What are normal forms in relational databases? Why are they used, that is, what are the advantages of putting data in higher normal forms?

Can you define the basic differences between first, second, and third normal forms?

Can you give an example of a functional dependency?

9 Basic Spatial Analyses

Introduction

Spatial data analysis involves the application of operations to coordinate and related attribute data. Spatial analyses are most often applied to solve a problem, for example, to identify high crime areas, to generate a list of road segments that need repaving, or to select the best location for a new business. There are hundreds of *spatial operations* or *spatial functions* that involve the manipulation or calculation of coordinates or attribute variables.

We note that the terms spatial operations and spatial functions are often used interchangeably. Some insist an operation doesn't necessarily produce any output, while a function does. However, in keeping with many authors, GIS practitioners, and software vendors, we will use the terms function and operation interchangeably.

Spatial operations may be applied sequentially to solve a problem. Each spatial operation may create output, and the output may serve as input to other spatial operations. A chain of spatial operations is often specified, with the output of each spatial operation serving as the input of the next (Figure 9-1). Part of the challenge of geographic analysis is the selection of the appropriate spatial operations, applied in the appropriate order.

The table manipulations we described in Chapter 8 are included by our definition of a spatial operation. Indeed, selection and modification of attribute data in spatial data layers are included at some time in nearly all complex spatial analyses. Many operations incorporate both the attribute and coordinate data, and the attributes must be further selected and modified in the course of a spatial analysis. Some might take issue with our inclu-

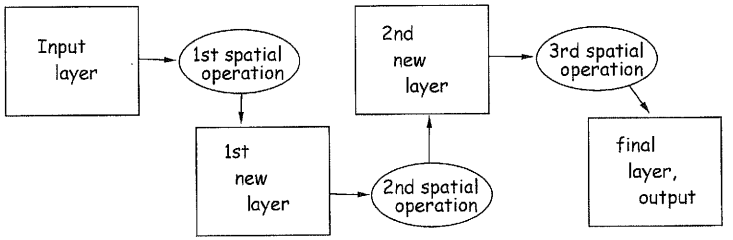


Figure 9-1: A sequence of spatial operations are often applied to obtain a desired final data layer.

sive definition of a spatial operation, in that an operation might be applied only to the "non-spatial" attribute data stored in the database tables. Attribute data are part of the definition of spatial objects, and it seems artificial to separate operations on attribute data from operations that act on only the coordinate portion of spatial data.

The discussion in the present chapter will expand on rather than repeat the selection operations treated in Chapter 8. This chapter describes spatial data analyses that involve sort, selection, classification, and spatial operations that are applied to both coordinate and associated attribute data.

Input, Operations, and Output

Spatial data analysis typically involves using data from one or more layers to create output (Figure 9-2). The analysis may consist of a single operation applied to a data layer, or many operations that integrate input data from many layers to create the desired output.

As shown in Figure 9-2, there may be single or multiple inputs or outputs from a spatial data operation. Many operations require a single data layer as input and generate a single output data layer. Vector to raster conversion is an example of an operation that typically takes a single input data layer and generates a single output data layer (Figure 9-3). There is a one-to-one correspondence between input and output layers.

There are also operations that generate several output data layers from an input, or that require several inputs to generate a single output data layer. Terrain analysis functions may take a raster grid of elevations as an input data layer and produce both slope (how steep each cell is) and aspect (the slope direction). Two outputs are generated for each input elevation data set. A layer average is an example of the use of multiple input layers to produce a single output layer. Mean annual grain production might be stored for 10 separate years in raster data layers. To calculate the average grain production over the 10 year period, the annual layers are combined and averaged on a cell-by-cell basis. This results in a single output data layer. Finally, there are some spatial operations that require many input layers and generate many output layers.

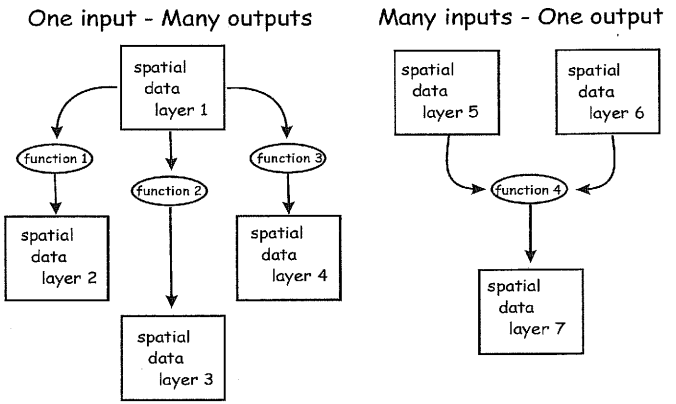


Figure 9-2: Much basic spatial data analysis consists largely of spatial operations. These operations are applied to one or more input data layers to produce one or more output data layers.

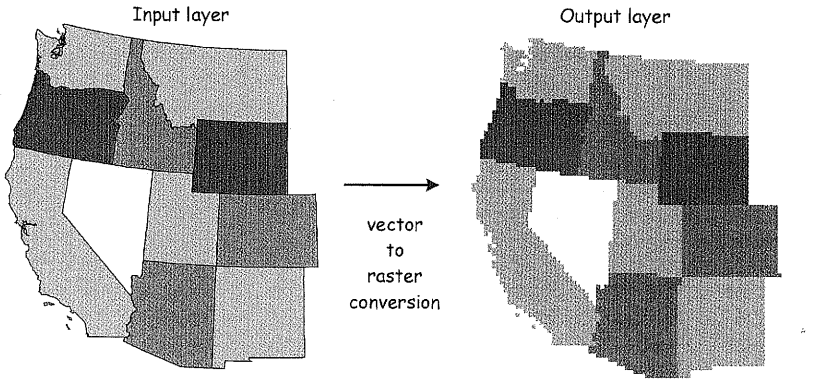


Figure 9-3: Vector to raster conversion, an example of a spatial data operation.

Spatial operations may be considered the basic components of spatial data analyses, and several operations are often used in quite long sequences when problem solving with spatial data. A single operation alone is usually not sufficient to solve a significant problem, and a series of operations is performed in most spatial data analyses. Most significant spatial analyses require multiple primary data inputs. Tens of spatial operations are applied, many of which generate new data layers, all of which may be used in creating output layers or scalars (single numbers).

The output from a spatial operation may be spatial, in that a new data layer is produced, or the output may be non-spatial, in that the spatial operation may produce a scalar value, a list, or a table, with no explicit geometric data attached. A layer mean function may simply calculate the mean cell value found in a raster data layer. The input is a spatial data layer, but the output is a scalar, the single number representing the mean raster value.

Other operations create aspatial output, for example, a list of all landcover types may be extracted from a data layer. The list indicates all the different types of landcover that can be found in the layer, but is non-spatial, because it does not attach each landcover to specific locations on the surface of the Earth. One might argue that the list is referenced in a general way to the area covered by the spatial data layer, but each landcover class in the list is not explicitly related to any specific polygons or points of interest in the data layer. Thus, a spatial input is passed through this "occurrence" operator and provides a list output.

Scope

Spatial data operations may be characterized by their *spatial scope*, the extent or area of the input data that are used in determining the values at output locations (Figure 9-4). There is often a direct

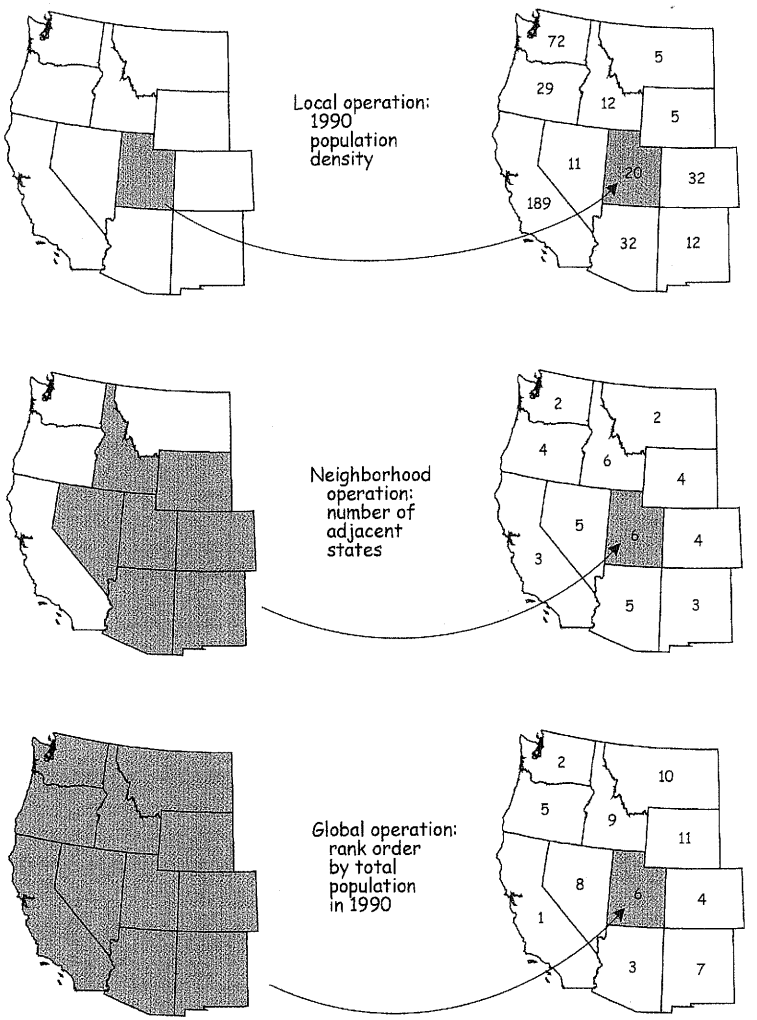


Figure 9-4: Local, neighborhood, and global operations. Specific input and output regions are shown for Utah, the shaded area on the right side of the figure. Shaded areas on the left contribute to the values shown in the shaded area on the right. Local operation output (top right) depends only on data at the corresponding input location (top left). Neighborhood operation output (middle right) depends on input from the local and surrounding areas (middle left). Global operation output (bottom right) depends on all features in the input data layer (bottom left).

correspondence between input data at a location and output data at that same location. The geographic or attribute data corresponding to an input location are acted on by a spatial operation, and the result is placed in a corresponding location in the output data layer. Spatial operations may be characterized as local, neighborhood, or global, to reflect the extent of the source area used to determine the value at a given output location.

Local operations use only the data at one input location to determine the value at a corresponding output location (Figure 9-4). Attributes or values at adjacent locations do not affect the operation or contribute to the value placed in an output location.

Neighborhood operations use data from both an input location plus nearby locations to determine the output value (Figure 9-4). The extent and relative importance of values in the nearby region may vary, but the value at an output location is influenced by more than just the value of data found at the corresponding input location.

Global operations use data values from the entire input layer to determine each output value. The value at each location depends in part on the values at all input locations (Figure 9-4).

The set of available spatial operations depends on the data model and type of spatial data used as input. Some operations, e.g., buffering or proximity operations (described later in this chapter) may be easily applied to raster or vector data, and to point, linear, or areal features. While the details of the specific implementation may change, the concept of the operation does not. Other operations may be possible in only one data model.

Characteristics of a data model will determine how a concept is applied. Analyses will vary depending on the model and the specific operations the GIS software developers chooses to include in their computer programs. Operation scope provides a good example. A local operation in a raster data layer typically matches input and output for a specific raster cell. The cell is uniformly defined in size, shape, and location. A local operation for a vector polygon data set is likely to have a variable size, shape, and location. In Figure 9-4 the local operation follows a state boundary. Therefore, the operation applies to a different size and shape for each state. Neighborhood analyses are affected by the shape of adjacent states in a similar manner. Summary values such as populations of adjacent states may be greatly influenced by changes in neighborhood size, so great care must be taken when interpreting the results of a spatial operation. Knowledge of the algorithm behind the operation is the best aid to interpreting the results.

While most operations might be conceptually compatible with most spatial data models, there are often significant differences among models in the ease with which a spatial operation may be implemented. Some operations are quite easy to program when using raster data models, and quite difficult when using vector data models. The reverse is true for other spatial data operations. In many instances it is more efficient to convert the data between data models and apply the desired operations, and if necessary, convert the results back to the original data model.

Selection and Classification

Selection operations involve identifying features that meet one to several conditions or criteria. The attributes or geometry of features are checked against criteria. Those that satisfy the conditions defined by the criteria are selected. These features may then be written to a new output data layer, or the geometry or attribute data may be manipulated in some manner.

Figure 9-5 contains an example of a selection operation that involves the attributes of a spatial data set. Two conditions are applied and the features that satisfy both conditions are included in the selected set. This example shows the selection of those states in the "lower 48" states of the United States of America that are a) entirely north of Arkansas, and b) have an area greater than 84,000 square kilometers. The complete set of features that will be considered is shown at the top of the figure. This set is comprised of the lower 48 states, with the state of Arkansas indicated by shading. The next two maps of Figure 9-5 shows those states that match the individual criteria. The second map from the top shows those states that are entirely north of Arkansas, while the third map shows all those states that are greater than 84,000 square kilometers. We are required to identify states that meet both conditions, so we need to identify states that are shaded in both the second and third maps. The bottom part of Figure 9-5 show those states that satisfy both conditions. This figure illustrates two basic characteristics of selection operations. First, there is a set of features that are candidates for selection, and second, these features are selected based singly or on some combination of the geographic and attribute data.

