

Modeling and Computing Ternary Projective Relations between Regions

Eliseo Clementini and Roland Billen

Abstract—Current spatial database systems offer limited querying capabilities beyond binary topological relations. This paper introduces a model for projective relations between regions to support other qualitative spatial queries. The relations are ternary because they are based on the collinearity invariant of three points under projective geometry. The model is built on a partition of the plane into separate zones that are obtained from projective properties of two reference objects: Then, by considering the empty/nonempty intersections of a primary object with these zones, the model is able to distinguish between 34 different projective relations. Then, the paper proposes original algorithms for computing the relations under the assumption that regions of the plane are stored as vector polygons in a spatial database. These algorithms run in optimal $O(n \log n)$ time.

Index Terms—Spatial databases, spatial queries, projective relations, geographic information systems.



1 INTRODUCTION

A formal geometric definition of spatial relations is needed to build reasoning systems on them and facilitate a standard implementation in spatial database systems. This is what happens for some models of topological relations, like the 9-intersection [12] and the calculus-based method—CBM [6]. These models provide formal definitions for the relations, establish reasoning mechanisms to find new relations from a set of given ones [11] and, as part of the OpenGIS Consortium (OGC) specifications [27] and ISO/TC 211 standard, have been implemented in several commercial geographic information systems (GISs) and spatial database systems.

Topological relations take into account an important part of geometric knowledge and can be used to formulate qualitative queries about the connection properties of neighboring spatial objects, like “retrieve the lakes that are *inside* Scotland.” Other qualitative queries that involve objects in close proximity cannot be formulated in topological terms, for example: “the cities that are *between* Glasgow and Edinburgh,” “the lakes that are *surrounded* by the mountains,” “the shops that are on the *right* of the road,” and “the building that is *before* the crossroad.” All these examples can be seen as semantic interpretations of underlying projective properties of spatial objects. As discussed in [4], geometric properties can be subdivided into three groups: *topological*, *projective*, and *metric*. Most qualitative relations between spatial objects can be defined in terms of topological or projective properties [39] with the exception of qualitative distance and direction relations (such as, *close*, *far*, *east*, and *north*) that are a qualitative

interpretation of metric distances and angles [5]. To have a qualitative understanding of projective relations, it is helpful to think about different two-dimensional views of a three-dimensional real-world scene of objects: changing the point of view, metric aspects such as distances and angles among the objects appear to be different, but there are properties that are common in all the views. These common properties are projective properties.

Invariants are geometric properties that do not change after a certain class of transformations: *topological invariants* are properties that are maintained after a topological transformation (a bicontinuous mapping or homeomorphism) and *projective invariants* are properties that are maintained after a projective transformation. Likewise, topological relations, which are defined by using the connectedness topological invariant, projective relations can be defined by using the *collinearity* projective invariant, which is the property of three collinear points being still collinear after an arbitrary number of projections. A main difference in the treatment of topological relations and projective relations is that, while basic topological relations are binary, basic projective relations are ternary because they are defined on the collinearity of three points.

In this paper, we propose a model for representing the projective relations between any three regions of the plane. The relations establish a jointly exhaustive and pairwise disjoint set of relations (JEPD). A preliminary version of this model was presented in [2], but the set of relations was not JEPD. One of the regions acts as the primary object and the other two as reference objects for the relation. We distinguish two cases based on whether the convex hulls of the reference objects are disjoint or not disjoint. In the first case, by using only projective concepts it is possible to partition the plane into five zones with respect to the reference objects; in the second case, the partition of the plane results in two zones. The model, called the *5-intersection*, is able to differentiate between 34 different projective relations that are obtained by computing the

• E. Clementini is with the Department of Electrical and Information Engineering, University of L'Aquila, I-67040 Poggio di Roio (AQ), Italy. E-mail: eliseo@ing.univaq.it.

• R. Billen is with the Department of Geography, University of Liege, 17 Allée du 6-Août, B-4000 Liege, Belgium. E-mail: rbillen@ulg.ac.be.

Manuscript received 16 May 2005; revised 26 Oct. 2005; accepted 13 Dec. 2005; published online 20 Apr. 2006.

For information on obtaining reprints of this article, please send e-mail to: tkde@computer.org, and reference IEEECS Log Number TKDE-0190-0505.

intersection of the primary object with the zones that are determined by the reference objects.

