Computing the Topological Relationship of Complex Regions

Markus Schneider[⋆]

University of Florida
Department of Computer & Information Science & Engineering
Gainesville, FL 32611, USA
`mschneid@cise.ufl.edu`

Abstract. *Topological predicates* between spatial objects have always been a main area of research on spatial data handling, reasoning, and query languages. The focus of research has definitely been on the design of and reasoning with these predicates, whereas implementation issues have been largely neglected. Besides, design efforts have been restricted to simplified abstractions of spatial objects like single points, continuous lines, and simple regions. In this paper, we present a general algorithm which is based on the well known plane-sweep paradigm and determines the topological relationship between two given *complex* regions.

1 Introduction

For a long time, *topological predicates* on spatial objects like *overlap*, *disjoint*, or *inside* have been a focus of research on spatial data handling and reasoning in a number of disciplines like artificial intelligence, linguistics, robotics, and cognitive science. They characterize the relative position between two (or more) objects in space. In particular, they support the design of suitable query languages for spatial data retrieval and analysis in geographical information systems (GIS) and spatial databases. Two important approaches are the 9-*intersection model* [4] based on point set topology as well as the *RCC model* [3] based on spatial logic. Whereas the predicates in these two models operate on simplified abstractions of spatial objects like simple regions, we are interested in the design and implementation of topological predicates for *complex* regions, which may consist of several components (faces) and which may have holes. Implementation issues for these predicates, regardless of whether they operate on simple or complex geometries, have so far been rather neglected. Hence, the goal of this paper is to show how topological predicates for complex regions can be efficiently implemented on the basis of the well-known plane-sweep paradigm. This algorithm copes with region objects whose segments are non-interesting or allowed to intersect.

Section 2 discusses related work. Section 3 describes a database-compatible data structure for complex regions. In Section 4 we present the algorithm *TopRel*

[⋆] This work was partially supported by the National Science Foundation under grant number NSF-CAREER-IIS-0347574.

which determines the topological relationship for a given pair of complex regions. Finally, Section 5 draws some conclusions.

2 Related Work

Spatial data types (see [7] for a survey) like *point*, *line*, or *region* have turned out to provide fundamental abstractions for modeling the structure of geometric entities, their relationships, properties, and operations. Whereas in older models the geometric structure of spatial data has been restricted (only simple regions, continuous lines, single points), in the meantime a few models also allow complex spatial objects which may consist of several disjoint components. Additionally, a region object may have holes. The reasons for defining complex geometric structures are the maintenance of closure properties for geometric operations (simple spatial data types are not closed under these opreations) and application-specific requirements (e.g., to be able to model Italy with its offshore islands and the Vatican as a hole).

A well known, mathematical model for characterizing topological relationships between simple regions is the 9-*intersection model* [4]. Eight well known topological predicates have been identified for two simple regions. They are named *disjoint*, *meet*, *overlap*, *equal*, *inside*, *contains*, *covers*, *coveredBy*. All predicates are mutually exclusive and cover all topological situations. This paper is based on a generalization of these eight predicates to *complex* regions [8]. The generalized predicates inherit the properties of the simple predicates but take into account multi-component regions possibly containing holes. Informally, two complex regions A and B are *disjoint* if they do not share any points. They *meet* if their boundaries share points and their interiors are disjoint. They are *equal* if both their boundaries and their interiors coincide. A is *inside* B (B *contains* A) if A is a proper subset of B and if their boundaries do not touch. A *is covered by* B (B *covers* A) if A is a proper subset of B and if their boundaries touch. Otherwise, A and B *overlap.*

The algorithm *TopRel* is based on the plane-sweep paradigm and on the ROSE algebra [6] which has the requirement of non-intersecting spatial objects. This is ensured by the so-called *realm* concept [5] providing a discrete, numerically robust, and consistent geometric basis upon which all spatial objects are constructed. Hence, different spatial objects are only allowed to have line segments meeting in end points or being disjoint. This makes an understanding and implementation much easier if we compare it, e.g., to the classical Bentley-Ottmann algorithm [2]. But our algorithm also covers the general case of intersecting segments by reducing it to the ROSE case.

