A SPATIAL DATA STRUCTURE FOR GEOGRAPHIC INFORMATION SYSTEMS

Robert M. Haralick

Department of Electrical Engineering
Virginia Polytechnic Institute and State University
Blacksburg, Virginia

ABSTRACT

In this chapter we describe a data structure that is equally natural for spatial data in the raster run length format and the vector format. This makes it easier to design a geographic information system that handles both data formats and that optimizes for itself whether any particular data set is in raster or vector format. This same data format is appropriate for intelligent query and inference uses as described in Shapiro's chapter of this book.

Brief mention is made of the topological consistency requirements for each data format. A bibliography of many of the fast algorithms for vector format data is provided at the end of the paper.

I. INTRODUCTION

Geographic information systems have requirements for high volume spatial data handling. Because the volume of spatial data is so high, it is particularly important for the geographic information systems to have efficient data storage formats and associated optimal algorithms.

ISBN 0-12-267180-5

Spatial data has two natural organizations: raster format and vector format. Arguments have been made in favor of each of them. It is the thesis of this paper that the particular format and associated algorithms by which a user's spatial data is processed is not relevant to the user. What is relevant is the cost of processing. This suggests that the geographic information system must be able to handle both the raster and vector formats using the particular format most cost effective for each individual's spatial data processing profile and spatial data set. Format conversions, if determined to be cost effective for a processing profile, are the responsibility of the geographic information system, not the user.

Because the vector data format and the raster data format are so different, one might expect that there is no logical data structure that naturally can contain these formats. From this it might follow that a geographic information system which had to use and keep track of the different formats would be excessively complex. It is the purpose of this paper to argue against such a conclusion.

We first discuss a logical spatial data structure which naturally and efficiently can contain the vector format data or raster format data. Second we describe some of the efficient processing algorithms associated with each of the formats. Note that the structure we discuss here is one that is suitable for the inference algorithms that enable high level queries to be answered. Its use in inference can be found in Shapiro's chapter of this book.

Rather than begin with a definition of the suggested spatial data structure, we begin with a toy example which has

in it perhaps just enough complexity to indicate the universality of the data structure for spatial data processing as well as inference work. The example emphasizes spatial relationships without exclusively using coordinate information. This kind of information is sometimes useful in performing limited inference tasks which do not have to depend on coordinates. In the example, we will give a written description of some spatial information about a place called Ecaf. Then we will encode it in a brief tabular form closely related to our logical spatial data structure and illustrate a map of Ecaf visually portraying the spatial information. Finally we will put the information about Ecaf into our suggested spatial data structure.

Example

Ecaf has a population of 1,252,140 and occupies a near circular area of 30,179 square mm. It is located in the uppermost part of Ydob. It has four main regional centers of activity: htuom, eson, and the twin seyes.

The twin seyes are located in the northern part of Ecaf, one in the west side and one in the east side. Each was planned and constructed identically by the same master architect. Each occupies a 200 square mm oval shaped area, elongated in the east west direction. Each has a circular downtown business area of about 20 square mm located in its west end. Lying on low land, east and west seye are protected on the north from the seasonal flooding by a barrier forest. The main function of the seyes is to act as the receiving center for the high bandwidth information that comes into Ecaf.

Eson is a city of 150,000 located in the center of Ecaf.

It has a thin triangular 300 square mm area elongated in the north-south direction and its northern tip is between the two seyes. It functions as the gas trade center of Ecaf.

Htuom is a city of 200,100 located in the southern part of Ecaf. It has a thin elongated 1500 square mm area stretching from southwest Ecaf to southeast Ecaf. Its east and west ends have been experiencing northward growth. Htuom is an industrial city importing raw materials as well as being an outgoing communication center of Ecaf.

Tables 1 through 6 list the information about Ecaf in tabular form. Some of the data such as population or area are just characteristics of Ecaf. Other data such as boundary give the name of the set of coordinates defining the boundary of the areas involved. Figure 1 gives a map of Ecaf, a familiar place: the human face.

We will now classify the kind of data in the tables. First, every table contains the name and type of the entity represented by that table. Entity type provides the key to the entity type prototype which indicates the kind of information that can be in the entity type and the meaning of the information. Shapiro's chapter in this book fully describes the notion of prototype. Second the table contains the attributes the entity can have; shape, area, population, and so on. Attributes can have values which are atomic (numbers, character strings, or N-tuples) or names of other entities. Finally,there are relations which may be simple lists (ordered or unordered) such as the unordered list (a unary relation) associated with "contains". A relation may be a set of ordered pairs such as the binary relation of coordinates associated with "boundary", or may be more complicated N-ary relations.

Entity Name	Ecaf
Entity Type	County
Population	1,252,140
Area	30,179 sq mm
Area Shape	Near Circular
Situated In	Northern Part of Ydob
Contains	Htuom, Eson, and Twin Seyes
Boundary	Ecaf Boundary

Table 1 lists the information available about Ecaf

Entity Name	Twin Seyes
Entity Type	City
Situated In	Northern Ecaf
Contains	Seyes East and Seyes West

Table 2 lists the information about Twin Seyes

Entity Name	Seyes East
Entity Type	City
Population	279,300
Area	200 sq mm
Area Shape	Oval, Elongated in East-West direction
Situated In	Northeast Ecaf
Contains	Downtown Seyes East
Boundary	Seyes East Boundary
Protected By	Forest East
Located On	Lowland
Functions As	Receiving Center

Table 3 lists the information about Seyes East

Entity Name	Downtown Seyes East
Entity Type	Central Business District
Area	20 sq mm
Area Shape	Circular
Situated In	West Side of Seyes East

Table 4 lists the information on Downtown Seyes East

Entity Name	Eson
Entity Type	City
Area	300 sq mm
Area Shape	Thin Triangular, North-South Elongated
Situated In	Central Ecaf
Contains	Twin Slartsons
Boundary	Eson Boundary
Functions As	Gas Trade Center

Table 5 lists information on Eson

Entity Name	Htuom
Entity Type	City
Population	200,120
Area	1500 sq mm
Area Shape	Elongated
Situated In	Southern Ecaf
Functions As	Industrial Center
Boundary	Htuom Boundary

Table 6 lists the information on Htuom

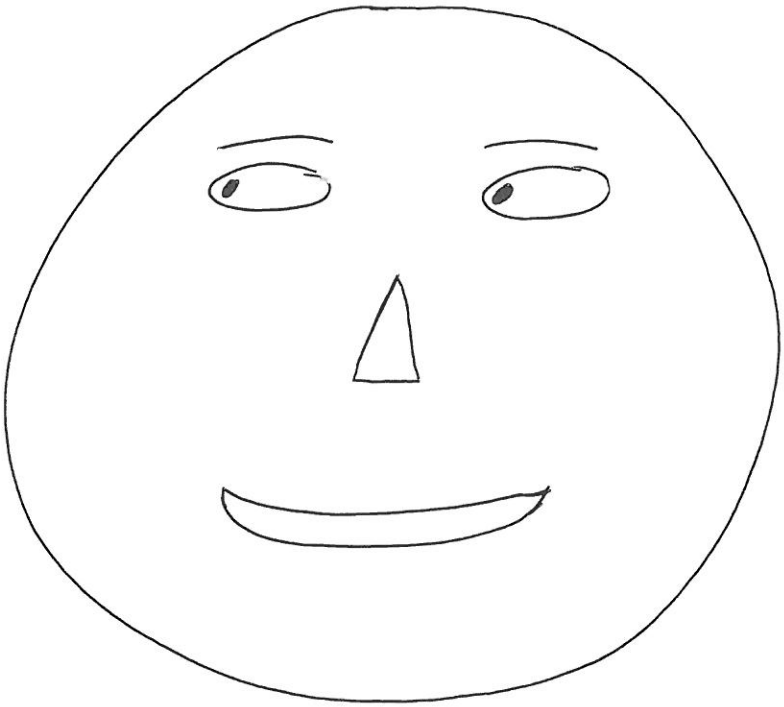


Figure 1 shows a map of the human face.

