SCHEMA TRANSFORMATION FOR SEMANTIC GEODATA TRANSLATION

Z. Xu[2], Y.C. Lee[1], Y.Q. Chen[3]

[1]Dept. of Geodesy and Geomatics Engineering
University of New Brunswick, Canada
[2,3] Dept. of Land Surveying and Geo-Informatics
The Hong Kong Polytechnic University, Hong Kong
[1]E-mail: yclee@unb.ca
[2]E-mail: 98900969r@polyu.edu.hk
[3]E-mail: lsyqchen@polyu.edu.hk

KEY WORDS: Data exchange, GIS, Interoperability, Object-oriented, Semantics

ABSTRACT

We address the problem of semantic geodata translation in a file-based data sharing environment. By Semantic translation, we mean the translation process that can resolve semantic heterogeneity between source and target data set. First we identify sources of semantic heterogeneity of spatial data. Then a classification of semantic heterogeneity using OO data model as the common framework is introduced to deal with semantic heterogeneity in a systematic way. We propose a schema mapping specification approach to resolve semantic heterogeneity, which is considered to be flexible and efficient for spatial data exchange. OQL is used to specify the schema transformation from source schema to target schema. Examples show that this approach is effective and able to deal with a wide range of semantic heterogeneity. Most of the discussion is also helpful to interoperable GIS.

1. INTRODUCTION

Geodata sharing is of great concern in the geographic information community since collecting geodata is a very costly and time-consuming process. To facilitate data sharing, many data translators have been developed to translate spatial data from one format to another. Such data translators convert data automatically but rather blindly, causing some important information to be lost during the translation. One of the main causes for data lost is due to differences in data model between systems, particularly in their differences in semantic richness. It is now widely recognized that semantic heterogeneities are great impediments to data sharing and interoperability.

There are several levels of interoperability [BI98]. Interoperability at the system level is now manageable thanks to developments in computer network and distributed computing platforms (DCP). In the GIS community, OGIS is the industry-wide effort to achieve both data and service sharing [OGC98]. We feel that most of current research works on GIS interoperability makes use of advances in DCP. In this study, we jump to semantic interoperability while we discuss file-based data sharing which makes no use of advances in system interoperability. The justification of such a treatment is that we believe file-base spatial data exchange will still play an important in spatial data sharing and no matter what form spatial data sharing take the problem of semantic data translation needs to be tackled.

By semantic data translation we mean a data translation process that caters to a special data set and is able to resolve potential semantic heterogeneity. Of course, it is unwise and impossible to write a translating program for every case of data sharing. We have designed a framework for semantic geodata translation, which makes semantic data translation possible and efficient [LX99]. In this paper, we will address the issue of resolving semantic heterogeneity of geodata translation in a file-based data sharing environment.

2. SOURCES AND CLASSIFICATION OF SEMANTIC HETEROGENEITY OF SPATIAL DATA

2.1 Sources of semantic heterogeneity of spatial data

The problem of what are semantic heterogeneities of spatial data has not yet been well answered. The rest of this section tries to answer this question in a systematic way with supporting examples.

Semantic heterogeneity includes differences in the way the real world is modeled in the database, particularly in the schemas of the databases [GSC96]. Before we classify semantic heterogeneity, it is helpful to identify source of semantic heterogeneity of spatial data. Here we have identified three sources for semantic heterogeneity of spatial data, two of which is specific to spatial data.

2.1.1 Semantic heterogeneity resulting from differences between spatial data models

System heterogeneity, and particularly DBMS heterogeneity, may cause semantic heterogeneity by forcing the adoption of different models (relational versus OO, for example). Because of that, a given conceptual relationship may have to be represented by different structures (referential integrity in the relational model versus aggregations in OO) or simply not represented if the model does not support it (specialization structures and methods in OO unsupported in the relational model) [GSC96].

Such system heterogeneity is amplified in the case of spatial data since there are many more spatial data models in GIS applications than traditional applications. A particular GIS might not support all the useful types, thus forcing users to use an inappropriate data type. Many GIS packages assume that a feature type is of the same geometric type. In other

systems, however, a feature type may have an abstract geometric data type that can be instantiated with different geometric attribute types. In yet some spatial data transfer formats one feature can have more than one geometric attribute.