3 Data Structures

For reasons of efficiency and performance, our region data structure does not use main memory pointer structures but a small number of memory blocks that can be transferred efficiently between secondary and main memory. It is represented by a *root record* which contains some fixed size components and one or more components which are references to arrays. Arrays are used to represent the varying size components of the data structure and are allocated to the required size. All internal pointers are expressed as array indices. Of course, the data structure supports plane-sweep algorithms very well.

In the following, we represent an array as a sequence $\langle \cdot \rangle$ with additional, direct access to its elements. A *subarray* shall denote a specific subrange (subsequence) within an array. Coordinates are given as rational numbers of a certain precision[1]. A value of type *point* is represented by a record $p = (x, y)$ where x and y are corrdinates. We assume the usual lexicographical order on points.

Conceptually, complex regions can be considered from a "structured" and a "flat" view. The structured view defines an object of type *region* as a set of edge-disjoint faces. A face is a simple polygon possibly containing a set of edge-disjoint holes. A hole is a simple polygon. Two spatial entities are edge-disjoint if they share perhaps single boundary points but not boundary segments. The flat view defines an object of type *region* as a collection of line segments which altogether preserve the constraints and properties of the structured view. For a formal definition see [6].

The implementation of a *region* object is given as an *ordered* sequence (array) of *halfsegments*. The idea of halfsegments is to store each segment twice. Let $S = \mathit{point} \times \mathit{point}$ be the set of segments. We normalize S by the requirement that $\forall s \in S : s = (p, q) \Rightarrow p < q$. This enables us to speak of a *left* and a *right end point* of a segment. We define $H = \{(s, d) \mid s \in S, d \in \{\mathit{left}, \mathit{right}\}\}$ as the set of *halfsegments* where flag d emphasizes one of the segments' end points, which is called the *dominating point* of h. If $d = \mathit{left}$, the left (smaller) end point of s is the dominating point of h, and h is called *left halfsegment*. Otherwise, the right end point of s is the dominating point of h, and h is called *right halfsegment*. Hence, each segment s is mapped to two halfsegments (s, left) and (s, right). Let dp be the function which yields the dominating point of a halfsegment. For two distinct halfsegments h_1 and h_2 with a common end point p, let α be the enclosed angle such that $0° < \alpha \leq 180°$ (an overlapping of h_1 and h_2 is excluded by the prohibition of self-intersections of *region* objects). Let a predicate *rot* be defined as follows: $\mathit{rot}(h_1, h_2)$ is *true* if and only if h_1 can be rotated around p through α to overlap h_2 in counterclockwise direction. We can now define a complete order on halfsegments which is basically the (x, y)-lexicographical order on dominating points. For two halfsegments $h_1 = (s_1, d_1)$ and $h_2 = (s_2, d_2)$ we obtain:

$$\begin{aligned} h_1 < h_2 \Leftrightarrow\ & \mathit{dp}(h_1) < \mathit{dp}(h_2) \lor \\ & (\mathit{dp}(h_1) = \mathit{dp}(h_2) \land \\ & ((d_1 = \mathit{right} \ \land\ d_2 = \mathit{left}) \ \lor\ (d_1 = d_2 \ \land\ \mathit{rot}(h_1, h_2)))) \end{aligned}$$

An example of the order on halfsegments is given in Figure 1. An ordered sequence of halfsegments is given as an array $\langle (h_1, a_1, \mathit{nsic}_1), \ldots, (h_m, a_m, \mathit{nsic}_m) \rangle$

[1] To ensure consistency among operations, we currently plan to replace the fixed size rationals by varying length rationals and explore the costs that this could have regarding the performance.