This suggests the following formal definition of a general spatial data structure: A spatial data structure S is a set of named relations $\{R_1,\ldots,R_k\}$ where each relation R_k is some N_k-ary relation whose tuples may have components whose values are atoms or spatial data structures.

To illustrate this structure, Figure 2 shows the Ecaf information in Table 1 put into the logical form of the spatial data structure. The entity name and type information actually appear in the directory of the spatial information system. The purpose of the directory is to give the "address" of any spatial data structure as well as the name of its prototype. Ecaf consists of four relations: an attribute value table, a contains relation, a situated relation, and a boundary relation. The attribute value table is a binary relation and has the attributes population, area, and shape with their corresponding values. The contains relation is a unary relation consisting of the simple set (unordered list) Htuom, Eson, and Twin Seyes. The situated in relation is a binary relation consisting of the set whose only member is the pair (northern, ydob). The character string "northern" means north part of and is really the name of a generic relation. The boundary for Ecaf is assumed to be a polygon specified by its vertices an ordered set of (x,y) coordinate pairs. We may think of this as an ordered binary relation or as a ternary relation.

Figure 2 only shows the logical structure. There are a wide range of choices for the physical implementation of this logical structure. Reasonable physical implementations would probably not explicitly represent entity name and type, attribute value table, contain, situated in, and boundary as

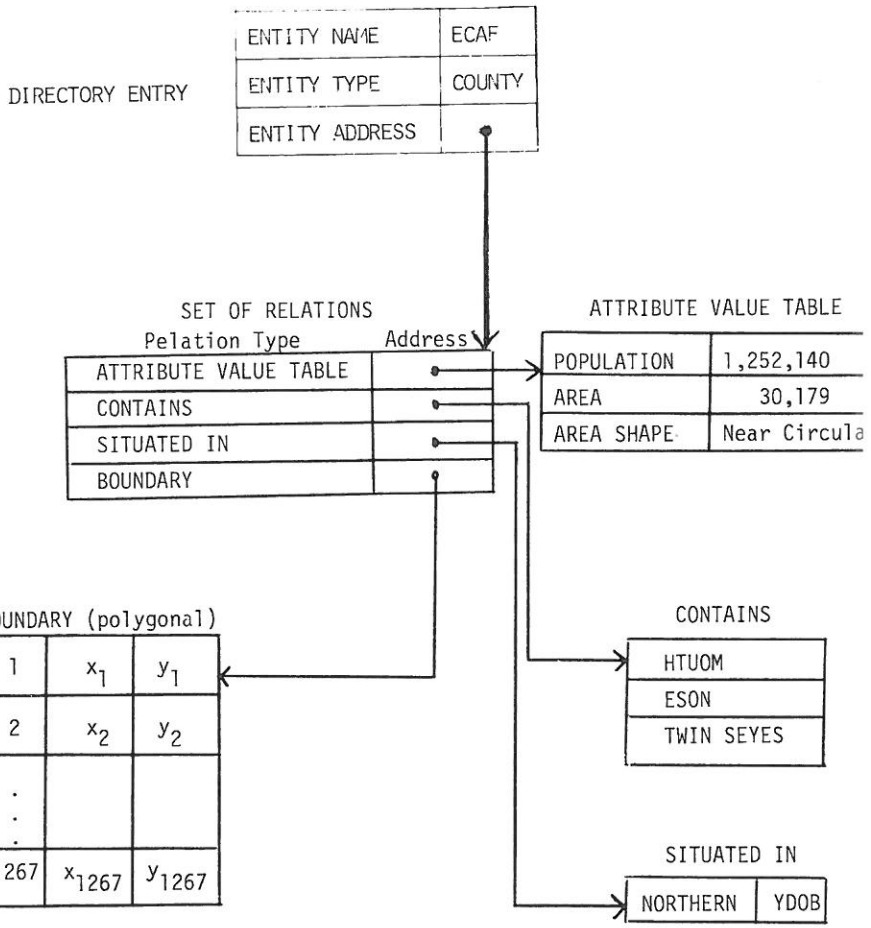


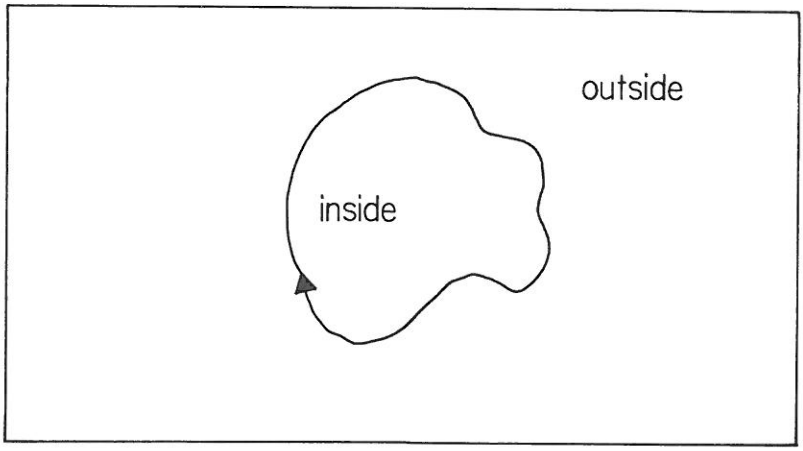
Figure 2 shows the Ecaf information of Table 1
put into the logical form of the spatial data structure

character strings. Conventions could establish the order in which entity name, type, and address occur in a directory. The types of relations, associated with their addresses, could be given by bit table, for example. The ordered relations such as boundary would not explicitly represent the indexing giving the order of the (x,y) coordinate pairs. The order would be implicitly in the natural ordering of words in memory.

II. VECTOR FORMAT SPATIAL DATA

In vector format spatial data, areas are represented by their boundaries. Simple boundaries are simple closed curves which separate the inside of an area from its outside. To enable the distinction of inside and outside to take place, we must consider the closed curve to be directed. As we travel on the closed curve in the designated direction there will be an area to our right and an area to our left. Each of these areas can be associated with their names, thereby allowing the designation of inside and outside to occur. See Figure 3.