Geometric data type differences are conventionally treated purely as data structure differences and when performing geodata translation, the interaction between spatial data model and application schema is ignored. Because of this, it is clear that spatial data model sometimes should be treated as part of the application schema to enable semantic data translation.

2.1.2 Semantic differences resulting from conceptualization differences

The main reason for semantic heterogeneity is the difference in conceptualization, which results in differences between application schemas. That is, even if the same data model is used to model an application, different application schemas can be created due to different perspectives on the problem, different application requirements, and the many different ways of expressing a concept. Such differences are common to both spatial applications and non-spatial applications. Examples of such differences can be found in the appendix and in section 3.

2.1.3 Semantic difference resulting from scale difference

Some researchers have identified that difference in the scales associated with spatial data is an important source of semantic difference. [UVO99] gave an example in which a road is modeled as a linear feature in one application and area feature in another and an algorithm is presented to convert area road to linear road. Although it is possible to extract the centerline of the road from an area presentation of it, we think that pure geometric processing can hardly give us the result we really want. [DPS98] gave an example in which two applications model the road network differently due to scale difference. A mapping language is designed to integrate the schemas. The advantage and defect of this method will be analyzed in section 4.

In general, we feel that scale difference is a special case of conceptualization difference specific to spatial data and should be treated accordingly. This is because scale difference leads to both spatial resolution difference and semantic resolution difference. To deal with scale difference, data need to be restructured as well as abstracted. Such an abstraction is different from data modeling in that in data modeling we abstract real word objects to database object while in data abstraction we abstract data to a higher level of abstraction. Some analysis on resolving scale differences is given in section 4.

2.2 classification of semantic heterogeneity

To deal with semantic heterogeneity in a systematic way, it is necessary to classify those heterogeneities. Different classifications have been proposed due to the different common data model employed [KCG+93] [KS91] [GSC96]. Classification of semantic heterogeneity of spatial data has been mentioned in [BI98] [OGC98]. However these classifications are conceptual and not enough examples are given to support their classifications. Here we base our discussion on the classification of [GSC96] since we believe OO data model is more appropriate for spatial data than relational data model and, to our knowledge, this is the most complete and systematic classification employing object oriented data model.

The detailed classification of semantic heterogeneities is given in next section together with methods to resolve them. We give here a brief introduction to this classification with highlights on those most significant to spatial data. The semantic heterogeneity between object-oriented schemas is examined from three aspects, heterogeneity between object classes, between class structures, and between object instances.

Class is the most important data element in an OO data model. Heterogeneity between classes includes differences in object extensions, differences in attributes/methods, differences in domains and differences in constraints, each of which has several cases. Differences in domains are distinguished from differences in attributes because domain can be a user defined type/class as well as atomic types in the OO model. The domain difference proliferates in spatial data since spatial data has geometric attributes whose domains are geometric data types.

Heterogeneity between class structures is complex and has many specific cases. This is because OO data model provides rich constructs for defining classes, including generalization/specialization, aggregation/de-aggregation and collection. In our study, we will pay much attention to such heterogeneity of geometric classes since a unique characteristic of spatial data lies is the complexity of geometric data. We think the heterogeneity between class structures together with semantic domain differences constitute major semantic differences of spatial data. Many examples resolving such heterogeneity will be discussed in section 3.

Heterogeneity at the data instance level, i.e. heterogeneity between object instances is an important issue in the case of data integration. However, currently we do not take into account this heterogeneity since we assume a data user is importing data and there is only one data source at a time. In the case of spatial data integration, the differences in spatial data quality will be a major source of data inconsistency.

The examples given in next section will show that this classification of semantic heterogeneity applies well to spatial data for both heterogeneities resulting from spatial data model differences and those resulting from conceptualization differences. As for semantic heterogeneity resulting from scale difference, we will address them in section 4.

