edition', and 'publisher' may together uniquely determine 'Book'. In this case, these three attributes as a whole will be an element of λ_S('Book').

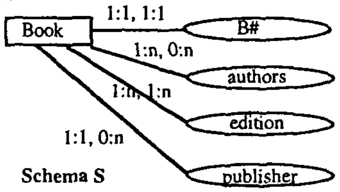


Fig.4 capture semantics of the entity 'Lecturer' from its attributes

3.2 Capture $\lambda_S(e)$ from relationships and their mapping constraints

Relationship types play an important role in stipulating the semantics of the entity types it relates, since a relationship type not only associates two entity types but also dominates them. In other words, a relationship needs a subject which carries out the action represented by the relationship and an object which accepts the action. Therefore the relationship, or the action, demands that its subject should possess some particular features which stipulate a semantic category (the category here means a set, each of whose elements can be a subject of this relationship) of which the subject is an element. Similarly, the object is also an element of a semantic category. The semantic category is intensionally a set of features, and extensionally, a group of things which possess these features. For instance, the relationship type 'attends' in the statements 'author attends a conference' and 'submitter attends a conference' imply that 'author' and 'submitter' as subjects should belong to the same semantic category, which has the feature 'can attend', on one hand. On the other hand, 'attends' stipulates that 'conference' should be an element of the semantic category having the feature 'can be attended'. Inversely, relationships are of course affected by their subjects and objects.

Definition 3.3 (Mapping constraint) Let **r** be a relationship type which has a domain e1 and a range e2, both being entity types. The mapping constraint of r with respect to e1 and e2, denoted [m1:m2, n1:n2], indicates that
1) for each element in **extension**(e1), there are at least m1 and at most m2 elements in **extension**(e2) participating in the relationship r; and
2) for each element in **extension**(e2), there are at least n1 and at most n2 elements in **extension**(e1) participating in the relationship r.

Definition 3.4 (Relationship semantic expansion) Let $\lambda_S(e)$ be the characteristic set of entity type e in a schema S. If the entity type e is related to another entity type e' with a relationship type **r** and the mapping constraint is [1:1, 1:1], then $\{name(r), \lambda_S(e')\} \in \lambda_S(e)$, where name(r) is the name of the relationship type **r** and $\lambda_S(e')$ is the characteristic set of entity type e'.

This definition states a fact that an entity type can be uniquely identified by a given relationship and a known entity type which participates in the relationship when there is a one-to-one mapping between the extensions of two entity types with respect to the relationship. For example, a schema contains such components as 'Dean administrates Department'. The mapping constraint of this particular 'administrates' is [1:1, 1:1]. Suppose that we have already known the key of the entity type 'Department' to be 'Dept#', according to the definition, we can uniquely identify the entity type 'Dean' by 'administrates' and 'Department'.

3.3 Capture $\lambda_S(e)$ from contexts

As we discussed above, the relationships of an entity can give some characteristics (or part of intension) of the entity, whereas the contexts of the entity can give more refined characteristics of the entity set which the entity is contained. A context of an entity e' is a set, whose element is an ordered pair, <r, e>, where r is a relationship and e is an entity. Obviously, a context of e', <r, e>, specifies a set of entities, while e' is an element of the entity set. Consider the example 'Driver drives a Car'. The relationship 'drives' and the entity 'Car' together determine a group of entities, each of whose elements has the property 'able to drive a car'.

Definition 3.5 (The context of an entity type) Let e be an entity in schema S and e is related to a set of entities e1, e2, ..., en through a set of relationships r1, r2, ..., rn. The context of e, denoted **Context$_S$**(e), is the set {<ri, ei> | i = 1, ..., n}.

Definition 3.6 (Context semantic expansion) Let $\lambda_S(e)$ be the characteristic set of entity type e in schema S and the context of e be **Context$_S$**(e). If **Context$_S$**(e) can uniquely identify the entity type e, then the set {<name(ri), λ_S(ei)> | i = 1, ..., n} $\in$ $\lambda_S(e)$, where name(ri) is the name of the relationship ri and λ_S(ei) is the characteristic set of entity type ei, for i = 1, 2, ..., n.