In first approximation, this work can be compared to research on qualitative relations dealing with relative positioning or cardinal directions [15], [16], [24], [26], [32], [33] and also path relations [22]. Most approaches consider binary relations to which a frame of reference is associated [17], [31]. But, most of them, even when explicitly related to projective geometry, never avoid the use of metric properties (minimum bounding rectangles, angles, etc.) and external frames of reference (such as a grid). To this respect, the main difference in our approach is that we only deal with projective invariants, independently of metric aspects such as distances and angles. Most work on cardinal directions deals with point abstractions of spatial features and limited work has been devoted to extended objects [16], [23], [38]. In [10], the authors use spheres surrounding the objects to take into account the shape of objects in relative orientation. In [16], the authors develop a model for cardinal directions between extended objects, where the partition of the plane is determined by the prolongations of the sides of the minimum bounding rectangle (MBR) of a reference object. Efficient algorithms for the implementation of such a model have been proposed in [35]. The projective relations that are introduced in our paper take into account the size, shape, and relative orientation of the three objects involved in a relation, thus enhancing the flexibility of the model since the acceptance areas of the relations are truly based on the projective properties of the objects. Early work on projective relations such as “between” was developed by [14]. Freksa’s double-cross calculus [13] is similar to our approach in the case of points. Such a calculus, as it has been further discussed in [18], [34], is based on ternary directional relations between points. However, in Freksa’s model, an intrinsic frame of reference centered in a given point partitions the plane into four quadrants that are given by the front-back and right-left dichotomies. This leads to a greater number of qualitative distinctions with different algebraic properties and composition tables. A smaller number of qualitative distinctions and an independence from the specific frame of reference would improve the possibility of extending this model to other spatial types besides points.

In this paper, after defining the model, we discuss the algorithms to compute the relations. In this way, we show how to implement the relations as new operators in a spatial database system. We develop the algorithms by assuming that regions are represented as polygons, where each polygon is a linked list of vertices. Thus, the implementation of the 5-intersection model can be carried out by taking advantage of many existing computational geometry algorithms: We propose different strategies only when it seemed appropriate to build a customized solution. We will show that the computation of the relations can be overall performed in optimal $O(n \log n)$ time when n is the sum of the number of vertices of the three polygons involved in a relation.

The paper is organized as follows: We start in Section 2 with developing a model for ternary projective relations between points: This is a natural starting point because the

collinearity invariant applies to three points. Such a model is a simplified version of the model for regions and is very useful to understand the plausibility of the relations. In Section 3, we introduce the 5-intersection model for ternary projective relations between regions, giving the definitions and examples of the geometric configurations. In Section 4, we describe the algorithms to implement the model in a vector data structure. In Section 5, we discuss the application to query languages with some examples of queries. In Section 6, we illustrate a Java-based implementation. In Section 7, we draw short conclusions and discuss further developments of the model.

2 MATHEMATICAL BACKGROUND

2.1 Basic Concepts

We consider ordinary objects of point-set topology, such as points and regions, which are embedded in the Euclidean two-dimensional space R^2 . We avoid using any metric properties of objects, such as lengths, areas, and angles, and restrict ourselves to use the minimal number of geometric concepts in order to remain inside the domain of projective geometry. We take an axiomatic view of projective geometry, where fewer axioms than in Euclidean geometry are assumed. The classification of geometries based upon the action of a group of allowable transformations on a set was introduced by Klein [21]. Therefore, projective geometry is defined by projective transformations. The names “projective transformation,” “homography,” “collineation,” and “projectivity” are all equivalent.

Definition 1. A projective transformation of the projective plane is defined as a mapping from the plane to itself such that the collinearity of any set of points is preserved. Such a mapping can be achieved with matrix multiplication by a nonsingular 3×3 matrix T . Each point p is transformed into a point p' : $p' = Tp$.

In the above definition, points are expressed in homogeneous coordinates¹ and equality must be intended up to a scale factor since scaling is unimportant [8]. Properties of objects that are maintained after a projective transformation are called projective invariants. Examples of projective invariants are the collinearity of three points, the extreme points of a set (e.g., the vertices of a polygon), the convexity of a region, the number of concavities of a region, etc. In the following, we list all the geometric concepts that will be used in the paper, together with the symbols that we will use, but for most of them we are not going to give any definition, assuming that they are well-known from elementary geometry.

Points will be indicated with small letters x, y, z , etc. Points are single elements that constitute the embedding space R^2 . Regions are bounded point-sets and will be indicated with capital letters A, B, C , etc. The interior of a region A is indicated with A° . The closure of a region A is indicated with \bar{A} . Regions are *simple* if they are regularly closed (i.e., $A = \bar{A}^\circ$) and without holes and disconnected

1. In homogeneous coordinates, we need three numbers to specify the coordinates of a point in the projective plane. It is a way of formally adding points at infinity to the Euclidean plane.

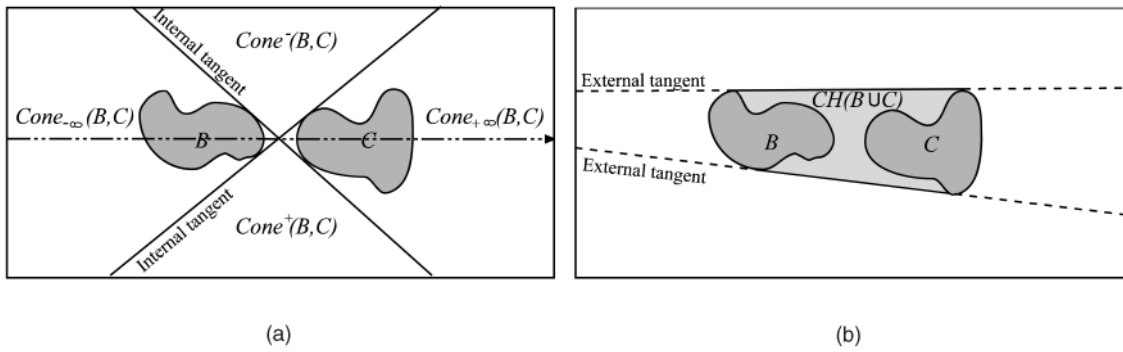


Fig. 1. (a) The internal common tangents partition the plane into four cones. (b) The external common tangents make up the convex hull of the union of regions B and C .

components. Regions are *complex* if they are regularly closed and have holes and disconnected components.

