SPATIAL OPERATORS FOR SELECTED DATA STRUCTURES

By Robert W. Claire
and
Stephen C. Guptill
U.S. Geological Survey
521 National Center
Reston, Virginia 22092

ABSTRACT

The discipline of mathematics is characterized by a small, fundamental set of operators which, when taken in concert, support complex manipulations and analyses. In a similar sense, higher levels of spatial applications programming are founded upon a fundamental set of spatial operators. The tendency to focus upon geometric elements and operators based upon the dimensionality of space is perhaps due to the straightforward manipulations of these elements on a conceptual level. Given the necessity for continuous geometric elements to conform to a discrete, sequential computer storage, our purview should be extended to a primal level that exploits the discrete qualities of spatial data representations.

INTRODUCTION

The U.S. Geological Survey (USGS) is currently compiling a Digital Cartographic Data Base (DCDB) consisting of digital elevation models, digital line graphs, and thematic data. Contained in the data base will be numerous categories of information relating to transportation features, hydrology, public land net, administrative boundaries, and land use and land cover. The potential size of the data base is extremely large; with complete coverage of the conterminous United States requiring several trillion bits of spatial data.

To complement its role as a generator of digital map data, there is an increasing realization of the need for the USGS to develop and expand capabilities for supporting applications programming. Although applications programming, per se, may be as varied as the individual contexts of the users, a fair degree of underlying commonality does exist among applications. The concept of spatial operators is thus an appealing one. Spatial operators, analogous to mathematical operators, perform fundamental manipulations on spatial data and can be used in combination to support more sophisticated types of analyses.

Spatial operators have been keyed to the dimensionality of space. The basic manipulations of point entities, linear entities, and areal entities, which are defined to be operators, are straightforward on a conceptual level. Implementation considerations, however, must address the representations of these conceptual spatial elements as

afforded by alternative spatial data structures. Herein lies the proverbial problem of spatial data handling; that is, "of managing a two-dimensional continuous view of reality in a natural and efficient way within the discrete and sequential computer storage" (Tamminen, 1980).

An approach is outlined below that explicitly extends the purview of spatial operators to primal spatial elements. A taxonomy is envisioned for the formulation of spatial operators ranging from a user-logic level of abstraction to a primal level. Primal elements are based upon components for the discrete representation of geometric entities. Primal operators are formulated about a minimal structure. They perform basic manipulations independent of higher order constructs and thus are suitable for alternative data structures.

SYNTAX AND TERMINOLOGY

An operation, as given in mathematics, is a process or action performed in a specified sequence and in accordance with specific rules of procedure. That which distinguishes an operator from an operation is the straightforward and fundamental nature of the action, such that an operator is typically referenced by a commonly accepted symbol or abbreviation.

The syntax is simple. Operators act upon operands to produce a result. Operands, in turn, are defined by the values that they may assume and the type of operators to which they may be subjected. Given operands of a certain type and an appropriate operator, the results generated are of an expected form.

In higher level programming languages, the constructs of operators and operands are handled explicitly. A variable is declared to be of a particular data type (e.g., LOGICAL, REAL, INTEGER, etc.). This variable, in turn, constitutes a valid operand for an operator which is defined to act upon its data type.

A case has been made for a spatial data type (Aldred, 1974), but the data types, and respective operators, inherent to programming languages are found not well suited for spatial data handling. Candidates for spatial data types typically reference the dimensionality of space; that is, nodes (point entities), arcs (linear entities), and polygons (areal entities). (Dimensions greater than two are not considered in this discussion for simplicity.) Existing mathematical operators such as addition, multiplication, exponentiation, and the like have little meaning for spatial entitities. Conversely, operations including logical polygon intersection, union, and difference are very relevant to spatial data handling but cannot be expressed by existing operators.