The simplest form of selection is an *on-screen query*. A data layer is displayed, and features are selected by a human operator. The operator uses a pointing device to locate a cursor over a feature of interest and sends a command to select, often via a mouse click or keyboard entry. On-screen (or interactive) query is used to gather information about specific features, and is often used for interactive updates of attribute or spatial data. For example, it is common to set up a process such that when a feature is selected the attribute information for the feature is displayed. These attribute data may then be edited, and changes saved.

Queries may also be specified by applying conditions solely to the aspatial components of spatial data. These selections are most often based on the attribute data tables for a layer or layers. These selection operations are applied to a set of features in a data layer or layers. The attributes for each feature are compared against a set of conditions. If the attributes match the conditions, they are selected; if the features fail to match the conditions, they are placed in an unselected set. In this manner selection splits the data into two sets: a selected set and an unselected set. Selected data are typically then acted on in some way. Selected data are often saved to a separate file, deleted, or changed in some manner.

Selection operations on tables were described in general in Chapter 8. The description here expands on that information and draws attention to specific characteristics of selections applied to spatially-related data. Table selections have spatial relevance because each record in a table is associated with a geographic features. Selecting a record in a table may also be used to select cells, points, lines, or areas. Spatial selections may be combined with table selections to identify a set of selected geographic features.

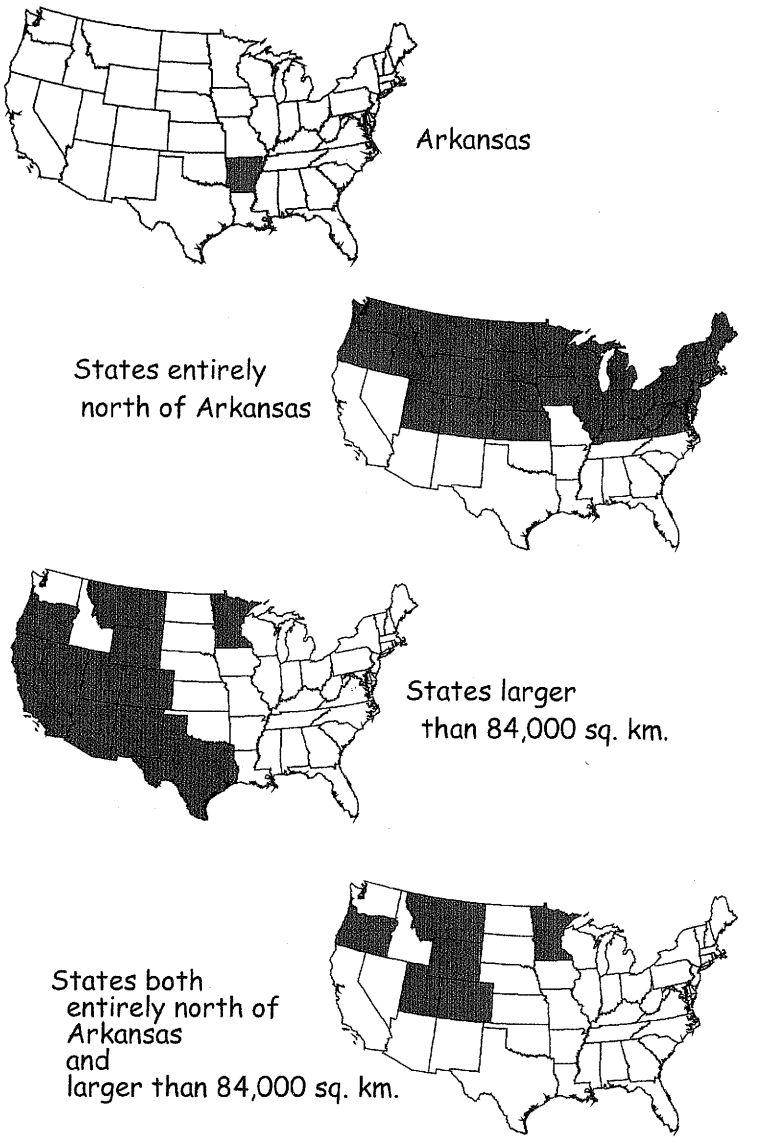


Figure 9-5: An example of a selection operation based on multiple conditions. All states both entirely north of Arkansas and larger than 84,000 square kilometers are selected. Arkansas is shown in the upper panel, followed by states matching each condition. States meeting both conditions are shown at the bottom of the figure.

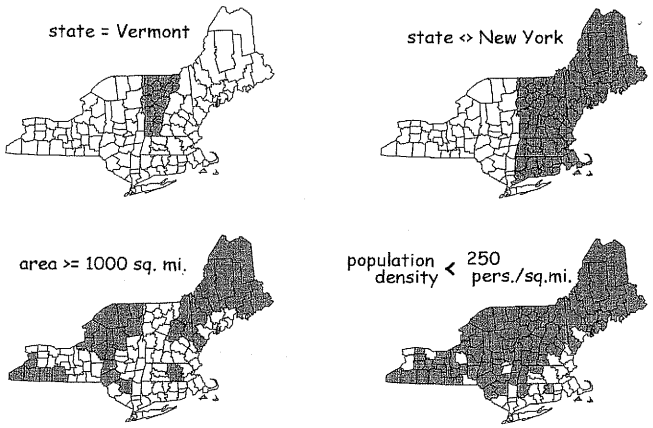


Figure 9-6: Examples of expressions in set algebra and their outcome. Selected features are shaded.

Set Algebra

Selection conditions are often formalized using *set algebra*. Set algebra uses the operations less than (<), greater than (>), equal to (=), and not equal to (< >). These selection conditions may be applied either alone or in combination to select features from a set.

Figure 9-6 shows four set algebraic expressions and the selection results for a set of counties in New England and New York. The upper two selections illustrated in Figure 9-6 show an equal to (=) and a not equal to (< >) selection. The upper-left shows all counties with a value for the attribute `state` that equals `Vermont`, while the upper right shows all counties with a value for `state` that are not equal to `New York`. The lower selections in Figure 9-6 show examples of ordinal comparisons. The left figure shows all counties with a size greater than or equal to (> =) 1000 square miles, while the right side shows all counties with a population density less than (<) 250 persons per square mile.

Set algebra operations greater than (>) or less than (<) may not be applied to nominal data, because there is no implied order in nominal data. Green is not greater than yellow, and red is not less than blue. Only the set algebra operations equal to (=) and not equal to (< >) apply to these or other nominal variables. All set algebra operations may be applied to ordinal data, and all are often applied to interval/ratio data.

Boolean Algebra

Boolean algebra uses the conditions `OR`, `AND`, and `NOT` to select features. Boolean expressions are most often used to combine set algebra conditions and create compound spatial selections. The Boolean expression consists of a set of Boolean operators, variables, and perhaps constants or scalar values.

Boolean expressions are evaluated by assigning an outcome, true or false, to each condition. Figure 9-7 shows three example Boolean expressions. The first is an expression using a Boolean `AND`. There are two arguments for the expression. The first argument specifies a condition on a variable named `area`, and the second argument a condition on a variable named `farm_income`. Features are selected if they satisfy both arguments, that is, if their `area` is larger than `100,000` `AND` `farm_income` is less than `10 billion`. Features meeting both conditions will become part of the selected set.

Boolean expressions

1. (area > 100,000)
   AND
   (farm_income < 10 billion)

2. NOT (state = Texas)

3. [(rainfall > 1000)
   AND
   (taxes = low)]
   OR
   [(house_cost < 65,000)
   AND
   NOT (crime = high)]

Figure 9-7: Examples of Boolean expressions

Expression 2 in Figure 9-7 illustrates a Boolean `NOT` expression. This conditions specifies that all features with a variable `state` which is not equal to `Texas` will return a true value, and hence be selected. `NOT` is also often known as the negation operator. This is because we might interpret the application of a `NOT` operation as exchanging the selected set for the unselected set. The argument of expression 2 in Figure 9-7 is itself a set algebra expression. When applied to a set of features, this expression will select all features for which the variable `state` is equal to the value `Texas`. The `NOT` operation reverses this, and selects all features for which the variable `state` is not equal to `Texas`, e.g., the other 49 states of the United States.

The third expression in Figure 9-7 shows a more complex Boolean expression. This is a compound expression that specifies multiple conditions. Boolean operators `AND`, `OR`, and `NOT` are used to combine four set algebra expressions. This example shows what might be a naive attempt to select areas for retirement. Our grandparent is interested in selecting areas that have high rainfall and low taxes (a gardener on a fixed income), or low housing cost and low crime.

The spatial outcomes of specific Boolean expressions are shown in Figure 9-8. The figure shows three overlapping circular regions, labeled `A`, `B`, and `C`. Areas may fall in more than one region, e.g., the center, where all three regions overlap, is in `A`, `B`, and `C`. As shown in the figure, Boolean `AND`, `OR`, or `NOT` may be used to select any combination or portions of these regions.

`OR` conditions return a value of true if either argument is true. Areas in either region `A` or region `B` are selected at the top center of Figure 9-8. `AND` requires the conditions on both sides of the operation be met; an `AND` operation results in a reduced selection set (top right, Figure 9-8). `NOT` is the negation operator, and flips the effect of the previous operations, i.e., it turns true to false and false to true. The `NOT` shown in the lower left portion of Figure 9-8 returns the area that is only in region `C`. Note that this is the converse, or opposite set returned by the comparative `AND`, shown in the top center of Figure 9-8. The `NOT` operation is often applied in combination with the `AND` Boolean operator, as shown at the bottom center of Figure 9-8. Again, this selects the converse (or complement) of the corresponding `AND`. Compare the bottom center selection to the top right selection in Figure 9-8. `NOT`s, `AND`s, and `OR`s may be further combined to select specific combinations of areas, as shown in the lower right of Figure 9-8.

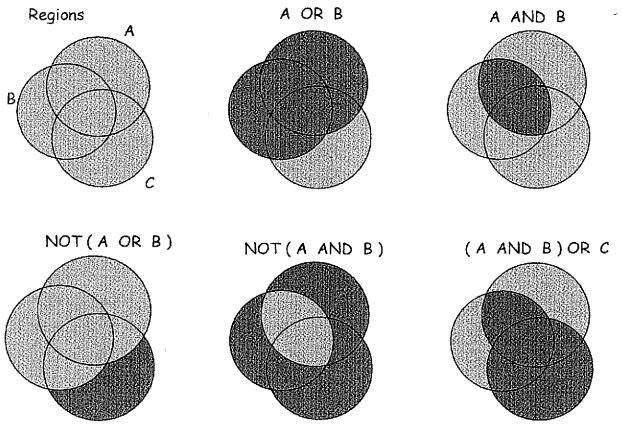


Figure 9-8: Examples of expressions in Boolean algebra, and their outcomes. Sub-areas of three regions are selected by combining AND, OR, and NOT conditions in Boolean expressions. Any sub-area or group of sub-areas may be selected by the correct Boolean combination.

Note that as with table selection discussed in Chapter 8, the order of application of these Boolean operations is important. In most cases you will not select the same set when applying the operations in a different order. Therefore, parentheses, brackets, or other delimiters should be used to specify the order of application. The expression *A AND B OR C* will give different results when interpreted as (*A AND B*) *OR C*, as shown in Figure 9-8, than when interpreted as *A AND* (*B OR C*). Verify this as an exercise. Which areas does the second Boolean expression select?

Spatial Selection Operations

Many spatial operations select sets of features. These operations are applied to a spatial data layer and return a set of features that meet a specified condition. Adjacency and containment are commonly used spatial selection operations.

Adjacency operations are used to identify those features that "touch" other features. Features are typically considered to touch when they share a boundary, as when two polygons share an edge or bounding line. A target or key set of polygon features is identified, and all features that share a boundary with the target features are placed in the selected set.

Figure 9-9a shows an example of a selection based on polygon adjacency. The state of Missouri is shaded on the left side of Figure 9-9a, and states adjacent to Missouri are shaded on the right portion of Figure 9-9a. States are selected because they include a common border with Missouri.

There are many ways the shared border may be detected. With a raster data layer an exhaustive cell-by-cell comparison may be conducted to identify adjacent pairs with different state values. Vector adjacency may be identified by observing the topological relationships (see Chapter 2 for a discussion of topology). Line and polygon topology typically records the polygon

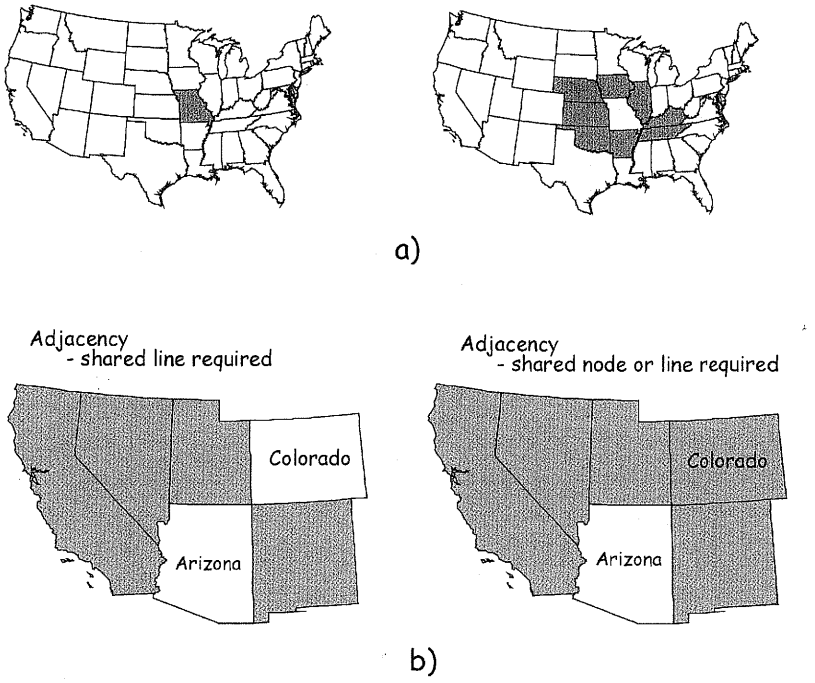


Figure 9-9: Examples of selections based on adjacency. a) Missouri, USA is shown on the left and all states adjacent to Missouri shown on the right. b) Different definitions of adjacency result in different selections. Colorado is not adjacent to Arizona when line adjacency is required (left), but is when node adjacency is accepted (right).

identifiers on each side of a line. All lines with Missouri on one side and a different state on the other side may be flagged, and the list of states adjacent to Missouri extracted.