Two points uniquely identify a (straight) line: Therefore, a line is indicated with two points lying on it, e.g., \overline{xy} , or alternatively, with a small letter such as l . A closed interval on the line is indicated with $[x, y]$, where x and y are the extreme points of the interval. An open interval is indicated with (x, y) . We consider an elementary concept of orientation on a line, which is provided by the order of two points: Therefore, if we assume that a point x comes before a point y ($x < y$), we obtain an oriented line indicated with \overrightarrow{xy} . Every line \overline{xy} may correspond to two oriented lines \overrightarrow{xy} or \overrightarrow{yx} . An oriented line implies a total order among its points: All the points of an oriented line can be ordered: $x < y < z < \dots$. Two distinct points x, y of an oriented line \overrightarrow{xy} subdivide the line itself in three intervals, two of which are unbounded, which we denote with $(-\infty_{xy}, x)$, $[x, y]$, and $(y, +\infty_{xy})$. In \mathbb{R}^2 , a line subdivides the space in two half-planes. Given an oriented line \overrightarrow{xy} , we can distinguish the open right half-plane and the open left half-plane, indicated with HP_{xy}^+ and HP_{xy}^- , respectively.

The convex hull of a region A is indicated as a function $CH(A)$. The convex hull of a region is always a simple region. If two regions A and B have disjoint convex hulls, then two pairs of common tangents are uniquely defined: the external common tangents and the internal common tangents. The internal common tangents intersect inside the convex hull of the union of A and B (Fig. 1a), while the external common tangents intersect outside the convex hull of the union of A and B (Fig. 1b). The internal common tangents subdivide the plane in four cones. In order to distinguish the four cones, we consider an oriented line from region B to region C and we call $Cone_{-\infty}(B, C)$ the cone that contains region B , $Cone_{+\infty}(B, C)$ the cone that contains region C , $Cone^+(B, C)$ the cone that is to the right of the oriented line, and $Cone^-(B, C)$ the cone that is to the left of the oriented line. We assume that each of these cones is an open set, to exclude limit cases of points that coincide with the external tangents.

2.2 Ternary Relations

A ternary relation among three objects A , B , and C is denoted with $r(A, B, C)$. While in some relations the role of the three objects can be exchanged without affecting the relation, in some relations if the arguments are exchanged,

also the relation changes. For binary relations, the role of the two objects can be exchanged if the relation is symmetric. While there is no definition of symmetry for ternary relations, we will point out later on when an exchange of the arguments is possible. We assume that the first object A involved in the relation r has the role of primary object and the second and third objects have the role of reference objects. Therefore, the relation $r(A, B, C)$ should be read as “ A is in the relation r with B and C .” The order of the reference objects B and C affects the orientation, that is, an orientation from reference object B to reference object C is assumed.

2.3 Properties of Ternary Relations

A full discussion of properties of ternary relations is outside the scope of this paper. The study of properties is fundamental for developing a reasoning system based on the relations. A first step toward this development was published in [35].

We can identify the following groups of properties for ternary relations:











1. properties obtainable for two or three coincident arguments, e.g., $r(A, A, A)$, $r(A, A, B)$,
2. properties obtainable by exchanging the order of arguments, e.g., $r(A, B, C) \Rightarrow r'(A, C, B)$, and
3. properties about inferences that can be done with relations applied to four objects, e.g., $r(A, B, C) \wedge r'(B, C, D) \Rightarrow r''(A, C, D)$.

A relation algebra of ternary relations has been introduced in [19]. Composition is an example of property of the third group, while converse and rotation are examples of the second group.

2.4 Projective Relations among Points

In Table 1, we summarize the definitions of projective relations among points. The basic projective relation is the *collinearity* of three points. Three points are *collinear* if they lie on the same line. If two of the points are coincident, the three points are trivially collinear since two points determine a line. If all three points are coincident, they are also trivially collinear because there are infinite lines incident in a point. We can exchange the order of the arguments: If $coll(x, y, z)$, then $coll(y, x, z)$, and so on. If three points are *aside*, they must be distinct points. We can also exchange the order of the arguments. The relation *aside* can