In complex spatial data, the total area is divided into many regions with many pairs of regions sharing some portion of their boundary. If we follow the data organization principle that each data item may only be represented once, then the closed curve bounding any area must be divided up into segments, each segment being a connected boundary arc whose left side is one region and whose right side is another region. In this manner each region can reference the same arc segment which is part of a shared boundary. Figure 4 illustrates a map composed of four regions, A, B, C, D, and with directed arcs a through m bounding these regions. Table



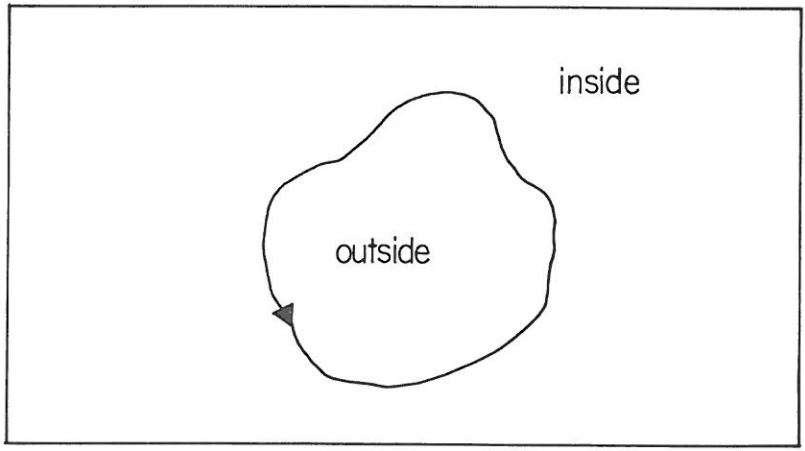


Figure 3 illustrates the directed closed curve which can label areas. In this example, the area to the right is the inside area.

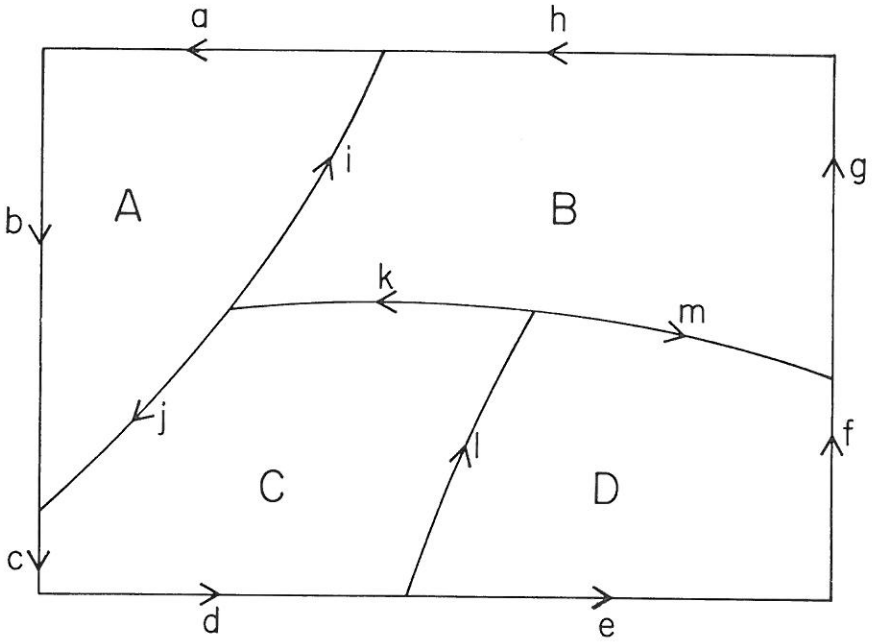


Figure 4 illustrates a map composed of four regions with directed arcs bounding these regions.

7 lists the relationship between arc name, its left region, and its right region for this example.

The boundary for region B then consists of the ordered sequence of arcs i, h, g, m and k. Traversing these arcs in the sequence given produces region B to the right of the traversed direction. Notice that at the place where two arcs meet it is not necessary that the arc directions be the same. Also,the direction in which we travel around a closed boundary does not have to agree with the direction specified in any of the arc segments of the boundary. The arc segment directions only give meaning to the region at the arc's left or right. From this example we see that a simple boundary is an ordered set of arcs forming a simple closed curve.

Another characteristic of complex spatial data is the fact that there may be some regions which have holes (regions which they entirely surround). Furthermore, the holes themselves may have holes which are part of the region. Such a region is necessarily disconnected and the determination of whether a point is in the region or not must make full use of the boundary information of the holes. Figure 5 illustrates a simple example.

This motivates the following tentative definition of boundary. A boundary is a simple boundary plus a (possibly empty) set of boundaries of the holes for the region it bounds.

This definition would be alright except for the fact that a region may have disconnected parts no one of which is a hole in another as illustrated in the map of Figure 6. This suggests that a boundary is a set of simple boundaries, each one of which bounds a connected region and has an associated

ARC NAME	REGION LEFT	REGION RIGHT
a	A	--
b	A	--
c	C	--
d	C	--
e	D	--
f	D	--
g	B	--
h	B	--
i	A	B
j	C	A
k	C	B
l	C	D
m	B	D

Table 7 shows the association of region left and region right to each directed arc of figure

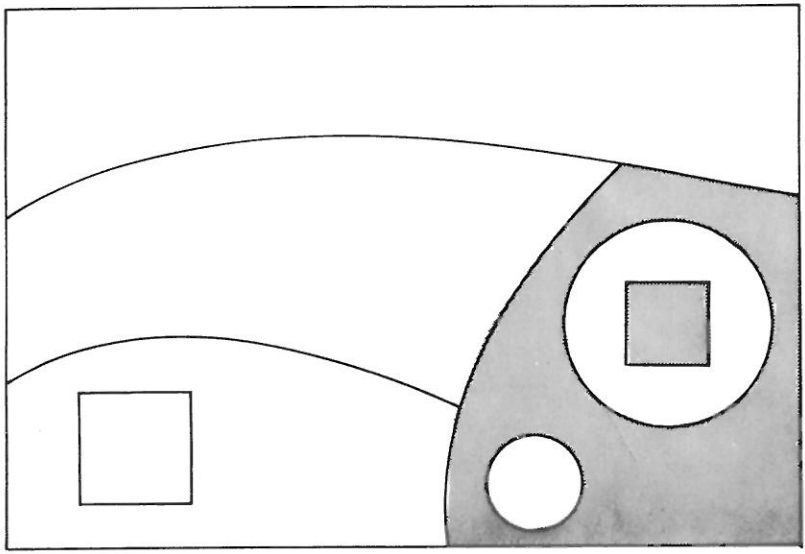


Figure 5 illustrates an example where regions have holes. Notice that one of the holes itself has a hole which is part of the region.

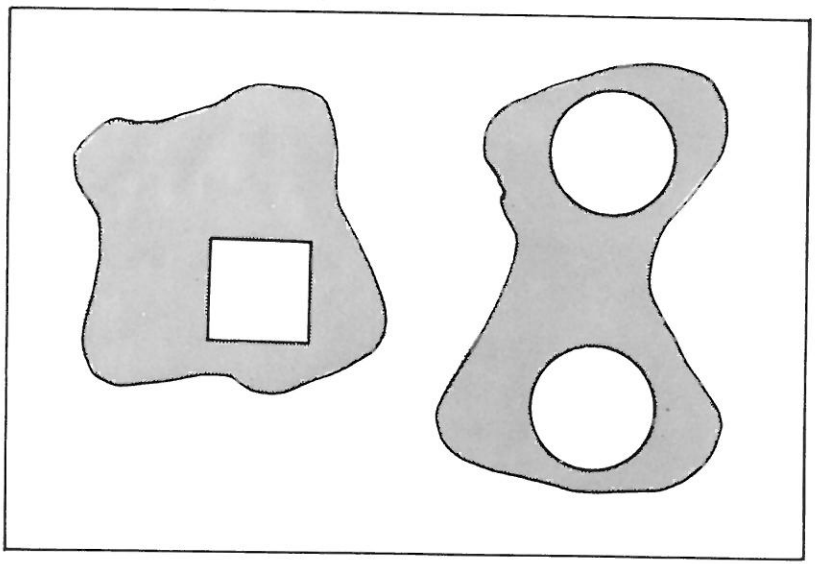


Figure 6 shows a map having a region which is disconnected and each connected part having at least one hole.

