16

HIGH-LEVEL SPATIAL DATA STRUCTURES FOR GIS

M J EGENHOFER AND J R HERRING

The chapter focuses on the semantics of spatial concepts and their formalizations. A framework is proposed consisting of spatial concepts, spatial data models (or high-level spatial data structures) and (low-level) spatial data structures. Examples of spatial data models include spaghetti, regular and irregular tessellations. The implementation of each of these in (low-level) data structures is discussed.

INTRODUCTION

The organization of spatial data in a computer system has drawn much attention in the field of GIS. Since its early beginnings, designers and developers of GIS have been looking for appropriate representations of geometrical objects. Peucker and Chrisman (1975), Nagy and Wagle (1979), Peuquet (1984), Burrough (1986) and, most recently, Samet (1989) have published comprehensive compilations of these previous studies. The most popular abstractions organize spatial data as points, lines and areas. These elements have also been incorporated as fundamental components of the cartographic data exchange standard (DCDSTF 1988). The goals of methods for organizing spatial data, often called spatial data structures, are: (1) to process geometrical user queries and other geometrical operations for application programs; and (2) to implement these operations so that they are consistently, robustly, and efficiently executed in a computer system. The latter is frequently associated with computational geometry (Preparata and Shamos 1985; Edelsbrunner 1987), while spatial query processing is of particular interest to spatial database management systems (Frank 1988; Egenhofer 1989b).

The selection of a specific spatial data structure may considerably influence the processing of spatial queries. For instance, in the case of a collection of records of house ownership deeds, a user may enquire about the owners of the parcels adjacent to the house lot '461 Ocean Boulevard'. Depending on the organization of parcels, the query may be processed in different ways:

1. Each parcel in the database could be checked to determine whether its bounding nodes have the same geodetic coordinates as one of the nodes of parcel '461 Ocean Boulevard' and the corresponding owner found. Of course, this sequential comparison will be quite inefficient as the number of parcels stored in the database increases because each parcel has to be retrieved and all its nodes need to be checked against all the nodes of the target parcel.
2. The performance of this operation improves if spatial objects can be accessed based on their spatial location. Such an access method allows for searching parcels within a given region, for example a rectangle, so that the set of objects to be tested against the target is relatively small. Then the actual neighbours can be found with the same type of checking as above, based on a much smaller number of parcels.
3. Another way to organize parcels is to record explicitly the neighbours for each parcel. The neighbourhood query is then reduced to accessing the related parcels based on the stored adjacency information.

The semantics of spatial concepts common to multiple implementations of spatial data structures and their formalizations are the focus of this chapter. The former specifies what operations can be performed on spatial objects, while the latter defines how these operations are implemented so that they can actually be executed in a computer. A similar separation has been proposed for the schema design of database management systems, distinguishing physical, conceptual, and user schemata or views (ANSI 1975). This distinction advocates data independence so that the implementation of a specific model of a mini-world may be updated without affecting the remaining software system (Codd 1982). Since the remainder of the software system relies upon the concepts implemented behind well-defined interfaces, changes in the physical schemata will be hidden from users at the logical level – except for performance or storage requirements. Likewise, in the realms of programming and query languages, there is often a differentiation between procedural languages – to describe how a function should be performed – and descriptive languages – to explain the goal without specifying how it should be achieved. Essentially, the same difference exists between data structures and data models – even if sometimes data models are explained in an exemplary fashion by describing how operations should be performed.

This chapter defines the terms 'data model' and 'data structure' which in the past have sometimes been thought of as high-level and low-level data structures. It proceeds with the definition of a framework to distinguish spatial data models from high-level and low-level spatial data structures. The next sections discuss formalizations of spatial concepts known from mathematics and examples of spatial data models and spatial data structures. The conclusion focuses on the integration of different data models under a common user interface.

A CONCEPTUAL FRAMEWORK FOR SPATIAL DATA MODELS AND STRUCTURES

Solutions for geometrical problems may appear to be trivial when determined using principles of Euclidean geometry, the popular school model for solving geometrical problems (see Gatrell 1991 in this volume). Euclidean geometry is based on a continuous space consisting of an infinite number of points. Analytical geometry is a convenient mapping to the coordinate space and relies on real numbers where between any two numbers another one exists. This is necessary to represent the property of Euclidean geometry in which between any two points another one can be inserted. Unfortunately, finite computers and their number systems cannot guarantee this assumption and the immediate use of floating point or integer arithmetic for Euclidean geometry frequently leads to undesired effects, thus contradicting our common sense of geometrical properties and violating some laws of geometry. Examples of such counter-intuitive effects of implementations of Euclidean coordinate geometry in computer systems are:

- the scaling of coordinates may change topology by moving points, initially close to a line, from one side to another (Franklin 1984);
- the intersection of two lines does not necessarily lie on both lines (Nievergelt and Schorn 1988); and
- the application of two inverse geometrical operations may generate a geometry which differs from the original geometry (Dobkin and Silver 1988; Hoffmann 1989).