3 RESOLVING SEMANTIC HETEROGENEITY THROUGH SCHEMA MAPPING SPECIFICATION

3.1 why schema mapping specification

There are several approaches to resolving semantic heterogeneity, which vary in the level of system coupling [ERS99]. The most tightly coupled approach is to build a global schema shared by all participating databases. The less coupled approached is a federated schema, which protects more autonomy. In both cases, one global schema or more federated schemas need to be built. Building an integrated schema is the bottleneck of these two approaches. The even less coupled approach is accessing multiple heterogeneous databases through multi-database language, in which the user resolves semantic heterogeneity using the multi-database languages. The approach using multi-database language is more flexible and practical but it provides less transparency than the other two.

File-based data sharing can be considered as an extreme form of data sharing in which data is shared without integrating databases. Therefore, it would be more efficient to resolve semantic heterogeneity without performing schema integration. In this study, we introduce a schema mapping specification approach to resolve semantic heterogeneity. The specification describes the transformation from source schemas to target schema.

Semantic translation cannot be achieved without cost, which is the need to resolve semantic heterogeneity by some kind of semantic enrichment since no current data models can capture the full semantics of data. However, what is important is that human interaction and effort should be minimized. We believe the approach of using a declarative language is flexible and efficient in the sense that the work of resolving semantic heterogeneity is simply to declaratively describe the mapping from the source schema to the target schema.

3.2 Assumptions

We rely on the user or data administrator to identify semantic differences and specify them. This is different from many schema integration tools that provide tools to assist the detection of schema differences and automate the process of schema integration and schema mapping specification. For instance, in [NS96] the user need only to assert correspondences between schema constructs and the tool will do the rest. Although such kind of automation is desirable, it is not a trivial task and a human operator is the best for detecting semantic differences and resolving them.

We assume that source data schema and target data schema are given as object oriented schemas. If the original data schemas are of other data models, it is assumed that they have been translated into an object oriented data model. For an introduction of schema translation, i.e. translation of schema from one data model to another data model, please refer to [PTR96]. What is left is the task to transform the source schemas into the target schema. This task is similar to defining the target schema as a view of source schemas.

3.3 Schema mapping specification using OQL

As have been discussed, we need a language to specify the mappings between source schemas and target schemas. A requirement for this language is that it should be declarative. This is an important character since any procedural language will make the specification too complicated to use. Database query languages are appropriate in this sense. For a lack of spatial query language standard, we turn to the Object Query Language (OQL) for the following reasons.

OQL is based on the OO model, which is the old adopted for reasons explained earlier. Most current spatial query languages are based on SQL, which cannot manipulate geometric data easily.

The integration of OQL and programming languages is tighter. OQL is intended as an extension to some object-oriented programming language such as C++, Java and Smalltalk. With this characteristic, supporting functions in the form of methods can be implemented in the programming language without extra effort for data transfer between OQL and programming languages. Such a characteristic is especially important to spatial data because operations on geometric data are usually so complex that functions/methods are needed to implement these operations.

OQL is an attempt to standardize query language for object-oriented database. The brief syntax of OQL is as follows:

select <result-set> *from* <set-definition> {, set-definition} { *where* <expression> }

There are other clauses and operators to assist the select-from-where statement to perform more sophisticated queries, such as *order by*, *group by*, *having* clauses and set/bag/list/array operators. For a complete reference of OQL, please see [CB97].

3.4 Resolving semantic heterogeneity

In this subsection, resolution of semantic heterogeneities using OQL is discussed. We will follow the structure of semantic heterogeneity classification of [GSC96]. For each semantic heterogeneity, a description is given and followed by methods to resolve them and examples where necessary with schema mapping specification in OQL.

A. semantic heterogeneity between object classes

A.1 differences in extensions

Differences in characterization of object: This difference rise when two schemas differ in criterias of what constitutes an object and what is the character that distinguishes object A from B. This difference can only be resolved if there are supporting information on deriving target objects from source objects. For example, if in the source schema a building

can have many towers while in the target schema a building with several towers is considered as several buildings, then we can split the source building into several target buildings if there is information on each tower of source building.