For instance (see Fig.5), schema S contains three entities 'Lecturer', 'Department' and 'Course'. The entity 'Lecturer' has the context **context$_S$**('Lecturer') = {<'works-at', 'Department'>, <'gives', 'Course'>}. Then λ_S('Lecturer') includes {<'works-at', 'Department'>, <'gives', 'Course'>}.

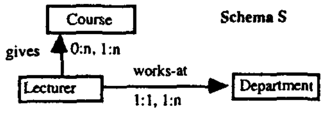


Fig.5 capture semantics of the entity 'Lecturer' from its contexts

3.4 Capture $\lambda_S(e)$ from the special relationships

3.4.1 Generalization

Definition 3.7 (Generalization semantic expansion) Let e1 and e2 be two entity types in schema S and λ_S(e1) and λ_S(e2) the characteristic sets of e1 and e2 respectively. If <e1, e2> $\in$ **Gen**, then λ_S(e1) is included in λ_S(e2).

This definition indicates the fact that a sub-concept inherits all the characteristics of its super-concept. For instance (see Fig.6), in the schema S, we have the entity 'Lecturer', which is sub-concept of the entity 'Employee'. Then it inherits all the characteristics of 'Employee'. That is, λ_S('Lecturer') is expanded to include {Em#, salary, <'works-for', 'Organization'>}.

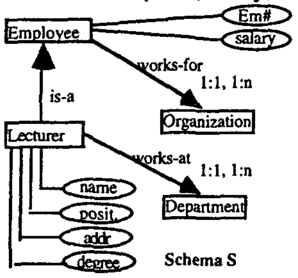


Fig.6 capture semantics from generalization

Note that during the process of this semantic expansion of λ_S('Lecturer'), a new semantic relationship between, for example, the entities 'Organization' and 'Department' may be discovered. Hence, the characteristic sets for 'Organization' and 'Department' should be formed for the identification of this new relationship. Possibly, the two entities have synonymous names, or one is the generalization of the other, etc.

3.4.2 Coverage

Definition 3.8 (Coverage semantic expansion) Let e be an entity and e1, e2, ..., en be a group of entities in schema S. $\lambda_S(e)$ is the characteristic sets of e and $\lambda_S(ei)$ the characteristic sets of ei, i = 1, 2, ..., n. If $<e, \{e1, e2, ..., en\}> \in$ **Cov**, then

1) $\lambda_S(e)$ is included in $\lambda_S(ei)$; and

2) $\cap\lambda_S(ei)$ is included in $\lambda_S(e)$, for i = 1, 2, ..., n.

The first conclusion of the definition indicates the fact that a sub-concept inherits all the characteristics of its super-concept. The second conclusion is that the characteristics that are possessed by all the sub-concepts should be the characteristics of the super-concept. As a matter of fact, it is still the inheritance put in another way. A schema appears in Fig.7, which model a part of a library information. In a library, we keep the publications which can be broken into three disjoint classes: books, journals, and newspaper. That is, the entity 'Publication' covers the entities 'Book', 'Journal', and 'Newspaper'.

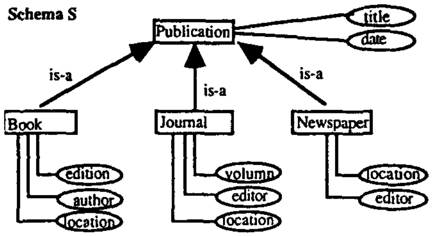


Fig.7 capture semantics from generalization

According to the definition 3.7, λ_S('Book') should be expanded to include λ_S('Publication'), and so should λ_S('Journal') and λ_S('Newspaper'). Since the attribute 'location' belongs to all the covered entities, the property {location} should be included in λ_S('Publication').

3.5 Capture $\lambda_F(e)$ from schema family

Conceptual schemata can be considered to be a very effective approach to the capturing of the characteristics of an object, where the objects related to the given object along with the relationships are taken into account for the contribution of the object characteristic set. The $\lambda_S(e)$ obtained from a single schema S for an entity is only reflecting a lopsided view of a concept, whereas the characteristic set $\lambda_F(e)$ obtained from a group of schemata **F** may reflect a relatively manifold views of a concept. This is rather crucial in schema integration. However, since several schemata are considered in order to obtain the $\lambda_F(e)$ for an entity, the pre-condition is necessary that these schemata should be consistent, i.e., no semantic conflicts existing between them.