Euclidean coordinate geometry is but one concept humans employ when dealing with space and geometry. Point-sets, for example, are another concept for dealing with spatial objects. A framework for the organization of spatial data proposed here consists of (1) spatial concepts, (2) spatial data models (or high-level spatial data structures), and (3) (low-level) spatial data structures.

Spatial concepts

Humans employ spatial concepts to organize and structure their perception of (geographical) space. For this purpose, they use different concepts depending on the observers' different experiences and the context in which a person views some situation. Humans have the particular ability to handle multiple concepts simultaneously, to select

the most appropriate for solving a specific task, and to switch between them whenever necessary or appropriate (Mark *et al.* 1989). For example, people may use an essentially Euclidean geometry when operating and reasoning in their immediate neighbourhood, for instance, on a desktop. On the other hand, they employ the spatial concept of a network when navigating a car.

Although these spatial concepts are frequently employed in everyday life, they are only informally defined. A few formalized spatial concepts, (e.g. Euclidean geometry) are based on assumptions, such as an infinitely precise numbering system, which restrict them from being correctly implemented on finite computer machines. Image schemata, for example, play an important role in humans' perception and understanding of space (Lakoff and Johnson 1980); however, they are defined in linguistic terms (Talmy 1983; Herskovits 1986) which lack formalism and mathematical rigour.

This topic is the focus of Frank and Mark (1991 in this volume) and, therefore, needs no further elaboration here.

Spatial data models

The semantics of spatial concepts are defined and formalized by spatial data models. Spatial data models are formalizations of the concepts humans use to conceptualize space. Such formalizations of spatial concepts are necessary, because computer systems are essentially formal systems that manipulate symbols according to formal rules. They do not understand – in the sense of human beings' understanding – the meanings of symbols or what they stand for; instead, they follow the instructions of their programs blindingly fast, but without any 'common sense'. The formalization of geometrical concepts employs well-understood mathematical findings to describe the meanings of the operations to be performed on spatial objects (Egenhofer and Frank 1990).

The common use of abstraction from concepts by humans helps to make these concepts implementable on computer systems, without the need to describe actual implementation details. The view of geometry in a spatial data model is, therefore, independent of implementation aspects. The role of a spatial data model is similar to the conceptual schema in the three-schema view: concepts get separated from the actual implementations, thus implementations of certain parts of the large GIS software system become more independent and may be updated without affecting the remaining software parts. Finally, the formalisms may serve as a means for users to verify that implementations of operations concur with their expectations.

A commonly employed approach in modelling with spatial properties is to separate them into geometrical and non-geometrical classes. Abstractions similar to these formalizations can be found in areas other than geometrical data handling as well. Probably best known is the formalization of the concept of a table in the relational data model (Codd 1970). Humans frequently employ the organization of data in the form of tables in business. The relational data model formally describes the operations on tables and their semantics. This formalization leads to a possible implementation of how humans deal with tables, but it does not yet describe the actual implementation of these operations in a computer system.

Low-level spatial data structures

Spatial data structures, also called low-level data structures, are the implementations of spatial concepts. In order to be a valid implementation of a spatial concept, a spatial data structure must follow the properties defined in the corresponding spatial data model. Several spatial data structures may be implemented for the same spatial data model. It should, therefore, be possible to exchange different spatial data structures for the same data model with only implementation changes and no observable effects for the user. Spatial data structures are concerned with the efficiency of the implementation, namely: (1) minimizing storage requirements to reduce the amount of space to hold the data; and (2) maximizing performance to improve the processing speed of geometrical operations. For these reasons, the representation of data internal to a computer system and its architecture are crucial factors to be considered in the design of spatial data structures. Spatial data structures are then built from standard data structures as developed in computer science (Knuth

1973), such as stacks, queues, lists, and trees, and specific index structures to provide fast access to spatial data based on a spatial location (Samet 1989; see also Franklin 1991 in this volume).

An example of different levels in the framework

The following example demonstrates the roles of the different levels in this framework:

- When dealing with land parcels, humans may employ the concept of a discrete 2-dimensional space in which the parcels form an ideal partition of the space.
- One formalization of a discrete space is the spatial data model of a raster of squares of equal size. Within a raster, adjacent cells can be determined and distances between points can be calculated in terms of raster units.
- A raster may then be implemented in various low-level spatial data structures, e.g. as quadtree (Samet 1984), linear quadtree (Gargantini 1982), run-length encoded (Burrough 1986), and so on. These implementations have the same behaviour, but they may have varying efficiency with respect to the data storage requirements and performance of operations.