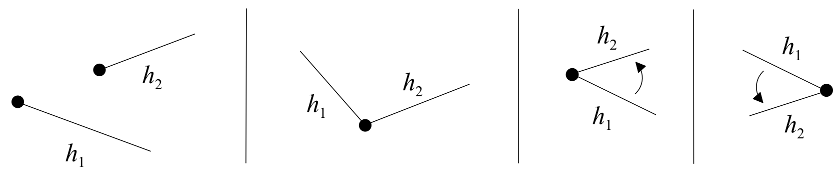

Fig. 1. Examples of the order on halfsegments: $h_1 < h_2$.

of m halfsegments $h_i \in H$ with $h_i < h_j$ for all $1 \leq i < j \leq m$. Each *left* halfsegment $h_i = (s_i, \mathit{left})$ has an attached set a_i of *attributes*. Attributes contain auxiliary information that is needed by geometric algorithms. Each left halfsegment h_i also has an additional pointer nsic_i (*n*ext *s*egment *i*n *c*ycle) indicating the array index of the next segment in the cycle it belongs to[2]. We assume clockwise order for the traversal of the segments of outer and hole cycles. An example is given in Figure 2. Those *left* halfsegments of a region object r carry an associated attribute *InsideAbove* where the interior of r lies above or left of their respective (normalized) segments.

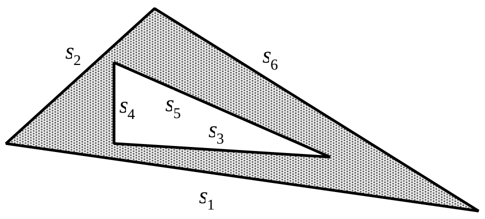

Fig. 2. Example of the halfsegment sequence of a region. If $h_i^l = (s_i, \mathit{left})$ and $h_i^r = (s_i, \mathit{right})$ denote the left and right halfsegments belonging to the segments s_i for $1 \leq i \leq 6$, then $\mathit{halfsegments} = \langle (h_1^l, \{I\}, 2), (h_2^l, \varnothing, 8), (h_3^l, \varnothing, 4), (h_4^l, \{I\}, 6), (h_4^r, \varnothing, 0), (h_5^l, \{I\}, 0), (h_2^r, \varnothing, 0), (h_6^l, \varnothing, 0), (h_5^r, \varnothing, 0), (h_3^r, \varnothing, 0), (h_6^r, \varnothing, 0), (h_1^r, \varnothing, 0) \rangle$. For right halfsegments, the second and third component of a triple are always the empty set and zero, so that they are omitted in our implementation. For left halfsegments, I stands for the attribute *InsideAbove* and a zero in the third component indicates the end of a segment list of a cycle. Index counting in an array is assumed to start at 1.

Since inserting a halfsegment at an arbitrary position needs $O(m)$ time, in our implementation we use an AVL-tree embedded into an array whose elements are linked in sequence order. An insertion then requires $O(\log m)$ time.

4 The Algorithm *TopRel*

The description of implementation strategies for topological predicates both on simple and complex spatial data types has been rather neglected in the literature.

[2] It is unnecessary to attach attributes and pointer information to *right* half segments since their existence in an ordered halfsegment sequence only indicates to plane sweep algorithms that the respective segment has to be removed from the *sweep line status*.

This fact is surprising since such an implementation is required especially in GIS and spatial databases for the realization of spatial joins and spatial selections.

Let T be the set of eight generalized topological predicates between two complex regions. We do not consider the implementation of a specific predicate $p \in T$ with $p : \mathit{region} \times \mathit{region} \rightarrow \mathit{bool}$ (leading to eight algorithms) but a single function $\mathit{TopRel} : \mathit{region} \times \mathit{region} \rightarrow T$ which determines the topological relationship for a given spatial configuration of two complex regions. An adaptation to an individual implementation of topological predicates can be easily performed since $p(r_1, r_2) \Leftrightarrow (\mathit{TopRel}(r_1, r_2) = p)$.

A specialty of our implementation is that it is able to process two regions with either possibly *intersecting* line segments (the general case) or with *exclusively non-intersecting* line segments (the ROSE case). The reason for our interest in the second case is that we integrate the algorithm *TopRel* into our ROSE algebra implementation [6], which so far only offers a rudimentary treatment of topological predicates. We will indicate the necessary modifications.

4.1 Geometric Preliminaries

The description of the algorithm for *TopRel* requires three concepts: (i) the *plane sweep* paradigm, (ii) the *parallel traversal* of two complex regions, and (iii) the concept of *overlap numbers*. We will not describe these well-known concepts in general but focus on some specialties of these concepts in our setting.

