Draft summary of *Geographic Information Systems and Cartographic Modeling* by C. Dana Tomlin, Prentice-Hall, Englewood-Cliffs, NJ, 1990.

Hanan Samet
Computer Science Department and
Center for Automation Research and
Institute for Advanced Computer Studies
University of Maryland
College Park, Maryland 20742

ABSTRACT

A review and critique is presented of the concepts of cartographic modeling as described in the text *Geographic Information Systems and Cartographic Modeling* by C. Dana Tomlin, Prentice-Hall, Englewood-Cliffs, NJ, 1990. This is coupled with a detailed discussion of the underlying representations and operations. The review is augmented by explanations of key concepts from cartography, spatial databases, and image processing. In addition, enhancements are proposed to deal with some perceived shortcomings and inconsistencies.

February 21, 1996

1. REPRESENTATIONS

The principal theme behind cartographic modeling (à la Tomlin) is one of overlay mapping. The user has a library of maps (all in registration - i.e., a common origin), and the goal is to perform a sequence of operations on them. The maps are inherently two-dimensional. From a spatial occupancy standpoint, a cartographic model consists of a hierarchy with a map layer at its highest level. Each map layer consists of zones. Each zone consists of locations. Each location consists of a pair of coordinate values that specify its address.

A map layer is usually described by its title, resolution, orientation, and its constituent zones. The distinction between a map layer and a conventional map is that on a conventional map each location is characterized in terms of a number of attributes (zero to many). On the other hand, each map layer is characterized in terms of exactly one attribute. Thus it captures the variation of just one variable. The set of locations having a particular value or range of values for each attribute form a zone. The area spanned by a particular zone need not be contiguous (i.e., when several non-connected regions in the map layer have the same attribute value). However, the zones must be mutually exclusive (i.e., disjoint) as well as all inclusive (i.e., span the entire map layer). A zone is also frequently referred to as a *class*.

A zone is described by its label, value, and location(s). A label is the written name of the zone (it can also serve as the name of the value). With the exception of the null value (which corresponds to a zone of whose characterization is unknown), values must be integers. A distinction is made between values and measurements. Values correspond to measurements which can correspond, in increasing complexity, to nominals (e.g., wheat, corn, rice, etc. in a crop coverage map layer), ordinals (e.g., first, second, third, etc.), ranges (i.e., intervals), or ratios (also includes numbers). Not all operations are meaningful when applied to some of the measurement types; however, all operations are applicable to every value.

Locations are the primitive elements of cartographic space. In particular, each location in a layer is associated with a geographic characteristic. The location corresponds to a grid square where all grid

squares are of a uniform size, shape, and orientation. A grid square is commonly known in image processing and computer graphics as a *pixel.* Locations can be aggregated in terms of rows, columns, and neighborhoods. Neighborhoods can be defined by distance and/or direction, as well as a range or ranges thereof. They differ from zones in that they are allowed to overlap whereas zones are disjoint.

There are a number of ways to implement the layer, zone, location hierarchy. The first is to associate a one-dimensional array with each location where each element of the array corresponds to a layer. The second is to have one two-dimensional array for each layer. The third is to have a hierarchy of records where there is one record for each layer. Each layer record consists of a set of zone records. Each zone record consists of a set of location records that comprise it. Whenever possible, our discussion is independent of the layer-zone-location implementation, although at times there is an implicit assumption that the second implementation is being used. The extension to three-dimensional data is straightforward. In the case of temporal data, time can either be treated as the third dimension in the case of two-dimensional data, or as the fourth dimension in the case of three-dimensional data.

Locations are specified by their coordinates. There are many ways of encoding the information stored in a location. The most straightforward approach is to record explicitly the pair of coordinate values for each location and the value. This is quite costly in terms of the storage that is required (i.e., three numbers for each location), and thus there is considerable interest in taking advantage of the regularity of the cartographic grid to make use of the implicit, rather than the explicit, associations between locations and their coordinate values. The nature of this association can be one of a simple ordering or an ordering coupled with an aggregation of similarly-valued locations. The associations can be based either on the interior or on the boundary of the space occupied by the similarly-valued locations.

Let us first look at associations based on the interior of the space occupied by the locations. The locations can be stored and processed in an order that corresponds to their positions in the cartographic grid. In essence, these orders are linearizations of the cartographic grid. The advantage of an ordering is that now only one number is necessary for each location (i.e., its value). The orders are also known as space-filling since they pass through every location in the grid. There are a number of choices available. The principal ones are raster-scan and Peano. The difference is that the former orders the grid by rows (or columns) while the latter orders it by blocks (i.e., quadrants of the grid are recursively exhausted before proceeding to the next quadrant).