Formalizations of spatial concepts

Spatial data models formally describe the semantics of concepts present in the geometrical data model. With the help of the mathematical concepts of topology, metric and order, various types of geometrical questions may be answered which are particularly important for GIS (Kainz 1989). Examples of such questions include 'How far is it from Bangor to Boston on Interstate I-95?', 'Which provinces in Canada share a common boundary with the state of Maine?', and 'What county contains Bangor?' These concepts are not completely unrelated since every metric space has a topology – and some different distance functions describe even the same topology. Likewise, the concept of inclusion/containment is a property which can be expressed in both topology and order.

Order

The concept of order allows for the comparison of two or more elements from a set. A relation R over a set X is a strict order relation if for all x, y, z elements of X, R is

asymmetric: $x\ R\ y \rightarrow \text{NOT}\ (y\ R\ x)$ [16.1]
transitive: $x\ R\ y\ \text{AND}\ y\ R\ z \rightarrow x\ R\ z$ [16.2]

Within strictly ordered sets, any two elements can be compared with each other, that is for all x, y elements of S: $x\ R\ y\ \text{AND}\ y\ R\ x$; therefore, a strict order relation establishes a hierarchy.

Strict order relations are different from order relations. To distinguish the two, the structure defined by an order relation is called a partially ordered set or poset. A relation R over a set X is an order relation if, for all x, y, z elements of X, R has the following three properties (Gill 1976):

reflexivity: $x\ R\ x$ [16.3]
anti-symmetry: $x\ R\ y\ \text{AND}\ y\ R\ x \rightarrow x = y$ [16.4]
transitivity: $x\ R\ y\ \text{AND}\ y\ R\ z \rightarrow x\ R\ z$ [16.5]

Order, both strict and partial, can be applied to GIS, for instance, to determine whether one spatial object is inside another – and its inverse operation whether a spatial object contains another one (Saalfeld 1985; Greasley 1988) – as well as for perspectives, such as left/right, in front/behind (Freeman 1975; Pullar and Egenhofer 1988). For example, the relation inside between cities and counties is an order relation, indicating that each city belongs to exactly one county, and that a county can have several cities. On the other hand, the relation between school districts and voting districts is a partial order: while some school districts may be completely included within a voting district, not all boundaries of the voting districts coincide with school district boundaries.

The application of order to sort non-geometrical property values derived from the geometry of spatial objects, such as the areas of regions or the lengths of lines, is the same as for conventional data, for example, integers or strings, and, therefore, does not introduce any new challenges due to handling geometrical data.

Topology

Topology is the study of the characteristics of geometrical objects that are independent of the

underlying coordinate system. Topology treats those properties that are preserved under topological transformations, characterized by a group of bijective (a function f: A -> B is bijective if there exists another function g: B -> A such that $g(f(A)) = A$ and $f(g(B)) = B$) and continuous (a function f: A -> B is continuous if for each element a of A every open set V in B is mapped on to an open set U in A containing a such that $f(U)$ is a subset of V) transformations which have also continuous inverses. Properties that are preserved under topological transformations are called topological invariants. Examples of topological transformations are translation, rotation, and scaling.

The major purpose of using topologically structured data is to improve the spatial analysis capability of GIS (Herring 1987; Pullar and Egenhofer 1988; Herring 1989b; Egenhofer 1989a). This is achieved using the techniques of problem translation and symbolic manipulation as used in algebraic topology (Alexandroff 1961) and results in the following:

- The intersection of two points or lines becomes the search for common nodes.
- The coincidence of two linear objects becomes the search for their common edges.
- The adjacency of a linear object to a region is the search for common edges (or nodes depending on the definition of adjacency) in the line and the boundary of the region.
- The adjacency of regions is reduced to determining the common edges or nodes in the boundaries of the regions, probably from opposite sides.
- Polygon overlay is the analysis of common faces.

Metric

A metric space is defined by a function $d(p,p)$ -> R calculating the distance between two points p. This distance function must obey the following four axioms:

1. The distance from a point to itself is zero (eqn [16.6]).
2. If the two points are not identical then the distance between two points is greater than zero (eqn [16.7]).
3. The distance is symmetrical (eqn [16.8]).
4. The sum of the lengths of two legs of a triangle is greater than or equal to the length of the third leg (eqn [16,9]).

$$d(p,p) = 0 \qquad [16.6]$$

$$d(p1,p2) > 0 \text{ if } p1 \neq p2 \qquad [16.7]$$

$$d(p1,p2) = d(p2,p1) \qquad [16.8]$$

$$d(p1,p3) \leq d(p1,p2) + d(p2,p3) \qquad [16.9]$$

A common metric space is the Euclidean space described by the Pythagorean distance. The metric of an n-dimensional Euclidean space is defined by the following distance function between two points $p1(x_1,\ldots,x_n)$, $p2(x'_1,\ldots,x'_n)$:

$$d(p1,p2) = ((x_1 - x'_1)^2 + \ldots + (x_n - x'_n)^2)^{1/2} \qquad [16.10]$$

Two other distance functions are sometimes used describing the taxicab metric and the max metric (Croom 1989; see also Gatrell 1991 in this volume). In the plane, the taxicab metric, also called the Manhattan or city block distance, is defined by a function d calculating the distance from point $p1$ to $p2$ as the sum of the lengths of a horizontal segment and a vertical segment joining $p1$ to $p2$ (eqn [16.11]). The max metric is defined by the chessboard distance as the largest of the absolute values of the coordinate differences between $p1$ and $p2$ (eqn [16.12]):

$$d(p1,p2) = \sum_{i=1}^{n} |x_i - x'_i| \qquad [16.11]$$

$$d(p_1,p_2) = \max \{|x_i, x'_i|\}_{i=1}^{n} \qquad [16.12]$$

Each of these functions is a different implementation which has the properties defined by the axioms of a distance function. Metrics are used in GIS applications to determine distances between objects, to find shortest paths, and to identify nearest neighbours.

EXAMPLES OF SPATIAL DATA MODELS

Various models have been developed to represent geometrical properties. These models differ in their powers and capabilities to guarantee the formalizations of spatial concepts. Their deficiencies become apparent when users try to modify the geometry or want to verify its consistency. Some structures are quite primitive so that without the help of Euclidean coordinate geometry certain operations cannot be performed. Whenever an operation, such as the intersection of two lines, the test for the inclusion of a point within a region, or the coincidence of a point with a line, requires coordinate calculations, the imprecisions of the underlying number systems affect the results and consistency cannot be guaranteed.

Spaghetti data structure

Maps, as a basic data type, hold all their information in the form of graphical representations. Roads are red, rivers are blue, and so on. This simple view of geographical data formed the basis for many of the earlier GIS, but was designed for the purpose of mapping rather than geographical analysis. The process of digitizing spatial data from maps was mimicked directly in the digital domain, and thus initiated what is sometimes called the spaghetti data structure, by analogy with a plate of disconnected, but intertwined and intertangled pasta. In such structures geometry is represented as sequences of straight edges, essentially imitating the manual or stream digitizing process. Thus, a line-string is represented as a sequence of n connected edges with two end points and $n-1$ breakpoints. This data structure is sufficient for graphical purposes, such as redrawing the digitized map. However, it provides insufficient information for most non-geometrical and all complex geometrical operations.

Spaghetti structures have their greatest usage in situations involving a large number of simple geometrical edits through a graphics interface. They are especially well suited for data capture scenarios in which connectivity is of prime importance. The simplistic nature of the data leads to simple and robust system designs and, consequently, this structure is popular in GIS.

To extend the usefulness of spaghetti structures, some simple steps are usually taken to annotate the geometry. These include feature type association by symbology (line colour, style, level if provided), or data segregation into separate files (one for roads, one for parcels, etc.). This need to separate data into different files, layers, or overlays is very similar to the manner in which the map colour lithographic process requires the separation of symbols by the colour of ink required.

Another option for spaghetti data is the association of external database linkages via a 'user-data' segment in the graphical record, which acts as a foreign key to an external attribute database. This is preferable in an environment in which the data held or the analysis needed goes beyond the manual, graphics-oriented environment.

While spaghetti data structures are useful in recording connectivity between points, they show major deficiencies for dealing with neighbourhood and inclusion (Frank 1984). Given these problems, most GIS systems take either another approach or an additional, parallel, approach to geometrical storage and analysis.

Regular tessellations

Regular tessellations come close to the perception of spatial objects when data are captured with remote sensing or scanning devices. Such regular tessellations, also called rasters, are based on aggregates of picture cells, called pixels, representing regions. A pixel is characterized by: (1) the area it covers and (2) one or several values describing nonspatial properties of the entire pixel. The simplest pixel value is a Boolean-type representation in which a pixel can have two values, 'on' or 'off'. Pixels assigned with the value 'on' represent parts of an object, while 'off' pixels stand for the empty embedding space. The most common shapes of pixels are squares, but equilateral triangles and hexagons are also occasionally used (Diaz and Bell 1986). A raster image is a collection of pixels which together form a contiguous, non-overlapping image or partition.

A first step toward a data model for regular tessellations has been accomplished with a semi-formal description of the associated operations, the MAP algebra (Tomlin 1990; see also Tomlin 1991 in this volume). By applying more rigid, mathematical methods, such as algebraic specifications (Guttag,

Horowitz and Musser 1978; Zilles 1984), a geometrical data model for regular tessellations can be fully formalized. An algebraic specification defines the finite set of objects, called SORTs, to be treated and characterizes the behaviour of the model in terms of the effects of related operations (OPNS) to change and observe instances within this model. For example, the spatial relationship 'inside' (Egenhofer 1989a) can be formally defined for a region, a finite set of pixels, as follows (the specifications of the sort set of pixels and the operations 'interior' and 'boundary' are omitted for the sake of clarity):

```
SORT    region
OPNS    create: set of pixels -> region
        intersection: region x region ->
                      region
        interior: region -> region
        boundary: region -> region
        inside: region x region -> boolean
VARS    r1, r2: region
EQNS    inside (r1, r2) IF intersection
        (r1,interior (r2)) = r1 and
        intersection (r1, boundary (r2)) = {}
```