Adjacency is defined in Figure 9-9a as sharing a boundary for some distance greater than zero. Figure 9-9b shows how a different definition of adjacency may affect selection. The left of Figure 9-9b shows the state of Arizona and a set of adjacent (shaded) western states. By the definition of adjacency used in Figure 9-9a, Arizona and Colorado are not adjacent, because they do not share a boundary along a line segment. Arizona and Colorado share a border at a point, called Four Corners, where they join with Utah and New Mexico. When a different definition of adjacency is used, with a shared node qualifying as adjacent, then Colorado is added to the selected adjacent set (right, Figure 9-9b). This is another illustration of an observation made earlier; there are often several variations of any single spatial

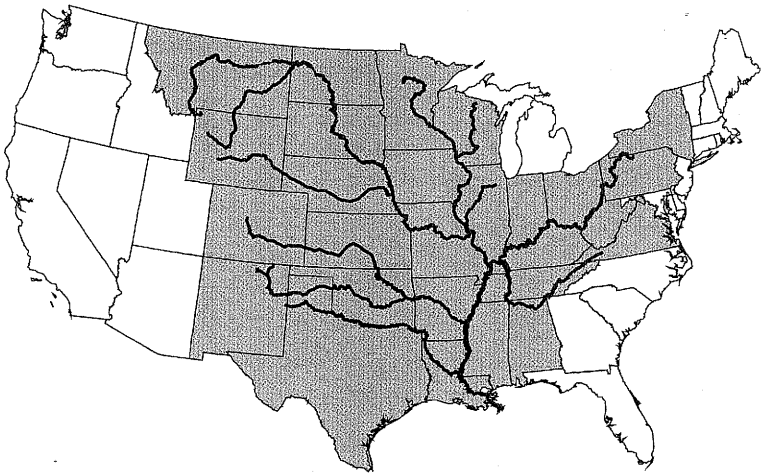


Figure 9-10: An example of a selection based on containment. All states containing a portion of the Mississippi River or its tributaries are selected.

operation. Care must be taken to test the operation under controlled conditions until the specific implementation of a spatial operation is well understood.

Containment is another spatial selection operation. Containment selection identifies all features that contain or surround a set of target features. For example, the California Department of Transportation may wish to identify all counties, cities, or other governmental bodies that contain some portion of Highway 99, because they wish to consider improving road safety. A spatial selection may be used to identify these governmental bodies.

Figure 9-10 illustrates a containment selection based on the Mississippi River in North America. We wish to identify states that contain some portion of the River and its tributaries. A query is placed, identifying the features that are contained, here the Mississippi River network, and the target features that may potentially be selected. The target set in this example consists of the lower 48 states of the United States. All states that contain a portion of the Mississippi River or its tributaries are shaded as part of the selected set.

Classification

Classification is a spatial data operation that is often used in conjunction with selection. A classification, also known as a *reclassification* or *recoding*, will categorize geographic objects based on a set of conditions. For example, all the polygons larger than one square mile may be assigned a `size` value equal to `Large`, all polygons from 0.1 to one square mile may be assigned a `size` equal to `Mid`, and all polygons smaller than 0.1 square miles may be assigned a `size` equal to `Small` (Figure 9-11). Classifications may add to or modify the attribute data for each geographic object. These modified attributes may in turn be used in further analyses, e.g., for more complex combinations in additional classification.

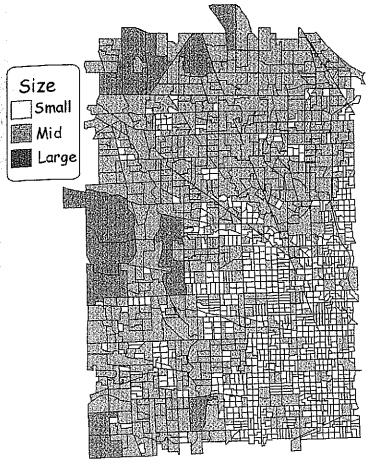


Figure 9-11: Land parcels re-classified by area.

Classification may be used for many other purposes. One common end is to group objects for display or map production. These objects have a common property, and the goal is to display them with a uniform color or symbol so the similar objects are identified as a group. The display color and/or pattern is typically assigned based upon the values of an attribute or attributes. A range of display shades may be chosen, and corresponding values for a specific attribute assigned. The map is then displayed based on this classification.

A classification may be viewed as an assignment from an existing set of classes to a new set of classes. The assignment from input attribute values to new class values may be defined manually, or the assignment may be defined automatically. For manual classifications, the class transitions are specified entirely by the human analyst.

Classifications are often specified by a table or array. The table identifies the input class, and the output class for each input class. Figure 9-12 illustrates the use of a classification table to specify class assignment. Input values of `A` or `B` lead to an output class value of `1`, an input value of `E` leads to an output value of `2`, and an input value of `I` leads to an output value of `3`. The table provides a complete specification for each classification assignment.

Figure 9-12 illustrates a classification based on a manually defined table. The table is manually defined in that each col-

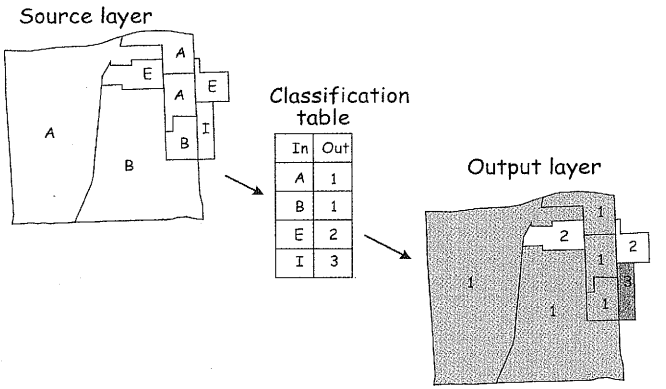


Figure 9-12: The classification of a thematic layer. Values for a specific attribute are used with a classification table to assign classes in an output layer.

umn entry is specifically assigned and entered by the human analyst. The analyst specifies the In items for the source data layer. She enters these values directly into the classification table, as well as the corresponding output value for each In variable. Out values must be specified for each input value or there will be undefined features in the output layer. Manual definition of the classification table provides the greatest control over class assignment. Alternatively, classification tables may be automatically assigned, in that a number of classes may be specified and some rule used to assign output classes to input classes.

A *binary classification* is perhaps the simplest form of classification. A binary classification places objects into two classes – 0 and 1, true and false, A and B, or some other two-level classification. A set of features is selected and assigned a value, e.g., one. The complement of the set, all remaining features in the data layer, is assigned the alternate binary value, e.g., zero.

A binary classification is often used to store the results of a complex selection operation. A large number of Boolean and set algebra expressions may be applied, resulting in a selected set. The selected set of features is assigned a unique value, and all features in the unselected set a different value. We may wish to record membership in the selected set, and so we create a new variable, and assign all selected records a value for this new variable, and all unselected records an alternate value.

For example, we may wish to select states west of the Mississippi River as an intermediate step in an analysis (Figure 9-13). We may be using this classification in many subsequent spatial operations. Thus, we wish to store this characteristic,

Classification table

state name	is_west
Alabama	0
Arizona	1
Arkansas	1
Colorado	1
Connecticut	0
....	...
Wyoming	1

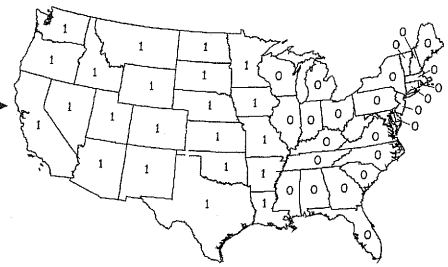


Figure 9-13: An example of a binary classification. Features are placed into two classes in a binary classification, west (1) and east (0) of the Mississippi River. The classification table codifies the assignment.

whether the state is west or east of the Mississippi River. States are selected based on location and reclassified. We record this classification by creating a new attribute and assigning a binary value to this attribute, e.g., one for those parcels that satisfy the criteria, and zero for those that do not (Figure 9-13). The variable is_west records the state location relative to the Mississippi River. Additional selection operations may be applied, and the created binary variable preserves the information generated in the initial selection.

The manual definition of the classification table may not always be necessary, and may be tedious or complex. Suppose we wish to assign a set of display colors to a set of elevation values. There may be thousands of distinct elevation values in the data layer, and it would be inconvenient at best to assign each color manually. Automatic classification methods are often used in these instances.

An automatic classification uses some rule to specify the input class to output class assignments. The input and output class boundaries are often based on a set of parameters used to guide class definition.

A potential drawback from an automated class assignment stems from our inability to precisely specify class boundaries. A mathematical formula or algorithm defines the class boundaries, and so specific classes of interest may be split. The analyst sacrifices precise control over class specification when an automated classification is used. Considerable time may be saved by automatic class assignment, but we may have to manually change some class boundaries as the only way to achieve a desired classification.

Figure 9-14 describes a data layer we will use to illustrate automatic class assignment. The figure shows a set of "neighborhoods" with populations that range from 0 to 5133. We wish to display the neighborhoods and populations in three distinct

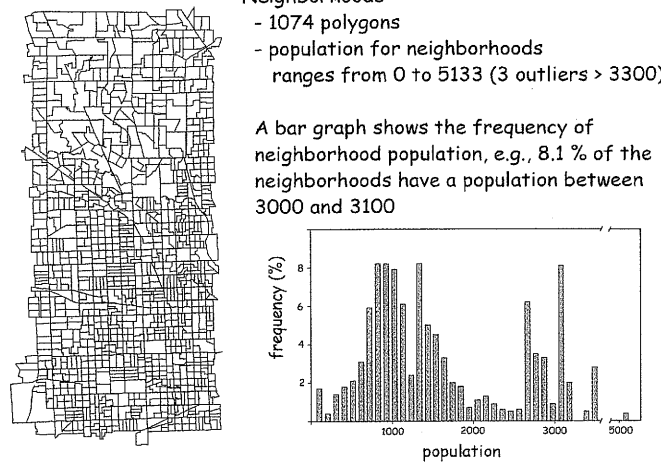


Figure 9-14: Neighborhood polygons and population levels used in subsequent examples of classification assignment. The populations for these 1074 neighborhoods ranges from 0 to 5133. The histogram at the lower right shows the frequency distribution. Note that there is a break in the chart between 3500 and 5000 to show the three "outlier" neighborhoods with populations above 3300.

classes, high, medium, and low population. High will be shown in black, medium in gray, and low in white. We must decide how to assign the categories - what population levels define high, medium, and low? In many applications the classification levels are previously defined. There may be an agreed-upon standard for high population, and we would simply use this level. However, in many instances the classes are not defined, and we must choose them.

Figure 9-14 includes a bar graph depicting the population frequency distribution, commonly called a histogram. The frequency histogram shows the number of neighborhoods that are found in each bin of a set of very narrow population categories. For example, we may count the number of neighborhoods that have a population between 3000 to 3100. If a neighborhood has 3037, 3004, 3088, or any other number between 3000 and 3100, we add one to our frequency sum for the category covering 3000 and 3100. We review all neighborhoods in our area, and plot the percentage of neighborhoods that have a population between 3000 and 3100 as a vertical bar on the histogram. Approximately 8.4 percent of the neighborhoods have a population in this range, so a vertical bar corresponding to 8.4 units high is plotted. We count and plot the historgram values for each of our narrow categories, e.g., the number from 0 to 100, from 100 to 200, from 200 to 300, until the highest population value is plotted (Figure 9-14).

We may view our class assignment problem as deciding where to place the class boundaries. Should we place the boundary between the low and medium population classes at 1000, or 1200? Where should the boundary between medium and high population classes be placed? The location of the class boundaries will change the appearance of the map, and also the resulting classification.

One common method for automatic classification specifies the number of output classes and requests equal interval classes over the range of input values. This *equal-interval* classification simply subtracts the lowest value of the classification variable from the highest value, and defines equal-width boundaries to fit the desired number of classes into the range.

Figure 9-15 illustrates an equal-interval classification for the population variable. Three classes assigned over the range of 0 to 5133 are specified. Each interval is approximately one-third of this range. This range is evenly divided by 1711. The small class extends from 0 to 1711, the medium class from 1712 to 3422, and the large class from 3423 to 5133. Population categories are shown colored accordingly on the map and the bar graph, with the small (white), medium (gray) and large (black) classes shown.