The semantic conflicts include 1) synonym, where two different words represent the same concept, 2) homonym, where one word represents two different concepts, and cyclic generalization, where the concept A is a sub-concept of the concept B in one schema while in another schema B is a sub-concept of A. When a semantic conflict is detected, a semantic dictionary or the user's assertion is required to resolve the conflict. In the semantic dictionary, we maintain a list of synonyms for looking up when necessary [13]. A detailed definition and discussion for these conflicts will be given in the next section.

After all the semantic conflicts detected among the schema family are eliminated, the $\lambda_S(e)$ of the entity e can be expanded to include $\lambda_{S_i}(e)$ into $\lambda_F(e)$, S_i is a schema in the schema family. The $\lambda_F(e)$ can be defined as follows.

Definition 3.9 (Collectives semantic expansion) Suppose that the entity e occurs in a group of schemata $S_1, S_2, \ldots, S_n$ and the characteristic sets of e in these schema are respectively $\lambda_{S1}(e), \lambda_{S2}(e), \ldots, \lambda_{Sn}(e)$. If $S_1, S_2, \ldots, S_n$ are consistent, then $\lambda_F(e) = \lambda_{S1}(e) \cup \lambda_{S2}(e) \cup \ldots \cup \lambda_{Sn}(e)$.

For instance, the entity Employee is common in two schema S1 and S2 in Fig 8. The characteristic set of Employee in schema 1 is λ_{S1}('Employee') = {<works-at, Department>, Em#}, and that of Employee in schema 2 is λ_{S2}('Employee') = {<works-for, Project>, Em#}. Hence, the characteristic set of Employee in the schema family (combining schema 1 and 2) is λ_F('Employee') = {<works-at, Department>, <works-for, Project>, Em#}.

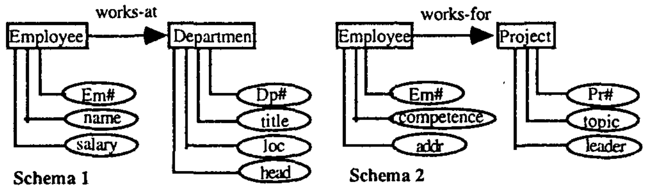


Fig. 8 Schemata of department and project

3.6 A summary of this section

We have observed that the intension, i.e., characteristics of an entity can be acquired from its attributes, its relationships with other entities, and its contexts in a single schema. Through eliminating the semantic conflicts among schemata and unifying the characteristic sets obtained from these schemata, the intension of a common entity can also be acquired. It is more significant to form the characteristic set of an entity by considering a group of schemata. In addition, by capturing the schema family semantics, we may maintain the schema design and integration history. That is, the schema family (assuming it is content after removing conflicts) can be used as a new semantic dictionary in broader sense for the detection of new schema design and integration. Therefore, the detection and resolution of the semantic conflicts become crucial in this sense. In the following section, we will discuss how to apply the characteristic sets to the conflict detection and hence the suggestions for the user's assertions when the conflicts are detected.

4 Some examples of using the semantic set

The aim of capturing the characteristic set of an entity is to support the resolution of the semantic conflicts which are found during schema integration. The semantic conflicts arise because of the diversity of the people's understanding of concepts and the diversity of their representations in names (or words) in schemata. Semantic conflicts can be seen as the incorrect or inappropriate usage of words for concepts which are established during people's percepting the reality (or part of the reality). The characteristic sets obtained by the methods discussed in the last section are mainly applied, in the schema comparison step, to detect the semantic conflicts. The semantic conflicts can break into these cases: synonym, homonym, and cyclic generalization. When the semantic conflicts are found some suggestions to indicate the causes resulting in the conflicts and the possible resolutions of the conflicts should be provided for the user's assertions.