TABLE 1
The Definitions of Projective Relations among Points

name	short name	definition	drawing
<i>collinear</i>	$coll(x,y,z)$	$\exists \text{ line } l : x \in l, y \in l, z \in l$	
<i>aside</i>	$aside(x,y,z)$	$\neg coll(x,y,z)$	
<i>rightside</i>	$rs(x,y,z)$	$x \in HP_{yz}^+$	
<i>leftside</i>	$ls(x,y,z)$	$x \in HP_{yz}^-$	
<i>inside</i>	$in(x,y,z)$	$x=y \wedge y=z$	
<i>outside</i>	$ou(x,y,z)$	$x \neq y \wedge y=z$	
<i>between</i>	$bt(x,y,z)$	$y \neq z \wedge x \in [y, z]$	
<i>nonbetween</i>	$nonbt(x,y,z)$	$coll(x,y,z) \wedge y \neq z \wedge x \notin [y, z]$	
<i>before</i>	$bf(x,y,z)$	$coll(x,y,z) \wedge y \neq z \wedge x \in (-\infty, y)$	
<i>after</i>	$af(x,y,z)$	$coll(x,y,z) \wedge y \neq z \wedge x \in (z, +\infty)$	

be divided in the relations *rightside* and *leftside*, for which we cannot exchange the order of the arguments. If we exchange the two reference objects in $rs(x, y, z)$, we obtain $ls(x, z, y)$. In the case when the two reference points are coincident, we refine the relation *collinear* in the relations *inside* and *outside*. In the case when the two reference points are distinct, we refine the relation *collinear* in the relations *between* and *nonbetween*. The relation *nonbetween* can be refined in the relations *before* and *after*.

Theorem 1. *The set of ternary relations among points* *rightside*, *leftside*, *between*, *before*, *after*, *inside*, and *outside* *is a JEPD set of relations.*

Proof. First part: The set of relations $\{rs, ls, bt, bf, af, in, ou\}$ is a jointly exhaustive set. We have to prove that, given any three points x, y, z , the projective relation among them must be one of the relations in the set. The decision tree in Fig. 2 gives the proof.

Second part: The set of relations

$$\{rs, ls, bt, bf, af, in, ou\}$$

is a pairwise disjoint set. We have to prove that if a given relation $r(x, y, z)$ holds, then the remaining relations in

the set are false. This is an immediate consequence of the partition of the plane that is implicit in the definition of the relations. Specifically, we distinguish the case $y = z$ and $y \neq z$. If $y = z$, the relation among points x, y, z can be either an *inside* or an *outside* (Fig. 3a). If $y \neq z$, the relation among points x, y, z can be only one among *between*, *before*, *after*, *leftside*, and *rightside* (Fig. 3b). \square

3 PROJECTIVE RELATIONS BETWEEN REGIONS

The results of Section 2 were useful to understand the hierarchy of ternary projective relations between three points. Such results immediately follow from the definition of collinearity in a projective space. In this section, we are going to define ternary projective relations between three objects of type region. The definitions for points will be a special case of the definitions for regions.

3.1 Definitions of Relations

The importance of semantics of collinearity relies on the fact that modelling all projective properties of spatial data can be done as a direct extension of such a property. The relation *collinear* among regions can be introduced as a

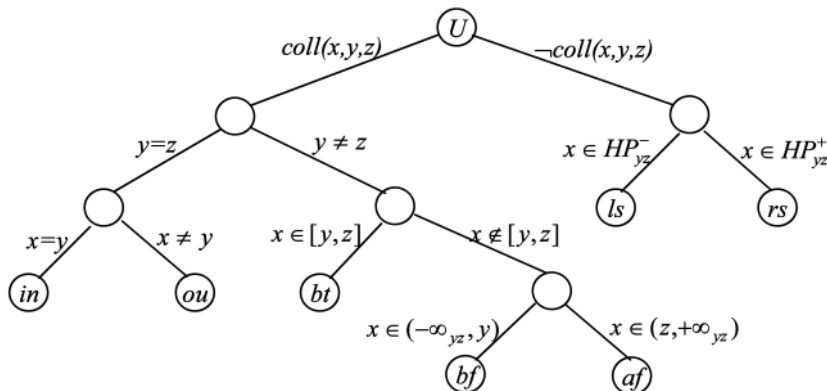


Fig. 2. A decision tree for the projective relations among points.

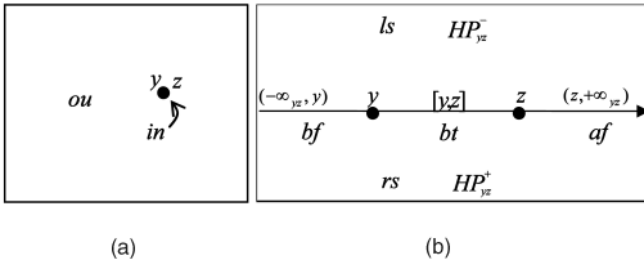


Fig. 3. (a) If y and z are coincident, the relation among points x, y, z can be either *inside* or *outside*. (b) If y and z are distinct, the relation among points x, y, z can be only one among *between*, *before*, *after*, *leftside*, *rightside*.

generalization of the same relation among points. Using different combinations of universal and existential quantifiers, we can obtain various definitions of collinearity for regions. The most useful definition is given in the sequel, while a discussion about other alternative definitions is given in [3].

Definition 2. Given three simple regions $A, B, C \in R^2$, $coll(A, B, C) \equiv_{def} \forall x \in A^\circ [\exists y \in B^\circ [\exists z \in C^\circ [coll(x, y, z)]]]$.