Note that the low population class shown in white dominates the map; most of the neighborhoods fall in this population class. This often happens when there are features that have values much higher than the norm. There are a few neighborhoods with populations above 5000 (to the right of the break in the population axis of the bar graph), while most neighborhoods have populations below 3000. The outliers shift the class boundaries to higher values, 1711 and 3422, resulting in most neighborhoods falling in the small population category.

Another common method for class assignment results in an *equal-area* classification (Figure 9-16). Class boundaries are defined to place an equal proportion of the study area into each of a specified number of classes. This usually leads to a visually-balanced map in that all classes have approximately equal extents. Equal-area classes are often desirable for reasons other than generating a balanced map. Resources may need to be distributed over equal areas, or equally-sized overlapping sales territories may be specified.

Note that the class width may change considerably with an equal-area classification. An equal-area classification sets class boundaries so that each class covers approximately the same area. A class may

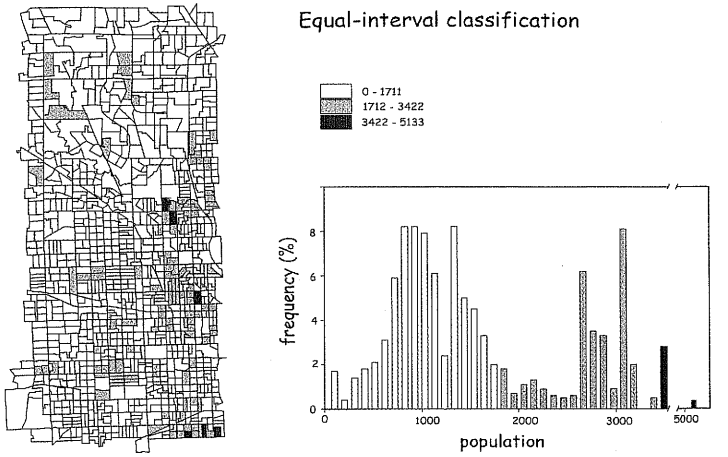


Figure 9-15: An equal interval classification. The range 0 - 5133 is split into three equal parts. Colors are assigned as shown in the map of the layer (left), and in the frequency plot (right). Note the relatively few polygons assigned to the high classes in black. A few neighborhoods with populations near 5000 shift the class boundaries upward.

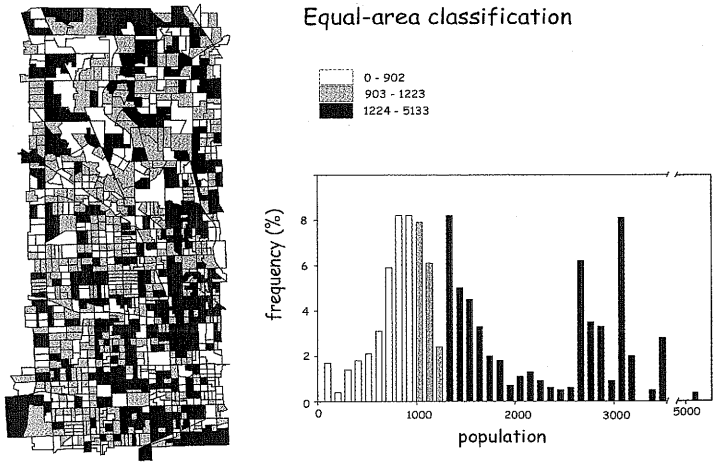


Figure 9-16: Equal-area classification. Class boundaries are set such that each class has approximately the same total area. This often leads to a smaller range when groups of frequent classes are found. In this instance the medium class spans a small range, from 903 to 1223, while the high population class spans a range that is almost 10 times broader, from 1224 to 5133.

consist of a few or even one large polygon. This results in a small range for the large-polygon classes. Classes also tend to have a narrow range of values near the peaks in the histogram. Many polygons are represented at the histogram peaks, and so these may correspond to large areas. Both of these effects are illustrated in Figure 9-16. The middle class of the equal-area classification occurs at population values between 903 and 1223. This range of populations is near the peak in the frequency histogram, and these population levels are associated with larger polygons. This middle class spans a range of approximately 300 population units, while the small and large classes span near 900 and 4000 population units, respectively.

Note that equal-area assignments may be highly skewed when there are a few polygons with large areas, and these polygons have similar values. Although not occurring in our example, there may be a relationship between the population and area for a few neighborhoods. Suppose in a data set similar to ours there is one very large neighborhood dominated by large parks. This neighborhood has both the lowest populations and largest area. An equal-area classification may place this neighborhood in its own class. If a large parcel also occurs with high population levels, we may get three classes: one parcel in the small class, one parcel in the high population class, and all the remaining parcels in the medium population class. While most equal-area classifications are not this extreme, unique parcels may strongly affect class ranges in an equal-area classification.

We will cover a final method for automated classification, a method based on *natural breaks*, or gaps, in the data. Natural breaks classification looks for "obvious" breaks. It attempts to identify naturally occurring clusters of data, not clusters based on the spatial relationships, but rather clusters based on an ordering variable.

There are various algorithms used to identify natural breaks. Large gaps in an ordered list of values are one common method. Baring gaps, low points in the frequency histogram may be identified. There is usually an effort to balance the need for relatively wide and evenly distributed classes and the search for natural gaps. Many narrow classes and one large class may not be acceptable in many instances, and there may be cases where the specified number of gaps does not occur in the data histogram. More classes may be requested than obvious gaps, so some natural break methods include an alternative method, e.g., equally-spaced intervals, for portions of the histogram where no natural gaps occur.

Figure 9-17 illustrates a natural break classification. Two breaks are evident in the histogram, one near 1300 and one between 1900 and 2600 persons per neighborhood. Small, medium, and large populations are assigned at these junctures.

Figure 9-15 through Figure 9-17 strongly illustrate an important point: you must be careful when producing or interpreting class maps, as the apparent relative importance of categories may be manipulated by altering the starting and ending values that define each class. Figure 9-15 suggests most neighborhoods are low population, Figure 9-16 that high population neighborhoods cover the largest areas and these are well mixed with areas of low and medium population, while Figure 9-17 indicates the area is dominated by low and medium population neighborhoods. Precisely because there are no objectively defined population boundaries we have great flexibility in manipulating the impression we create. The legend in class maps should be scrutinized, and the range between class boundaries noted. A histogram, as shown in these figures, and an indication of the highest and lowest data values are valuable when interpreting the legend.

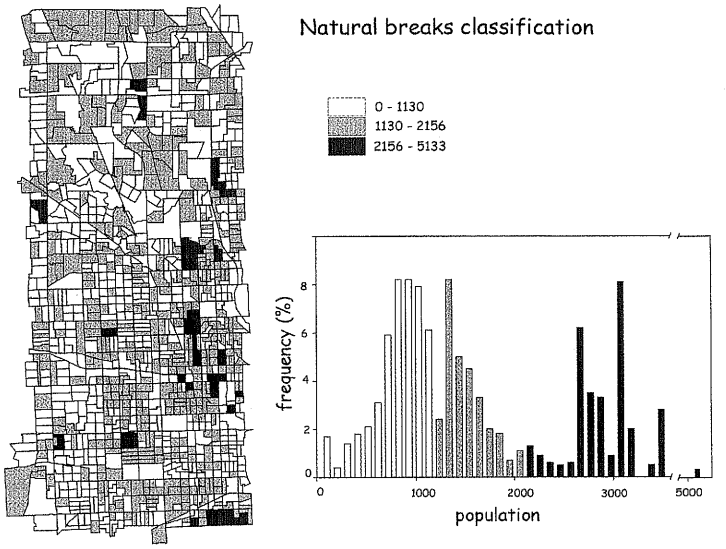


Figure 9-17: Natural breaks classification. Boundaries between classes are placed where natural gaps or low points occur in the frequency distribution.

Dissolve

A *dissolve* function has the primary purpose of combining like features within a data layer. Adjacent polygons may have identical values for an attribute. For example, a wetlands data layer may specify polygons with several sub-classes, e.g., wooded wetlands, herbaceous wetlands, or open water. If an analysis requires we identify only the wetland areas vs. the upland areas, then we may wish to dissolve all boundaries between adjacent wetlands. We are only interested in preserving the wetland/upland boundaries.

Dissolve operations are usually applied based on a specific "dissolve" attribute associated with each feature. A value or set of values is identified that belong in the same grouping. Each line that serves as a boundary between two polygon features is assessed. The values for the dissolve attribute are compared across the boundary line. If the values are the same, the boundary line is removed, or dissolved away. If the values for the dissolve attribute differ across the boundary, the boundary line is left intact.

Figure 9-18 illustrates the dissolve operation that produces a binary classification. This classification places each of the contiguous United States into one of two categories, those entirely west of the Mississippi River (1) and those east of the Mississippi River (0). The attribute named `is_west` contains values indicating location. A dissolve operation applied on the

variable is_west removes all state boundaries between similar states. This has the result of reducing the set from 48 polygons to two polygons.

Dissolve operations are often needed prior to applying an area-based selection in spatial analysis. For example, we may wish to select areas from the natural breaks classification shown on the left of Figure 9-19. We seek polygons that are greater than three square miles in area and have a medium population. The polygons may be composed of multiple neighborhoods. We typically must dissolve the boundaries between adjacent, medium-sized neighborhoods prior to applying the size test. Otherwise two adjacent, medium population neighborhoods may be discarded because both cover approximately two square miles. Their total area is four square miles, above the specified threshold, yet they will not be selected unless a dissolve is applied first.

Dissolves are also helpful in removing unneeded information. After the classification into small, medium, and large size classes, many boundaries may become redundant. Unneeded boundaries may inflate storage and slow processing. A dissolve has the primary advantage of removing unneeded geographic and tabular data, thereby improving processing speed and reducing data volumes and complexity.

Figure 9-19 illustrates a dissolve of the natural breaks classification described previously. Each line that separates two polygons is inspected. If the same size class occurs on both sides of the line, the line is removed. Otherwise, the line is retained. New spatial and attribute data must be generated for the layer once all appropriate lines have been dissolved. This example illustrates the space saving and complexity reduction common when applying a dissolve function. The number of polygons is reduced approximately nine-fold by the dissolve, from 1074 on the left to 119 polygons on the right of Figure 9-19.

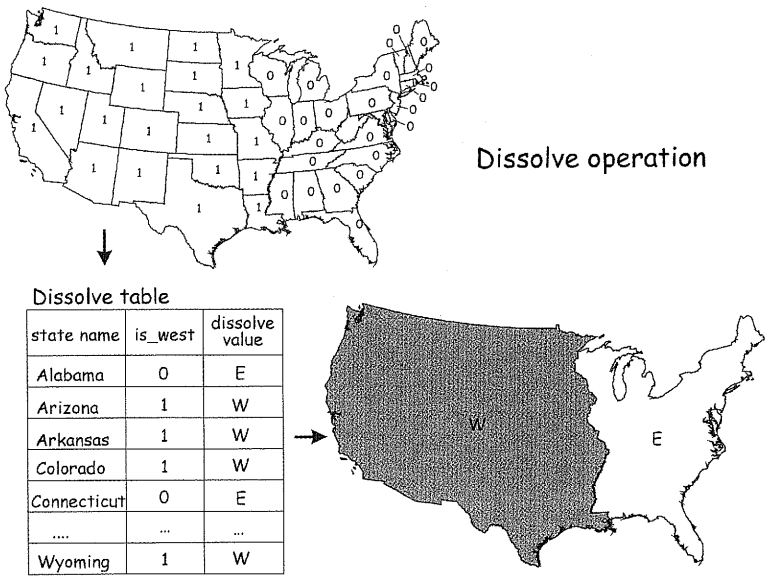


Figure 9-18: An illustration of a dissolve operation. Boundaries are removed when they separate states with the same value for the dissolve attribute is_west.

Figure 9-19: An example of a dissolve operation. Note the removal of lines separating polygons of the same size class. This greatly reduces the number of polygons.

Proximity Functions and Buffering

Proximity functions or *operations* are among the most powerful and common spatial analysis tools. Many important questions hinge on proximity, the distance between features of interest. How close are schools to an oil refinery, what neighborhoods are far from convenience stores, and which homes will be affected by an increase in freeway noise? These and other questions regarding proximity are often answered through analyses in a GIS.

Proximity functions modify existing features or create new features that depend in some way on the distance from existing features. For example, one simple proximity function creates a raster of the minimum distance to a set of features (Figure 9-20). The figure shows a distance function applied to water holes in a wildlife reserve. Water is a crucial resource for many animals, and the reserve managers may wish to ensure that most of the area is within a short distance of water. In this instance

point features are entered. The point features represent the location of permanent water. Water holes are represented by individual points, and rivers by a group of points set along the river course. The distance function calculates the distance to all water points for each raster cell. The minimum distance is selected and placed in an output raster data layer. The distance layer is illustrated in Figure 9-20 along with water locations. The short distances are shown in white and longest distances in black. The distance function creates a mosaic of what appear to be overlapping circles. Although the shading scheme shows apparently abrupt transitions, the raster cells contain a smooth gradient in distance away from each water feature.

Distance values are calculated based on the Pythagorean formula (Figure 9-21). These values are typically calculated from cell center to cell center when applied to a raster data set. Although any distance is possible, the distances between adjacent cells change in discrete intervals related to the cell size. Note that distances are not restricted to even multiples of the cell size, because distances measured on diagonal angles are not even multiples of the cell dimension. There may be no cells that are exactly some fixed distance away from the target features, however there may be many cells less than or greater than that fixed distance.