(possibly empty) set of boundaries for the holes of the connected region it bounds.

The definition of a boundary then is perhaps more complex than one might initially think. But it does bring out quite nicely a very important and interesting property of spatial information; spatial information at a high level is relational and at a lower level is more data coordinate oriented. Boundary is not just a simple ordered binary relation of (x,y) coordinate pairs. Boundary is a full and complete spatial data entity in its own right.

Illustrating this idea with an example from Ecaf, consider the boundary of Twin Seyes. Twin Seyes is a disconnected region with two pieces Seyes East and Seyes West. Seyes East and West are connected regions each having a simple boundary and each having a hole. Let us suppose that the name of the boundary for Twin Seyes is Twin Seyes Bndry. It is a spatial data structure having an attribute value table and Boundary of relation. The Boundary of relation is a unary relation, a set, whose members are Seyes East and Seyes West. These are, of course, spatial data structures of regions having boundaries. As can be seen from Figure 7 the spatial data structure for Seyes East has a unary relation of type B. The singleton member of this set is Seyes East B, the name of the spatial data structure for the boundary of Seyes East. Referring to Figure 8, Seyes East B is a spatial data structure having for its relations an attribute value table, a simple boundary, and a set for the boundaries of its holes. The simple boundary is an ordered set of five arcs.

The circularly ordered list of five arcs may actually be

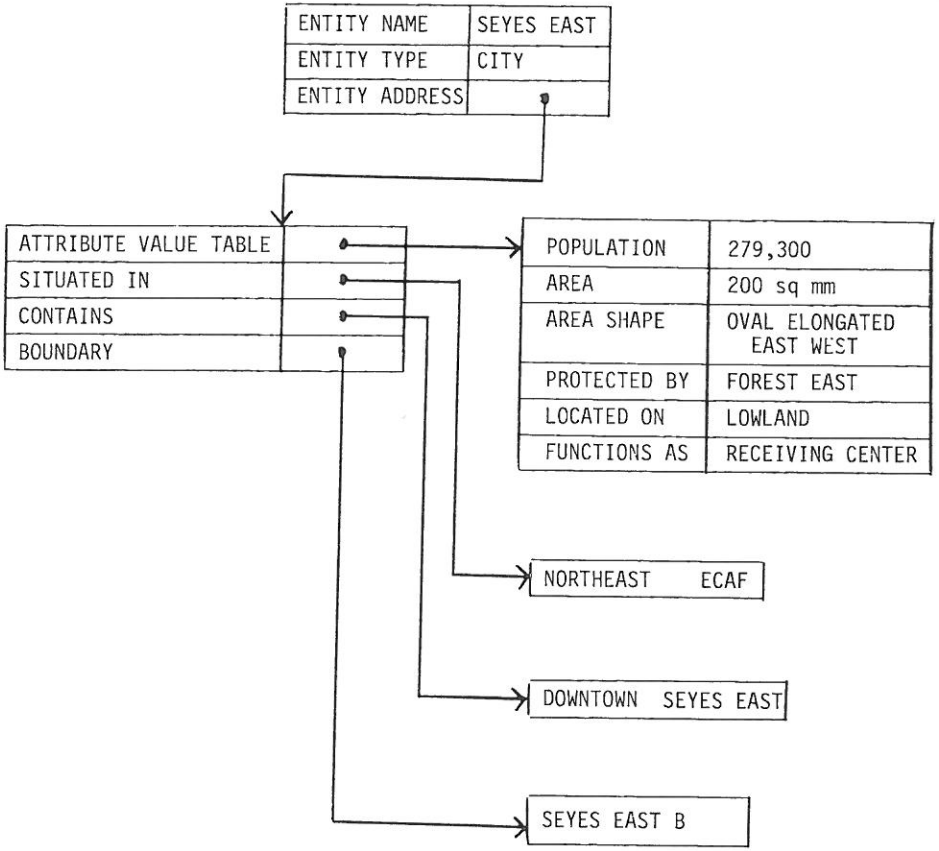


Figure 7 illustrates the logical form of the spatial data structure for Seyes East

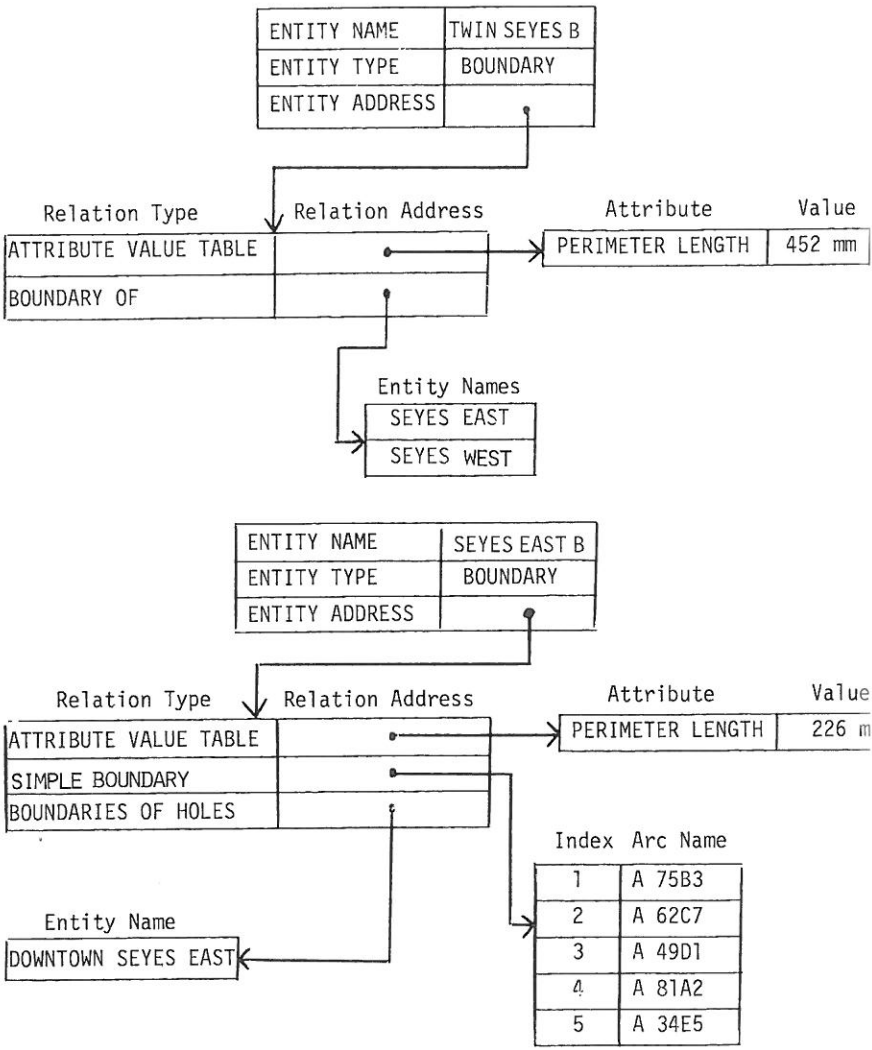


Figure 8 shows the logical structure for the boundary of the disconnected region called Twin Seyes in the Ecaf example.

stored in the logical form shown in Figure 8 or be stored by having each arc point to its next arc left and next arc right as in Tomlinson's CGIS system. In this latter case, the relation type "Simple Boundary" points to any arc in the sequence of 5 with a designation as to whether the region is to the arc's left or right.