Further space saving can be obtained by augmenting the ordering with an aggregation based on similarity of values. The nature of the aggregation depends on the ordering. These are known as runlength representations and can be one-dimensional (e.g., by rows or columns) or two-dimensional (e.g., by squares or blocks of arbitrary size). Quadtrees and medial axis transforms (i.e., MATs) are examples of the latter.

Associations based on the interior of the space occupied by the locations are the most general because they can be used effectively even without resorting to similarity-valued aggregation. In contrast, the effectiveness of associations based on the boundary does depend on similarity-valued aggregation, as otherwise all we have is a collection of boundaries of individual locations. In particular, associations based on boundaries are most useful when used in conjunction with zones whose interiors consist of sets of locations where the cardinality of the sets is much greater than one. In such a case, the ordering is relative in the sense that each location in the sequence is described by its relationship to an adjacent location in the sequence.

The chain code is an example of a boundary representation. Recall that a boundary encloses a region consisting of similarly-valued locations. Instead of recording the coordinate values of the locations, it records the coordinate values of one location that is adjacent to the boundary (usually so that the boundary is to the right of the location) and the sequence of unit vectors which comprise the boundary. Each unit vector has one of four directions - i.e., one for each possible boundary of a location (i.e., a grid square). Aggregation can be taken advantage of by examining the locations where the boundary changes direction. The most appropriate aggregation technique is similar to the runlength representation

discussed above. The difference is that the aggregation is now based on the constancy of the direction of the boundary rather than on the value of the locations. In particular, no two consecutive elements on the aggregated boundary have the same direction. This is implemented by recording a number with each direction that indicates its length. An alternative is to store just the locations where the directions change. The actual direction can be obtained by comparing the coordinate values of adjacent locations. Our discussion has assumed that the layer (consisting of regions in this case) is four-connected. This means that if two similarly-valued locations are only adjacent along a corner, then they are in different regions. If they were in the same region, then we would say that the layer is eight-connected.

An encoding that makes use of the boundary of a region (or sequence of locations) is known as a *vector* format because each element of the boundary usually has a magnitude and a direction. In contrast, an encoding that makes use of the interior of a region (or sequence of locations) is known as a *raster* format. These formats differ in how they affect the resolution of the cartographic space and in the definition of location that they support. In the case of the raster format, the resolution becomes finer as the size of the grid squares is decreased. However, the volume of the data also increases dramatically. On the other hand, to increase the resolution of the vector format we only need to increase the precision with which the coordinate values are specified. The result is that the percentage of locations that have meaningful data on a raster map layer is typically several orders of magnitude smaller than on a vector map layer.

The raster format is based on a decomposition of cartographic space into discrete units (termed *grid squares* or *pixels* in two dimensions, and *voxels* in three dimensions). This is also true when the vector format is used to designate locations such as point data and the endpoints of lineal data. However, this discretization is inapplicable to the rest of the lineal data as the resulting partition of space need not conform to the underlying grid. The result is that aggregates of locations can no longer be used to represent all of the spatial characteristics. In particular, lineal, areal, and surficial conditions are now represented as vectors, polygons, and polyhedra, respectively.

The discretization of the space that is inherent to the raster format also affects the quality (i.e., accuracy) of the measurements that can be obtained. This is especially noticeable for the approximation of a point by the center of the grid square in which it lies. In particular, the error in measuring the distance between two points may be as large as the length of the diagonal of a grid square. Similarly, the error in measuring the slope of the line connecting two points may be as large as 90 degrees. Increasing the resolution will reduce the magnitude of the errors in distance measurement. However, it will not reduce the slope error.

In general, when making a transition from the discrete locations of the raster format to the specialized locations (i.e., points, vectors, polygons, and polyhedra) of the vector format, we are moving from a more general unit of cartographic space to more specialized units. It is interesting to note that defining cartographic space as a set of discrete locations (i.e., grid squares) can be supported by data encodings based on raster and vector formats. The support of the raster format is clear. In the case of the vector format, each line (i.e., vector) is interpreted as a sequence of consecutive grid squares which should be, of course, of a fine resolution. For areal and surficial data, the interpretation is as a group of connected grid squares.

The vector format permits spatial variability in the sampling process as sampling only occurs at locations of interest. In contrast, the raster format samples at regular intervals and thus its use may result in a waste of resources in areas where no variation takes place. In terms of data acquisition, the raster format is more appropriate for automated methods while the traditional paper map is in the vector format. In terms of output, the raster format is closely related to photographic images, while the vector format is like a line drawing, and the ease with which it can deal with symbolic representations explains why it has been traditionally associated with cartography. In terms of data interpretation, the raster format lends itself to queries that are location-based while the vector format is more feature-based. An interesting and appropriate characterization of the two formats is that the raster format associates features with locations, while the vector format associates locations with features.