4.1 The detection of synonym

The semantic conflict case - synonym occurs when the same concept is represented by two or more words (names).

Definition 4.1 Let w1 and w2 be two different words of the word set W respectively. If there exists the equation **symbolize**(w1) = **symbolize**(w2), the words w1 and w2 are considered to be synonymous.

An example appears in Fig.9, where two schemata are designed to describe the departments in a university.

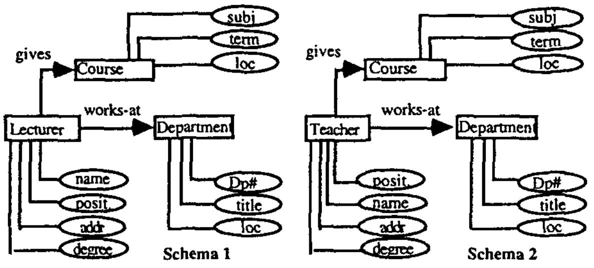


Fig.9 Example of synonym

The words 'Lecturer' and 'Teacher' are adopted to represent the same group of people who have degrees, work at a department, and advise courses, etc. (the same concept in the real world). Keeping two distinct entities in the integrated schema would result in modelling a single concept by means of two different entities.

1) Detection of synonym. Synonymous words can be detected by comparing the semantic sets of the words. Suppose that $\lambda_1(e_1)$ and $\lambda_2(e_2)$ are two characteristic sets and e_1 and e_2 are two different entity types in two different schemata 1 and 2. If $\lambda_1(e_1) = \lambda_2(e_2)$, e_1 and e_2 have the great possibility to be synonymous since they share almost all the characteristics in common. Consider again the above example, the semantic set of the entity 'Lecturer' is λ_1('Lecturer') = {<name, position, addr, degree>, <works-at, Dept>, <gives, Course>}, and the semantic set of the entity 'Teacher' is λ_2(Teacher) = {<name, position, addr, degree>, <works-at, Dept>, <gives, Course>}. Here we assume that <name, position, addr, degree> can determine 'Lecturer'.

Since for each element in λ_1('Lecturer'), there is a same element existing in λ_2('Teacher'), we may conclude that the entities 'Lecturer' and 'Teacher' are synonymous. However, It is quite often that not all elements in one semantic set can be found in another semantic set and vice verse. That is, for example, λ_1('Lecturer') and λ_2('Teacher') have majority of common elements. Then, the suggestion that 'Lecturer' is synonymous to 'Teacher' is proposed to the user for confirmation.

2) Suggestion. After the synonym conflict is detected, the suggestion that two entities be synonymous is presented, along with their characteristic sets, to the user for confirmation. When the user asserts that the two entities are same, then one preferable name is required to replace the names of the two entities and a new item is recorded in the semantic dictionary with the fact that the two entities are synonymous. At last, the two entities are integrated with the union of their characteristic sets.

4.2 The detection of homonym

The second semantic conflict case is homonym, which arises when the same name (word) is used to represent two different concepts in different schemata.

> **Definition 4.2** Let w be a word in the word set W and c1 and c2 be two concepts in the concept set C. If w is mapped into two different elements of C, i.e., **symbolize**(w) = c1 and **symbolize**(w) = c2, then w is a homonym.

Consider the example in Fig.10, where two entities 'Equipment' in schema 1 and 'Equipment' in schema 2 obviously refer to different concepts, but are represented by the same name. Merging the two entities in the integrated schema would result in producing a single entity for two distinct concepts.

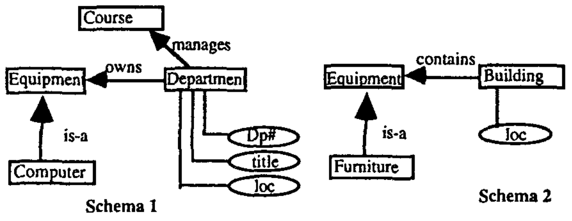


Fig.10 Example of homonym

1) Detection. By analyzing the semantic sets of the entities with the same name, we can find out whether the homonym case occurs. Suppose that $\lambda_1(e1)$ and $\lambda_2(e2)$ are two characteristic sets and e1 and e2 have the same name in two different schemata. If $\lambda_1(e1) \neq \lambda_2(e2)$, e1 and e2 can be considered to be two distinct concepts since they have no characteristics in common. Consider again the above example, the semantic set of the entity 'Equipment' in schema 1 is λ_1('Equipment') = {<Department, owns>, <has-sub-concept, Computer>} and the semantic set of the entity 'Equipment' in schema 2 is λ_2('Equipment') = {<Building, contains>, <has-sub-concept, Furniture>}.