II.1 Topologic Consistency

One of the important and non-trivial operations that must be performed on any vector format spatial data is topologic consistency. Topologic consistency demands that:

1) every point in the plane be in some region bounded by a sequence of arcs in the data base; and
2) there are no extraneous arcs in the data base.

There are two ways of performing consistency checks. In the first way, all entities must be searched for all boundaries. In each simple boundary, it must be verified that in its circularly ordered list of arc names, one end point of every arc must match one end point of the successor arc. As each arc in succession is being checked, reference can be made to the (arc, region left, region right) relation to verify that the current region being circled indeed appears to the arc's right when traversing the arc in the direction of the ordered arc sequence bounding the region. Also, each arc must have some region which references it and is to its left and must have some region which references it and is to its right.

The second way of topologic consistency checking requires that a list of arc end points be prepared. In this list, each end point, called a node, must point to all the arcs having it for one of its end points.

The consistency checking is accomplished by searching through all nodes. For each node begin with an arbitrary associated arc. Then circle the node by jumping from arc to arc. Walking along the arc in a direction towards the node, determine who the region on its left is. Then search the list of remaining arcs to determine which arc has that region on its right (again, taking the sense of direction to be that of walking toward the node). Repeat the process of jumping from arc to arc until the first arc selected is returned to. There is a topologic inconsistency if we cannot return back to the initial arc. And if all arcs have not been touched, then there is either garbage arcs or possible errors. Since each arc has two ends, each arc will be accessed twice in this scheme too.

III. RASTER FORMAT SPATIAL DATA

We assume that the raster data is compressed into a run-length variant known as Y partitions (Merrill, 1973). In this data structure, for each connected region, we must store the first row r_1 and the last row r_N having some pixel in the region. Then for each row r, $r_1 \leq r \leq r_N$, we must maintain an ordered list l(r). Each element of l(r) is a maximal horizontal strip which is contained in the given region. The list l(r) has as elements column pairs of the form (b,e), where b designates the beginning pixel and e designates the ending pixel of an interval in the region. Thus, if $(b,e) \in l(r)$, then $\{(r,b), (r,b+1),\ldots,(r,e)\}$ must be contained in the region, and (r,b-1) and (r,e+1) do not lie in the region.

Figure 9 illustrates Seyes East again, but this time using the raster format data representation. Notice that

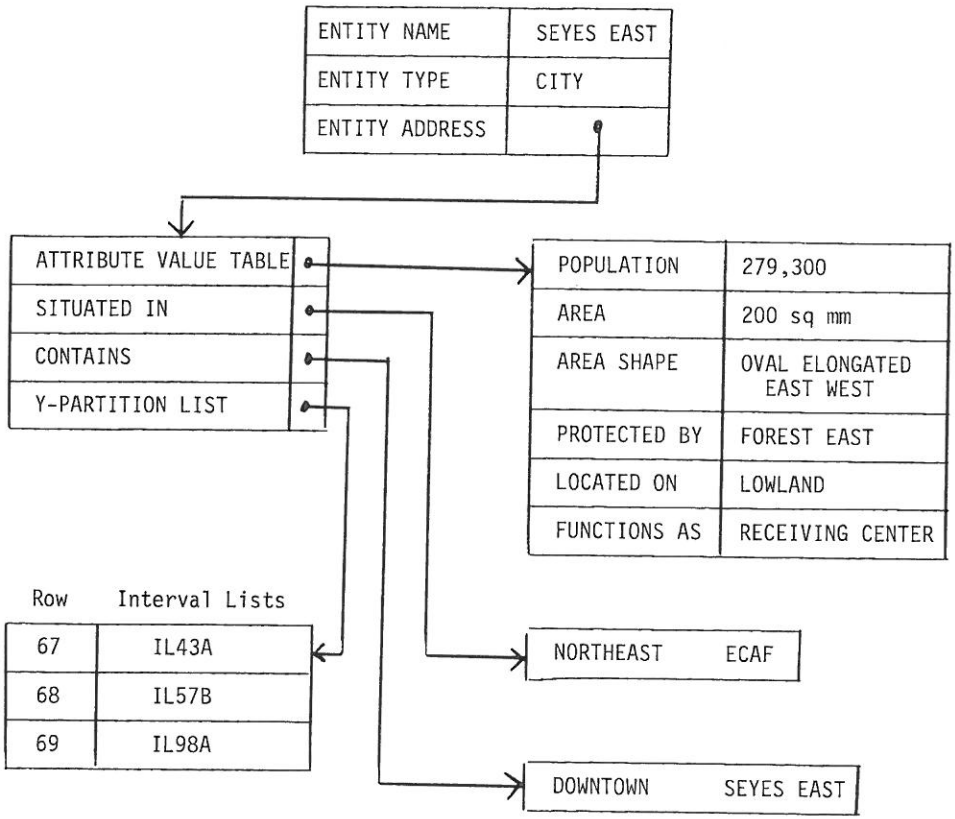


Figure 9 illustrates the logical form of the spatial data structure for Seyes East when the data format is raster

except for the replacement of boundary by partition list, little has changed compared to Figure 7. Figure 10 illustrates the logical structure of a Y-partition list. It is shown as a binary relation having three (row index, interval lists) pairs. Each interval list is a spatial data structure itself containing its intervals as a binary relation of ordered pairs whose first component is beginning pixel and whose last component is ending pixel of an interval. In its attribute value table is its associated region name. Finding all the pixels in a given region is done by going through all the interval lists in the Y-partition of the region. Each interval list specifies pixels contained in the region.

Another convenient data structure for raster format data is a master set of intervals relation shown in Figure 11. This relation stores for each row all the interval lists in the row. The master set of intervals relation can be created from the Y-partitions in the following way. Search through all regions and their Y-partitions. For each Y-partition pick up the interval lists and their corresponding row indexes. Simply add the intervals in the interval list to the set of intervals in the master on the corresponding row.

Finding what region a pixel (r,c) belongs to is done by consulting the master list under row r and searching all the associated intervals to find that one containing c. The attribute value table then gives the region name.

Verifying topologic consistency amounts to verifying that for every row r of the master list, the associated set of intervals constitutes a partition of the pixels in row r. Pixels which are missing from all intervals in a row or which appear in more than one interval are errors.