Plane Sweep The well-known plane-sweep works as follows: A vertical *sweep line* sweeping the plane from left to right stops at special points called *event points* which are generally stored in a queue called *event point schedule*. The event point schedule must allow one to insert new event points discovered during processing; these are normally the initially unknown intersections of line segments. The state of the intersection of the sweep line with the geometric structure being swept at the current sweep line position is recorded in vertical order in a data structure called *sweep line status*. Whenever the sweep line reaches an event point, the sweep line status is updated. Event points which are passed by the sweep line are removed from the event point schedule.

Whereas a general plane sweep requires an efficient, fully dynamic data structure to represent the event point schedule and often an initial sorting step to produce the sequence of event points in (x, y)-lexicographical order, in the ROSE case an explicit event point schedule is implicitly given by the two ordered half-segment sequences of the operand objects. The event point schedule is even static since no two segments intersect within their interiors. There is even no initial sorting necessary, because the plane sweep order of segments is the base representation of *region* objects anyway.

For the general case, to preserve these benefits, it is our strategy to reduce it to the ROSE case. In case that two segments from different *region* objects intersect, partially coincide, or touch each other within the interior of a segment, we pursue a splitting strategy which we call *realmification*. If the segments intersect, they are split at their common intersection point so that each of them is

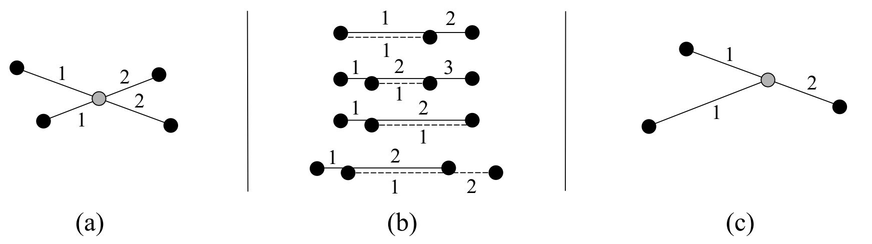

Fig. 3. Splitting of two intersecting segments (a), two partially coinciding segments (without symmetric counterparts) (b), and a segment whose interior is touched by another segment (c). Digits indicate part numbers of segments after splitting.

replaced by two segments (i.e., four halfsegments) (Figure 3a). If two segments partially coincide, they are split each time the endpoint of one segment lies inside the other segment. Depending on the topological situations, which can be described by Allen's thirteen basic relations on intervals [1], each of the two segments either remains unchanged or is replaced by up to three segments (i.e., six halfsegments). From the thirteen possible relations, eight relations (four pairs of symmetric relations) are of interest here (Figure 3b). If an endpoint of one segment touches the interior of the other segment, the latter segment is split and replaced by two segments (i.e., four halfsegments) (Figure 3c).

For each *region* object, besides a static part, the halfsegments obtained by the realmification are stored in an additional dynamic part of the halfsegment sequence. This part is also organized as an AVL tree which is embedded in an array and whose elements are linked in sequence order. Assuming that k splitting points are detected during the plane sweep, we need additional $O(k)$ space, and to insert them requires $O(k \log k)$ time.

Parallel Object Traversal During the planesweep we traverse the halfsegment sequences of both region operands. We know that each sequence is already in halfsegment sequence order. Hence, it is sufficient to check the current halfsegments of both sequences and to take the lower one with respect to halfsegment sequence order for further computation.

The order definition on halfsegments given in Section 3 is referring to the halfsegment sequence of *individual* regions. In the ROSE case, this order definition is also sufficient for the parallel object traversal of two regions since any two halfsegments stemming from different regions may not intersect each other or partially coincide. For the general case, however, Figure 4 shows additional situations. The first three constellations are covered by the existing order definition on halfsegments. The fourth constellation requires an extension of the definition. For two halfsegments $h_1 = (s_1, d_1)$ and $h_2 = (s_2, d_2)$ we define:

$$\begin{aligned} h_1 < h_2 \Leftrightarrow\ & dp(h_1) < dp(h_2)\ \vee\ (dp(h_1) = dp(h_2)\ \wedge \\ & ((d_1 = \mathit{right}\ \wedge\ d_2 = \mathit{left})\ \vee\ (d_1 = d_2\ \wedge\ \mathit{rot}(h_1, h_2))\ \vee \\ & (d_1 = d_2\ \wedge\ \mathit{collinear}(h_1, h_2)\ \wedge\ \mathit{len}(h_1) < \mathit{len}(h_2)))) \end{aligned}$$