There are four types of geographic features: points, lines, areas, and surfaces (actually volumes as well, but we do not dwell on them here). Problems arise because the cartographic space of grid squares does not bear much resemblance to the true world of geographic space. Some examples of problems include the determination of what part of a grid square to associate with a point. Should it be the center or a corner? For example, if it is the center, then a change of the resolution of the map means that the point no longer lies in the center of a grid square. How is lineal data such as a road represented? One approach is to associate a grid square with each part of the space that is occupied by the lineal feature. This has a large potential for ambiguity. For example, what happens when a lineal feature passes through four adjacent grid squares that form a 2×2 block?

Problems with areal data arise with respect to measuring its size. Do we use the perimeter or the area? Use of the perimeter is not so simple (aside from the issues of scale - e.g., the fractal dimension) since there are three possible measurements. We can measure the number of grid squares on the outer boundary, the number of grid squares on the inner boundary, or the number of boundary segments between grid squares. Another problem pertains to the location of the edges of the zones. Are the edges restricted to be rectilinear? Should the corners of the grid squares be beveled according to the values associated with their neighbors, thereby overcoming the rectilinear restriction?

Problems also arise with surface data. Surface data is expressed by points representing positions in a third dimension, termed *vertical*, which are perpendicular to the horizontal plane consisting of the locations in the zones of the layer (also termed the *cartographic plane*). A distinction must be made between true three-dimensional data and surface data (termed *surficial*) in that surfaces do not include objects such as polyhedra, spheres, boxes, etc. since their surfaces are usually not single-valued (i.e., there can be more than one vertical position associated with each location in the cartographic plane). Associating a single vertical position with each grid square (usually its center) leads to discontinuities at the grid square's boundaries and results in a surface that looks like a skyline (or a collection of pencils). Thus interpolation should be used at the edges and corners of the grid squares.

The above problems are rooted in the fact that only rarely does the geographic data fit into the boundaries of the grid square. Often, the use of a location to represent the zone only means that the geographic feature occupies a part of the location's grid square. In particular, forcing the geographic features to lie within the grid squares leads to a number of issues whose resolution often results in the elimination of certain spatial configurations (e.g., 5 line segments cannot meet at a point). Of course, many of these issues could be overcome by use of spatial indexing. In such a case, the grid squares do not correspond to the actual geographic features. Instead, the grid squares just contain pointers to an alternative representation which captures the features in a more precise manner.

Tomlin introduces a number of techniques for dealing with the issues that arise because the point, line, area, and surface data do not fit neatly into the rectilinear compartmentalization of space that is induced by the grid squares. They consist of refinements in the form of additional map layers where each location is set to a value that characterizes the geographic feature (i.e., point, line, area, or surface) that is represented by the location, or neighborhood of the location, in the original layer.

Tomlin infers lineal data from a set of grid squares under the assumption that if two similarly-valued grid squares are adjacent along an edge or a corner, then a line exists between their centers. However, if three similarly-valued grid squares are adjacent along both an edge and a corner so that the result is in the shape of an 'L', then the lateral (i.e., horizontal or vertical) adjacency takes precedence over the diagonal adjacency. The adjacencies are classified into 47 legal configuration patterns. A map layer that is to be interpreted as lineal data is usually accompanied by an additional map layer which associates the identity of the appropriate configuration with each location.

Tomlin infers non-rectilinear region data (henceforth termed *areal*) by permitting the corners of the grid squares to be interpreted as being beveled. The areal shapes are defined by examining the values of the four locations that share each corner of a grid square. Each location can have 4 possible values (black, black-gray, gray-white, and white). This leads to 4^4=256 possible configurations patterns. Using

symmetry, rotations, and reflections, this is reduced to 7 sets of basic configuration patterns. A map layer that is to be interpreted as non-rectilinear areal data is usually accompanied by additional map layer indicating which of its corners are beveled. This information plus the colors of the relevant neighbors are sufficient to describe any location. Of course, the result is just an approximation. To obtain more refined inferences, we would have to look at more than four locations that are share a corner.

The 7 sets of basic configuration patterns are derived as follows. There are 4 configuration patterns when all locations are a single color. There are 48 configuration patterns when 3 locations are one color and the remaining location is another color. There are 24 configuration patterns when 2 laterally adjacent locations are one color and the remaining 2 locations are another color. There are 12 configuration patterns when 2 diagonally adjacent locations are one color and the remaining 2 locations are another color. There are 96 configuration patterns when 2 laterally adjacent locations are one color and the remaining 2 locations are each of a different color. There are 48 configuration patterns when 2 diagonally adjacent locations are one color and the remaining 2 locations are each of a different color. There are 24 configuration patterns when each location is of a different color.

Tomlin models surficial data by rectangular volume elements whose surface consists of 8 triangular facets. The facets are formed by associating an elevation with the center of each grid square. The elevations of the corners of each grid square are set to the average elevation values of its four diagonally adjacent neighbors. The elevations of the midpoints of the sides of each grid square are set to the average elevation values of its two laterally adjacent neighbors.