Buffers

Buffering is one of the most commonly used proximity functions. A *buffer* is a region that is less than or equal to a specified distance from one or more features (Figure 9-22). Buffers may be determined for point, line, or area features, and for raster or vector data. Buffering is the process of creating buffers. Buffers typically identify areas that are "outside" some given threshold distance vs. those "inside" some threshold distance.

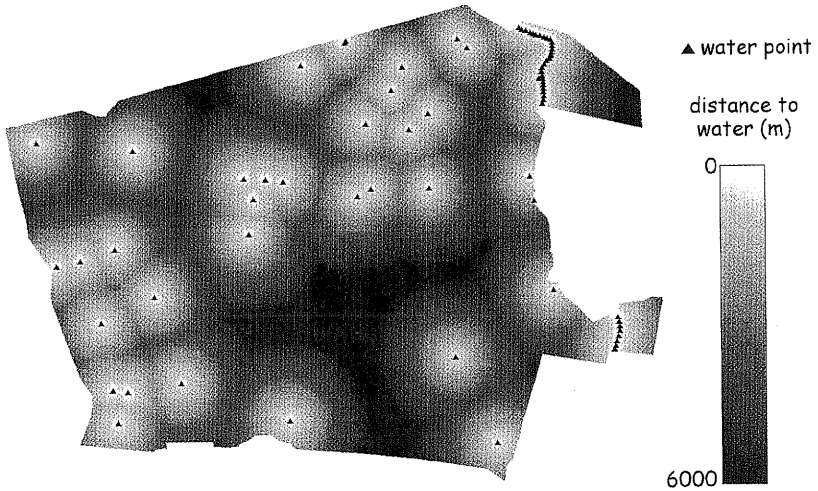


Figure 9-20: An example of a distance function. This distance function is applied to a point data layer and creates a raster data layer. The raster layer contains the distance to the nearest water feature.

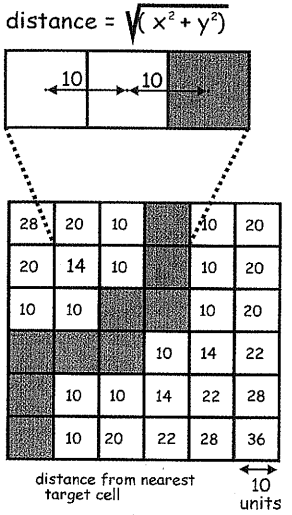


Figure 9-21: A distance function applied to a raster data set.

Buffers are used often because many spatial analyses are concerned with distance constraints. For example, emergency planners might wish to know which schools are within 1.5 kilometers of an earthquake fault, a park planner may wish to identify all lands more than 10 kilometers from the nearest highway, or a business owner may wish to identify all potential customers within a given radius of her store. All these questions may be answered with the appropriate use of buffering.

Raster Buffers

Buffer operations on raster data entail calculating the distance from each source cell center to all other cell centers. Output cells are assigned an in value whenever the cell-to-cell distance is less than the specified buffer distance. Those cells that are further than the buffer distance are assigned an out value (Figure 9-23).

Raster buffers may be viewed as a combination of a minimum distance function and a binary classification function. A minimum distance function calculates the

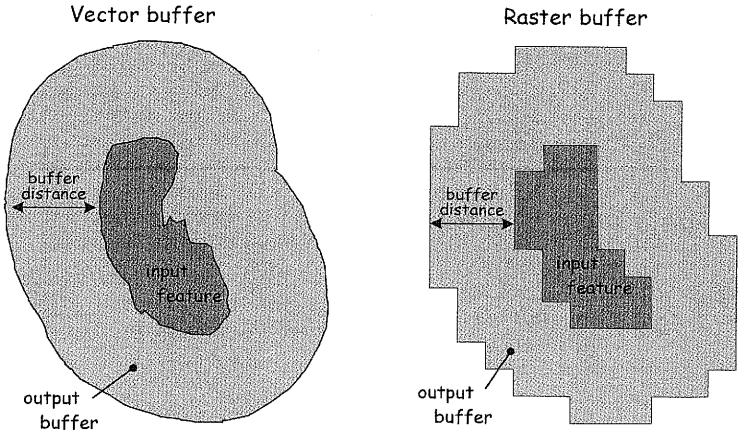


Figure 9-22: Examples of vector and raster buffers derived from polygonal features. A buffer is defined by those areas that are within some buffer distance from the input features.

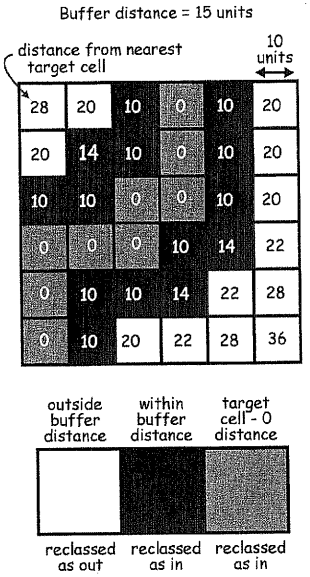


Figure 9-23: Raster buffering as a combination of distance and classification.

shortest distance from a set of target features and stores this distance in a raster data layer. The binary classification function splits the raster cells into two classes: those with a distance greater than the threshold value, and those with a distance less than or equal to a threshold value.

Buffering with raster data may produce a "stair-step" boundary, because the distance from features is measured between cell centers. When the buffer distance runs parallel and near a set of cell boundaries, the buffer boundary may "jump" from one row of cells to the next (Figure 9-23). This phenomenon is most often a problem when the raster cell size is large relative to the buffer distance. A buffer distance of 100 meters may be approximated when applied to a raster with a cell size of 30 meters. A smaller cell size relative to the buffer distance results in less obvious "stair-stepping". The cell size should be small relative to the spatial accuracy of the data, and small relative to the buffer distance. If this rule is followed, then stairs-stepping should not be a problem, because buffer sizes should be many times greater than the uncertainty inherent in the data. Buffer distances on the same order of magnitude as the spatial accuracy should be avoided.

Vector Buffers

Vector buffering may be applied to point, line, or area features, but regardless of input, buffering always produces an output set of area features (Figure 9-24). There are many variations in vector buffering. *Fixed distance buffering*, the simplest and most common form of vector buffering delineates areas a fixed distance from the input features (Figure 9-24). Fixed distance, or simple buffering separates the areas farther from any input features than the specified buffer distance from those regions that are closer than the buffer distance from an input feature. Simple buffering does not distinguish between regions that are close to one, two, three, or more features. A location is either near any feature, or farther away. Simple buffering also uses a uniform buffer distance for all features. A buffer distance of 100 meters specified for a roads layer may be applied to every road in the layer, irrespective of road size, shape, or location. In a similar manner, buffer distances for all points in a point layer will be uniform, and buffer distances for all area features will be fixed.

Buffering on vector point data is based on the creation of circles around each point feature in a data set.

The equation for a circle with an origin at x=0, y=0 is:

$$r = \sqrt{x^2 + y^2} \qquad (9.1)$$

where r is the buffer distance. The more general equation of a circle with a center at x_1, y_1, is:

$$r = \sqrt{(x - x_1)^2 + (y - y_1)^2} \qquad (9.2)$$

Equation (9.2) reduces to equation (9.1) at the origin, where $x_1 = 0$, and $y_1 = 0$. The general equation creates a circle centered on the coordinates x_1, y_1, with a buffer distance equal to the radius, r. Point buffers are created by applying this equation of a circle successively to each point feature in a data layer. The x and y coordinate locations of each point feature are used for x_1 and y_1, placing the point feature at the center of a circle (Figure 9-25). This generates a set of circles, one for each point feature.

Circles may overlap. In simple buffering, the circle boundaries that occur in overlap areas are removed. For example, areas within 10 kilometers of hazardous waste sites may be identified with the creation of a buffer layer. We may have a data layer in which hazardous waste sites are represented as points (Figure 9-26a). A circle with a 10 kilometer radius is drawn around each point. When two or more circles overlap, internal boundaries are dissolved, resulting in non-circular polygons (Figure 9-26b).

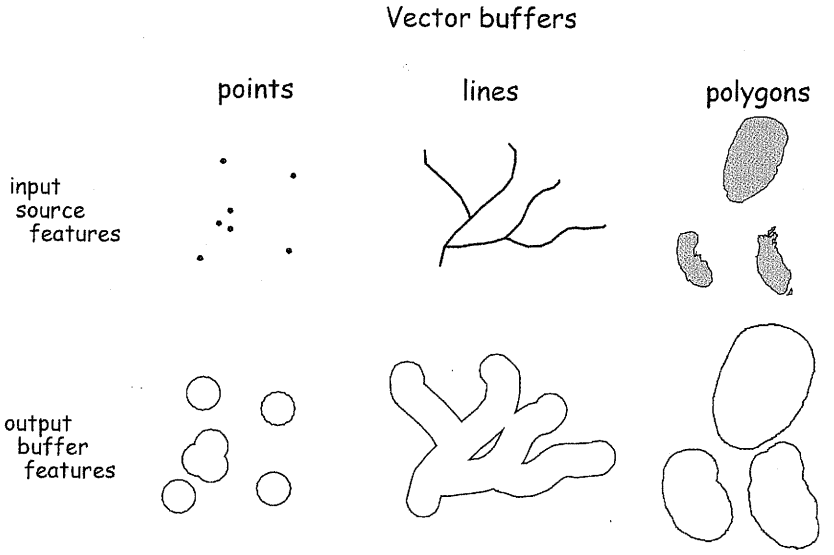


Figure 9-24: Vector buffers produced from point, line, or polygon input features. In all cases the output is a set of polygon features.

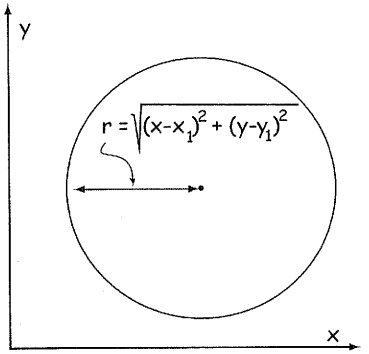


Figure 9-25: The formation of a point buffer boundary.

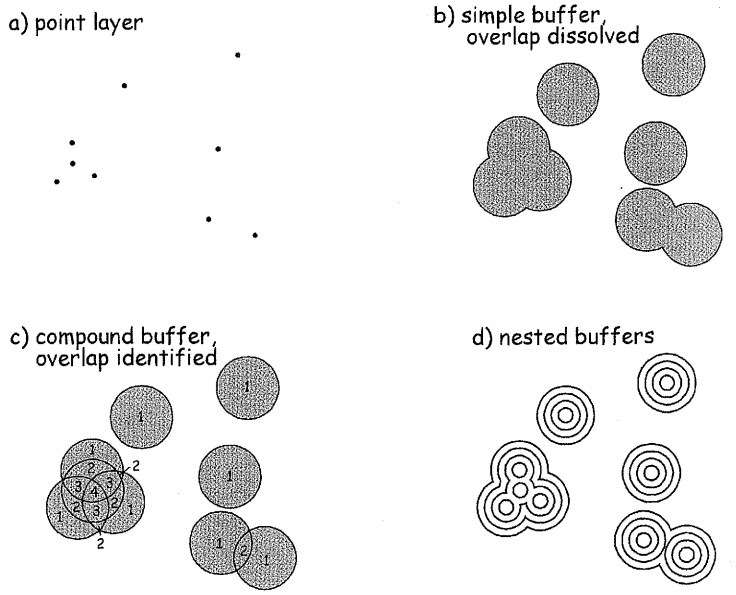


Figure 9-26: Various types of point buffers. Simple buffers dissolve areas near multiple features, more complex buffers do not. Multi-ring buffers provide distance-defined zones around each feature.

More complex buffering may be applied. These may identify buffer areas by the number of features within the given buffer distance, or apply variable buffer distances depending on the characteristics of the input features. We may be interested in areas that are near multiple hazardous waste sites. These zones may entail added risk and therefore require special monitoring or treatment. This in turn mandates the identification of all areas within the buffer distance of a hazardous waste site, and the number of sites. In most applications the majority of areas will be close to one site, but some will be close to two, three, or more sites. The simple buffer, described above, will not provide the required information. This simple buffering discards the boundaries specified by overlapping buffers.

A buffering variant, referred to here as *compound buffering*, provides the needed information. Compound buffers maintain all overlapping boundaries (Figure 9-26c). All circles defined by the fixed-radius buffer distance are generated. These circles are then intersected to form a planar graph. An attribute is created for each area that records the number of features within the specified buffer distance.

Nested (or multi-ring) buffering is another common buffering variant (Figure 9-26d). We may require buffers at multiple distances. In our hazardous waste site example, suppose threshold levels have been established with various actions required for each threshold. Areas very close to hazardous waste sites require evacuation, intermediate distance require remediation, and areas further away require monitoring. These zones may be defined by nested buffers.

Buffering on vector line data is also quite common. The formation of line buffers may be envisioned as a sequence of steps. First, circles are created that are centered at each node or vertex (Figure 9-27). Tangent lines are then generated. These lines are parallel to the input feature lines

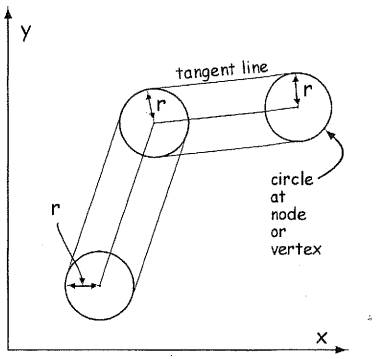


Figure 9-27: The creation of a line buffer at a fixed distance r.

and tangent to the circles that are centered at each node or vertex. The tangent lines and circles are joined and interior circle segments dissolved.