Irregular tessellations

During the nineteenth and twentieth centuries, topologists developed a theoretical tool to study the invariance of properties under groups of transformations by decomposing the geometrical objects, called manifolds, into relatively simple geometrical shapes, called cells, which were joined along their common borders much like a biological organism is a collection of living cells. By decomposing the manifold into simple cells, topologists reduced complex geometrical problems into equivalent combinatorial problems, based upon intercellular relationships, attackable through algebraic methods (Alexandroff 1961; Munkres 1966; Spanier 1966; Artin and Brown 1969; Lefschetz 1975; Switzer 1975).

In geographical terms, such a decomposition results in an organization of space based upon irregularly shaped polygons for surface modelling or polyhedra for solids modelling. The storage of polygons is generally vector, although, topologically, it does not matter whether the polygons are composed of line-strings or of curvilinear function segments, such as spline curves, but much of their manipulation is raster-like. For example, given two topologically structured overlays or map coverages, a finer topological coverage can be produced that divides the cells of each of the input coverages into finer cells in the output coverage. The attributes of these subdivisions can be combined in any manner analogous to the combinations of values or attributes in raster overlays. Thus a topological data structure can support the same type of map algebra supported by classical grid-based GIS systems (Tomlin 1990).

EXAMPLES OF (LOW-LEVEL) SPATIAL DATA STRUCTURES

(Low-level) spatial data structures are the implementations of geometrical concepts. In contrast to spatial data models, spatial data structures focus on efficiency, for both storage and performance. Several spatial data structures may be the implementation of the same data model. For example, the raster data structure may be implemented as a matrix, run-length encoded, as a quadtree, as a linear quadtree, and so on. Each of these data structures follows the formally defined data model of a regular tessellation and is distinguished only by its storage requirements and its speed when processing raster operations.

Spatial data structures are supported by spatial index structures to provide fast access to spatial data stored in primary and secondary memory. In order to reduce the bottleneck in fast processing of large amounts of spatial data, they cluster spatial data according to their neighbourhood (Frank 1981) and organize the storage of spatial data such that the number of disk accesses is minimized. A large number of storage and index structures for spatial data have been proposed, such as the popular grid file (Nievergelt, Hinterberger and Sevcik 1984) and R-tree (Guttman 1984). For an overview over other structures and performance evaluations, see Samet (1989) and Kriegel *et al.* (1989), respectively.

Implementations of regular tessellations

Raster data structures implement the regular tessellation data model. Generally, their

implementations are simple and versatile and some of them reduce storage requirements and increase performance considerably. Most popular are implementations of data structures for tessellations with square pixel shape.

Two-dimensional matrices

Rasters can be simply stored as two-dimensional matrices in which each pixel occupies a specific position (column and row) according to its spatial location. By convention, 'true' is used to represent filled elements, while 'false' describes empty elements. Variations of these binary representations distinguish multiple values for the representation of objects.

Matrices are convenient because they are: mathematically well defined and simple to comprehend; generally usable because for each pixel the same operation can be applied; simple to implement because two-dimensional arrays are immediately supported by most high-level programming languages; and easy to use because operators to loop over matrices are immediately available in programming languages. The disadvantages of using matrices for raster representation include:

- detail – the whole image is represented in the same way and no advantage is taken from the existence of larger areas covered by the same type of pixel;
- abstraction – in order to get less detailed representations, all the data have to be considered and no different levels of abstraction can be achieved without checking all details;
- storage capacity – potential waste of storage space;
- size of data sets – transmission of raster data is limited by the bandwidth of the transmission channel and matrix representation is an impediment when transferring collections of rasters at a high rate; and
- inefficiency – the whole image has to be kept in main memory which can be critical for very large images.

The storage of raster images in matrix form is space consuming and introduces considerable overheads. The storage space required depends on the extension of the area to be represented and the resolution, but is independent of the actual objects contained. Particularly, for regions with a low 'population' matrices are an inefficient data structure. Besides the argument for less storage space, there is also the need for increased processing speed of raster operations. Matrix operations are inherently dependent on the size of the matrix, that is, the area of the space modelled, and make no use of any information about the objects in this space. More efficient storage techniques are necessary, especially for sparse distributions of pixels. These processes are frequently referred to as data compression and their goal is to reduce the amount of space required to store a raster without losing information. Two structures have become popular, less in GIS than in image processing and robotics: run-length encoding and quadtrees.

Run-length encoding

Run-length encoding is a technique that makes use of the fact that objects frequently extend over areas larger than a single pixel. Instead of recording the values of each pixel, run length encoding groups the rows of a raster into blocks with an identical value. A binary image, for instance, may be horizontally segmented into black and white intervals whose lengths are recorded.