Although our presentation is in terms of layers, at this point it is worthwhile to contrast it briefly with an object approach to modeling. The layer approach yields a continuous view of the world, while the object approach yields a discrete view. The objects are usually points, lines, areas, etc. The description of an object is usually combined with a specification of the applicable operations. The layer approach is more general; however, it is somewhat inefficient when attributes are defined only over a limited geographic area or classes of objects.

2. OPERATIONS

Given the basic map layers for a cartographic model, we can apply a variety of operations to combine these layers into new ones. The operations involve combinations of maps (e.g., Boolean or set-theoretic), and the application of particular functions to the maps. These functions can either measure, search, or reclassify (i.e., recode) the data. They are expressed in terms of the individual locations (local), the zones (zonal), or the neighborhoods of the locations (focal). An alternative differentiation of the function types is that the involved locations are coincident in position (local), related thematically (zonal), or related spatially (focal). Examples of measurements include sum, mean, difference, diversity (e.g., standard deviation and moments), maximum, minimum, product, ratio, majority, minority, variety (also known as cardinality), etc. An additional set of functions (incremental) is also introduced to cope with the fact that much cartographic data does not fit into the rectilinear compartmentalization of space that is induced by grid squares (i.e., to infer other characteristics of it such as size and shape). These functions are a combination of local and focal operations. Note that all operations are specified in terms of their effect at the grid square level. As such, they lend themselves to an environment where massive parallelism is present (e.g., a SIMD architecture) since every operation is defined in terms of its effect on every grid square in the map layer - i.e., it is applied to every grid square in the map layer.

Local operations have the effect that each location in the destination map layer is set to a value that is a function of the location's existing values in one or more source map layers. At least one of the source layers corresponds to an actual map layer. The remaining source layers may be specified as numbers, in which case they correspond to map layers where every location's value is the specified number.

Local operations have four variants - those that measure, those that search, those that correspond to a recoding or a reclassification, and those that correspond to variants of the polygon overlay operation (also known as composition or superposition). We do not elaborate further on the first two here. In the case of recoding or reclassification, one of the source operands is usually a map layer, while the second source operand, if present, is usually a number (e.g., a ratio or product operation where the result is a rescaling). When the reclassification can not be expressed so concisely, a LocalRating operation is used to specify an explicit correspondence from existing values or ranges of values to new values. All values that are not explicitly reclassified are left alone. The LocalRating operation (as well as its focal and zonal variants) can also specify a correspondence with a map layer instead of a value. In this case, the location's value in the map layer corresponding to the second source operand serves as the new value reclassifying the value designated in the primary source operand. This feature makes it very easy to implement a Boolean set operation. For example, the union of map1 and map2 is ''LocalRating of map1 with 1 for 1 and map2 for 0'' while the intersection of map1 and map2 is ''LocalRating of map1 with 0 for 0 and map2 for 1.''

Polygon overlay operations take the values associated with a location in two or more source layers and calculate (according to the function) a new value which is stored in the destination layer. When the new value cannot be expressed by one of the predefined mathematical functions (e.g., sum, product, minimum, etc.), then a LocalRating operation is used to specify an explicit correspondence in the form of a new destination layer value for each possible combination of input layer values or combination of ranges of input layer values. Unfortunately, when the range of values is large, the LocalRating function becomes cumbersome. Moreover, many of the combinations never arise. For example, given two maps with 4 and 5 zones (i.e., values) respectively, there is a maximum of 20 destination layer values that may need to be specified. The LocalCombination operation overcomes this problem by simply assigning a unique new value to each combination of existing values that actually occurs.

Zonal operations facilitate a special case of the polygon overlay operation. A typical zonal operation involves three map layers. Two of the layers serve as source operands while a third layer serves as the destination. One source layer (termed the *input layer*) contains input values (one for each location, although they need not be distinct), while the second source layer partitions the space into zones and serves as a mask (termed the *mask layer*). The zones behave like attributes in the sense that they usually correspond to nonspatial information such as crop types, soil types, etc. If the mask layer is omitted, then it is assumed to consist of one zone. The result is a partition of the destination layer into zones corresponding to the mask layer, where the value of each location in a zone of the destination layer is the result of the application of the operation to all the locations of the zone in the input layer. Thus we see that the operation is zonal with respect to the partition induced by the mask layer (thereby ignoring the partition induced by the input layer) while involving the values in the input layer (thereby ignoring the values in the mask layer). In contrast, in the general polygon overlay operation, the destination layer consists of a set of new zones which correspond to a subset of the Cartesian product of the zones in the two source layers, and hence does not ignore the values in either of these layers. Using the terminology of relational databases, polygon overlay is a *join* operation where the condition is coverage of the same group of grid squares.