What constitutes a spatial operator may lie in the eyes of the invoker; however, some frequently referenced spatial operators are listed in Table 1. A spatial operator is

classified as monadic or dyadic depending upon whether it acts upon a single or a pair of geometric operands. Dyadic operators are further classified as symmetric or asymmetric. Symmetric operators are commutative; the order of the operands is not significant. With asymmetric operators, the order is important. Results of spatial operators are categorized as numeric, boolean, geometric, or a particular type of geometric entity.

Table 1. Commonly referenced spatial operators

N: node, A: arc, P: polygon, G: geometric (i.e., N,A,P), Nu: numeric, B: boolean, Mon: monadic, Dya: dyadic, Sym: symmetric, Asy: asymmetric.

Operator	Class	Operand	Result	Comments
Length	Mon	A	Nu	
Area	Mon	P	Nu	
Boundary	Mon	P	A	
Box	Mon	G	P	Bounding rectangle.
Extend	Mon	G,Nu	P	Search areas about N,A or P.
Union	Dya,Sym	G	G	
Intersection	Dya,Sym	G	G	
Difference	Dya,Asy	G	G	
Separation	Dya,Sym	G	Nu	Shortest distance.
Equality	Dya,Sym	G	B	
Inequality	Dya,Sym	G	B	
Contain	Dya,Asy	G	B	
Share	Dya,Sym	G	B	

A spatial operator can be defined in a manner sufficiently concise and sufficiently general by adopting a recursive definition:

A spatial operator is:

(a) A straightforward manipulation of a spatial entity with the result in an expected form; or
(b) A sequential set of spatial operators; or
(c) An iterative application of a spatial operator.

The issue arises as to the definition of a spatial entity. Definitions based upon the dimensionality of space appear only to address the issue from one perspective, that of an object geometry. To a user, whose applications are to be supported by spatial operators, individual polygons, arcs, and nodes are but abstractions of the user's view of spatial reality. Spatial entities, and respective operators, may be defined for a user-logic level.

Moreover, polygons, arcs, and nodes represent entities in a continuous two-dimensional space, and they, in turn, may be abstracted to conform to the discrete nature of computer storage. The digital representations of geometric entities are comprised of lower level spatial entities that are referred to here as primal spatial elements.

These multiple perspectives suggest an approach where spatial operators are formulated about entities at different levels--a user-logic level, a geometric level, and a primal level.

USER-LOGIC LEVEL

The purpose of spatial operators is to provide for the structured manipulation of spatial entities, which, in turn, supports some higher level, user-specific application. At a user level of abstraction, entities such as nodes, arcs, and polygons are not likely to have much meaning. Rather, spatial reality from the user's view is comprised of point features, linear features, and areal features, each with their respective attributes. These geographic features comprise spatial entities at the user-logic level.

Spatial operators may be formulated about entities at the user-logic level which need not access the object geometry. Instead, references to spatial entities are limited to an indexing system as it would be for an aspatial type of observation. The following are examples modeled after simple query types given by Martin (1975):

Ai(Ej) = ??		What is the value of attribute i for spatial entity j?
	>	
	<	
Ai(??) = Vk		What spatial entities have values for
	≠	attribute i that are equal to (etc.) a value k?
	>	
	<	
??(Ej) = Vk		What attributes of spatial entity j
	≠	have values equal to (etc.) a value k?
??(Ej) = ??		What are the values for all attributes of spatial entity j?
Ai(??) = ??		What are the values of attribute i for all spatial entities?
	>	
	<	
??(??) = Vk		What are the attributes of all
	≠	spatial entities that have a value equal to (etc.) a value k?

These simple operators can be used in combination to formulate more complex operators. For example, the set intersection of all spatial entities with an attribute 'i' less that a value 'k' AND an attribute 'r' greater than a value 's'. The limitations of indexing-type operators,

however, become apparent when user queries address spatial characteristics and locational aspects of the entities. To accommodate fundamental types of explicitly spatial queries, operators are formulated about a geometric level.