Since there is no element which belongs to both λ_1('Equipment') and λ_2('Equipment'), we may conclude that 'Equipment' is homonymous and a suggestion should be given to rename one the entities.

2) Suggestion. When a homonym conflict is detected, the user is required to assert whether the two entities refer to different concepts (perhaps they refer to the same concept). The assertion making is of course supported by presenting the user with the characteristic sets of the entities. After the user's confirmation that the two entities are homonymous, a rename operation is triggered to give a more suitable name to either of the entities.

4.3 The detection of cyclic generalization

The semantic conflict, cyclic generalization, happens when in one schema <e1, e2> $\in$ **Gen** and in another schema <e2, e1> $\in$ **Gen**, and the two schemata are merged into one This case often occurs, in particular, when the schemata are designed by different designers with different perspectives and views on the reality. This conflict must be eliminated during schema integration; otherwise a cyclic generalization will appear in the integrated schema, which apparently violates the transitivity of the generalization.

> **Definition 4.3** Let e1 and e2 be two entities. When that there exist both the generalization relations <e1, e2> $\in$ **Gen**[5] and <e2, e1> $\in$ **Gen**, a cyclic generalization occurs.

1) Detection. Suppose that $\lambda_{S1}(e1)$ and $\lambda_{S1}(e2)$ are two characteristic sets and <e1, e2> $\in$ **Gen** in schema S1, and $\lambda_{S2}(e1)$ and $\lambda_{S2}(e2)$ two characteristic sets and <e2, e1> $\in$ **Gen** in schema S2. According to the Definition 3.7, $\lambda_{S1}(e1)$ should be included in $\lambda_{S1}(e2)$ and $\lambda_{S2}(e2)$ should be included in $\lambda_{S2}(e1)$. By comparing $\lambda_{S1}(e1)$ and $\lambda_{S2}(e1)$ (assuming that $\lambda_{S1}(e1) = \lambda_{S2}(e1)$ before the ***generalisation expansion***), immediately, we have the contradiction that $\lambda_{S1}(e1)$ is a subset of $\lambda_{S2}(e1)$. The same contradiction happens when comparing $\lambda_{S1}(e2)$ and $\lambda_{S2}(e2)$ (assuming that $\lambda_{S1}(e2) = \lambda_{S2}(e2)$ before the generalisation expansion).

2) Suggestion. As soon as a cyclic generalization is found, this generalization chain will be presented to the user along with all the characteristic sets associated with the entities in the

[5]According to the theorem 1, e1 can be an indirect generalization of e2. That is, there may exist a entity e', so that <e1, e'> $\in$ **Gen** and <e', e2> $\in$ **Gen**.

generalization chain. Due to misuse of some concepts, and/or some words, the user's assertions are necessary to either re-consider more precise concepts or rename some words which will precisely represent the concepts.
In conclusion, by applying the characteristic sets of entities, not only can the semantic conflicts be detected, but some suitable suggestions can be presented to the users for confirmation as well. In particular, when such characteristic sets of entities are acquired from a family of schemata, the conflict detection proposed will be more powerful and the suggestions presented to the users will be more precise.