The most typical zonal operations involve properties of the entire zone. Some examples include maximum, minimum, mean, product, and sum. Note that unlike local operations, the value of a zone in the destination layer is a function of all the values in the zone's corresponding locations in the input layer. For example, ZonalSum is the sum of the different values associated with the zone's locations in the input layer. The number of different values is given by the ZonalVariety operation. An interesting operation, which is somewhat related to the polygon overlay operation, is called ZonalCombination. It indicates for each zone in the mask layer the collection of different values associated with the corresponding locations in the input layer.

Actually, zonal operations are more general than described here. Our discussion has assumed that as a result of a zonal operation, all locations in a particular zone will have the same value. This is not necessarily so for all zonal operations. For example, a zonal operation could be used to contrast the

value of each location l, to a statistic that summarizes the values of all locations in the zone containing l. This can be seen by examining the ZonalRanking operation. Given location l with value v in zone z, ZonalRanking determines how many values less than v are associated with locations in z and assigns this number as the value of l in the destination layer. Note that in essence this operation results in ranking partial zones in z where the partial zones correspond to locations with the same value in z. Other zonal operations with a similar meaning include ZonalPercentage and ZonalPercentile.

One of the problems with zonal operations is that they don't always yield the desired results when the zones are not contiguous. For example, suppose that the values stored in the input layer are elevations and we wish to obtain the maximum difference in elevation between two locations in the same zone. In this case, we want all the locations in the zone to be contiguous. Also, at times, it may be desired to define operations that deal with the interrelationship between a zone and its non-contiguous components. The problem is that the concept of a zone serves to indicate both spatial contiguity and non-spatial contiguity, and we need to distinguish between the two. This can be done operationally or conceptually. Operationally, we execute a procedure that uniquely labels each similarly-valued location in a contiguous area (see the discussion of the FocalInsularity function below). Conceptually, and the approach we follow, we define an additional grouping hierarchy, termed a *clump*, that lies between a location and a zone. This hierarchy consists of contiguous locations.

Use of clumps implies a two-level aggregation hierarchy. In order to cope with this two-level hierarchy we redefine the zonal operation to involve as many as four map layers. In all cases, we have an input layer and a destination layer. Instead of one mask layer partitioning the space into zones, we now have one or two mask layers. One mask layer partitions the space into clumps (termed the *clumpmask layer*) while the second mask layer aggregates the clumps into zones (termed the *zonalmask layer*). Zonal operations could be defined as ''ZonalFunction of InputLayer [in ClumpmaskLayer] [within ZonalmaskLayer]''. The result is that some operations only require a clumpmask layer while the more general ones require both clumpmask and zonalmask layers. The clumpmask layer could be omitted in some cases with only a zonalmask layer being specified. Given a zonalmask layer, there is no need for a clumpmask layer since the information provided by it can be obtained by applying a FocalInsularity operation. However, having both a clumpmask and a zonalmask layer means that we can determine quickly if two distinct locations that are in the same zone are in the same clump (i.e., component) of the zone. This is useful if we want to know how many non-contiguous regions make up each zone.

Neighborhood operations (also termed *focal*) are concerned with the relationships between locations on the same map layer rather than between instances of a location on different layers. A neighborhood is a set of one or more locations in the same map layer that are within either a specified distance or direction (also termed *bearing)* of a given location. This location is termed the *neighborhood focus* and hence the origin of the qualifier ''focal'' used in naming the operations.

There are a number of ways of measuring neighboring distance that include physical separation, reachability, trafficability, time, line of sight, etc. A distinction between operations can be drawn on the basis of the extent of the neighborhood. Some operations are only meaningful for the immediate neighborhood of the focus (i.e., its adjacent locations), in which case no distance or direction value is specified. Other operations are only meaningful for an extended vicinity. Some operations are applicable to both types of neighborhoods. In fact, neighborhoods could also be defined in terms of zones. This definition is somewhat cumbersome when the zones are not contiguous since it may involve a search.

The most common operations are those that are applied to the adjacent locations. They bear a close resemblance to their image processing counterparts such as convolution, filtering, smoothing, etc. Many of the operations are very similar to the local and zonal operations discussed earlier (e.g., maximum, minimum, mean, product, sum, combination, variety, etc.). However, the difference is that the focal operations are applied to the values of multiple locations in the same layer instead of to the same location on multiple or mask layers, as is the case for their local counterpart. For example, a FocalCombination operation indicates the combination of values that occur within each location's neighborhood. A FocalRating operation reclassifies each location's value according to the combination of values

occurring within its neighborhood. A null value is associated with all unspecified combinations. This operation is equivalent to a FocalCombination followed by a LocalRating (i.e., a composition). Some operations have the effect of summarizing the neighborhood's values. These are known as filtering operations. The summaries can either accentuate the differences (e.g., minimum, maximum), or try to smooth the differences (e.g., mean, sum, product, rating).

