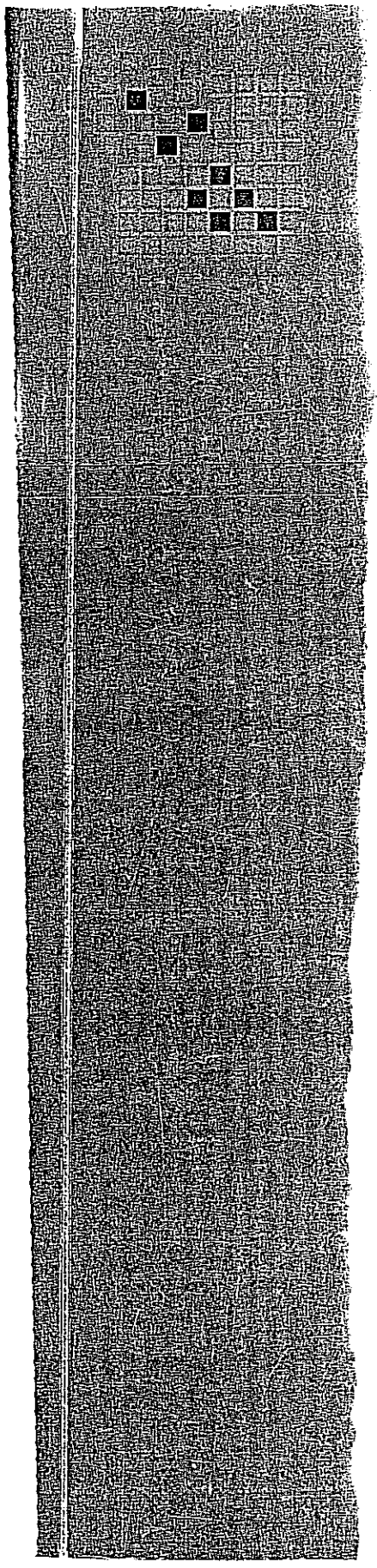


PROCEEDINGS OF THE

4TH INTERNATIONAL SYMPOSIUM ON SPATIAL DATA HANDLING

1990
ZURICH, SWITZERLAND

Vol. 2

USING CATEGORY THEORY TO MODEL GIS APPLICATIONS

by

John R. Herring
Intergraph Corporation
Huntsville, AL 35894-0001, USA
e-mail: ingr!b17a!mobius!jrh@uunet.uu.net
and
Max J. Egenhofer and Andrew U. Frank
National Center for Geographic Information and Analysis
University of Maine
Orono, ME 04469, USA
e-mail: max@mecanl.bitnet frank@mecanl.bitnet

ABSTRACT

The theory of models is extremely important to any scientific endeavor. We build formal models of real-world phenomena, analyze these models, and draw conclusions about the real world. In a GIS, analysis can occur within several conceptual models at once, each model representing a different view or paradigm of the data, such as Euclidean geometry, topological, metric, raster or object models. Their integration into a unifying view is necessary so that users of spatial information may deal with geometric concepts in a manner close to their perception of the "real world."

This paper describes a theory of the internal structure of GIS applications based upon the manner in which they integrate different spatial paradigms into a single unified system. Such a description is useful in the investigation both of GIS software functionalities and data structures. This research resulted from discussions among the authors, and others, during the NCGIA's specialist meeting on "Languages for Spatial Relations", which identified a Category Theory as a promising approach to research this integration. The application of this theory lies in (1) GIS and GIS application design and implementation (2) spatial reasoning and (3) query processing.