Let us examine the geometric realization of collinearity among regions. Similar to points, where we had a degenerate case of collinearity for coincident reference points, we have a degenerate case of collinearity among regions if reference regions have nondisjoint convex hulls: A region A is always collinear to regions B and C if the intersection $CH(B) \cap CH(C)$ is nonempty (see Fig. 4a). If the intersection $CH(B) \cap CH(C)$ is empty, we can identify a part of the plane where a region A that is completely contained into it satisfies the relation *collinear* (see Fig. 4b). Let us call this part of the plane the *collinearity zone* of B and C , $Coll(B, C)$. Such a zone can be built by considering all the lines that are intersecting both B and C .

In general, the zone of the plane where a region A that is completely contained into it satisfies a relation r is called the *acceptance area* of r . The collinearity zone and all acceptance areas of relations are open sets: this corresponds to considering the interior of regions in all the definitions of relations. This choice allows us to avoid limit cases: A point x in the boundary of region A that is falling in the boundary of an acceptance area does not influence the relation (Fig. 5a). If we had made the opposite choice, the point x would have contributed at the same time to the *collinear* and *aside* relations.

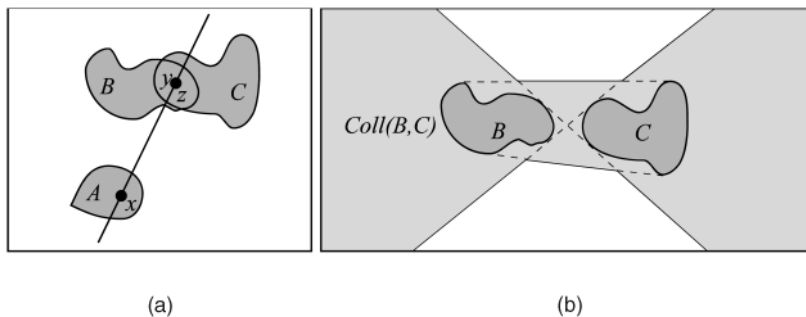


Fig. 4. (a) For nondisjoint convex hulls of regions B and C , region A is always collinear. (b) For disjoint convex hulls, the collinearity zone of B and C can be defined.

The definition of the relation *collinear* can be extended to complex regions by considering the convex hulls of the reference objects in place of the reference objects (Fig. 5b). The definition of *collinear* together with other projective relations for regions is given in Table 2. Besides each definition, also the corresponding acceptance area is given. Similarly to relations among points, the relations *rightside* and *leftside* are refinements of the relation *aside*. The relation *collinear*, in the case the two reference regions have not disjoint convex hulls, can be refined in two relations that are called *inside* and *outside*. The relations *between* and *nonbetween* are refinements of the relation *collinear* in the case the two reference regions have disjoint convex hulls. The relation *nonbetween* can be refined in the two relations *before* and *after*, respectively.

3.2 The 5-Intersection and 2-Intersection Model

In this section, we use the basic relations *rightside*, *leftside*, *between*, *before*, *after*, *inside*, and *outside* to build a model for all projective relations between three regions of the plane. The zones corresponding to the first five relations make a partition of the plane in the case the two reference regions have disjoint convex hulls (Fig. 6a). If the two reference regions have not disjoint convex hulls, the plane is partitioned in two zones, corresponding to relations *inside* and *outside* (Fig. 6b).

Let us consider the following matrix of empty/nonempty intersections of a region A with the five zones of Fig. 6a:

	$A \cap \text{Leftside}(B, C)$	
$A \cap \text{Before}(B, C)$	$A \cap \text{Between}(B, C)$	$A \cap \text{After}(B, C)$
	$A \cap \text{Rightside}(B, C)$	

We call this matrix the *5-intersection*. In the matrix, a value 0 indicates an empty intersection, while a value 1 indicates a nonempty intersection. The 5-intersection can have $2^5 - 1$ different configurations. Each configuration corresponds to a projective relation among three regions A, B , and C , where $CH(B) \cap CH(C) = \emptyset$. The matrix with all five values equal to zero does not correspond to a relation. For $CH(B) \cap CH(C) \neq \emptyset$, we consider the following *2-intersection* matrix:

$A \cap \text{Inside}(B, C)$	$A \cap \text{Outside}(B, C)$
------------------------------	-------------------------------

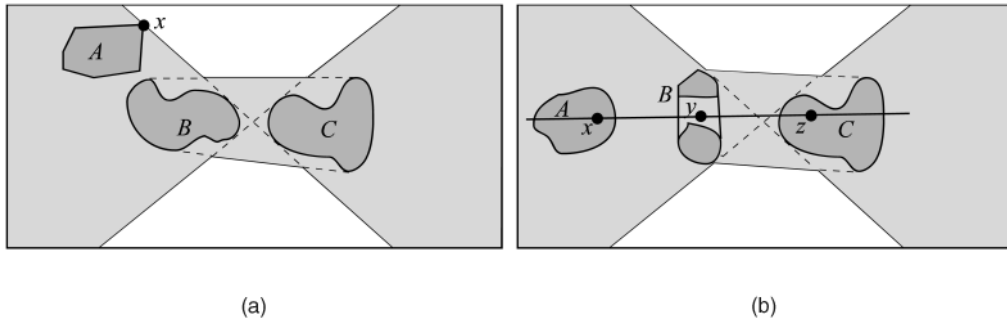


Fig. 5. (a) Points in the boundary do not affect relations among regions. (b) The relation *collinear* for complex regions, e.g., region *B* has two components.

which can assume the values (0 1), (1 0), and (1 1). Overall, we obtain a model that is able to distinguish among a set of 34 projective relations among three regions of the plane.