5 The conclusion and future work

To acquire the semantic of schemata is difficult but significant a research area both for schema design and integration. By investigating the surroundings of schema objects to capture the schema semantics is one of the approaches to the schema semantic capturing. In this paper, we have discussed some methods to acquire the semantics of schema objects in schemata in terms of their attributes, their relationships and their contexts, and hence to form the characteristic sets for the schema objects. In our opinion, this approach is feasible to explore, to a large extent, the semantic associations of objects existing in schemata.
The contributions in the paper are 1) to investigate the semantic association of schema objects; 2) to present an approach to the obtainment of the object semantics and to the expansion of the semantics; 3) to use the characteristic sets obtained for the detection of the semantic conflicts; and 4) to pave a road to the further study of the semantic knowledge acquisition for schema design and integration.
There are still a lot to do in the semantic knowledge search for schema integration. Our future work is intended to deeply explore the semantics existing in the schema objects, not only entity types but relationship types and attribute types as well, and to form a semantic dictionary based on the semantic knowledge so obtained. Such semantic knowledge acquisition, in our opinion, can be done not only from the schema families, but also from the users assertions during schema integration.
The semantic dictionary, which was discussed in [1, 13], can be considered to be a powerful tool for schema integration since it plays the role of a carrier both for semantic knowledge explored in schema design and integration, as well as for semantic knowledge obtained from the user's assertions for the schema integration. However, the forming and structure of the semantic dictionary have not yet been well studied.
In addition, little focus has been put on the quantitative analysis of the semantic similarity of the integrating schemata. This is perhaps another bottleneck problem (schema semantic knowledge acquisition is one) of the generation of semi-automatic, even automatic schema integration.

The author would like to thank Prof. Janis A. Bubenko, jr. for his valuable comments on the previous version of the paper.

Reference

1. Batini, C., G. Di Battista and G. Santucci. Representation Structures for Data Dictionaries. Dept. of Informatics and Systematics, The university of Rome. Report No. 09.91. Sept. 1991.
2. Batini, C., M. Lenzerini and S. B. Navathe. A Comparative Analysis of Methodologies for Database Schema Integration. ACM Computing Surveys. 18(4): 323-364, 1986.
3. Bouzeghoub, M. and I. Comyn-Wattiau. View Integration by Semantic Unification and Transformation of Data Structures. the 9th Int'l Conf. on Entity-Relationship Approach. Lausanne, Switzerland. 1990
4. Bubenko, J. J. Knowledge for Schema Restructuring and Integration Tools. SYSLAB, DSV. Internal Working Note. SYSLAB IWN No.1. Dec. 1985.

5. Comyn-Wattiau, I. and M. Bouzeghoub. Constraint Confrontation: An Important Step in View Integration. the 4th Int'l Conf. CAiSE. Manchester, UK. 1992
6. Howe, G. A. A Collision of Semantics. DP&D. 93(2) pp.54-61, 1993.
7. Hull, R. and R. King. Semantic Database Modelling: Survey, Applications and Research Issues. ACM Comput. Surveys. 19(3): 201-260, 1987.
8. Johansson, B.-M. and C. Sundblad. View Integration: A Knowledge Problem. Department of Computer and Systems Sciences, University of Stockholm. Working Paper. SYSLAB WP No. 115. 1987.
9. Larson, J., S. Navathe and R. Elmasri. A Theory of Attribute Equivalence in Database with Application to Schema Integration. IEEE TOSE. SE-15(4): 1989.
10. Navathe, S. and S. G. Gadgil. A Methodology for View Integration in Logical Database Design. the 8th Int'l Conf. on VLDB. Mexico City, Mexico. 1982
11. Pirri, F. and C. Pizzuti. Data Dictionary Design: A Logic Programming Approach. The 11th Int'l Conf. on the Entity Relationship Approach. Karlsruhe, Germany. 1992
12. Song, W. W. and P. Johannesson. Schema Integration: Present and Future. ICYCS'93. Beijing. 1993
13. Song, W. W., P. Johannesson and J. J. Bubenko. Semantic Similarity Relations in Schema Integration. the 11th Int'l Conf. on the Entity Relationship Approach. Karlsruhe, Germany. 1992
14. Sowa, J. F. Conceptual Structures: Information Processing in Mind and Machine. Addison-Wesley Publishing Company. 1984.
15. Spaccapietra, S. and C. Parent. View Integration: A Step Forward in Solving Structural Conflicts. Laboratoire de Bases de Donnees, Dept. d'informatique, Ecole Polutechnique de Lausanne. Research Report. Report No. 1990.