Example 1. The two corresponding classes are:

sBuilding { string name; integer numOfTowers; SET(MIF_Point) poistions;}
tBuilding { string name; MIF_Point poisition;}

The mapping from sBuilding to tBuilding would be:

select tBuilding(name:s.name, position:x) *from* sBiulding *as* s, s.positions *as* ps *where for all* x *in* ps : TRUE

This example shows the resolution of three kinds of semantic heterogeneity. First is the naming difference in both attributes and classes, which is resolved using the mechanism of structuring result in OQL. Second is the multi-valuation difference between the s.Building.positions and t.Building.position. Third is the difference in the characterization of objects. This example also shows that we benefit a lot from OQL's capability to manipulate complex object. Note that we have de-aggregated a sBuilding to a set of tBuildings by addressing the point elements of the set attribute of a sBiulding object. If necessary, the coordinates of the points can be addressed.

Differences in objects included: Two corresponding classes differ in their sets of objects since the contexts are different. There are several kinds of extension relationship: disjoint, overlap, and inclusion, the last can be further divided into equivalence and strict inclusion. This kind of difference is not considered in this study since we assume users are importing data to extent data extension.

Naming differences: Corresponding classes are named differently, synonymy; and different concepts are names identically, homonym. See example 1 for resolving method.

Multiplicity of object differences: One class of database may correspond to several classes in database B. This is similar to the vertical partition case in relational database. This kind of difference can be avoid if B is first intra- integrated before integration with A. This difference can be resolved by selecting attributes of several related source classes into one target class. The prerequisite to do so is that the objects of source classes can be matched by keywords.

A.2 differences in attributes and methods

Present/absent attribute difference Some attributes in a class is missed in its corresponding class and/or vice versa. This difference can be resolved by coercing nonexistent attributes to null, or excluding extra attributes from the *select* clause, or including an expression for missing attribute whose value can be derived or has default.

Temporal differences: The corresponding attributes of corresponding class refer to different time of object states. Currently, we don't consider this difference.

-arity differences: A class attributes corresponds to a method of another class. This difference is not applicable to our study because we don't take methods into account.

Multi-valuation differences: An attribute may be uni-valued in one database, while the corresponding attribute is multi-valued; or although both are multi-valued but the maximum numbers of values are different. Example 1 illustrates one case of how to resolve this difference by de-aggregation. It is also possible in some cases that one of the multi-values is picked up for a single-value attribute. The following example shows the possibility of assign a single-value attribute to a multivalue attribute using a *set* keyword. It is also possible to make a set (containing only one element) from an element using the *element* keyword

Example 2.

s2Building { string name; set(MIF_Point) positions;}

The mapping from tBuilding to s2Building:

Select s2Building(name, positions:set(t.position)) *From* tBuildings *as* t

Null differences: An attribute allows null while its counterpart does not. This difference can be resolved by providing default or dropping the objects.

Default differences: The corresponding attributes have different default values. This difference can be resolved by changing the default.

A.3 Differences in the domains of attribute/results of methods

A.3.1 Semantic domain differences

Differences in object extension : There are three cases of extensional differences as introduced in A.1, namely object characterization, objects included and multiplicity of objects. The resolution of the last difference has been shown in examples 1 and 2. As have been explained, the second case is not relevant. The first case needs extra attention when the object is referenced by other objects. Consider the following example, which is an extension of example 1.

sResident { string name; integer id#; sBuilding liveIn; }
tResident { string name; integer id#; tBuilding liveIn; }

When we derive tResident from sResident it should be noted that the *liveIn* attributes of each class have a semantic domain difference, which is in particular an object characterization difference. Noting this, it is found that we cannot precisely map the sBuilding.liveIn to tBuilding.liveIn because the mapping from sBuilding to tBuilding is one-to-many. Therefore, one needs to make a decision. One possible decision is that we select one from the several tBuilding objects whose name is the same of the source sBuilding objects. The corresponding mapping specification would be:

select distinct tResident(name:sr.name, id#:sr.id#, liveIn:tb)
from sResidents *as* sr, tBuilding *as* tb
where tb.name=sr.liveIn.name

Differences in object identifiers: This inconsistency happens when different schemas use different strategy to manage identifiers. In one case, identifiers may use system-generated characters or integers while another may use keywords. Here, we simply assume that all data formats use keywords as identifiers and the source and target schemas use same keywords.