For the sake of conciseness, we also use a linear notation for the relation by listing seven bits that represent the intersection of region *A* with *Leftside*(*B, C*), *Before*(*B, C*), *Between*(*B, C*), *After*(*B, C*), *Rightside*(*B, C*), *Inside*(*B, C*), and *Outside*(*B, C*), respectively. In this notation, the basic relations are expressed as follows:

$$\begin{aligned}
 ls(A, B, C) &= (10000|00), \\
 bf(A, B, C) &= (01000|00), \\
 bt(A, B, C) &= (00100|00), \\
 af(A, B, C) &= (00010|00), \\
 rs(A, B, C) &= (00001|00), \\
 in(A, B, C) &= (00000|10), \\
 ou(A, B, C) &= (00000|01).
 \end{aligned}$$

Other relations correspond to matrices with more than one nonempty value: for example, a relation that is a combination of two basic relations, such as a “before and rightside,” is indicated as: $bf:rs(A, B, C) = (01001|00)$. Previously defined relations *nonbetween*, *collinear*, and *aside* can be expressed in terms of the model with disjunctions, such as:

$$\begin{aligned}
 coll(A, B, C) &= (01000|00) \vee (00100|00) \vee (00010|00) \\
 &\vee (01100|00) \vee (01010|00) \vee (00110|00) \\
 &\vee (01110|00) \vee (00000|10) \vee (00000|01) \\
 &\vee (00000|11),
 \end{aligned}$$

and $aside(A, B, C) = (10000|00) \vee (00001|00) \vee (10001|00)$.

In Fig. 7, Fig. 8, Fig. 9, Fig. 10, and Fig. 11, we show examples of the 34 projective relations. All relations can hold among simple regions or complex regions, with the exception of few of them that can hold for complex regions only: The relations $Tbf:af(A, B, C) = (01010|00)$ and $ls:rs(A, B, C) = (10001|00)$ need region *A* to have at least two

TABLE 2
The Definitions of Projective Relations among Regions

relation	definition	acceptance area
$coll(A, B, C)$	$\forall x \in A^\circ [\exists y \in CH(B)^\circ [\exists z \in CH(C)^\circ [coll(x, y, z)]]]$	$Coll(B, C) = Cone_{-\infty}(B, C) \cup Cone_{+\infty}(B, C) \cup (CH(B \cup C))^\circ$
$aside(A, B, C)$	$\forall x \in A^\circ [\forall y \in CH(B)^\circ [\forall z \in CH(C)^\circ [aside(x, y, z)]]]$	$Aside(B, C) = Cone^+(B, C) \cup Cone^-(B, C) - (CH(B \cup C))$
$rs(A, B, C)$	$\forall x \in A^\circ [\forall y \in CH(B)^\circ [\forall z \in CH(C)^\circ [rs(x, y, z)]]]$	$Rightside(B, C) = Cone^+(B, C) - CH(B \cup C)$
$ls(A, B, C)$	$\forall x \in A^\circ [\forall y \in CH(B)^\circ [\forall z \in CH(C)^\circ [ls(x, y, z)]]]$	$Leftside(B, C) = Cone^-(B, C) - CH(B \cup C)$
$in(A, B, C)$	$CH(B) \cap CH(C) \neq \emptyset \wedge A^\circ \subseteq (CH(B \cup C))^\circ$	$Inside(B, C) = (CH(B \cup C))^\circ$
$ou(A, B, C)$	$CH(B) \cap CH(C) \neq \emptyset \wedge A^\circ \cap (CH(B \cup C)) = \emptyset$	$Outside(B, C) = R^2 - (CH(B \cup C))$
$bt(A, B, C)$	$CH(B) \cap CH(C) = \emptyset \wedge A^\circ \subseteq (CH(B \cup C))^\circ$	$Between(B, C) = (CH(B \cup C))^\circ$
$nonbt(A, B, C)$	$CH(B) \cap CH(C) = \emptyset \wedge A^\circ \subset (Cone_{-\infty}(B, C) \cup Cone_{+\infty}(B, C)) - CH(B \cup C)$	$NonBetween(B, C) = (Cone_{-\infty}(B, C) \cup Cone_{+\infty}(B, C)) - CH(B \cup C)$
$bf(A, B, C)$	$CH(B) \cap CH(C) = \emptyset \wedge A^\circ \subset Cone_{-\infty}(B, C) - CH(B \cup C)$	$Before(B, C) = Cone_{-\infty}(B, C) - CH(B \cup C)$
$af(A, B, C)$	$CH(B) \cap CH(C) = \emptyset \wedge A^\circ \subset Cone_{+\infty}(B, C) - CH(B \cup C)$	$After(B, C) = Cone_{+\infty}(B, C) - CH(B \cup C)$