GEOMETRIC LEVEL

Associated with entities in the user's frame of reference is an object geometry; that is, the nodes, arcs, and polygons that relate to spatial features in the user logic. It is the object geometry that must be accessed to handle explicitly spatial queries. For example, an indexing structure cannot handle, say, 'the area within a distance 'd' of some linear feature'.

A one-to-one mapping is not likely between entities at the geometric level and entities at the user-logic level. The State of Michigan is typically given as an example of a single user-logic entity (State jurisdiction) corresponding to two geometric entities (the upper and lower peninsulas). Various hierarchial and lattice pointer structures can be established to track the topological relationships between the user-logic and geometric levels.

Complexities in formulating spatial operators at the geometric level focus upon two factors--the distinction between objects of dimensionality, and the lack of distinction between continuous spatial entities and their respective representations within the discrete and sequential computer storage.

Consider the node, arc, and polygon typology in the context of the intersection of two polygons. It is not difficult to conceive of a resulting configuration that yields not only common subareas of the polygons, but also common arc segments and common nodes. This result does not conform to the predefined categories of the typology.

A suggested approach (Cox and others, 1980) would define a 'geometric' data type that would accommodate mixed configurations of nodes, arcs, and polygons. It is not resolved as to whether the geometric data type would constitute a single data type or coexist with node, arc, and polygon data types. In the latter case, a new operator, 'dimensional select,' can be defined to extract a node, arc, or polygon data type, and ignore extraneous data.

Consider again the intersection of two polygons. On a conceptual level where polygons can be viewed as continuous, two-dimensional elements, their intersection is easily identified. Similarly, given representations that allow for the definition of continuous spatial elements, the intersection is easily specified. For example, the circular regions about two points, (x_1, y_1) and (x_2, y_2), with radii of r_1 and r_2 respectively can be expressed in equation form. The intersection of the two areas is the set of points (X,Y) such that:

$$[(X-x_1)^2 + (Y-y_1)^2 \leq r_1^2] \text{ .AND.}$$
$$[(X-x_2)^2 + (Y-y_2)^2 \leq r_2^2]$$

An .OR. condition would indicate the union of the two areas.

Only the most regular of spatial entities are easily defined by mathematical equations. Given the irregularity characteristic of spatial entities, a coordinate representation must be employed that conforms to the discrete computer storage.

The conceptual elegance of spatial operators at the geometric level is quickly lost at implementation time when the once continuous elements are treated in a discrete manner. A straightforward algorithm for polygon intersection based upon a vector representation is feasible only for the simplest of configurations. The number of required comparisons between vectors approach a near-exponential growth. Referencing available topological relations and invoking sorting routines reduces the permutations, but the sense of a straightforward, basic manipulation is lost.

In contrast, Merrill (1973) has demonstrated the elegance of raster implementations for polygon overlay. Rasters, however, do not represent a complete solution given the loss of definition for the logical spatial elements.

Burton suggests that a spatial operator not be tied to any particular data representation (Burton, 1979). Instead, a spatial operator could be invoked for the representation that best suits it. The circular polygon that results from extending a node, for example, can be represented by the equation of the circle. Implicit in this spatial operator/data representation independence are the capabilities for the effective conversion among different representations and the intelligence for determining their applications.

Perhaps some of the complexities described above can be alleviated by transplanting manipulations of the discrete elements which are used to approximate continuous spatial entities from the geometric level to a subordinate level. Operators at the geometric level may focus upon the manipulation of geometric entities as continuous elements. For example, given a circular polygon represented in equation form, geometric operators can be defined to return polygon area, boundary, perimeter, extreme coordinates, and the like without referencing a discrete representation of the polygon. The same polygon in raster or vector form can be passed to a subordinate, primal level for action by respective primal operators.

In addition, operators at the geometric level can be formulated about relations between geometric entities that do not require direct consideration of the respective image data. For example, an operator that, given a polygon, returns adjacent polygons.

Capabilities for converting from one representation to another, for invoking efficiency steps such as sorting and the like, and for identifying relevant elements of a geometric configuration (e.g., dimensional select), would form an interface between the geometric and primal levels.