INTRODUCTION

Geographic Information Systems (GIS) are in Database Management Systems (DBMS) for both spatial and spatially-associated attribute information (Frank-81, Orenstein-88). As such, they supply decision support through both spatial and non-spatial reasoning, and they are capable of interacting with the user at both descriptive and geometric levels. For user interface, performance, and analysis reasons, GIS's support multiple spatial representations simultaneously, often including one or all of the following:

- a combinatorial topology representation to support spatial analysis (Egenhofer-89, Herring-88),
- a Euclidean geometry representation for simple manipulations and editing (Kemper-87),

- a raster structure for the display and manipulation of graphics and digital imagery (Nagy-79, Samet-84),
- a non-Euclidean geometric representation based upon ellipsoidal and surface models for complex measurements, and
- a unified spatial and attribute query language to support a proposition-based user interface (Chang-80, Ooi-89, Egenhofer-89, Herring-88).

Each method of spatial representation has a fundamental set of spatial concepts, and associated methods for manipulating and organizing spatial objects and the relationships that exist between them. Some of these views are quite precise in their meanings and formal in their syntax (the geometries, formal query languages), while others are informal and fuzzily defined and thus subject to more variation in interpretation (natural languages). The models underlying existing GIS software systems demonstrate difficulties (Guting-88) for instance when reasoning about spatial phenomena which require more than one concept or representation. In such a complex system, a unifying theory is needed to integrate the multiple spatial paradigms.

In investigating the theory of spatial relations, there are three areas for each representational view or paradigm that must be researched. These areas are:

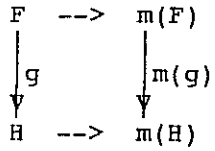
- the logical view of the spatial representations, the way in which space is organized
- the features/objects that are representable within this view
- the spatial operations/relations that can exist for these features within this view, and the rules for their combination, and manipulation (their algebra)

In systems that deal with more than one representational model, the mappings (morphisms, meaning "structure preserving correspondence) between concepts must also be understood, for it is the morphisms that allow questions and answers to pass between the different models. Morphisms further form the basis for the combination of multiple paradigms into a single model, and the mapping of conceptual design models to implementation design models.

CATEGORY THEORY

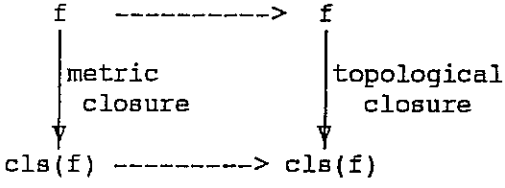
This section sets forth a basic category theory for handling multiple paradigms. This version of category theory is a similar to that used in modeling classical homological algebra (see Casti-89 and Cartan-56).

A "category" is a collection of primitive element types, a set of operations upon those types, and an operator algebra which is capable of expressing the interaction between operators and elements. A "morphism" or, more properly a "functor," between categories associates elements and operations (not necessarily primitive) from one category to those of another which preserves the operator algebra. The usual visualization for such morphisms are commutative diagrams:



This diagram implies that for f in F, m(g)(m(f)) = m(g(f)), that is the two paths from the upper left to the lower right are equivalent.

For example, "metric spaces" form a category whose primitive elements are "points" and whose primitive operation is "distance" and which can be embedded via a natural morphism in "topological spaces" (a category whose primitive elements are "sets" and whose primitive operations include "boundary", "interior" and "complement"). It is this metric-topology morphism that allows topological methods to be used in the spatial analysis of features, represented as sets within a Euclidean metric space (Egenhofer-90). In both categories, there is an operation called "closure". The existence of a morphism between topological spaces and metric spaces, says that given a set, the topological equivalent of its metric closure is the same as the topological closure of its topological equivalent. In a diagram:



Applications of these morphisms are made to spatial query processing and query-optimization theory. Each query-processing goal is achievable by different algorithms within different paradigms. The efficiency of a system depends upon the efficiency of the paradigms, with their related data structures and algorithms, and the efficiency with which the system can translate problems between these paradigms. For example in a GIS supporting geometry and a topological data structure, the efficiency of complex spatial operators depends upon the efficiency with which the system can utilize the topological paradigm in query processing.