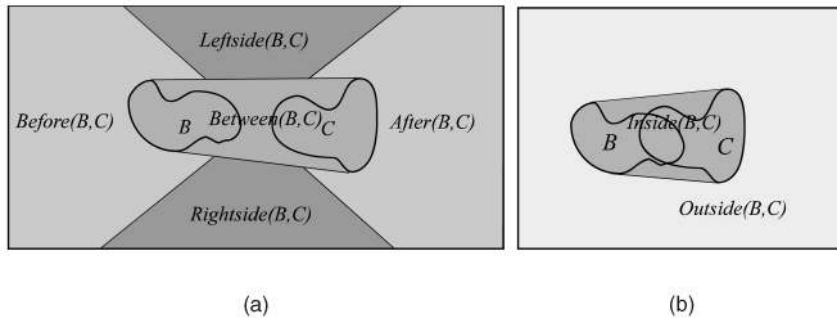


Fig. 6. (a) The partition of the plane into five zones for disjoint convex hulls of reference regions. (b) The partition of the plane into two zones for non-disjoint convex hulls of reference regions.

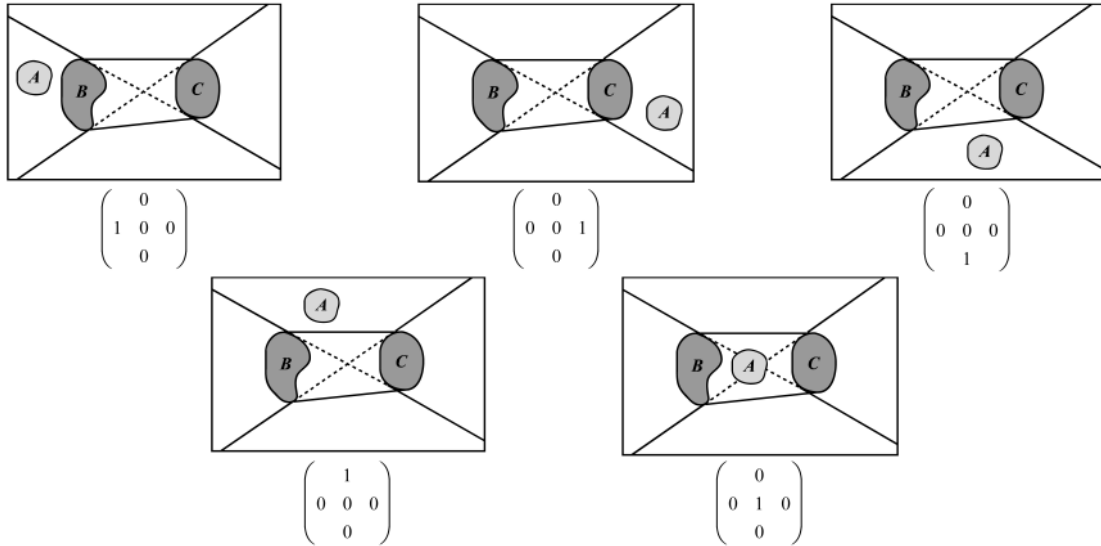


Fig. 7. The projective relations with object A intersecting only one zone of the partition.

separate components. When the intersection with $Between(B, C)$ is 1, region A can overlap regions B and C . However, to make the drawings clearer, we avoided drawing overlapping regions unless a complex region would have been necessary to interpret the relation: For example, in the case $ls : bt : af : rs(A, B, C) = (10111|00)$, we drew a region A nonoverlapping B and C ; in contrast, in the case $bf : bt : af(A, B, C) = (01110|00)$, we drew a simple region A overlapping B and C to avoid to draw a complex region A with three components.

Theorem 2. *The set of 34 ternary relations among regions is a JEPD set of relations.*

Proof. First part: The set of 34 relations is a jointly exhaustive set. We have to prove that, given any three regions A, B , and C , the projective relation among them must be one of the relations in the set. Two cases may apply: $CH(B) \cap CH(C) = \emptyset$ or $CH(B) \cap CH(C) \neq \emptyset$. In the first case, the plane can be partitioned in five zones:

1. $Rightside(B, C)$,
2. $Leftside(B, C)$,
3. $Between(B, C)$,
4. $Before(B, C)$, and
5. $After(B, C)$.

In the second case, the plane is partitioned in two zones: $Inside(B, C)$ and $Outside(B, C)$. Since in both cases the

zones cover the entire plane,² region A must intersect one or more zones: Therefore, one of the 34 relations will hold.

Second part: The set of 34 relations is a pairwise disjoint set. We have to prove that if a given relation $r(A, B, C)$ holds, then the remaining relations in the set are false. Since the zones of the partitions are disjoint, a region A can be split in parts such that each of these parts falls in exactly one zone: Only one of the 34 relations will hold. \square

4 ALGORITHMS FOR COMPUTING THE RELATIONS

In this section, we develop the algorithms to compute the relations. We refer to a specific vector data model, where a point is represented by its coordinates x and y and a region is represented by a polygon made up of a sequence of vertices. We restrict the treatment to simple polygons, i.e., polygons without holes and disconnected components.