PRIMAL LEVEL

The key underpinning of the primal level is that constructs are basic and structure is minimal. Perhaps by adhering to this requirement, a set of primal operators, together with primal data types, can be defined which will serve various applications regardless of higher level data structures.

A primal level is defined about the most fundamental construct of spatial data--a point, any higher order constructs (e.g., arcs, polygons) are reducible to relationships among points. And the manner in which continuous spatial entities are treated in discrete, machine-compatible form can be exploited to define primal spatial elements (Figure 1).

NODE SEGMENT:	A point which is distinct, or unrelated to adjacent points, or related to itself.
VECTOR SEGMENT:	A set of points, delineated by two end points, that are related along a single dimension in two dimensions.
HALF-PLANE SEGMENT:	A set of points, delineated by two end points, that in part define the limit of a set of points related in two dimensions.
RASTER SEGMENT:	A set of points, delineated by two end points, that in part define the extent of a set of points related in two dimensions.

Each of the primal elements is based upon a set of points that may be delineated by two points (a null relation for node segments). This allows for a uniform logical record format for the primal elements that can simply be expressed as [ET PT1 PT2], where ET is an element type code, and PT1 and PT2 are the coordinate pairs of the two endpoints.

For nodes, PT1 and PT2 are identical. For half-plane segments, the order of the endpoints can be defined such that related points are always to the right (or left) of the segment. A similar rule may be useful for raster segments; the first entry being the leftmost point. With primal vector segments, the order of endpoints is irrelevant. The required order of endpoints for half-plane or raster segments gives the segment an implicit direction. Any explicit treatment of direction (e.g.

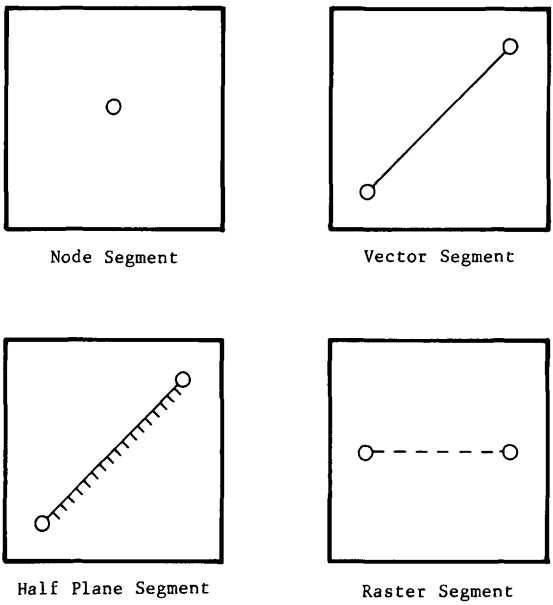


Figure 1. Primal level elements

streamflow, one-way streets) is considered an attribute of the segment and should be treated separately in the data structure.

The uniform format allows for sets of primal elements to be easily grouped into 'stacks'. An arc entity can be represented by a stack of vector segments. Alternatively, a stack may represent a network of arcs. Similarly, a polygon, or a group of polygons, can be encoded as a stack of half-plane or raster segments.

Given the desire for minimal structure, the order of the individual elements within a stack is considered arbitrary. Moreover, stacks may consist of primal elements of differing types. The objective at the primal level is to develop operators that act upon pairs or individual primal elements such that their repeated application for a stack(s) supports respective higher level geometric operators.

Some obvious candidates for monadic primal operators include the following:

LENGTH: The length of each vector segment is computed and accumulated.

AREA: The area below each half-plane segment (to some arbitrary lower limit) is computed and accumulated. The area figure is added and subtracted from the accumulator for segments in a +X and -X direction respectively.

BOUNDARY: The element type code for each half-plane segment is changed to indicate a vector segment.

PERIMETER: Invoke BOUNDARY and LENGTH operators for each half-plane segment.