Numerical/non-numerical differences: A numerical domain corresponds to a non-numerical domain. This kind of differences can be resolved using a mapping function.

Differences in dimension, unit of measure and scale: One may use weight or volume to measure the quality of oil. Different units may be used. The value may be scaled by a factor. This kind of difference can usually be resolved using arithmetic expressions in the select clause. Function can also be used if necessary.

A.3.2 Syntactic domain differences

Syntactic domain difference can be difference in types, length, numerical length, integer length, precision etc. These differences can usually be resolved through type conversion or using functions if the domains are compatible. Type conversion is allowed in OQL since it is strongly typed.

A.4 Differences in constraints

Constraints on one attribute include multi-valuation limits, uniqueness, null allowance and subtypes of a main type as discussed in A.3.1 and A.3.2.

Constraints on several attributes: In generally, this difference can be resolved similarly to constraints on one attribute. Complex constraints result in complex mapping specifications.

B. Semantic heterogeneity between class structures

B.1 Inconsistency along the generalization/specialization dimension

Inconsistency in specialization criteria: two corresponding superclasses may have different criteria of subclassing. This inconsistency can be resolved if there is supporting information in the source schema for reclassification of the superclass into target subclasses. The criteria for reclassification can usually be expressed as a selection statement. The point here is that we go one level up along the class hierarchy. If the target schema is not predefined, some kind of dynamic classes can be generated using the object/type/function schemas proposed by [KL93]. However, this requires extension to OQL.

Inconsistency in specialization degree and characterization: Two corresponding suplcasses may have different numbers of subclasses and they may also differ in characterization of subclassing. Such inconsistency is usual for geometric types. For example, one spatial format has more area types than another does. Example of resolving such inconsistency will be given later.

Inconsistency in the specialization kind: There are four kinds of specialization according to the relationships among extensions of superclass and subclasses: disjoint, complementary, alternative, general. This kind of inconsistency can be treated similar to inconsistency in specialization difference. Still, one can go one level up and reclassify the superclass of the classes being considered.

B.2 inconsistency along the aggregation/decomposition dimension

Simple class versus aggregate class: A class is a simple class while its corresponding class is an aggregated class. For classes with geometric attributes, the class is always an aggregated class. When mapping simple classes to aggregated classes, this inconsistency can be resolved by recursively structuring the output in the *select* clause. In the reverse mapping, the component classes should be flattened using path expression.

Inconsistency in the kind of aggregation: There are three kinds of aggregation: simple aggregation, composition and collection. For geometric classes, the inconsistency in aggregation kind is often encountered. For example, both the SHAPE format and MIF format employ composition aggregation on geometric classes and prohibit simple aggregation. For instance, a line shared by two polygons will be recorded in both polygons. However, SAIF supports the notion of object sharing. However, SAIF does not enforce sharing. Still, some data formats supporting topological relationship may enforce a simple aggregation on topological data types.

It is relative easy to convert simple aggregation to composition aggregation by duplicating the referred objects. However, the conversion from composite aggregation to simple aggregation needs special consideration for spatial data.

For non-spatial data, it would be all right to make the embedded objects exist by their own and substitute them in the aggregated object by their OID or keys. For spatial data, simple composition usually means that topological relationship is built. Simply substituting embedded objects by their OID or keys would violate those implicit topological constraints. At the current stage of our study, we do not consider those cases that need to build topological relationship. Yet, we do think translation of spatial data from format with topological relationship to format without topological relationship or to format with comparable and lower-level topological relationship is simply a kind of schema transformation.