The location of the tangent lines and their intersections with the circles are based on a complicated set of rules and algebra. These operations identify the buffer segments that define the boundaries between in and out areas. These segments are saved to create a buffer for an individual line. Buffers for separate lines may overlap. With simple line buffering the internal boundaries in the overlap areas are dissolved.

Note that three different types of areas may be found when creating simple line buffers (Figure 9-28). The first type of area is within the buffer distance of the line features. An example of this area is labeled inside buffer in Figure 9-28. The second type of area is completely outside the buffer. This area is labeled outside buffer in Figure 9-28. The third type of area is labeled enclosed area in Figure 9-28. This type of area is further than the buffer distance from the input line data, but completely enclosed within a surrounding

buffer polygon. These enclosed areas occur occasionally when buffering points and polygons, but enclosed areas are most frequent when buffering line features.

Area buffers are developed in a manner similar to line buffers. Each polygon is defined by a set of lines. Circles are calculated for each vertex and node, and tangent lines placed and intersected. A bounding polygon is defined by these geometric figures, and assembled for each polygon. Polygon buffering may be simple, in which case overlapping lines between separate polygon buffers are dissolved. Area buffers may also be more complex, maintaining location and number of overlaps.

Variable-distance buffers are another common variant of vector buffering. As indicated by the name, the buffer distance is variable, and may change among features. The buffer distance may increase in steps, for example, we may have one buffer distance for a given set of features, and a different buffer distance for the remaining features. In contrast, the buffer distance may vary smoothly, for example, the buffer distance around a city may be a function of the population density in the city.

There are many instances for which we may require a variable-distance buffer. Public safety requires different zones of protection that are dependant on the magnitude of the hazard. We may wish to specify a larger buffer zone around large fuel storage facilities when compared to smaller fuel storage facilities. We often require more stringent protections further away from large rivers compared to small rivers, and give large landfills a wider bearth than small landfills.

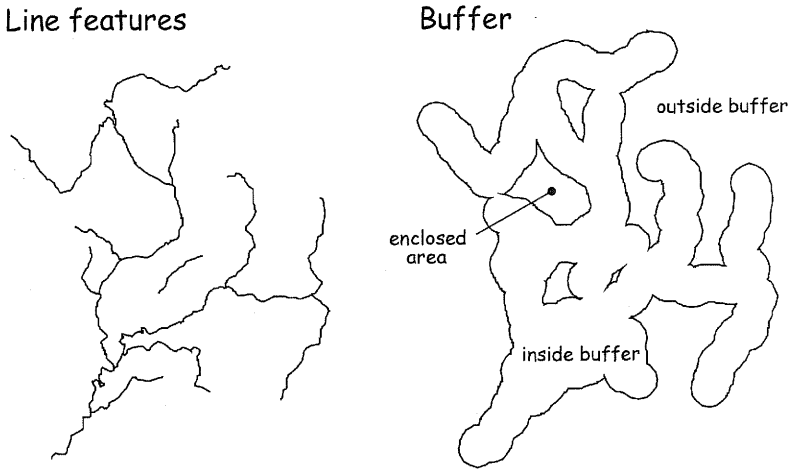
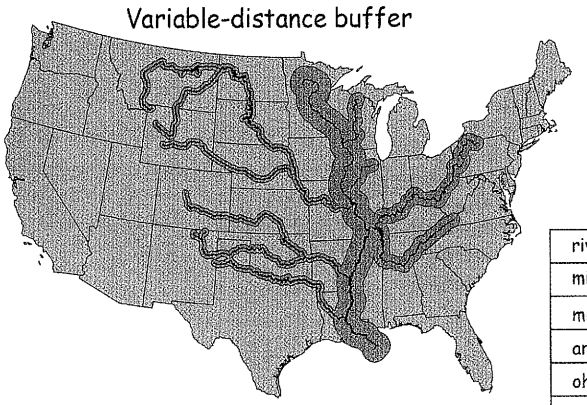


Figure 9-28: Simple buffers that are based on line features may result in three types of areas. Areas are defined inside the buffer distance. Those areas outside the buffer area may be either completely outside the buffer distance, or outside the buffer distance, but enclosed within surrounding buffer areas.



river_identifier	buffdist
mississippi	100
missouri	50
arkansas	50
ohio	75
tennessee	75
st. croix	75
illinois	75
wisconsin	75

Figure 9-29: An illustration of a variable-distance buffer. A line buffer is shown with a variable buffer distance based on a river_identifier. A variable buffer distance, buffdist, is specified in a table and applied for each river segment.

Figure 9-29 illustrates the creation of buffers around a river network. These buffers may be used to analyze or restrict land use near rivers. We may wish to increase the buffer distance for larger rivers. The increase in distance may be motivated by an increased likelihood of flooding downstream, or an increased sensitivity to pollution, or a higher chance of bank erosion as river size increases. We may specify a buffer distance of 50 kilometers for small rivers, 75 kilometers for intermediate size rivers, and 100 kilometers for large rivers. There are many other instances when variable distance buffers are required, e.g., larger distances from noisier roads, smaller areas where travel is difficult, or bigger buffers around larger landfills.

The variable buffer distance is often specified by an attribute in the input data layer. This is illustrated in Figure 9-29. A portion of the attribute table for the river data layer is shown. The attribute table contains the river name in river_identifier and the buffer distance stored in buffdist. The attribute buffdist is accessed during buffer creation, and the size of the buffer adjusted automatically for each line segment. Note how the buffer size depends on the value in buffdist.

Overlay

Overlay operations are powerful spatial analysis tools, and were an important driving force behind the development of GIS technologies. Overlays involve combining spatial and attribute data from two or more spatial data layers, and are among the most common and powerful spatial data operations (Figure 9-30). Many problems require the overlay of thematically different data. For example, we may wish to know where there are inexpensive houses in good school districts, where whale feeding grounds overlap with proposed oil drilling areas, or the location of farm fields that are on highly erodible soils. In the later example a soils data layer may be used to identify highly erodible soils, and a current land use layer used to identify the locations of farm fields. The boundaries of erodible soils will not coincide with the boundaries of the farm fields in most instances, so these soils and land use data must somehow be combined. Overlay is the primary means of providing this combination.

An overlay operation requires that data layers use a common coordinate system. Overlay uses the coordinates that define each spatial feature to combine the data from the input data layers. The coordinates for any point on Earth depend on the coordinate system used (Chapter 3). If the coordinate systems used in the various layers are not exactly the same, the features in the data layers will not align correctly.

Overlay may be viewed as the vertical stacking and merger of spatial data (Figure 9-30). Features in each data layer are set one "on top" another, and the points, lines, and area feature boundaries merged into a single data layer. The attribute data are also combined, so that the new data layer includes information contained in each input data layer.

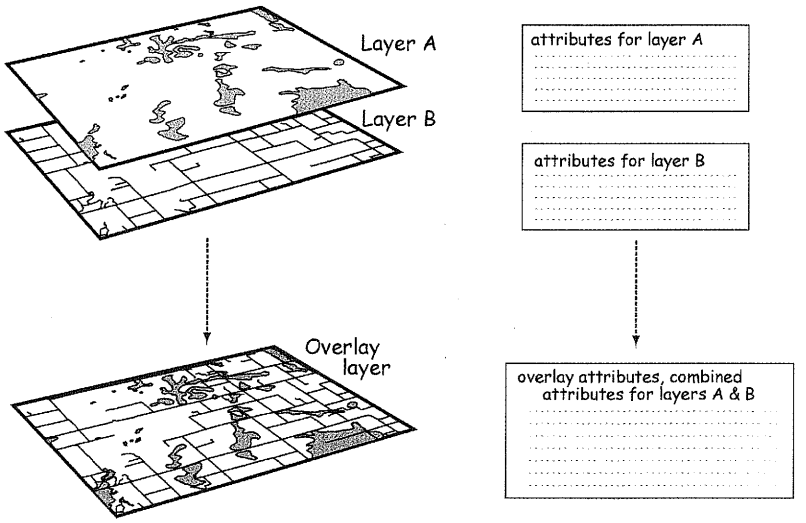


Figure 9-30: Spatial data overlay. Overlay combines both the coordinate information and the attribute information across different data layers.

Raster Overlay

Raster overlay involves the cell-by-cell combination of two or more data layers. Data from one layer in one cell location correspond to a cell in another data layer. The cell values are combined in some manner and an output value assigned to a corresponding cell in an output layer.

Raster overlay is typically applied to nominal or ordinal data. A number or character stored in each raster cell represents a nominal or ordinal category. Each cell value corresponds to a category for a raster variable. This is illustrated in the input data sets shown at the left and center of Figure 9-31. Input Layer A represents soils data. Each raster cell value corresponds to a specific soil value. In a similar manner input Layer B records land use, with values 1, 2, and 3 corresponding to particular land uses. These data may be combined to create areas with combinations of the two input layers, cells with values for both soil type and land use.

There are as many potential categories as there are possible combinations of input layer values. In Figure 9-31 there are two soil types in Layer A, and three land use types in Layer B. There are potentially six different combinations in the output layer. Not all combinations will necessarily occur in the overlay, as shown in Figure 9-31. In this example only four of the six overlay combinations occur. Unique identifiers must be generated for each observed combination, and placed in the appropriate cell of the output raster layer.

The number of possible combinations is important to note because it may change the number of binary digits or bytes required to represent the output raster data layer. Raster cells typically hold a number or character, and may be one-byte integer, two-byte integer, or some other size. Raster data sets typically use the smallest required data size. As discussed in Chapter 2, one unsigned byte may store up to 256 different

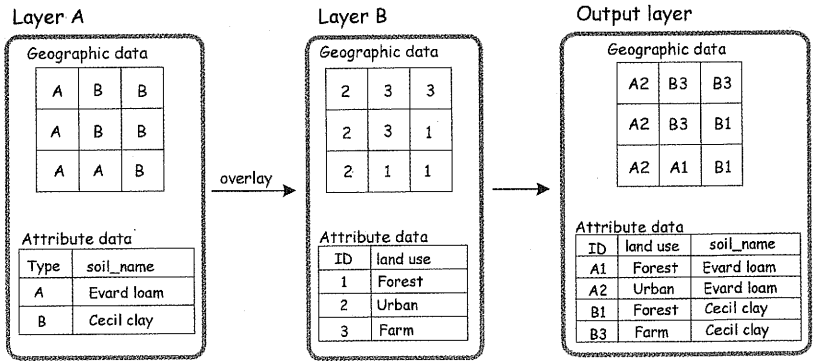


Figure 9-31: Cell-by-cell combination in raster overlay. Two input layers are combined in raster overlay. Nominal variables for corresponding cells are joined, creating a new output layer. In this example a soils layer (Layer A) is combined with a land use layer (Layer B) to create a composite Output layer.

values. Raster overlay may result in an output data layer that requires a higher number of bytes per cell. Consider the overlay between two raster data layers, one layer which contains 20 different nominal classes, and a second layer with 27 different nominal classes. There is a total of 20 times 27, or 540 possible output combinations. If more than 256 combinations occur the output data will require more than one byte for each cell. Typically two bytes will be used. This causes a doubling in the output file size. Two bytes will hold more than 65,500 unique combinations; if more categories are required then four bytes per cell are often used.

Raster overlay requires the input raster systems be compatible. This typically means they should have the same cell dimension and coordinate system, including the same origin for x and y coordinates. If the cell sizes differ, there will likely be cells in one layer that match parts of several cells in the second input layer (Figure 9-32). This may result in ambiguity when defining the input attribute value to use.

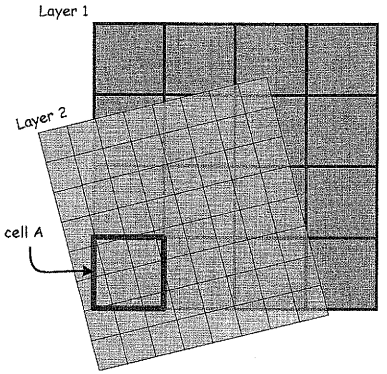


Figure 9-32: Overlain raster layers should be compatible to ensure unambiguous overlay. Cell orientation should be coincident and cell size should be compatible. In the overlay depicted here it is not clear which cells from Layer 2 should be combined with cell A in Layer 1.

Overlay may work if the cells are integer multiples with the same origin, e.g., the boundaries of a 1 by 1 meter raster layer may be set to coincide with a 3 by 3 raster layer, however this rarely happens. Data are normally converted to compatible raster layers before overlay. This is most often done using a coordinate transformation and/or resampling, as described in Chapter 4. In our example, we might choose to resample Layer 2 to match Layer 1 in cell size and orientation. Values for cells in Layer 2 would be combined through a nearest neighbor, bilinear interpolation, cubic convolution, or some other resampling formula to create a new layer based on Layer 2 but compatible with Layer 1.

Vector Overlay

Overlay when using a vector data model involves combining the point, line, and polygon geometry and associated attribute data. This overlay creates new geometry. Overlay involves the merger of both the coordinate and attribute data from two vector layers into a new data layer. The coordinate merger may require the intersection and splitting of lines or areas and the creation of new features.