In research, the model helps define the possible range of single and multiple paradigm algorithms. In application design, the model gives a basis for data structure definition to support multiple paradigms, diverse cross-paradigm algorithm integration and optimization, user-interface design, and query language implementations. Special emphasis should be placed upon the integration of multiple paradigms in a single system, integrated user interfaces and data-access, query optimization techniques.

Primitive Objects

The features or objects within a geographic model depend on the paradigm (e.g. Gaussian geometry deals with geodesics, curves of

locally shortest arc length curves, instead of straight lines). Some fundamental commonalty of concept applies across many of the models, and most of the paradigms deal at least with the following concepts:

- o points, locations or simple 0-dimensional objects
- o curves or simple 1-dimensional objects
- o polygons, areas or simple 2-dimensional objects
- o complex, homogeneous and non-homogeneous collections of any other features (possibly recursive in definition but not in realization)

Some of this commonalty derives from the embedding of the conceptualizations, such as any Euclidean space is Gaussian, both of them are also metric spaces, and all of them are topological spaces. Other commonalty derives from the morphism and often generates what at first may seem like oxymorons, as in "a raster line" (the morphism image of a line in raster space). Morphism based upon such common geometric or set concepts can be considered as natural.

The Operator Algebra

The operators that exist within a paradigm can be classified according to the domain of their input and returned value. One of the most common operator type is the Boolean or relational type; those that input two simple objects and return a truth value (either TRUE or FALSE). For example, the proposition "A intersects B" can be thought of as an operator equation "intersect(A,B) = TRUE". As Boolean operators, they can be operated on by the standard Boolean combiners AND, OR, XOR, and NOT. Inferences of spatial reasoning can thus be expressed as equations or propositions in Boolean logic, for example:

```
if
  "A contained in B" AND "C disjoint from B"
then
  "A disjoint from C"
```

or in equivalent equation form

```
(contained_in(A,B) AND disjoint_from(C,B)) OR
                    NOT disjoint_from(A,C) =     TRUE
```

Other operators input simple objects or sets of objects, returning sets of objects as in "which rivers intersect this road?" (inputs a "road" and the set of all "rivers", returns a subset of "rivers"). Others may input objects or sets of objects and return sets of geometry as in "where does this river intersect any road?" (may return a set of points or point objects, nodes in a topological paradigm, if the input objects were represented by curves). Other still may return ordered tuples of objects and geometry as in "which rivers intersect which roads and where do these intersections occur?" (returns ordered triples consisting of a "river", a "road" and a piece of geometric information, such as a node in a topological paradigm).

Of course, these classes of operators are inter-related. Statements concerning these inter-relationships form part of the operator algebra. For example, one could use the relationship between the operators "intersect" and "intersection" to make the following propositional algebra statement (IFF is "if and only if" or propositional equivalence):

intersect(river, road) = TRUE IFF
intersection(river, road) is not the empty set

Such equations from the operator algebra can be used to define one operator in terms of other more primitive ones or to support systems for spatial inference or reasoning. Integrity rules, which are often included in model descriptions (Date-83) can be considered as algebraic identities and theorems in the operator algebra.

Morphisms

The translation properties that define how one paradigm can be mapped onto another are collectively called the "morphism." These morphisms must describe how objects, relations, operators and operator algebras map to one another. A morphism must be defined for each pair of views that coexist within a single GIS.

The simplest case of this occurs in a topological GIS, where the topology is calculated form the input geographic coordinate information. During topological structuring, distinguished points and the intersections of the 1-dimensional input elements are associated to nodes, the resulting connected strings to edges, and the polygons to faces. This establishes the geometry-topology morphism for simple objects. The morphism for features and other complex objects is derived from this basic geometric one, so that area features are mapped to collections of faces, line features are mapped to collections of edges, and point features are mapped to collections of nodes.