COMPLEMENT: Reverse order of endpoints for each half-plane segment.

In each case, the order of the elements in a stack is irrelevant, as is the number of geometric entities represented by the stack.

Formulating dyadic primal operators, not surprisingly, is more complex. There are numerous ways in which a pair of primal elements may relate to one another; some of the permutations are illustrated in Figure 2. Primal operators can be developed to test for these types of relations and return the appropriate logical or graphic result.

In some instances, the result of a primal operator can be directly related to a geometric relationship. For example, 'true' results for a test of relations (C.2), (D.3), or (D.4) in Figure 2 would indicate that a node falls upon a polygon boundary.

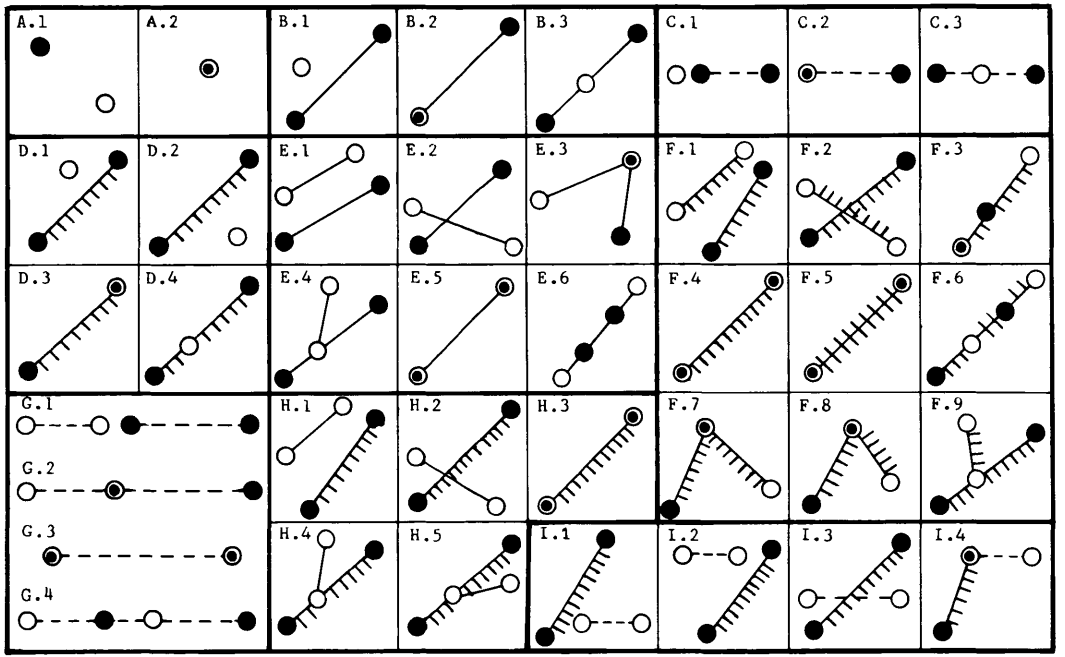
Determining if a node falls within a polygon is a straightforward test for polygons represented by raster segments, as illustrated in relation (C.3). For half-plane segments, the test requires more steps. A ray can be extended in one direction from the node and a count maintained of the number of intersections with half-plane segments. If odd, the node is inside; otherwise the node is outside.

This process of determining the parity of intersections with half-plane segments serves also to determine if other disjoint primal elements are within polygons represented by half-plane segments. The process is simply applied to each endpoint.

Primal operators for raster-to-raster relations (G.1) to (G.4) can be modeled after standard raster processing methods. Steps for determining union, intersection, difference and the like are straightforward.

Primal operators to test for vector-to-vector relations (E.1) to (E.6) are less straightforward than that for raster segments. A bounding rectangle test would determine the possibility for intersection. Endpoint equality test would detect relations (E.3) and (E.5). Finally, solving the two simultaneous linear equations would determine any interim points of intersection (e.g., (E.2),(E.4), and (E.6)).