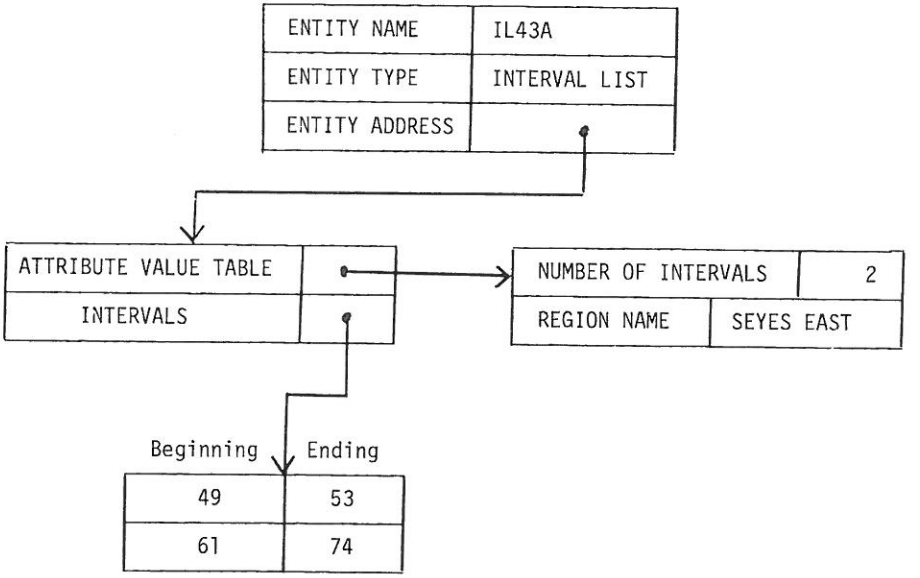


Figure 10 illustrates the logical structure of the interval lists

Row	Interval List
1	IL 10Z
1	IL 15A
1	IL 9B
2	IL 17C
.	.
.	.
.	.
67	IL 43A
.	.
.	.
.	.
512	IL 77D

Figure 11 illustrates the master interval list relation data structure

IV. RASTER FORMAT VS. VECTOR FORMAT

Raster format data has an advantage over vector format data in that the algorithms to determine in what region a pixel lies,to determine the boundary of a given region or to do polygon overlay, are simpler algorithms than the corresponding algorithms for vector format data. However, it is not necessarily the case that the simpler algorithm for raster format data will execute faster than the corresponding algorithm for vector format data because raster data typically has a much higher data volume. Which algorithm will execute quicker depends on the characteristics of the data set being operated on. The thesis of this chapter is that the data format question and its associated algorithm should not be the worry of a user of a spatial information system. Rather it should be the worry of the system itself.

This point of view which admits the possibility of data sets executing quicker in vector format is possible because of the recent advances made in the field of computational geometry. This research has shown that the typical brute force searching algorithms for vector format data are not the most efficient. More efficient algorithms exist. Each finds some way to use the order in a data point set to increase the search efficiency.

A bibliography of papers discussing and describing these fast geometric search algorithms for vector format data is given at the end of this chapter. Because a complete review of these algorithms would be quite lengthy, in the next few paragraphs we give the reader a feel for them by describing

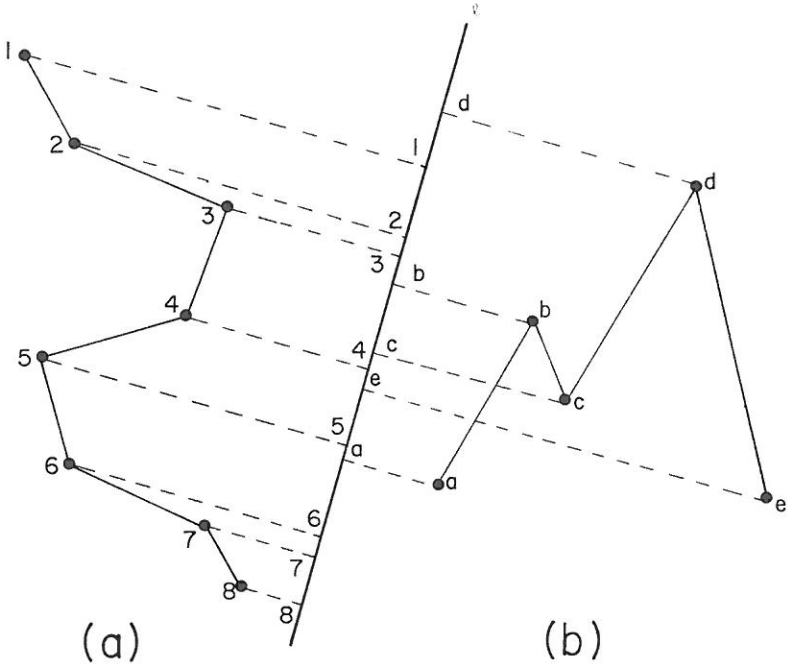
only one of them, an algorithm to solve the point in region problem published by Lee and Preparata (1977).

The point in region problem is given a point, determine which region it is in. The worst algorithm to use in solving this problem is the simplest one: search through all regions and determine the first region in which the given point lies. The algorithm described by Preparata and Lee is based on organizing the line segments of the region boundaries into monotone chains.

A chain of line segments is said to be monotone with respect to a line ℓ if the order of the verticies of the chain and the order of the vertices of the chain as projected onto line ℓ are identical. Figure 12 illustrates an example of a monotone chain and a non-monotone chain.

A quick algorithm exists for determining on which side of a monotone chain (with respect to the line ℓ) a point lies. Take the projection of the point to the line ℓ. The monotonicity of the chain implies that each interval on the projected line will correspond to exactly one line segment of the chain. Determine that line segment containing the projected point by a binary search and solve the problem on which side of the line segment does the point lie. The point lies on the same side of the chain as this side of the line segment.

Preparata and Lee decompose the set of boundary line segments into a monotone set of chains. In this set each chain is monotone with respect to the same line, each boundary line segment belongs to at least one chain, and the vertices of a chain C which are not in a chain C' all lie on the same side of C'. Figure 13 illustrates a set of regions decomposed



Monotone with respect to ℓ Not monotone with respect to ℓ

Figure 12 illustrates a monotone and non-monotone chain with respect to the line ℓ

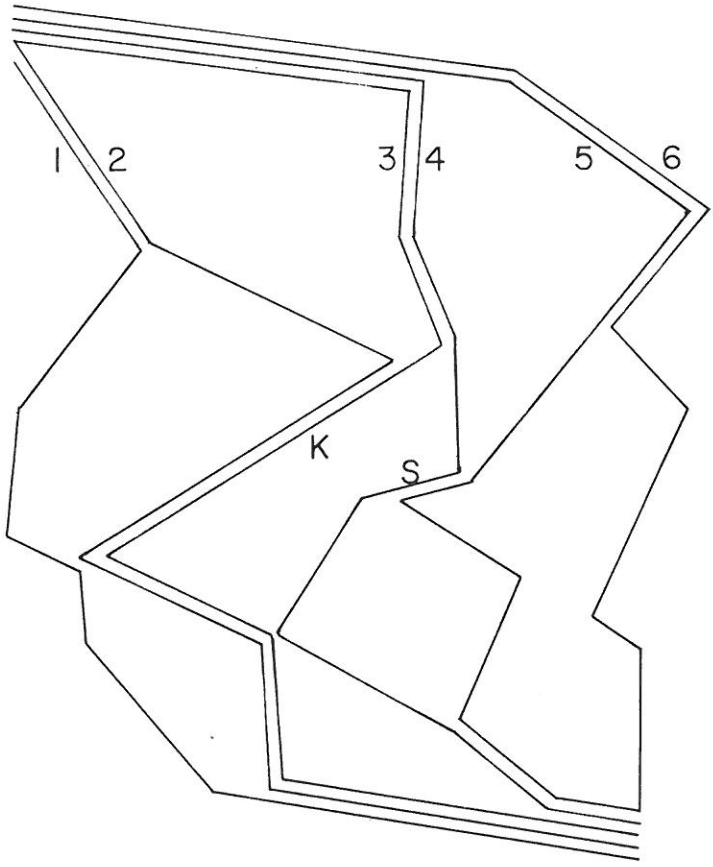


Figure 13 illustrates a set of regions decomposed into 6 chains monotone with respect to the y-axis.

into 6 chains all monotone with respect to the y-axis. Notice that because the chains are not allowed to cross one another (although they are allowed to share line segments), the chains can be simply ordered in a left to right fashion.