Filtering operations are common in image processing. They are achieved by moving a window across the entire image (i.e., map). The window is often square and typically consists of a 3×3 array of pixels. The value of a location in the middle of the window is often a weighted average of the remaining locations in the window. There are two types of filters. The first is known as a *low pass filter*. It smoothes the value by removing or reducing local detail. Two 3×3 filters are given below. The filter on the left results in severe smoothing, while the filter on the right achieves a slight amount of smoothing.

.11	.11	.11	.06	.06	.06
.11	.11	.11	.06	.52	.06
.11	.11	.11	.06	.06	.06

The second is known as a *high pass filter*. It enhances edges by exaggerating the differences or local detail. An example 3×3 filter is given below. It results in some enhancement by removing the effect of the neighbors.

-.06	-.06	-.06
-.06	1.48	-.06
-.06	-.06	-.06

At times, we may wish to identify all adjacent locations with the same value. This is achieved by the FocalInsularity operation. It results in a unique number being assigned to each group of spatially contiguous locations that have the same value. This operation is common in image processing and is known as *connected component labeling*.

Many of the above operations are also applicable to arbitrary (i.e., extended) neighborhoods. Of course, often the result is that the effect of the operation is exaggerated or smoothed. The extent of the neighborhood that is being examined can be limited by specifying a distance and a direction. When no direction is specified, the neighborhood extends in all directions. On the other hand, when no distance is specified, the neighborhood is limited to the immediately adjacent locations. The distance and direction parameters can be numbers or the titles (i.e., names of existing map layers). The use of map layers for the distance and direction parameters increases the generality of the focal operations since it means that the neighborhood can vary from location to location.

The following four extended neighborhood operations are frequently used in the implementation of queries that require locating nearest neighbors. The FocalProximity operation determines the distance to the nearest location whose value is not null. Specifying a distance value d with the operation results in constraining the search neighborhood, so that when no location is found, the result is d. This operation is commonly known as computing a buffer or a corridor (also dilating an image). It is very useful in searching. The FocalBearing operation determines the bearing (ranging from 1 to 360) of the nearest location whose value is not null. The FocalNeighbor operation differs from the FocalProximity and FocalBearing operations by returning the value of the nearest neighbor rather than its distance or bearing. The resulting decomposition of space is called a *Voronoi diagram* (also known as a *Thiessen polygon*). The effect of this operation is one of cartographic interpolation. The result is analogous to an estimate of unknown values where the estimate is restricted to be one of a number of predetermined values. Frequently, it is desired that the estimate be weighted by the neighboring values. The FocalGravitation operator yields such an estimate by weighting each known value by the inverse of the square of its distance from the neighborhood focus.

Besides restricting a neighborhood on the basis of distance and direction, a neighborhood can also be restricted on the basis of visual contact or the cost (i.e., in time, energy, money, etc.) of actually moving from one location to another. In the case of visual contact, it is assumed that the neighborhood focus contains a light source which radiates light to the rest of the neighborhood (there can be more than one foci). For visual contact to occur between the neighborhood focus f, and a location l, there must exist an unobstructed line of sight between the vertical positions of l and f. This is achieved by the *radiating* variation of the focal operations. In this case, the extent of the neighborhood is determined by unobstructed lines of sight. Thus the obstruction of light is now the basis of the distance calculation. These operations find application in tasks involving surveillance, siting of unsightly facilities, etc. Their proper implementation requires the specification of a number of map layers in addition to the cartographic plane.

First of all, a surface map layer must be specified to indicate the vertical position (i.e., topographic elevation) of each grid square. The remaining map layers all contain vertical positions that are relative to those of the surface layer. Thus to obtain the true vertical position at a location, one of its additional map layer values must be added to its corresponding surface map layer value. A transmission map layer indicates a set of values for establishing the vertical position of a neighborhood focus (i.e., the light source). An obstruction map layer indicates a set of values for establishing the vertical positions of the obstacles (e.g., heights of buildings, vegetation, trees, etc.). The line of sight cannot pass through these locations to further locations at lower vertical positions. Finally, a reception map layer indicates a set of values for establishing the maximum vertical positions at which light (i.e., a line of sight) can be received. The reception layer indicates the maximum three-dimensional volume that could be illuminated by the combination of light sources. For example, it is cone-shaped for a covered light source while it is a dome for a light source with no cover. In essence, the reception layer records the relative elevation value of the highest location visible at each grid cell. Notice that if there is more than one light source, then we need to merge the individual reception layers into one before applying the operation. This result in a uniform illumination model (i.e., each location has the same intensity or brightness regardless of whether or not there are two or more light sources in close proximity). If we want to deal with a situation with varying intensities, then the operation must be performed for each light source separately after which the results are combined (i.e., the intensity values of the individual locations are added). Note that the extent of a reception layer r is a function of the intensity of a light source, and that the r may have a value even for the grid cell containing the corresponding light source. If any of the map layers are not specified, then they are assumed to be zero. The conditions specified by the transmission and reception layers have no effect on the visual obstructions.