Inconsistency in the aggregated classes: For compositions and simple aggregation only. For two corresponding complex classes, their component classes are not corresponding classes. It has three cases as follows:

Inconsistency by specialization in the aggregated class: A component class is a subclass of corresponding component class. It is always possible to map a class to its superclass. It is also possible to map a class to its subclass depending on the kind of specialization. The following example is the former case:

sBuilding { string name; rectangular position; }
tBuilding { string name; polygon position; }

The mapping from sBuilding to tBuilding could be:

select tBuilding(name, position:polygon(position)) *from* sBuildings

Inconsistency by collection in the aggregated class: A component class is a collection of corresponding component class. Example 1 illustrates the resolution of such a consistency in one direction. The following example illustrates the mapping in the reverse direction*:*

Select sBuilding(name:n, numOfTowers:(*select* count(*) *from partition* p),
positions:(*select* p.t.position *from partition* p))
From tBuildings *as* t *Group by* t.name:n

Inconsistency by composition in the aggregated class: A component class is a composition of corresponding component classes.This inconsistency can be resolved by aggregating the component classes, or by de-aggregating in the inverse case. The following example illustrate mapping in both directions:

SHAPE_PointM { SHAPE_Point position; real measure;}
sWell { integer id#; SHAPE_PointM position ; }
tWell { integer id#; MIF_Point position; real depth; }

Forward mapping: *Select* tWell(id#, position:(position.position).2MIF_Point, depth:position.measure) *From* sWell

Backward mapping: *Select* sWell(id#, position:(SHAPE_PointM((position).2SHAPE_Point:position, depth:measure))) *From* tWell

Where 2MIF_Point and 2SHAPE_Point are function names.

Geometric type conversions need dedicated functions to be defined and implemented. It is important that functions developed for translating data between two specific formats such as MIF and SHAPE can be reused in other cases of translation such as translation between MIF and SAIF and between SHAPE and SAIF. We have suggested in [LX99] and [XL99] a mechanism of defining an integrated spatial schema for reusing these functions. In this paper, we concern only their interaction with application schema.

Inconsistency in the subkind of collection: For collections only. Collection can be of one of four subkinds: disjoint, covering, partitioning and general. A class may be a collection of one subkind while its corresponding class may be a collection of another kind. Consider the following example:

TYPEDEF enum featureType={ tourism, culture, economic}
sFeaturedCounties { featureType feature; set(sCounty) hasCounties; }
sCounty { string name; polygon position; }
tFeaturedCounties { featureType feature; set(tCounty) hasCounties ; }
tCounty { string name; polygon position; }

Suppose that in the source schema a county belongs to one and only one sFeaturedCounties object while in the target schema, a county can belong to zero to three tFeaturedCounties objects. This is one case of inconsistency in the subtype of collection, which is partition versus general. Generally, this kind of inconsistency usually means differences in characterization of aggregated objects and can be resolved if there are supporting information by re-collecting component objects. Actually, the mapping between such source and target classes does not make much sense.

Inconsistency in the component class of collection: For collections only. The class constituting corresponding collections may be different. Consider the following example:

sProvince { string name; set(sMunicipalRegion) hasMunicipalRegion; }
sMunicipalRegion { string name; set(sCounty) hasCounty; }
sCounty { string name; polygon position; }
tProvince { string name; set(tCounty) hasCounties; }

tCounty { string name; polygon position; }

Suppose the collections here are all partition collections. The sProvince and tProvince has such an inconsistency in the sense that sProvince consists of sMunicipalRegion and tProvince consists of tCounty while sMunicipalRegion is not the corresponding class of tCounty. The mapping from sProvince to tProvince is obvious and therefore omitted.

B.3 Schematic discrepancies

It may happen that what is seen as data or value in one data set is considered as meta data or part of schema in another data set. This is called schematic discrepancy. Schema discrepancy is hard to deal with when performing schema integration. In the file-based data exchange case, this can be relatively easily handled. We give examples of such discrepancies without mapping specifications due to length limitation.