To determine in what region a point lies, do a binary search on the chain set. Begin with a chain in the middle and determine whether the point is to the left or right of the middlechain. Continue the binary search until it is determined that the point lies between two successive chains. Then take the left chain and obtain the line segment of the chain having a point with same y-coordinate as the given point. If the line segment is directed upward, the desired region is on the line segment's right. Otherwise it is on its left.

Figure 14 illustrates the binary nature of the search tree. If there are K chains, the binary search will search through $\log_2 K$ chains to find the two chains the given point is between. If there are N vertices per chain, each side of chain search will take $\log_2 N$ operations. Thus, the number of operations required to do the point in region problem is $(\log_2 K)(\log_2 N)$.

Preparata and Lee also point out that even though line segments can be shared between chains, in the binary search of chains, each line segment need be accessed only once. If the test for chain B is below the test for chain A in the tree of figure 14 then any line segment on chain A which is also on chain B will only appear in the test for chain A. For example, line segment S is shared on chains 4 and 5. Since chain 5 is above chain 4 on the search tree line segment S only need be tested on chain 5. Hence for the point shown in Figure 14, it lies to the right of chain 3 since it lies to the right of

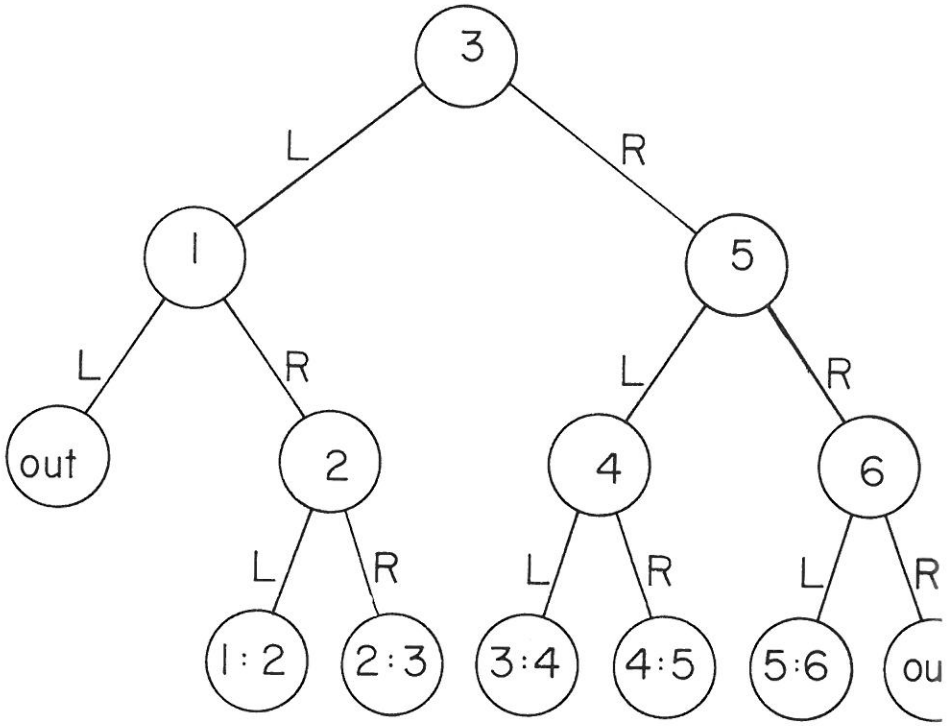


Figure 14 illustrates a binary search tree which can be used to determine between which pairs of successive chains in figure 13 a given point lies. The notation x:y means to the right of chain x and to the left of chain y.

line segment K of chain 3. And of all chains that line segment K participates in, chain 3 is the rightmost one. According to the tree, we must next test to determine on which side of chain 5 the point lies. We find that it lies to the left of chain 5 since it lies to the left of line segment S. Of all chains that line segment S participates in, chain 4 is the leftmost one. Hence the given point lies between chains 3 and 4. It is to the right of line segment K and to the left of line segment S. This information is sufficient to determine the region.

Figure 15 illustrates how the spatial data structure can store the data required for the Preparata and Lee algorithm. For the purpose of point in region problems it has two relations. The first relation is a unary relation consisting of the set of chains, each of which is a spatial data structure. The second relation is a sept-ary relation of the line segments. The components of the 7-tuples in this relation are line segment name, beginning point, ending point, region left, region right, chain left, and chain right. Chain left is the leftmost chain the line segment participates in and chain right is the rightmost chain the line segment participates in. Figure 16 illustrates the spatial data structure for a monotone chain.

V. CONCLUSION

We have described a data structure that naturally handles both raster and vector format spatial data. We have referenced the many fast vector format algorithms recently developed and have described one in detail, illustrating how our spatial data structure can fulfill its requirements. We