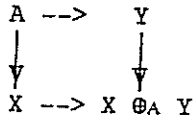
With this in mind, the process of a simple spatial query can be interpreted as a sequence of subprocesses, each within a single paradigm, interspersed with the use of morphisms to map from one paradigm to another. For example, beginning with "find all places where roads intersect rivers, report on the location, and place a bridge at those points" begins in a propositional form. Remaining in this form, a list of roads and rivers are obtained. Translating via the morphism to topology, the roads and rivers become lists of edges. Using the topological paradigm, nodes are found that form the ends of these edges. The road-node list is compared to the river-node list for common points (in yet another propositional paradigm, this process is to as a spatial join, since it can be implemented via the relational join operator). Each such common node represents a location where a river and a road cross, and mapping to the geometric paradigm, the coordinate location, and orientation of the road at that point are determined (enough for the report requested). Returning with this information to the propositional paradigm, a bridge can be created with this location

(associated to the node found) and orientation. Of course some of these morphism shifts are so natural, that implementors seldom realize that they have changed paradigm through the use of a morphism.

The Advantages of the Various Category Model Approaches

There are two basic ways to use the category theory as presented above: synthesis and analysis-translation.

In the synthesis approach, multiple paradigms are combined based upon a common submodel. This process froms a tensor sum of the two base paradigms, with the common submodel parts identified. In terms of morphisms, the following diagram commutes:



This synthesis allows the designer to determine the full conceptual design of the combined category by defining the subalgebra and its morphisms. This is equivalent to defining subclasses and using multiple inheritance in an object-oriented system.

In the analysis-translation approach, a complex paradigm is factored into independent subalgebras. The multiple paradigms are implemented separately with translation paths provided as needed. Procedures are factored among the various categories as appropriate to the implementation efficiency within each category and the translation efficiency between categories. A particular procedure is optimized by determining the path through the paradigms, that, with the appropriate algorithmic and translation cost, minimize total cost.

The costs of shifting from one spatial paradigm to another would at first glance seem to suggest that dealing with them separately might be of little value. This is misleading for two simple but important reasons.

First is complexity. The combined paradigm is sufficiently complex as to make single-approach very difficult. A standard algorithmic technique when presented with such a complex problem is "divide and conquer," based on the assumption that it is often easier to attach a set of simpler problems and aggregate their solution, than to solve the more complex one. By separating each of the simple systems from the main body, a set of problems susceptible to straight forward theoretical attack becomes available. In the case of the more mathematically define paradigms (the geometries and topology) there are large, well-developed and fairly complete mathematical theories and implementations for them. For the propositional paradigms, both data base management and linguistic approaches seem appropriate. Thus, this approach has the advantage that it can take advantage of existing implementations of individual paradigms.

Second is efficiency. Each of the paradigms may result in significantly different data structures, and extremely different algorithms for the solution of the various types of spatial problems. In determining whether of not to support a particular paradigm within an application system, you must decide if the advantages of the paradigm's algorithms outweighs the expense of supporting and maintaining (or calculating) the requisite structures. To make this cost-benefit analyst accurate, you must be able to estimate the cost of maintaining the structures, translation between supported paradigms, and solving homeomorphic problems in the separate systems. For example, supporting real-time topology requires a certain amount of storage overhead for the boundary relation structures, and a processing overhead for maintenance while not supporting it costs heavily in processing time for the repeated calculation of geometric intersection and overlay for spatial analysis, and represents an hidden cost in system flexibility.

Application Design

Standard software system design requires the definition of at least three categories (models): the conceptual model containing the high level system concepts; the implementation model containing the data structures and procedural definitions realizing the conceptual model; and the user interface model defining how the user interacts with the data. Each of these models is connected to the others through "natural" morphisms.

Between the conceptual model and the implementation model, the morphism is defined by the system software design. This morphism must map the semantic operations of the conceptual design to the syntactic operations of the implementing system. This morphism must be an embedding of the conceptual model into the implementation model (or else the system does not fulfill the conceptual model).