The same steps used for vector segments are applicable for testing relations among half-plane segments, but with an additional consideration for direction. For example,



A: node-node relations
B: node-vector relations
C: node-raster relations
D: node-half plane relations
E: vector-vector relations
F: half plane-half plane relations
G: raster-raster relations
H: vector-half plane relations
I: raster-half plane relations

Figure 2. Sample relations between primal elements

relation (E.5) represents coincidence for the vector segments. A respective result for half-plane segments requires a subsequent step to determine actual coincidence (F.4) or adjacency (F.5).

Primal operators for half-plane segments that in part support polygon overlay can be illustrated with relation (F.2). Given relation (F.2), operators for union, intersection, and difference would return the subdivided half-plane segments as illustrated in Figure 3. These steps would have to be augmented to detemine the disposition of disjoint half-plane segments (i.e. inside, outside) in a manner similar to that for node segments described above.

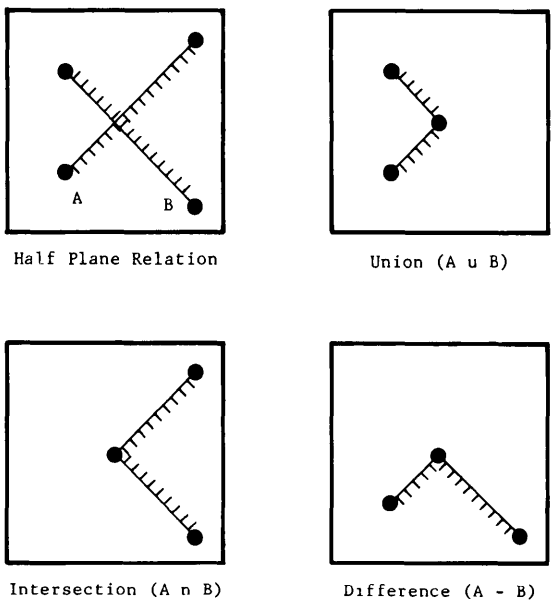


Figure 3. Primal operators related to polygon overlay

By extending the formulation of primal operators to handle elements of different types, tests can be made for relations illustrated by relations (H.1) to (I.4). This capability would serve to support the overlay manipulations of polygons in alternative representations and the overlay of polygons with network configurations and point distributions. The stacks may be designed to accommodate primal elements of different types. Two such stacks can be manipulated given primal operators defined for elements of different types to produce a resultant stack.

Considered in total, such stacks are much like the geometric data type discussed above. A dimensional select-type operator can sift through the resultant stack, piecing together desired higher level constructs (e.g., arc, polygon).

SUMMARY

The tendency to focus upon geometric elements and operators based upon the dimensionality of space is perhaps due to the straightforward manipulations of these elements on a conceptual level. Given the necessity for continuous geometric elements to conform to a discrete, sequential computer storage, our purview should be extended to a primal level that exploits the discrete qualities of spatial data representations. By formulating a primal operator that has geometric significance, yet employs a minimal structure, its utility is independent of any particular data structure.

REFERENCES

Aldred, B.K., 1974, The Case for a Locational Data Type: United Kingdom Scientific Centre, IBM United Kingdom Limited, County Durham

Burton, W., 1979, Logical and Physical Data Types in Geographic Information Systems, Geo-Processing, Vol. 1, pp. 167-181

Cox, N.J., Aldred, B.K., and Rhind, D.W., 1980, A Relational Data Base System and a Proposal for a Geographical Data Type: Geo-Processing, Vol. 1, pp. 217-229

Martin, J., 1975, Computer Data-Base Organization: Prentice-Hall Inc., Englewood Cliffs, New Jersey

Merrill, R.D., 1973, Representation of Contours and Regions for Efficient Computer Search: Communications of the ACM, Vol. 16, pp. 69-82

Tamminen, M., 1980, Management of Spatially Referenced data: Report-HTKK-TKO-B23, Helsinki University of Technology, Finland