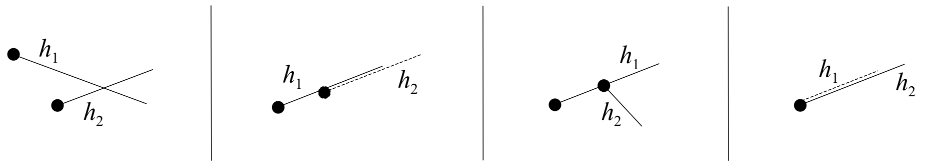

Fig. 4. Some examples of additional constellations of two halfsegments from different regions in the general case for $h_1 < h_2$.

The function *collinear* checks whether h_1 and h_2 are located on the same infinite line, and the function *len* computes the length of a halfsegment.

Overlap Numbers Overlapping of region parts is important for determining the topological relationship between two complex regions. For this purpose we introduce the concept of *overlap numbers*. A point obtains the overlap number 2 if it is covered by (or part of) two *region* objects. This means that for two intersecting simple polygons the area outside of both polygons gets overlap number 0, the intersecting area gets overlap number 2, and the other areas get overlap number 1. Since a segment of a region separates space into two parts, an inner and an exterior one, each (half)segment is associated with a pair (m/n) of overlap numbers, a lower (or right) one m and an upper (or left) one n. The lower (upper) overlap number indicates the number of overlapping *region* objects below (above) the segment. In this way, we obtain a *segment classification* of two *region* objects and speak of (m/n)-segments. Obviously, $m, n \leq 2$ holds. Of the nine possible combinations only seven describe valid segment classes. This is because a (0/0)-segment contradicts the definition of a *region* object, since then at least one of both regions would have two holes or an outer cycle and a hole with a common border. Similarly, (2/2)-segments cannot exist, since then at least one of the two regions would have a segment which is common to two outer cycles of the object. Hence, possible (m/n)-segments are (0/1)-, (0/2)-, (1/0)-, (1/1)-, (1/2)-, (2/0)-, and (2/1)-segments. An example is given in Figure 5.

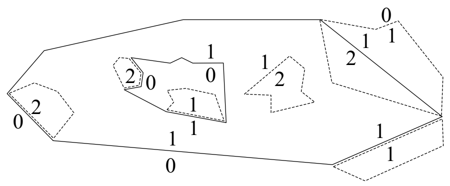

Fig. 5. Example of the segment classification of two *region* objects.

4.2 The Algorithm *TopRel*

Using the three aforementioned concepts, we perform a characterization of the eight topological predicates that is based on the segment classification of both argument objects and is a unique translation of the respective predicate specifications shown in [8]. The characterizations of all predicates together are also complete since their specifications in [8] are complete. The question is here: To which segment class(es) must the segments of both objects belong so that a given topological predicate is fulfilled? The computation of the segment classes itself is performed "on the fly" during the plane sweep. Let F and G be two *region* objects, and let $H(F)$ and $H(G)$ be functions which yield the elements of the halfsegment sequences of F and G, respectively, as sets. For a halfsegment h let (m_h, n_h) be its associated pair of overlap numbers. A function *point_in_common* checks whether two halfsegments stemming from two different *region* objects share at least a common point. In the ROSE case, this can only be a common end point. In the general case, this can in addition also indicate a touching point (i.e., an end point of one halfsegment touches the interior of another halfsegment), an intersection point, or a partial overlapping of two halfsegments. We obtain the following characterization for the predicates:

$$\begin{aligned}
\mathit{disjoint}(F,G) \Leftrightarrow\ & \forall\, h \in H(F) \cup H(G) : (m_h, n_h) \in \{(0/1), (1/0)\} \wedge \\
& \forall\, h \in H(F)\ \forall\, g \in H(G) : \neg \mathit{point\_in\_common}(h, g) \\
\mathit{meet}(F,G) \Leftrightarrow\ & \forall\, h \in H(F) \cup H(G) : (m_h, n_h) \in \{(0/1), (1/0), (1/1)\} \wedge \\
& (\exists\, h \in H(F) \cup H(G) : (m_h, n_h) \in \{(1/1)\} \vee \\
& \exists\, h \in H(F)\ \exists\, g \in H(G) : \mathit{point\_in\_common}(h, g)) \\
\mathit{inside}(F,G) \Leftrightarrow\ & \forall\, h \in H(F) : (m_h, n_h) \in \{(1/2), (2/1)\} \wedge \\
& \forall\, g \in H(G) : (m_g, n_g) \in \{(1/0), (0/1)\} \wedge \\
& \forall\, h \in H(F)\ \forall\, g \in H(G) : \neg \mathit{point\_in\_common}(h, g) \\
\mathit{contains}(F,G) \Leftrightarrow\ & \mathit{inside}(G, F) \\
\mathit{equal}(F,G) \Leftrightarrow\ & \forall\, h \in H(F) \cup H(G) : (m_h, n_h) \in \{(0/2), (2/0)\} \\
\mathit{coveredBy}(F,G) \Leftrightarrow\ & \forall\, h \in H(F) : (m_h, n_h) \in \{(1/2), (2/1), (0/2), (2/0)\} \wedge \\
& \forall\, g \in H(G) : (m_g, n_g) \in \{(1/0), (0/1), (0/2), (2/0)\} \wedge \\
& \exists\, h \in H(F) : (m_h, n_h) \in \{(1/2), (2/1)\} \wedge \\
& \exists\, g \in H(G) : (m_g, n_g) \in \{(1/0), (0/1)\} \wedge \\
& ((\exists\, h \in H(F) : (m_h, n_h) \in \{(0/2), (2/0)\} \wedge \\
& \exists\, g \in H(G) : (m_g, n_g) \in \{(0/2), (2/0)\}) \vee \\
& \exists\, h \in H(F)\ \exists\, g \in H(G) : \mathit{point\_in\_common}(h, g)) \\
\mathit{covers}(F,G) \Leftrightarrow\ & \mathit{coveredBy}(G, F) \\
\mathit{overlap}(F,G) \Leftrightarrow\ & \exists\, h \in H(F) : (m_h, n_h) \in \{(2/1), (1/2)\} \wedge \\
& \exists\, g \in H(G) : (m_g, n_g) \in \{(2/1), (1/2)\}
\end{aligned}$$

The predicate *disjoint* yields *true* if both objects do not share common areas, common segments, and common points. The *meet* predicate is similar to *disjoint* but, in addition, it requires the existence of at least one (1/1)-segment or a common point. Note that each object cannot exclusively consist of (1/1)-segments so that each object must automatically have (0/1)- or (1/0)-segments. For the *inside* predicate we must especially ensure that no segment of F shares a common point with any segment of G. The evaluation of the *equal* predicate amounts to a comparison of the two halfsegment sequences which have to be identical and hence all to be (0/2)- or (2/0)-segments. The specification of the *coveredBy* predicate is a little more complicated. The reason is that we must ex-

n_s	1	2	1	2	1	2	0	1	2	0	1	2	0	1	2	0	1	0	1
m_s	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	2	2	2	2
n_p	–	–	0	0	0	0	1	1	1	1	1	1	1	1	1	2	2	2	2
m_p	–	–	1	1	2	2	0	0	0	1	1	1	2	2	2	0	0	1	1

Table 1. Possible segment class constellations between two consecutive segments in the sweep line status.

clude the case that all segments from both objects are (0/2)- or (2/0)-segments, which would correspond to equality. The *overlap* predicate requires a test for the existence of a (2/1)- or (1/2)-segment in each object.

The characterizations can be read in both directions. If we are interested in a specific topological predicate, we have to look from left to right and check the respective right side of the predicate's characterization. This corresponds to an explicit implementation of each individual predicate. In our case, where we aim at deriving the topological relationship from a given spatial configuration of two regions, we have to look from right to left. That is, we traverse the two regions in parallel by a plane sweep, simultaneously register the arising segment classes of halfsegments as well as possibly common points, and match the result against the right sides of all predicate characterizations. For the characterization that matches we look on its left side to obtain the name of the predicate.