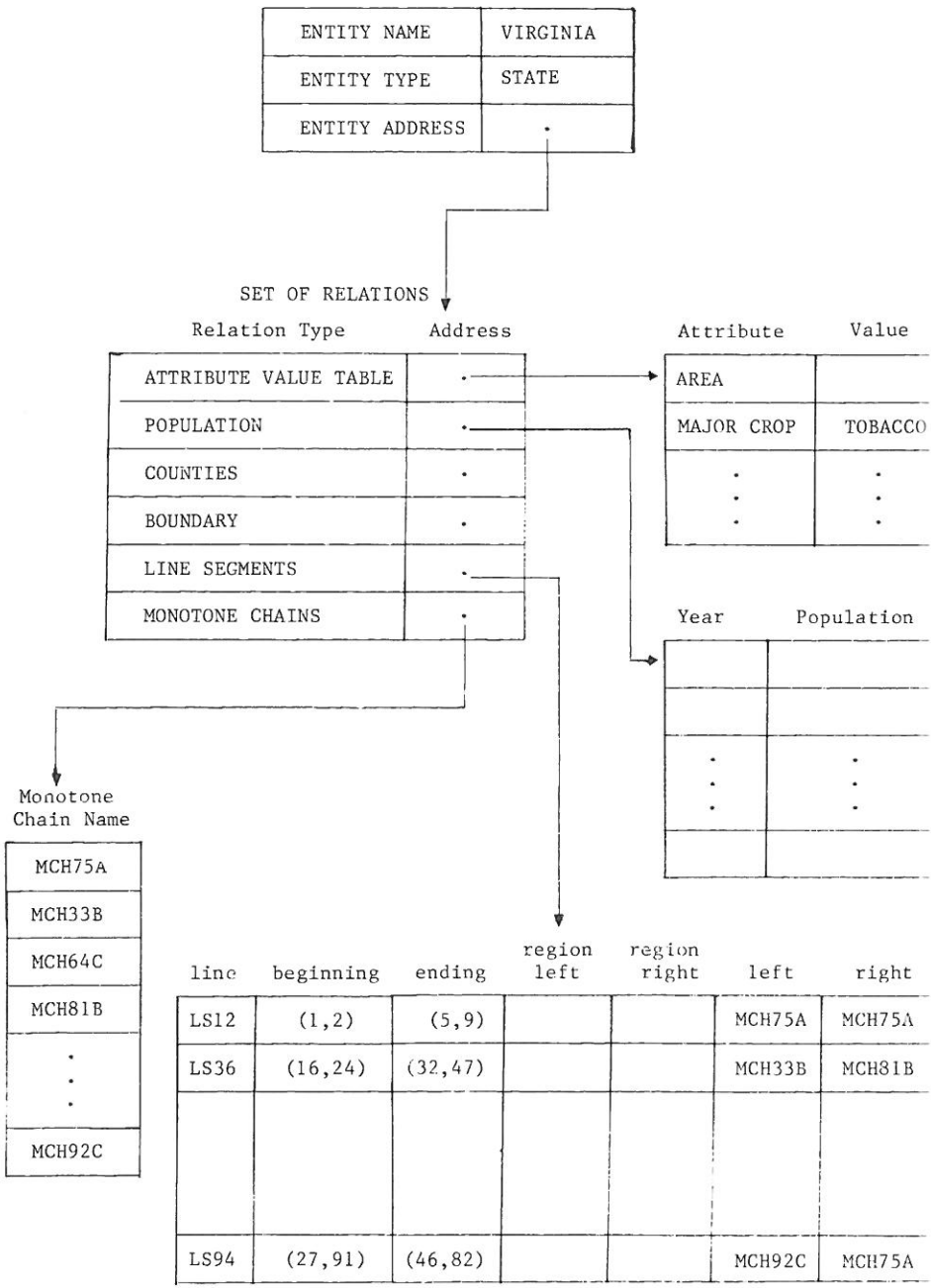


Figure 15 illustrate the spatial date structure for a region using the monotone chain algorithm for determining the region any point is

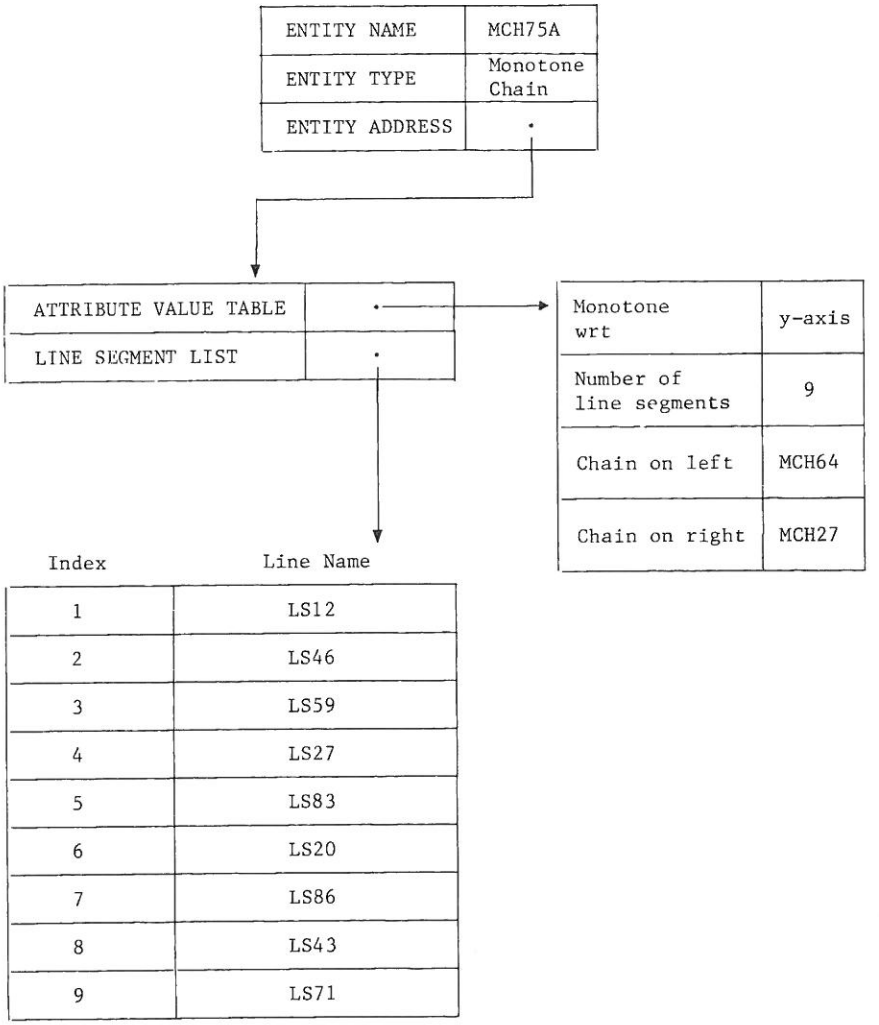


Figure 16 illustrates the spatial data structure for a monotone chain

have argued that the choice of data format should not be the user's worry, but should be left up to the geographic information system which can optimize the choice of data structure based on the statistics of each particular data set and on the distribution of queries mode of that data set. We have left for another paper a description of how the geographic information can optimize computer resources by appropriate choice of data format for each particular data set.

REFERENCES

Akl, S. G. and Toussaint, G. T. (1979). "Efficient Convex Hull Algorithms for Pattern Recognition Applications," 4th IJCPR, Chicago, August 1979, 483-487.

Bentley, J. L. and Stomat, D. F. (1975). "Analysis of Range Searches in Quad Trees," Information Processing Letters, Vol. 3, No. 6, July 1975, 170-173.

Lee, D. T. and Preparata, F. P., (1977). "Location of a Point in a Planar Subdivision and Its Applications," SIAM Journal of Computing, Vol. 6, No. 3, September 1977, 594-606.

Lee, D.T. and Preparata, F. P., (1979). "An Optimal Algorithm for Finding the Kernel of a Polygon," JACM, Vol. 26, No. 3, July 1979, 415-421.

Lee, D. T. and Wong, C. K. (1977). "Multidimensional Binary Search Trees and Balanced Quad Trees," Acta Informatica, Vol. 9, 23-29.

Lee, D. T. and Preparata, F. P. (1978). "The All Nearest-Neighbor Problem for Convex Polygons," Information Processing Letters, Vol 7, No. 4, June 1978, 189-192.

Little, J. J. and Peucker, T. K. (1979). "A Recursive Procedure for Finding the Intersection of Two Digital Curves," CGIP, Vol. 10, 159-171.

Merrill, R. D. (1973). "Representation of Contours and Regions for Efficient Computer Search," CACM, Vol. 16, No. 2, February 1973, 69-82.

Nagy, G. (1978). "Advances in Computational Geometry," Proceedings of the Twenty-first Midwest Symposium on Circuits and Systems, August 14-15, 1978.

Preparata, F. P. (1979). "An Optimal Real-Time Algorithm for Planar Convex Hulls," CACM, Vol. 22, No. 7, July 1979, 402-405.

Shamos, M. I. and Hoey, D. J. (1975). "Closest-point Problems," Proceedings of the Sixteenth Symposium on Foundations of Computer Science, October 1975.

Shamos, M. I. and Bentley, J. L. (1977). "Optimal Algorithms for Structuring Geographic Data," An Advanced Study Symposium on Topological Data Structures for Geographic Information Systems, October 1977.

Shamos, M. I. (1975). "Geometric Complexity," Proceedings of the 7th Annual Symposium in Theory of Computing, May 1975.

Yang, C. C. and Lee, D. T. (1979). "A Note on All Nearest-Neighbor Problems for Convex Polygons," Information Processing Letters, Vol. 8, No. 4, April 1979, 193-194.