Restricting the neighborhood on the basis of accessibility and the cost of making the actual motion is accounted for by the *spreading* variation of the focal operations. The key is that the minimum distance (or cost) between two locations is not necessarily the length of the straight line between them (i.e., the Euclidean distance). Instead, the distance is measured in terms of the lengths of the links that are traversed in proceeding from the neighborhood focus to the appropriate location. This means that we take into account obstructions in the sense that we need to move above (i.e., in the vertical) and around (analogous to the diffraction of light) them, and trafficability in the sense that we need to change the speed of the motion (analogous to the refraction of light). At times, these operations require the specification of a number of map layers in addition to the one representing the cartographic plane.

A friction layer indicates the incremental cost of proceeding through each location. It acts very much like an impedance factor. In essence, it indicates the number of incremental units of travel cost (time, money, etc.) that will accrue as a result of proceeding through each location. The actual cost of a link from a location to its neighbor is computed by averaging the friction layer values of the two locations (and multiplying by a factor of 1.414 in case the link is diagonal). A surface layer indicates the vertical position (i.e., topographic elevation) of each location. This has the effect of increasing the actual distance between a location and its adjacent neighbor by the product of their vertical separation and the secant of the vertical angle between them. A network layer indicates which of each location's 8 adjacent neighbors are reachable. This is in the form of an 8 bit vector where each bit corresponds to one of the 8 directions (multiples of 45 degrees). If no additional map layers are specified, then the

distance is simply the minimum of the sum of the lengths of the links traversed from the neighborhood focus to the location. We use the minimum since often there is more than one path.

Incremental operations are introduced to augment the set of focal operations when the contents of the grid squares are interpreted to contain data other than a square two-dimensional region. The nature of the operation and the neighborhood depend on the feature that is associated with the location (e.g., point, line, area, or surface). The rules that underly this interpretation were discussed earlier. The result is a new map layer. Incremental operations are only applicable to the immediate neighbors of a neighborhood focus, and in all directions. Hence, no restriction on direction is permitted.

Point data does not require additional operations. For lineal data, IncrementalLinkage infers the nature of the lines that pass through the location, while IncrementalLength indicates the total length of whatever portions of the lines pass through the location. For areal data, IncrementalPartition infers the shape of the areal element corresponding to the location, IncrementalArea indicates the planar area of the inferred areal element, and IncrementalFrontage indicates the length of the inferred edges between the inferred areal element and its immediate neighbors that are not in the same zone. The input to IncrementalLength, IncrementalArea, and IncrementalFrontage is a map layer where the inference of the lineal and areal shapes has already been performed. In addition, they can also take surface information into account, in which case a surface map layer is also specified.

Four incremental operations are defined to handle surface data. In this case, a surface map layer containing vertical positions (i.e., elevations) must be specified. An additional map layer may also be included to specify areal conditions under the surface. If none is specified, then all locations are assumed to lie in the same zone (and hence have the same value). IncrementalVolume calculates the volume under the surface for each location. IncrementalGradient yields the slope of the plane inferred (by use of an appropriate interpolation method) from the surface layer values of the location and all immediate neighbors that have the same zonal value. This value is in degrees, where a horizontal surface has a slope of zero and a vertical surface has a slope of 90. IncrementalAspect yields the compass direction of the steepest descent for the plane inferred (by the IncrementalGradient operation) from the surface layer values of the location and all immediate neighbors that have the same zonal value. The direction ranges from 0 to 360 degrees, where a nonsloping surface has a value of 0. IncrementalDrainage indicates which of a location's adjacent neighbors lie upstream (i.e., the watershed) on a surface inferred from the surface layer values of the location and those of its immediate neighbors that have the same zonal value. Note that the upstream neighbors of a given location are the ones from which the location is in a direction of steepest descent. The result is specified in the form of an 8 bit vector where each bit corresponds to a neighbor in one of the 8 directions (multiples of 45 degrees).

It is interesting to note that the above four incremental operations are applicable to all types of surfaces (not just topography). Moreover, IncrementalGradient and IncrementalAspect can be used to measure the rate of change or direction of change in any continuous quantity over space. Actually, these quantities are not really continuous since they are measured at discrete points (i.e., the grid squares).

Tomlin also provides a language (or more appropriately a syntax) for the statements that invoke the operations and the layers affected by them. The effect is very similar to that achieved by SQL (structured query language) for relational databases.