Figure 9-33 illustrates the overlay of two vector polygon data layers. This overlay requires the intersection of polygon boundaries to create new polygons. The overlay also entails the combination of attribute data during polygon overlay. The data layer on the left is comprised of two polygons. There are only two attributes for Layer 1, one an identifier (ID), and the other specifying values for a variable named class. The second input data layer, Layer 2, also contains two polygons, and two attributes, ID and cost. Note that the two tables have an attribute with the same name, ID. These two ID attributes serve the same function in their respective data layers, but they are not related. A value of 1 for the ID attribute in Layer 1 has nothing to do with the ID value of 1 in Layer 2.

Vector overlay of these two polygon data layers results in four new polygons. Each new polygon contains the attribute information from the corresponding area in the input data layers. For example, note that the polygon in the output data layer with the ID of 1 has a class attribute with a value of 0 and a cost attribute with a value of 10. These values come from the values found in the corresponding input layers. The boundary for the polygon with an ID value of 1 in the output data layer is a composite of the boundaries found in the two input data layers. The same holds true for the other three polygons in the output data layer. These polygons are a composite of geographic and attribute data in the input data layers.

The topology of vector overlay output will likely be different from that of the input data layers. Vector overlay functions typically identify line intersection points during overlay. Intersecting lines are split and a node placed at the intersection point. Thus topology must be re-created if it is needed in further processing.

Any type of vector feature may be overlain with any other type of vector feature, although some overlay operations rarely provide useful information and are performed infrequently. Point-in-polygon overlays are quite common, point in line much less so, and point-on-point nonsensical in most cases. In theory points may be overlain on point, line, or polygon feature layers, lines on all three types, and polygons on all three types. Point-on-point or point-on-line overlay rarely results in intersecting features, and so are rarely applied. Line-on-line overlay are sometimes required, for example when we wish to identify the intersections of two networks such as road and railroads. Overlays involving polygons are the most common by far.

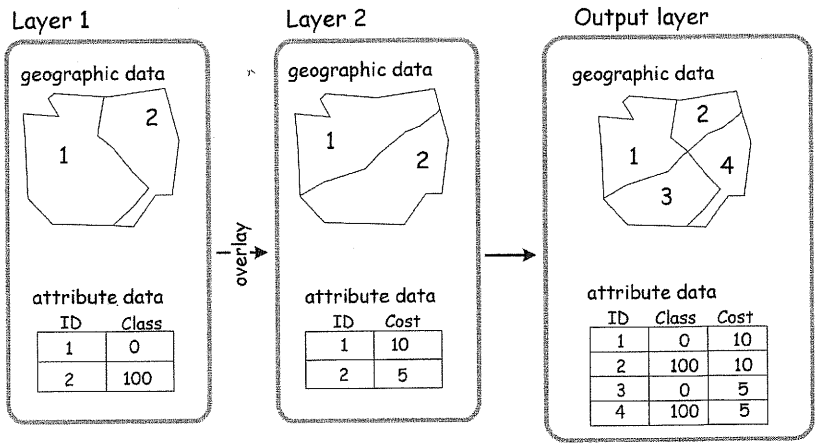


Figure 9-33: An example of vector polygon overlay. In this example output data contain a combination of the geographic (coordinate) data and the attribute data of the input data layers. New features may be created with topological relationships distinct from those found in the input data layers.

Overlay output typically takes the lowest dimension of the inputs. This means point-in-polygon overlay results in point output, and line-in-polygon overlay results in line output. This avoids problems when multiple lower-dimension features intersect with higher-dimension features.

Figure 9-34 illustrates an instance where multiple points in one layer fall within a single polygon in an overlay layer. Output attribute data for a feature are a combination of the input data attributes. If polygons are output (Figure 9-34, right, top) there is ambiguity regarding which point attribute data to record. Each point feature has a value for an attribute named class. It is not clear which value should be recorded in the output polygon, the class value from point A, point B, or point C. When a point layer is output (Figure 9-34, right, bottom), there is no ambiguity. Each output point feature contains the original point attribute information, plus the input polygon feature attributes.

One method for creating polygon output from point-in-polygon overlay involves recording the attributes for one point selected arbitrarily from the points that fall within a polygon. This is usually not satisfactory because important information may

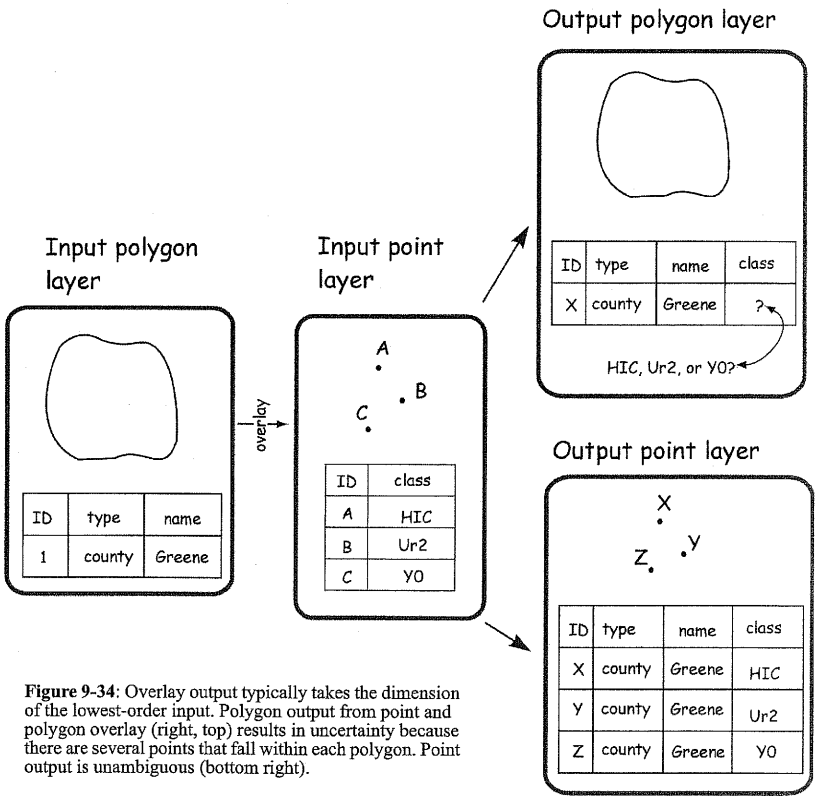


Figure 9-34: Overlay output typically takes the dimension of the lowest-order input. Polygon output from point and polygon overlay (right, top) results in uncertainty because there are several points that fall within each polygon. Point output is unambiguous (bottom right).

be lost. An alternative solution involves adding columns to the output polygon to preserve multiple points per polygon. However this would still result in some ambiguity, e.g., what should be the order of duplicate attributes? It may also add a substantial number of sparsely used items, thus increasing file size inefficiently. Forcing the lower order output during overlay avoids these problems, as shown in the lower right of Figure 9-34.

Note that the number of attributes in the output layer increases after each overlay. This is illustrated in Figure 9-34, with the combination of a point and polygon layer in an overlay. The output point attribute table shown in the lower right portion of the figure contains four items. This output attribute table is a composite of the input attribute tables.

Large attribute tables may result if overlay operations are used to combine many data layers. When the output from an overlay process is in turn used as an input for a subsequent overlay, the number of attributes in the next output layer will usually increase. In rare instances the number of attributes in an output layer will be the same as the larger of the two input layers. As the number of attributes grows, tables may become unwieldy, and there may be a need to delete redundant or superfluous attributes. Processes that require the overlay of many layers often include the removal of unneeded attributes.

Figure 9-35 illustrates the most common types of vector overlay. Each row in the figure represents an overlay operation with point, line or polygon input. The input data layers are arranged in the left and middle columns and the output data layer in the right-hand column. Rather than show the complete attribute tables, labels are used to represent the combination of attributes. The bottom row illustrates this for polygon overlay. The two polygons in the first input data layer are labeled 100 and 200. The two polygons in the second input layer (middle panel) are labeled R and S. The resultant composite polygons in the output layer are shown on the left with labels 100R, 100S, 200R, and 200S to represent the combination of attributes from both input data layers.

Common characteristics of various types of overlay are apparent in Figure 9-35. Point-on-line overlay is shown in the top row. This results in joining line data for only one point, 2, which in the overlay layer is labeled 2B. The other points do not intersect a line, as indicated by the minus in the labels denoting attributes, e.g., 3-. This is common with point-in-line overlay. Points have no dimension, and lines have zero width. The likelihood of points and lines intersecting are quite low for most data sets. Only in special circumstances such as when point data are restricted to fall on a linear network (e.g., points representing accident locations) are there likely to be frequent point and line intersections.

Point features result from point-in-polygon overlay, as shown in the second row of Figure 9-35. Points take the attributes of the coincident polygon. Point location is not changed nor are any geographic data from the polygon features typically incorporated into output point features.

Vector line-on-line overlay results in a line data layer (middle row, Figure 9-35). A planar graph is produced, meaning nodes (open circles) are placed at each line intersection. Each respective line segment maintains the attributes from the source data layer. Node attribute tables, if they exist, may contain information that originated from lines in each of the input data layers.

Line-on-polygon overlay is shown in the fourth row of Figure 9-35. This type of overlay typically produces a vector line output layer. Each line in the output data layer contains attributes from both the original input line data layer and the coincident polygon attribute layer. Line segments are split where they cross a polygon boundary, e.g., the line segment labeled 10 in Input layer 1 is split into two segments in the

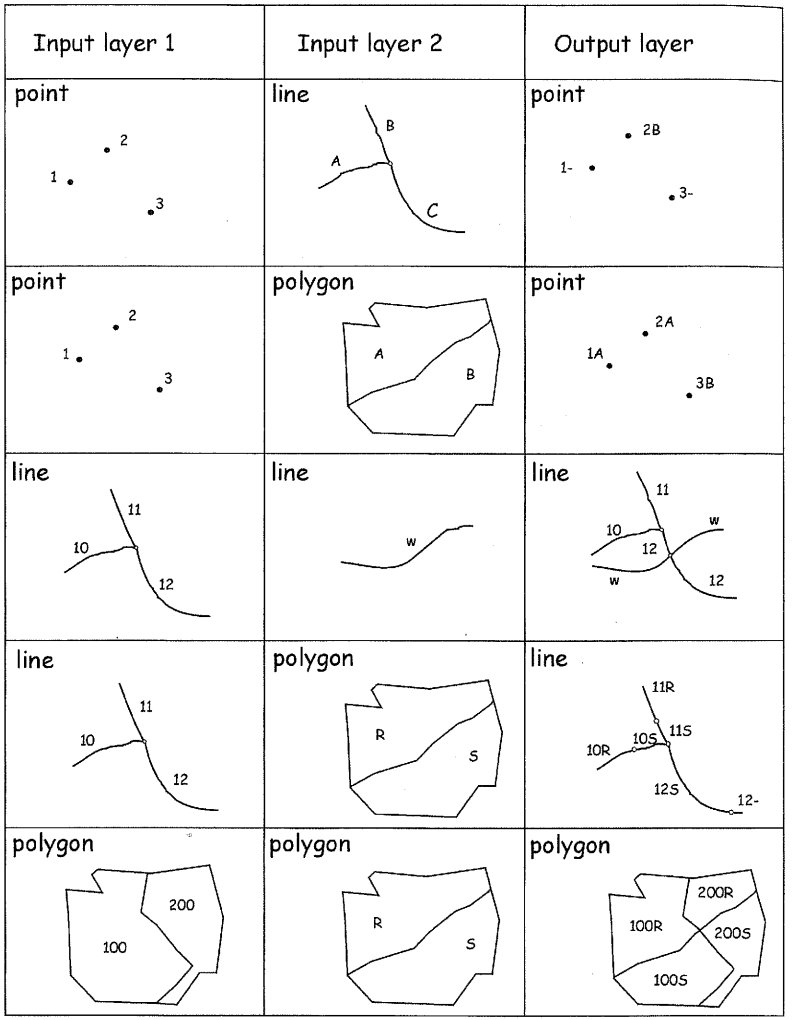


Figure 9-35: Examples of vector overlay. In this example, point, line, or polygon layers are combined in an inclusive overlay. The combination results in an output layer that contains the attribute and geographic data from input layers. The output data are typically the minimum order of the input, e.g., if point data are used as input, the output will be point. If line and line or line and polygon data are used, the output will be a line data layer.

Output layer. Each segment of this line exhibits a different set of attributes: 10R and 10S. Note that not all line segments may contain a complete set of attributes derived from polygons. Line segments falling outside all polygons contain a null value for polygon attributes, such as the segment at the lower right of line-on-line output panel. This segment is labeled 12-. The minus (-) denotes that polygon attributes are not recorded. These "outside" line segments typically contain null or flag values in the attribute table for the polygon items.

A polygon-on-polygon overlay is shown in the bottom row of Figure 9-35. The combination of polygon features has been discussed previously, and will not be described further here.

Overlays that include a polygon layer are most common. We are often interested in combination of polygon features with other polygons, or in finding the coincidence of point or line features with polygons. What counties include hazardous waste sites? Which neighborhoods does one pass through on E Street? Where are there shallow aquifers below cornfields? All these examples involve the overlay of area features, either with other area features, or with point or line features.