Schema specialization discrepancy: Also called value-entity discrepancy. The following example shows such a case. In the source schema class sRoadNode has two subclasses, sTraficLight and sBridge while in the target schema the RoadNode has no subclass but it has an attribute *ofType* to indicate of what type a RoadNode object is.

sRoadNode { point: position; }
sTrafficLight: sRoadNode{…}; sbridge: sRoadNode { …};
TYPEDEF enum RNType={TrafficLight, Bridge}
RoadNode {RNType: ofkind; point: position;}

Schema composition discrepency: Also called value-attribute discrepancy. We have not found sensible example of spatial data for this and next kinds of discrepencies. For example of non-spatial data please see [GS96].

Schema composition and specialization discrepency: Also called attribute-entity discrepancy.

Schematic collection and specialization discrepancy: What is seen as collection in one database is considered as subclass in another database. Followed is such an example. In the source schema a sProvince is a collection of stCounties while in target schema a subclass of stCounty is defined for each province and the example shows only one subclass:

stCounty {…};
sProvince { string: name; counties: set(country);…}
tCountyOfGuangdong::county {…}

Schematic collection and specialization kind discrepancy: In the case of collection versus subclass discrepancy, the kind of collection may be different from the kind of specialization into subclasses.

C. Semantic heterogeneity between object instances/ data inconsistency

We don't take into account this heterogeneity for the reason specified in subsection 2.2.

Based on the discussion and examples given in this subsection, it can be concluded that the semantic heterogeneities identified in subsection 2.2 can generally be resolved using OQL. This is attributed to two facts. First, semantic heterogeneities of spatial data can be well tackled using OO data model as the common framework. Second, OQL is powerful in manipulating complex data, which is based on the so-called object algebra [YM98].

4. RESOLVING SCALE DIFFERENCE USING SCHEMA MAPPING SPECIFICATION

The previous section shows that many semantic heterogeneities of spatial data can be specified using OQL. However, as will be seen, the capability of OQL to deal with semantic heterogeneities resulting from scale difference is limited. The difficulty there are many implicit spatial constraints that guide spatial data generalization, which usually cannot be incorporated into OQL statement. For example, in the context of road network generalization, we may want to eliminate some minor roads by selecting only those that are significant in length or grade as the following statement specifies

select * *from* Road *where* grade<=2 *and* length>5

Although the predicates seem reasonable, this selection may result in a road "network" with some roads not connected to other, violating the basic rule that road network should be connected.

As another example, consider the generalization of land cover information. At a larger scale, there may be four types of land use, which are to be reclassified into two types. Suppose we can easily reclassify the four types of land use into two by examining some of their thematic attributes as the following statement does:

select "AB", position *from* Landuse *where* luType = "A" *or* luType = "B"
union
select "CD", position *from* Landuse *where* luType = "C" *or* luType = "D"

Although the above reclassification is accurate, it does not merge neighbor land parcels that are of same land type.

Such difficulties are also faced by other approaches that deal with some kind of spatial data abstraction. [DPS98] presented a specification language for integrating spatial data of different scales. Their emphasis is on finding corresponding data instances and does not consider the implicit constraints associated with spatial data. To have some

idea of the difficulty of automating spatial data generalization, please refer to [MLW95]. It could be concluded that semantic difference resulting from scale difference is more a data abstraction/generalization problem than a schema transformation problem.

5. CONCLUSIONS

Semantic geodata translation is discussed in this paper. We have identified three sources of semantic heterogeneity, which are spatial data model difference, conceptualization difference, and scale differences. For the first two sources of semantic heterogeneity, we have shown that the classification using OO data model provides the base for tackling semantic heterogeneity in a systematical way. For performing semantic geodata translation, we suggest to use the schema mapping specification approach to resolve semantic heterogeneity between source and target schemas. Such an approach is suitable and effective in a file-based data sharing environment. OQL is employed as the schema mapping specification language for its declarativeness, its tight integration with programming language, and its power to manipulate complex data. For semantic differences resulting from scale difference, we have shown that the major difficulty is to preserve the implicit constraints on spatial data. This is more a data abstraction/generalization problem than a schema transformation problem. The discussion on semantic heterogeneity of spatial data and resolution of these heterogeneities are also helpful to interoperable GIS.