3. MODELING

Tomlin subdivides cartographic modeling into two classes. The first is *descriptive* and is associated with queries of the form ''what is'' or ''what could be''. The second is *prescriptive* and is associated with queries or statements of the form ''what should be''. Descriptive modeling techniques can be differentiated on the basis of whether they are analytic or synthetic. Analytic methods decompose the data into finer levels of meaning, while the synthetic methods recompose or aggregate the data with the goal of discovering new meanings. Analytic methods can be further decomposed into those that

analyze cartographic position and those that analyze cartographic form.

The position of a cartographic condition is usually expressed in terms of measurements indicating its absolute or relative position. The computation of the centroids of the zones of a map layer is an example of absolute position. Relative positions are usually expressed in terms of the distance or direction between locations. An example of an operation involving relative position is the computation of the gradient. In fact, Tomlin draws an interesting analogy between operations that involve relative position and calculus. An operation such as ''FocalProximity of X spreading in FRICTIONLAYER'' is similar to integration since the result is like the area under a curve. The similarity lies in the fact that the values generated by the FocalProximity operation yield the area under the FRICTIONLAYER surface along a minimum cost path. Similarly, the IncrementalGradient operation, which yields the slope at each location, is analogous to differentiation. Thus, applying an IncrementalGradient to the result of ''FocalProximity of X spreading in FRICTIONLAYER'' yields something similar to the original FRICTIONLAYER. We have a similarity rather than an exact match because of the two-dimensional nature of the X layer. If the X layer was one-dimensional, then the result would indeed be identical to FRICTIONLAYER.

The analysis of cartographic form is usually in terms of its size and shape. This means that it is dependent on the nature of the geographic feature that is being modeled. Point data by its very nature has no shape. However, shapes of point clusters may be a meaningful measure. For lineal data, form can be length. For areal data, form can be roundness, genus, number of holes, etc. For surface data, form can be topographic inflection (i.e., slope of slope), shadiness, narrowness, etc.

Although analytic techniques are often sufficient, sometimes synthetic techniques are need to characterize cartographic data. The purpose of synthesis is not only to expose significant facts in the data but also to express the meaning which the modeler may wish to attach to some of these facts. Analytic techniques are characterized as being objective while synthetic techniques tend to be subjective. This makes synthetic techniques more difficult to characterize. Another difference between these two techniques lies in the type of question that is being posed. Analytic methods are designed to ferret out ''what is significant'', while synthetic methods deal with the issue of ''how something is significant''. The issue of ''how'' can be further decomposed into ''what makes it significant'', and a ''value judgement of its significance''.

Tomlin separates the synthesis of a descriptive cartographic model into a formulation phase and an implementation phase. The formulation phase is very subjective and hard to pinpoint. The implementation phase is less subjective but is nevertheless more subjective than its analytic counterpart. The key issue is how to express the subjective judgement that is our goal. There are a number of choices. One is to make use of a LocalRating operation to characterize or filter a set of implications. An alternative, which is the most prevalent, is to make use of a LocalCombination operation. This has the advantage of synthesizing all the values in a way that avoids (or defers) making an explicit judgement. Another alternative is to use a common statistic such as LocalMajority, LocalMinority, or LocalVariety. These statistics are appropriate for all types of measurements. For all but nominal measurements, these statistics can be combined with operations such as LocalMaximum or LocalMinimum. LocalSum, LocalDifference, and LocalMean can be used with range and ratio measurements, while operations such as LocalProduct and LocalRatio can only be used with ratio measurements.

Prescriptive modeling techniques generally deal with cartographic allocation - i.e., the selection of locations that satisfy some criteria. The result is a solution of a problem. This process is inverted from the one used with descriptive modeling where we are exploring ''how things are''; here we are exploring ''how things will be''. Tomlin draws a distinction between what he terms *atomistic* and *holistic* allocation problems. In short, they differ on the basis of whether the allocation can be expressed in terms of individual locations of space (atomistic) or only in terms of the entire geographic space (holistic).

Atomistic allocation is usually a function of location. The criteria are usually very precise. For example, we may have a criterion that the air quality be above a certain value in the selected regions. The

solution process usually involves local operations. It is often the case that we have too many solutions, in which case the problem is most likely underconstrained. This means that the solution process must constrain the criterion further, and another iteration is performed.

Holistic allocation is usually a function of a neighborhood whose elements have similar properties. The criteria are varied (e.g., minimum cost paths, connectedness, size, etc.), imprecise, and often quite vague (e.g., sparseness, roundness, etc.). For example, we may wish to locate a site for an airport, which means that we must examine topological criteria for the positioning and grouping of one or more runways. In this case, the shape and the grouping pattern are as important as the location. The solution process usually involves focal and zonal operations. The inherent imprecision and vagueness of the problem means that we must often resort to heuristics to establish a starting point or existing condition which is subsequently iterated upon.

ACKNOWLEDGEMENTS

I have benefitted greatly from discussions with Kathleen Romanik.