The algorithm for determining the topological relationship between two complex regions runs in three phases. In the first phase, in O(1) time, we apply a bounding box test as a filter condition to the two *region* argument objects. If the bounding boxes do not intersect, the two regions must be disjoint.

Otherwise, in the second phase, we have to perform a more detailed analysis. For each region operand, we define a vector $v(i, j, k)$ which stores a boolean value for each triple (i, j, k). If region k contains an (i/j)-segment, $v(i, j, k)$ is set to *true* and otherwise to *false*. In this way, v records the segment classes found during the parallel traversal. The parallel object traversal visits the halfsegments of both *region* argument objects according to their halfsegment order and computes for each left halfsegment its segment class. In order to be able to determine the correct segment class, each (half)segment is inserted into the sweep line status according to the y-order of its dominating point and the y-coordinates of the intersection points of the current sweep line with the segments momentarily in the sweep line status. The possible 19 segment class constellations between two consecutive segments in the sweep line status are shown in Table 1. The table shows which segment classes (m_s/n_s) a new segment s to be inserted into the sweep line status can get, given a certain segment class (m_p/n_p) of a predecessor segment p. The first two columns show the special case that at the beginning the sweep line status is empty and the first segment is inserted. This segment can either be shared by both region objects ((0/2)-segment) or stems from one of them ((0/1)-segment). In all these cases $n_p = m_s$ must hold.

If F has l and G m halfsegments, at most $n = l + m$ halfsegments are visited in the ROSE case and $n = l + m + k$ times in the general case (where k is the number of splitting points detected during the plane sweep). Each of these (half)segments are inserted into and removed from the sweep line status. Since at most n elements can be contained in the sweep line status, the worst time complexity of the parallel object traversal and the plane-sweep is $O(n \log n)$.

The third and last phase is the evaluation phase. The vector v contains the found segment classes. By checking its different boolean values, the matching topological relationship can be identified in $O(1)$ time. This is done by comparing with the aforementioned characterizations of the predicates and excluding all other cases, i.e., v must yield false for all kinds of halfsegments not belonging to a predicate's characterization. In total, the time complexity of the algorithm *TopRel* is $O(n \log n)$.

5 Conclusions

In this paper, we have presented an efficient algorithm, called *TopRel*, which computes the topological relationship between two regions. It is general in the sense that (i) it operates on complex regions, (ii) it is applicable both to intersecting as well as exclusively non-intersecting regions, and (iii) it can be easily leveraged to explicitly compute a single topological relationship. This implementation is part of SPAL2D which is a *spatial algebra* under development for two-dimensional applications.

References

1. J. F. Allen. Maintaining Knowledge about Temporal Intervals. *Communications of the ACM*, 26:832–843, 1983.
2. J.L. Bentley and T. Ottmann. Algorithms for Reporting and Counting Geometric Intersections. *IEEE Trans. on Computers*, C-28:643–647, 1979.
3. Z. Cui, A. G. Cohn, and D. A. Randell. Qualitative and Topological Relationships. *3rd Int. Symp. on Advances in Spatial Databases*, LNCS 692, pp. 296–315, 1993.
4. M. J. Egenhofer, A. Frank, and J. P. Jackson. A Topological Data Model for Spatial Databases. *1st Int. Symp. on the Design and Implementation of Large Spatial Databases*, LNCS 409, pp. 271–286. Springer-Verlag, 1989.
5. R. H. Güting and M. Schneider. Realms: A Foundation for Spatial Data Types in Database Systems. *3rd Int. Symp. on Advances in Spatial Databases*, LNCS 692, pp. 14–35. Springer-Verlag, 1993.
6. R. H. Güting and M. Schneider. Realm-Based Spatial Data Types: The ROSE Algebra. *VLDB Journal*, 4:100–143, 1995.
7. M. Schneider. *Spatial Data Types for Database Systems - Finite Resolution Geometry for Geographic Information Systems*, volume LNCS 1288. Springer-Verlag, Berlin Heidelberg, 1997.
8. M. Schneider. A Design of Topological Predicates for Complex Crisp and Fuzzy Regions. *Int. Conf. on Conceptual Modeling*, pp. 103–116, 2001.