Quadtrees

The most common hierarchical organization of a raster is the quadtree. The quadtree is a maximal block representation in which the blocks have standard sizes and positions, that is, powers of two (Samet 1989; see also Gatrell 1991 in this volume). Its principle is divide and conquer, the successive subdivision of an image array into quadrants, each of which in turn can be subdivided into another four quadrants, and so on. This structure forms a tree with nodes representing heterogeneous areas and leaves for homogeneous areas with a single value. A quadtree in general reduces considerably the space needed to represent a raster image, because it makes use of aggregates of neighbouring pixels of the same kind.

Implementations of irregular tessellations

The data structures above share one common overriding attribute: they all divide space into a

collection of essentially regular geometrical shapes. In topological data structures this is not necessarily the case. There are two basic types of topological decomposition techniques depending on the prototypical cell type. The algebraically and logically simpler one is based on cells homeomorphic to the convex hulls of $(n+1)$ independent points in a Euclidean n-space, called a simplex, giving a simplicial decomposition. An equivalent, but algebraically more complex approach, uses cells homeomorphic to the unit disc and unit sphere in the same Euclidean n-space, called simply a cell, giving a cellular decomposition.

Simplicial topology

The simplest of the basic tools of topology is the simplex (Alexandroff 1961; Frank and Kuhn 1986; Egenhofer, Frank and Jackson 1989). In the most general of terms, an n-dimensional simplex, or n-simplex, is the convex hull of $n+1$ points, in general position, embedded in a space of dimension n or greater. Thus, for the modelling of real three-dimensional space a 0-simplex is a single point, a 1-simplex is a line joining two distinct points, a 2-simplex is a filled triangle joining three non-collinear points, and a 3-simplex is a solid tetrahedron joining four non-coplanar points. The boundary of an n-simplex contains $n+1$ $(n-1)$-dimensional simplices. For example, a 2-simplex has three 1-simplex faces. An n-dimensional simplicial complex is a collection of n-simplices sharing common boundaries. Most geographers are familiar with two-dimensional simplicial complexes in the form of the triangulated irregular network (TIN) used in terrain modelling (Peucker and Chrisman 1975; Frank, Palmer and Robinson 1986; see also Weibel and Heller 1991 in this volume), and with one-dimensional simplicial complexes in the form of path–node graph models used in network analysis.

In terms of implementation, the simplicial complex approach has the advantage of consistency throughout dimensions, making software development much easier, especially if linear interpolation is used between the vertices in a simplex.

Cellular topology

A more complicated notion in topology which is logically equivalent to the simplicial complex is the cellular complex. A homeomorphism between two geometrical objects is an invertible function from one to the other such that both the function and its inverse are continuous. The two objects are said to be homeomorphic (from Greek *homoiómorphos*, meaning 'of similar form'). An n-disc is the set of points in Euclidean n-space with distance from the origin less than or equal to 1. An n-cell is any object homeomorphic to an n-disc. The surface of the $(n+1)$-disc, that is, all points at a distance exactly 1.0, is called the n-sphere. The boundary of an n-cell is that portion of the n-cell mapped on to the $(n-1)$-sphere by any homeomorphism. An n-dimensional cellular complex, or n-complex, is a collection of n-cells, such that portions of their boundary, homeomorphic to $(n-1)$-cells, have been identified with one another.

Comparison

The fundamental approach to using these topological structures within a GIS to describe space is to view the underlying coordinate space as a union of disjoint, fundamental topological entities. In a two-dimensional simplicial implementation, there would be points, open lines, and the interior of triangles. In a two-dimensional cellular implementation, there would be points, simple curves, and connected areas possible with holes – in some implementations, phantom lines (1-cells) are used to remove holes from the areas and make them 2-cells.

The critical decision is over which method of storage is most appropriate to an automated GIS environment. The simplicial approach has the advantage of simplicity and easy generalization of code to higher and higher dimensions. It is appropriate for models with a great deal of microstructure, such as models of surface and subsurface geology. The cellular approach, as used in vector GIS such as ARC/INFO and TIGRIS, and exchange structures, such as DLG, has the advantage of reduced data set sizes – fewer entities, more coordinates per entity.

CONCLUSIONS

Current commercial systems implement only a single data model; however, a modern GIS should integrate several spatial concepts and provide the

corresponding geometrical data models. The properties of the different data models are quite different.

The extension to multiple concurrent data models requires not only their coexistence, but also the interconnections between data. Like humans switching between different spatial concepts, a GIS should select the representations which are most appropriate to solve a specific task. A framework for such a system has been proposed with the category model (Herring 1989a; Herring. Egenhofer and Frank 1990) as a unifying theory integrating multiple spatial paradigms.