Clip, Intersect, and Union: Special Cases of Overlay

There are three common ways overlay operations are applied: as a *clip*, an *intersection*, or a *union* (Figure 9-36). The basic layer-on-layer combination is the same for

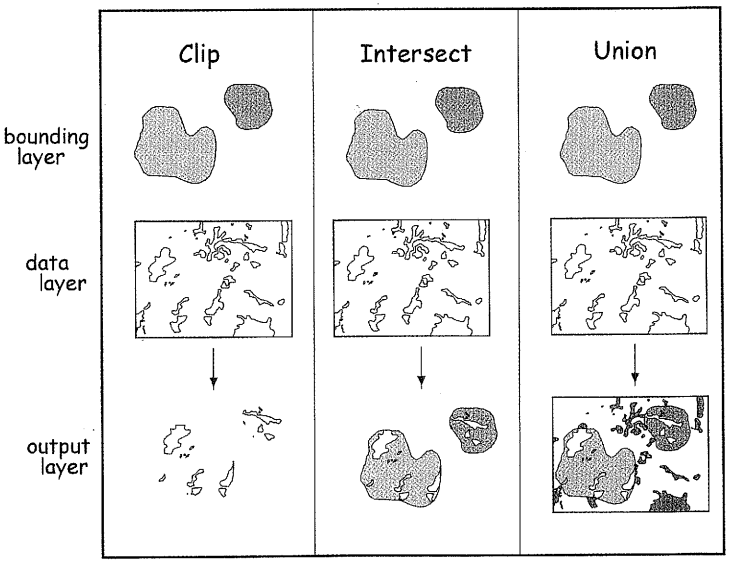


Figure 9-36: Common variations of overlay operations. Clip, intersect, and union operations are used to combine data from separate layers. The clip preserves information only from the data layer and only for the area of the bounding layer. An intersect is restricted to the area of the bounding layer, but combines data from both input layers. The union combines data from both input layers over the combined extent of both layers.

all three. They differ in the geographic extent for which vector data are recorded, and in how data from the attribute layers are combined. Intersection and union are derived from general set theory operations. The intersection operation may be considered in some ways to be a spatial AND, while the union operation is related to a spatial OR. The clip operation may be considered a combination of an intersection and an elimination. All three are common and supported in some manner as stand-alone functions by most GIS software packages.

A *clip* may be considered a "cookie-cutter" overlay. A bounding polygon layer is used to define the areas for which features will be output. This bounding polygon layer defines the clipping region. Point, line, or polygon data in a second layer are "clipped" with the bounding layer. In most versions of the clip function the attributes for the clipping layer are not included in the output data layer. Only features in the second input data layer are contained in the output data layer.

An example of a clip is shown on the left side of Figure 9-36. The bounding data layer consists of two polygons, and the data layer contains many small wetland boundaries. The presence of polygon attributes in the bounding layer is indicated by the different shades for the two polygons in that layer. The output from the clip consists of those portions of wetlands in the area contained by the bounding layer polygons. Note that the polygon boundaries defining the bounding layer are not included in the output data layer.

An *intersection* may be defined as an overlay that combines data from both layers but only for the region where both layers contain data. This is illustrated in the central panel of Figure 9-36. Data from the polygons of the bounding layer and data from the data layer are combined at the bottom of the central panel. Note that all or parts of polygons in the data layer that are outside the bounding layer are clipped and discarded. A spatial intersection operation may differ from a clip in that data from the bounding layer are also included in the output layer. Each polygon in the output layer may include attributes defined in either the bounding layer or the data layer.

A *union* is an overlay that includes all data from both the bounding and data layers. A union for our example is shown on the right of Figure 9-36. No geographic data are discarded in the union operation, and corresponding attribute data are saved for all regions. New polygons are formed by the combination of coordinate data from each data layer.

Many software packages support additional variants of overlay operations. Some support a complement to the clip function, in which the area covered by the input layer are "cut out" or erased from the bounding layer. Other software packages support other variants on unions or intersections. Most of these specialized overlay operations may be created from the application of union or overlay operations in combination with selection operations.

A Problem in Vector Overlay

Polygon overlays often suffer when there are common features that are represented in both input data layers. We define a common feature as a different representation of the same phenomenon. Figure 9-37 illustrates this problem. A parcel boundary may coincide with a change in a vegetation type. However the parcel boundary and vegetation data layers were digitized from different source materials, at different times, and using different systems. Thus, these two representations may differ even though they identify the same boundary on the Earth surface.

In most data layers the differences will be quite small, and will not be visible except at very large display scales, i.e., when the on-screen zoom is quite high. The differences have been exaggerated in Figure 9-37. When the vegetation and parcels

data layers are overlain, many small polygons are formed along the boundary. These polygons are quite small, but they are often quite numerous.

These "sliver" polygons cause problems because there is an entry in the attribute table for each polygon. One-half or more of the polygons in the output data layer may be these slivers. Slivers are a burden because they take up space in the attribute table but are not of any interest or use. Analyses of large data sets are hindered because all selections, sorts, or other operations must treat all polygons.

There are several methods to reduce the occurrence of these slivers. One method involves identifying all common boundaries across different layers. The boundary with the highest coordinate accuracy is substituted into all other data layers, replacing the less accurate representations. Replacement involves considerable editing, and so is most often used as a strategy when developing new data layers.

Another method involves manually identifying and removing slivers. Small polygons may be selected, or polygons with two bounding arcs, as commonly occurs with sliver polygons. Bounding

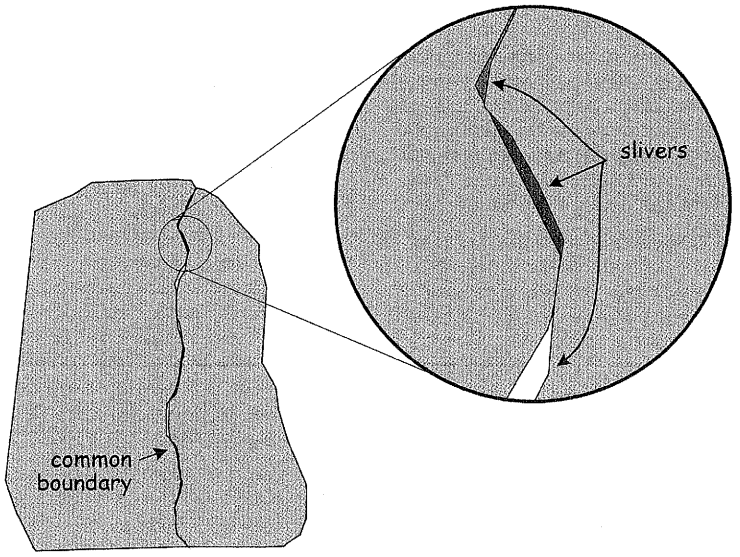


Figure 9-37: Sliver polygons may occur when two representations of a feature are combined. A common boundary between two features has been derived from different sources. The representations differ slightly. This results in small gaps and "sliver" polygons along the margin between these two layers. While slivers may be removed during initial data development, they often occur as a result of overlay operations. Some form of automated or manual sliver removal is often required after multi-layer overlay.

lines may then be adjusted or removed. Manual removal is not practical for many large data sets due to the high number of sliver polygons.

A third method for sliver reduction involves defining a snap distance during overlay. Much as with a snap distance during data development (described in chapter 4), this forces nodes or lines to be coincident if they are within the specified snap distance during overlay. As with data entry, this snap distance should be small relative to the spatial accuracy of the input layers and the required accuracy of the output data layers. If the two representations of a line are within the snap distance then there will be no sliver polygons. In practice, not all sliver polygons are removed, but their numbers are substantially reduced, thereby reducing the time spent on manual editing.

Summary

Spatial analysis, along with map production, is one of the most important uses of GIS. They are often the reason we obtain GIS and invest the substantial time and money required to develop a working system. Any analytical operation we perform on our spatial or associated attribute data may be considered as spatial analysis.

Spatial operations are applied to input data and generate output data. Inputs may be one to many layers of spatial data, as well as non-spatial data. Outputs may also number from one to many. Operations also have a spatial scope, the area of the input data that contributes to output values. Scopes are commonly local, neighborhood, or global.

Selection and classification are among the most often-used spatial data operations. A selection identifies a subset of the features in a spatial database. The selection may be based on attribute data, spatial data, or some combination of the two. Selection may apply set or Boolean algebra, and may combine these with analyses of adjacency, connectivity, or containment. A selected set may be classified in that variables may be changed or new variables added that reflect membership in the selected set.

Classifications may be assigned automatically, but the user should be careful in choosing the assignment. Equal-area, equal-interval, and natural-breaks classifications are often used. The resulting classifications may depend substantially on the frequency histogram of the input data layer, particularly when outliers are present.

A dissolve operation is often used in spatial analysis. Dissolves are routinely applied after a classification, as they remove redundant boundaries that may slow processing.

Proximity functions and buffers are also commonly applied spatial data operations. These functions answer questions regarding distance and separation among features in the same or different data layers. Buffering may be applied to raster or vector data, and may be simple, with a uniform buffer distance, or complex, with multiple nested buffers or variable buffer distances.

Overlay is a final spatial operation described in this chapter. Overlay involves the vertical combination of data from two or more layers. Both geometry (coordinates) and attributes are combined. Any combination of points, lines, and area features is possible, although overlays involving at least one layer of area features are most common. The results of an overlay usually take the lowest geometric dimension of the input layers.

Overlay sometimes results in the generation of gaps and slivers. These occur most often when a common feature occurs in two or more layers. These gaps and slivers may be removed via several techniques.

Suggested Reading

Aronoff, S., Geographic Information Systems, A Management Perspective, WDL Publications, Ottawa, 1989.

Batty, M and Xie, Y., Model structures, exploratory spatial data analysis, and aggregation, *International Journal of Geographical Information Systems*, 1994, 8:291-307.

Bonham-Carter, G. F., Geographic Information Systems for Geoscientists: Modelling with GIS, Pergamon, Ontario, 1996.

Carver, S. J., Integrating multi-criteria evaluation with geographical information systems, *International Journal of Geographical Information Systems*, 1991, 5:321-340.

Chou, Y. H., Exploring Spatial Analysis in Geographic Information Systems, Onword Press, Albuquerque, 1997.

Cliff, A. D and Ord, J. K., Spatial Processes: Models and Applications, Pion, London, 1981.

DeMers, M., Fundamentals of Geographic Information Systems, 2nd Edition. Wiley, New York, 2000.

Heuvelink, G. B. M. and Burrough, P. A, Error propagation in cartographic modelling using Boolean logic and continuous classification, *International Journal of Geographical Information Systems*, 1993, 7:231-246.

Laurini, R. and Thompson, D., Fundamentals of Spatial Information Systems, Academic Press, London, 1992.

Malczewski, J., GIS and Multicriteria Decision Analysis, Wiley, New York, 1999.

Martin, D., Geographical Information Systems and their Socio-economic Applications, 2nd Edition. Routledge, London, 1996.

Monmonier, M., How To Lie With Maps, Chicago Press, Chicago, 1993.

Steinitz, C. P. and Jordan, L., Hand-drawn overlays: their history and prospective uses, *Landscape Architecture*, 1976, 56:146-157.

Worboys, M. F., GIS: A Computing Perspective, Taylor and Francis, London, 1995.

Study Questions

Can you define and give examples of local, neighborhood, and global spatial operations?

What are selection and classification operations?

Can you describe set and Boolean algebra?

Can you describe three different classification methods?

What is a dissolve operation? What are they typically used for?

What is the basic principle behind buffering? What are some of the buffering variants that are applied?

How are raster proximity functions different from vector proximity functions?

How are point, line, and area buffers constructed?

What is the basic concept behind layer overlay?

Can you diagram and contrast raster and vector overlay?

Why are output features in vector overlay typically set to the minimum dimensional order (point, line or polygon) of the input features?

How are clip, intersection, and union overlay different?

What is the sliver problem in vector layer overlay? How might this problem be resolved?

10 Topics in Raster Analysis

Introduction

Raster analyses range from the simple to the complex, largely due to the early invention, simplicity, and flexibility of the raster data model. Raster structures are based on two-dimension arrays, constructs supported by many of the earliest programming languages. The raster row and column format is among the easiest to specify in computer code, and the structure is easily understood, thereby encouraging modification, experimentation, and the creation of new raster operations. Raster cells may store nominal, ordinal, or interval/ratio data, so a wide range of variables may be represented. Complex constructs may be built from raster data, including connected cells to form networks, or groups of cells to form areas.

The flexibility of raster analyses has been amply demonstrated by the wide range of problems they have helped solve. Raster analyses are routinely used to predict the fate of pollutants in the atmosphere, the spread of disease, animal migration, and crop yields. Time varying and wide-area phenomena are routinely analyzed using raster data, particularly when remotely-sensed inputs are available. Raster analyses are routinely used at the small parcel level, for example by the U.S. Environmental Protection Agency in hazard analysis of urban superfund sites, to global-scale estimates of forest growth. Local, state, and regional organizations have used raster analyses at many scales in between.

Numerous research projects have expanded and embellished the basic raster data structure, as well as investigated a general set of raster tools for spatial data analyses. Yale University, the Ohio State University, the Idrisi Project of Clark University, and the Harvard School of Design are among public institutions that developed raster analysis packages for research over the past four decades. Commercial products have been developed by a number of companies.

The long history and level of interest in raster analyses have resulted in a basic set of tools that should be understood by every GIS user. Many of the tools are based on a common conceptual basis, and form a set of generic methods that may be adapted to several types of problems. In addition, specialized methods have been developed for less frequently encountered problems. The GIS user may more effectively apply raster data analysis if she has developed an understanding of the underlying concepts and has become acquainted with a range of specialized raster analysis methods.