ACKNOWLEDGEMENTS: The work described in this paper was substantially supported by UGC No. B-Q195, the Hong Kong Polytechnic University.

REFERENCES

[BI98] Bishr, Y.A., 1998, *Overcoming the semantic and other barriers to GIS interoperability*, Int. J. Geographic Information Science, Vol. 12, No.4, pp299-314.

[CB97] Cattell, R.G.G. and Barry, D. K. (*eds*), 1997, *The object database standard: ODMG 2.0*, Morgan Kaufmann Publishers, Inc.

[CW94] Chomicki, J. and Litwin W., 1994, W., *Declarative definition of object-oriented multidatabase mappings*, Morgan Kaufmann Publishers, Inc.

[DPS98] Devogele, T. Parent, C. and Spaccapietra, S., 1998, *On spatial database integration*, Int. J. Geographic Information Science, Vol.12, No.4, pp335-352

[ERS99] Elimagarmid, A., Rusinkiewicz, M. and Sheth, A. (*eds*), 1999, *Management of heterogeneous and autonomous database systems*, Morgan Kaufmann Publishers, Inc.

[ES98] ESRI, 1998, *ESRI Shapefile Technical Description* ---An ESRI White Paper

[GSC96] Garcia-Solaco, M, Saltor, F. and Castellanos, M., 1996, *Semantic heterogeneity in multidatabase systems*, In Object-oriented Multidatabase System---A Solution for Advanced Applications (Bukhres, O.A. and Elmagarmid, A.K., Eds.), Prentice-Hall Inc., pp129-202

[GD95] Geographic Data BC, 1995, *SAIF (Spatial Archive and Interchange Format: Formal Definition Release 3.2)*, http://www.elp.gov.bc.ca/gdbc

[KCG+93] Kim, W., Choi, Y., Gala, S. and Scheevel, 1993, *On resolving schematic Heterogeneity in Multidatabase systems*, Distributed and Parallel Database, 1:3, pp. 251-279.

[KS91] Kim, W. and Seo, J., 1991, *Classifying schematic and data heterogeneity in multidatabase systems*, IEEE Computer, 24:12, pp. 12-18.

[LX99] Lee, Y.C. and Xu, Z., 1999, *Design of a geodata translation system*, Dynamic and Multi-Dimensional GIS, IAPRS Vol32. Part 4W12, Joint ISPRS Commission workshop, Oct. 1999, Beijing, China, pp181-188.

[MI96] MapInfo Corp., 1996, *MapInfo Reference Manual*.

[MLW95] Muller, J.C. Lagrange, J.P. and Weibel, R., 1995, *GIS and generalization: methodology and practice*, Taylor & Francis, London

[NS96] Navathe, S. and Savasere, A., 1996, *a schema integration facility using object-oriented data model*, In Object-oriented Multidatabase System---A Solution for Advanced Applications (Bukhres, O.A. and Elmagarmid, A.K., Eds.), Prentice-Hall Inc., pp105-127

[OGC99] OGC, 1999, *The OpenGIS Abstract Specification Model*, http://www.ogis.org

[OGC98] OGC, 1998, *The OpenGIS Guide* (Third Edition, Eds. Buehler, K. and McKee, L.), http://www.ogis.org

[PTR96] Papazoglou, M., Tari, Z. and Russell, N., 1996, *Object-oriented technology for Interschema and language mappings*. In Object-oriented Multidatabase System---A Solution for Advanced Applications (Bukhres, O.A. and Elmagarmid, A.K., Eds.), Prentice-Hall Inc., pp203-250

[UVO99] Uitermark, H, Vogels, A. and Oosterom, P., 1999, *Semantic and geometric aspects of integrating road networks*, Interoperating geographic information systems------Proceedings of Second International Conference INTEROP'99, Zurich, Switzerland, 1999

[YM98] Yu, C.T. and Meng, W.Y., 1998, *Principles of database query processing for advanced applications*, Morgan Kaufmann Publishers, Inc.

[XL99] Xu, Z. and Lee, Y.C., 1999, *A geodata translation system, project report*, The Hong Kong Polytechnic University.