Once this mapping has been defined, further reference to the implementation model is unnecessary, since a mapping to the conceptual model can always be extended to the implementation model through morphism composition. In fact a completed system is usually several layers of implementation, each model playing the role of a conceptual model to the implementation below it.

At this point, the conceptual model becomes an implementation model. Between the user model and the conceptual-implementation model another mapping is defined in direct parallel the above. This mapping must be definable and controllable by the user. The usual implementation of this mapping is the data dictionary, mapping user defined concepts onto the conceptual model. DBMS (database management system) and CASE (computer aided system/software engineering) tools are, in fact, based upon a variant this concept.

In a usual DBMS implementation, the morphism form the user paradigm to the database is dependant strictly upon the data type mappings, all processing models (the category's algebra) are strictly inherited from the DBMS system-supplied suite of processes.

In a true CASE scenario, the data and the process model (the full category specification) is mapped, with allowances for the user to define new processes for his data types (category entities) through specification via the implementation language of the conceptual model (the conceptual model's algebra), or the language of some lower level of implementation. The degree to which the user has access to lower and lower levels of implementation varies from system to system.

RDB (relational databases) usually restrict the level of access to their query language, although more and more of them are making allowances for access to lower level procedural languages. OOPS (object oriented-programming systems) as base implementations are exactly the opposite, providing no high level language and allowing unlimited access to the lowest levels of implementation models. As a OOPS develops, basic classes becoming available, inheritance of methods allows for more use of the higher level system functionality.

Spatial Example

Having described the general theory, it is helpful to see how it would be applied to the design and implementation of a particular system. For this example, we shall use TIGRIS (Herring-86, Herring-88) since we have the most familiarity with it, it supplies one of the richest set of examples, and it illustrates how OOPS may be used to parallel levels of conceptual models.

The basic conceptual model (F) for TIGRIS consist of 1) features with associated geometric representations and 2) the fundamental vector-based geometric paradigms. At this level of the conceptual model, features act as standard graphics objects for commands such as move, copy, digitize, add-vertex, delete-vertex, thin, etc. Messages defined at this level are inherited from the underlying graphics package I/DRAFT. The first and simplest addition is the tensor sum of a projection and coordinate system, which reinterprets the coordinates of the graphics as geographic based systems.

The next step is the introduction of a relational model to handle attributes. The embedding is done by using the ERA (entity-relation-attribute) model (E) and its natural morphisms to combine the feature model and the relational model (R), giving us a first integrated system model $S_1 = F \oplus_E R$. The relational query language is inherited directly, but the concepts of entity integrity and referential integrity become embedded in the feature representation from the underlying object system (Herring-87 and Herring-89a).

Spatial query in the system is accomplished through a combination of topological and geometric processes (Egenhofer-90). The geometric processes were included in the feature model F. The topological model T is summed in through a natural morphism based upon the underlying Euclidean geometric model (G), giving us a second integrated system model $S_2 = S_1 \oplus_G T$. This extends the query language to include the message functions inherited from the feature-object model F (Herring-86), the relational algebra

inherited from R (Herring-88), and the spatial operator system inherited from T (Egenhofer-90).

In the complete system, further paradigms for raster-imagery (I) and grid-based data for surface modeling (M) are incorporated. This final system design is given by:

$$TIGRIS = F \oplus_E R \oplus_G T \oplus_G I \oplus_G M$$

Two points need to made here. First, most of the tensor sums in the system model are made over a natural morphism based upon the underlying geometric or ERA paradigm. Second, the addition of new models to the system is usually accomplished by an additional subclassing from the base paradigms.

Summary

This paper has described the basic framework for a category theory of spatial representations, relations and applications built upon them. This machinery can be used in several distinct manners.

First, it may be used, as in the above TIGRIS example, to design new systems that both encompass and integrate multiple paradigms. Second, it may be used to integrate existing products built on similar paradigms into unified systems. Either type of system, judiciously designed, will inherit and integrate the advantages and functionality of their base subsystems. Within these systems, process and query optimization can take advantage of the multiple paradigms to produce automatically efficient execution plans.