REFERENCES

Alexandroff P (1961) *Elementary Concepts of Topology*. Dover Publications, New York

ANSI Study Group on Database Management Systems (1975) Interim report. *SIGMOD* **7**

Artin E, Brown H (1969) *Introduction to Algebraic Topology*. Charles E Merrill Publishing Company, Columbus

Burrough P A (1986) *Principles of Geographical Information Systems for Land Resources Assessment*. Oxford University Press, Oxford

Codd E F (1970) A relational model for large shared data banks. *Communications of the ACM* **13** (6): 377–87

Codd E F (1982) Relational database: a practical foundation for productivity. *Communications of the ACM* **25** (2): 109–17

Croom F (1989) *Principles of Topology*. Saunders College Publishing, Philadelphia

Diaz B, Bell S (eds.) (1986) *Spatial Data Processing using Tesseral Methods*. Natural Environment Research Council, Reading UK

DCDSTF (Digital Cartographic Data Standards Task Force) (1988) The proposed standard for digital cartographic data. *The American Cartographer* **15** (1): 9–140

Dobkin D, Silver D (1988) Recipes for geometry and numerical analysis, part 1: an empirical study. *Proceedings of Fourth ACM Symposium on Computer Geometry*, pp. 93–105

Edelsbrunner H (1987) *Algorithms in Combinatorial Geometry*. Springer-Verlag, New York

Egenhofer M J (1989a) A formal definition of binary topological relationships. In: Litwin W, Schek H-J (eds.) *Third International Conference on Foundations of Data Organization and Algorithms (FODO), Paris* (Lecture Notes in Computer Science, Volume 367). Springer-Verlag, New York, pp. 457–72

Egenhofer M J (1989b) *Spatial query languages*. Unpublished PhD Thesis, University of Maine, Orono

Egenhofer M J, Frank A U (1990) Lobster: combining AI and database techniques for GIS. *Photogrammetric Engineering and Remote Sensing* **56** (6): 919–26

Egenhofer M J, Frank A U, Jackson J (1989) A topological data model for spatial databases. In: Buchmann A, Gunther O, Smith T, Wang Y (eds.) *Symposium on the Design and Implementation of Large Spatial Databases* (Lecture Notes in Computer Science, Volume 409). Springer-Verlag, New York, pp. 271–86

Frank A U (1981) Applications of DBMS to land information systems. In: Zaniolo C, Delobel C (eds.) *Proceedings of Seventh International Conference on Very Large Data Bases, Cannes, France*. Morgan Kaufmann Publishers, Los Altos, pp. 448–53

Frank A U (1984) Computer assisted cartography – graphics or geometry. *Journal of Surveying Engineering* **110** (2): 159–68

Frank A U (1988) Requirements for a database management system for a GIS. *Photogrammetric Engineering and Remote Sensing* **54** (11): 1557–64

Frank A U, Kuhn W (1986) Cell graph: a provable correct method for the storage of geometry. *Proceedings of the 2nd International Symposium on Spatial Data Handling, Seattle*. International Geographical Union, Columbus Ohio, pp. 411–36

Frank A U, Mark D M (1991) Language issues for GIS. In: Maguire D J, Goodchild M F, Rhind D W (eds.) *Geographical Information Systems: principles and applications*. Longman, London, pp. 147–63, Vol 1

Frank A U, Palmer B, Robinson V (1986) Formal methods for the accurate definition of some fundamental terms in physical geography. In: Marble D (ed.) *Proceedings of Second International Symposium on Spatial Data Handling, Seattle*. International Geographical Union, Ohio, pp. 583–99

Franklin W R (1984) Cartographic errors symptomatic of underlying algebra problems. In: *Proceedings of International Symposium on Spatial Data Handling, Zurich*. International Geographical Union, Zurich Irchel, pp. 190–208

Franklin Wm R (1991) Computer systems and low-level data structures for GIS. In: Maguire D J, Goodchild M F, Rhind D W (eds.) *Geographical Information Systems: principles and applications*. Longman, London, pp. 215–25, Vol 1

Freeman J (1975) The modelling of spatial relations. *Computer Graphics and Image Processing* **4**: 156–71

Gargantini I (1982) An effective way to represent quadtrees. *Communications of the ACM* **25** (12): 905–10

Gatrell A C (1991) Concepts of space and geographical data. In: Maguire D J, Goodchild M F, Rhind D W (eds.) *Geographical Information Systems: principles and applications*. Longman, London, pp. 119–34, Vol 1

Gill A (1976) *Applied Algebra for the Computer Sciences*. Prentice-Hall, Englewood Cliffs New Jersey
Greasley I (1988) Data structures to organize spatial subdivisions. *Proceedings of ACSM-ASPRS Annual Convention, St. Louis*, pp. 139–48
Guttag J, Horowitz E, Musser D (1978) Abstract data types and software validation. *Communications of the ACM* **21** (12): 1048–64
Guttman A (1984) R-trees: a dynamic index structure for spatial searching. *Proceedings of the Annual Meeting ACM SIGMOD, Boston*, pp. 47–57