Third, in a more academic light, this theory forms the basic languages for a meta-model (a model of models and their relations). This model can be used to suggest research topics, and how their results of this research might integrated into existing systems. (Herring-89b).

References

Cardelli, L.; 1984; "A Semantics of Multiple Inheritance" in G. Kahn et. al., editors; Lecture Notes in Computer Science; pp51-67; Springer-Verlag; New York, New York.

Casti, John L., 1989, Alternate Realities: Mathematical Models of Nature and Man, John Wiley & Sons, New York, New York.

Cartan, Henri and Samuel Eilenberg, 1956 Homological Algebra, Princeton University Press, Princeton, New Jersey.

Chang, N. S. and K. S. Fu; 1980; "Query-by-Pictorial-Example"; IEEE Transactions on Software Engineering, SE-6(6); November 1980.

Date, C. J.; 1985; An Introduction to Database Systems, Volume II; Addison Wesley Publishing Company; Reading, Massachusetts; 1982, reprinted with corrections 1985.

Egenhofer, Max; 1989; "A Formal Definition of Binary Topological Relationships" in W. Litwin, H-J. Schek, editors; Third International Conference on Foundations of Data Organization and Algorithms (FODO); Paris, France; Lectures Notes in Computer Science, vol 367, pp457-472, Springer-Verlag, New York, New York, June 1989.

Egenhofer, Max and J. Herring; 1990; "A Mathematical Framework for the Definition of Topological Relations"; these proceedings.

Frank, Andrew; 1981; "Applications of DBMS to Land Information Systems"; in C. Zaniolo and C. Delobel, ed.; Seventh International Conference on Very large Spatial Data Bases; pp448-453; Cannes, France; 1981

Herring, John R.; 1987; "TIGRIS: Topologically Integrated Geographic Information System"; Proceedings of AutoCarto 8; ASPRS/ACSM; March 1987; Baltimore, Maryland, pp 282-291.

Herring, John R., R. C. Larsen, J. Shivakumar; 1988; "Extensions to the SQL Query language to Support Spatial Analysis in a Topological Database"; GIS/LIS-88; ASPRS, ACSM, AAG, URISA; San Antonio, Texas, November 1988.

Herring, John R.; 1989a; "A Full Integrated Geographic Information System"; Proceedings of AutoCarto 9; ASPRS/ACSM; March 1989; Baltimore, Maryland; pp 828-837.

Herring, John R.; 1989b; "The Category Theory of Spatial Paradigms"; David M. Mark, et. al., editors; Languages of Spatial Relations: Initiative Two Specialist Meeting Report; NCGIA Technical Paper 89-2, May 1989, pp47-52.

Herring, John R.; 1990; "The Definition and Development of a Topological Spatial Data System"; Otto Kobel ed.; Photogrammetry and Land Information Systems; Presses Polytechniques Romandes, Lausanne, Switzerland; 1990.

Kemper, A and M. Wallrath; 1987; "An Analysis of Geometric Modeling in Database Systems"; ACM Computing Surveys, 19(1). March 1987.

Nagy, G. and S. Wagle; 1979; "Geographic Data Processing"; ACM Computing Surveys, 11(2); June 1979.

Ooi, B. C. et al.; 1989; "Extending a DBMS for Geographic Applications"; IEEE Fifth International Conference on Data Engineering, Los Angeles, California; February 1989.

Orenstein, J. and F. Manola; 1988; "PROBE Spatial Data Modeling and Query Processing in an Image Database Application"; IEEE Transactions on Software Engineering, 14(5); May 1988.

Samet, Hanan; 1984; "The Quadtree and Related Hierarchical Data Structures"; ACM Computing Surveys, 16(2); pp 187-260, June 1984.