Herring J R (1987) TIGRIS: topologically integrated geographic information systems. *Proceedings of AUTOCARTO8*. ASPRS, Falls Church, pp. 282–91
Herring J R (1989a) The category model of spatial paradigms. In: Mark D M, Frank A U, Egenhofer M J, Freundschuh S, McGranaghan M, White R M (eds.) *Languages of Spatial Relations: Report on the Specialist Meeting for NCGIA Research Initiative 2*. National Center for Geographic Information and Analysis, Santa Barbara, pp. 47–51
Herring J R (1989b) A fully integrated geographic information system. *Proceedings of AUTOCARTO9*. ASPRS, Falls Church, pp. 828–37
Herring J R, Egenhofer M J, Frank A U (1990) Using category theory to model GIS applications. *Proceedings of the 4th International Symposium on Spatial Data Handling, Zurich*. International Geographical Union, Columbus Ohio, pp. 820–9
Herskovits A (1986) *Language and Spatial Cognition: an interdisciplinary study of the prepositions in English*. Cambridge University Press, Cambridge
Hoffmann C (1989) The problems of accuracy and robustness in geometric computation. *IEEE Computer* **22** (3): 31–42

Kainz W (1989) Order, topology, and metric in GIS. *Proceedings of ASPRS-ACSM Annual Convention, Baltimore*. ASPRS/ACSM, Falls Church, pp. 154–60
Knuth D (1973) *The Art of Computer Programming*. Addison-Wesley Publishing Company, Reading Massachusetts
Kriegel H-P, Schiwietz M, Schneider R, Seeger B (1989) Performance comparison of point and spatial access methods. In: Buchmann A, Gunther O, Smith T, Wang Y (eds.) *Symposium on the Design and Implementation of Large Spatial Databases* (Lecture Notes in Computer Science, Volume 409). Springer-Verlag, New York, pp. 89–114

Lakoff G, Johnson M (1980) *Metaphors We Live By*. University of Chicago Press, Chicago
Lefschetz S (1975) *Applications of Algebraic Topology*. Springer-Verlag, New York

Mark D M, Frank A U, Egenhofer M J, Freundschuh S, McGranaghan M, White R M (1989) Languages of spatial relations: report on the specialist meeting for NCGIA Research Initiative 2. *Technical Report 89–2*, National Center for Geographic Information and Analysis, Santa Barbara
Munkres J (1966) *Elementary Differential Topology*. Princeton University Press, Princeton,

Nagy G, Wagle S (1979) Geographic data processing. *ACM Computing Surveys* **11** (2)
Nievergelt J, Hinterberger H, Sevcik K (1984) The grid file: an adaptable, symmetric multi-key file structure. *ACM Transactions on Database Systems* **9** (1): 38–71
Nievergelt J, Schorn P (1988) Line problems with supra-linear growth (in German). *Informatik Spektrum* **11** (4)

Peucker T, Chrisman N (1975) Cartographic data structures. *The American Cartographer* **2** (2): 55–69
Peuquet D J (1984) A conceptual framework and comparison of spatial data models. *Cartographica* **21**: 66–113
Preparata F, Shamos M (1985) *Computational Geometry*. Springer-Verlag, New York
Pullar D, Egenhofer M J (1988) Towards formal definitions of topological relations among spatial objects. *Proceedings of the 3rd International Symposium on Spatial Data Handling*. International Geographical Union, Columbus Ohio, pp. 225–42

Saalfeld A (1985) Lattice structure in geography. *Proceedings of AUTOCARTO7*. ASPRS, Falls Church, pp. 482–97
Samet H (1984) The quadtree and related hierarchical data structures. *ACM Computing Surveys* **16** (2): 187–260
Samet H (1989) *The Design and Analysis of Spatial Data Structures*. Addison-Wesley Publishing Company, Reading Massachusetts
Spanier E (1966) *Algebraic Topology*. McGraw-Hill Book Company, New York,
Switzer R (1975) *Algebraic Topology-Homotopy and Homology*. Springer-Verlag, New York

Talmy L (1983) How language structures space. In: Pick H, Acredolo L (eds.) *Spatial Orientation: Theory, research, and application*. Plenum Press, New York
Tomlin C D (1990) *Geographic Information Systems and Cartographic Modeling*. Prentice-Hall, Englewood Cliffs
Tomlin C D (1991) Cartographic modelling. In: Maguire D J, Goodchild M F, Rhind D W (eds.) *Geographical Information Systems: principles and applications*. Longman, London, pp. 361–74, Vol 1

Weibel R, Heller M (1991) Digital terrain modelling. In: Maguire D J, Goodchild M F, Rhind D W (eds.) *Geographical Information Systems: principles and applications*. Longman, London, pp. 269–97, Vol 1

Zilles S (1984) Types, algebras, and modelling. In: Brodie M, Mylopoulos J, Schmidt J (eds.) *On Conceptual Modelling*. Springer-Verlag, New York, pp. 441–50