4.1 Relations between Points

Computing the projective relations between points is straightforward from basic computational geometry notions (see, e.g., [28]). Given three points p_1, p_2 , and p_3 , we indicate their coordinates with (x_1, y_1) , (x_3, y_3) , and

2. Since the zones have been defined as open sets, they cover all the plane except the zones' boundaries. However, this exception does not influence the proof being region A two-dimensional.

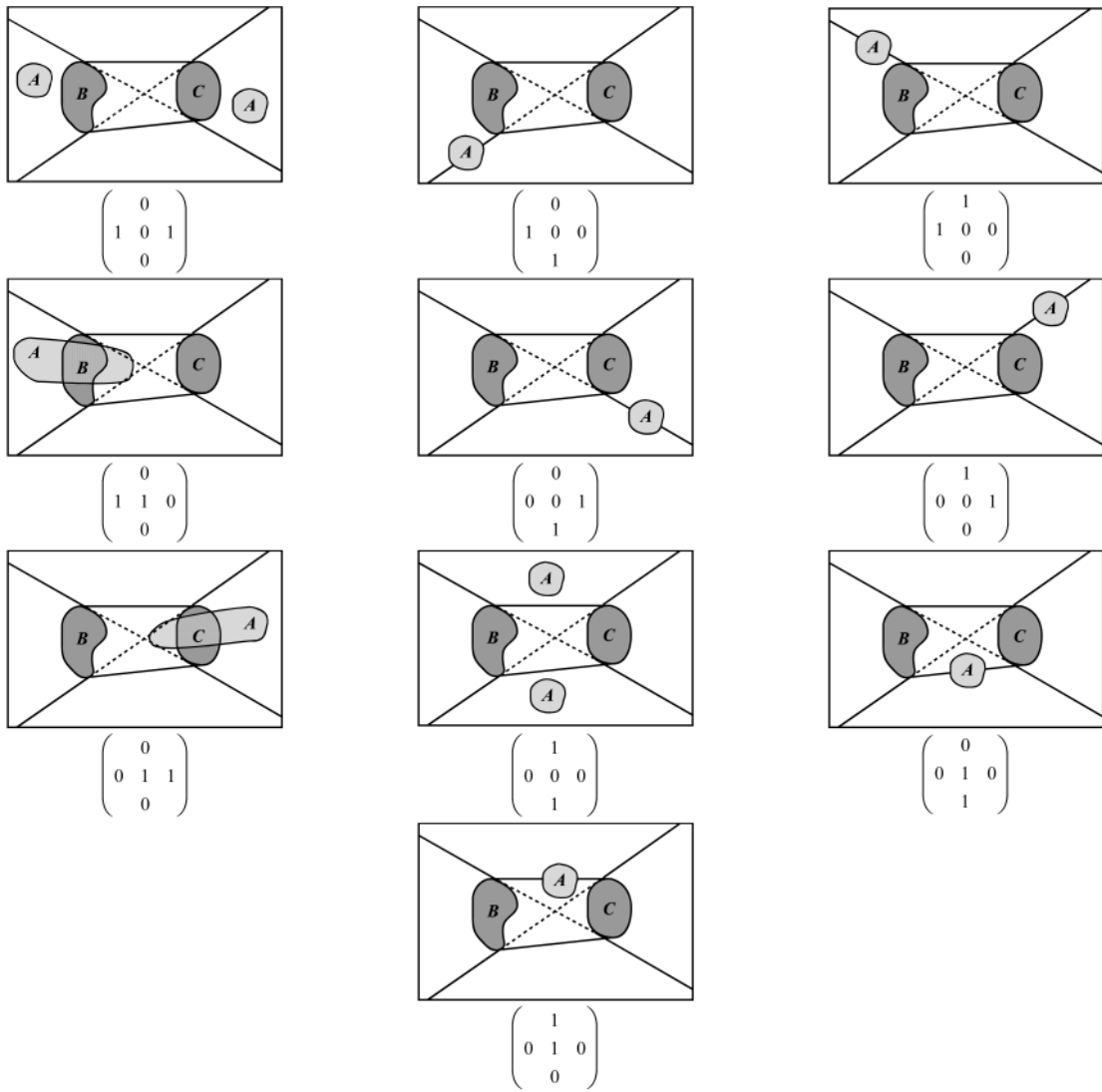


Fig. 8. The projective relations with object A intersecting two zones of the partition.

(x_3, y_3) . We indicate the triangle formed by the three points with $p_1p_2p_3$. The area of the triangle $p_1p_2p_3$ is given by half of the magnitude of the cross product between vectors $\overrightarrow{p_2p_3}$ and $\overrightarrow{p_2p_1}$:

$$2 \cdot \text{area}(p_1p_2p_3) = |\overrightarrow{p_2p_3} \times \overrightarrow{p_2p_1}|.$$

In terms of the coordinates of points:

$$2 \cdot \text{area}(p_1p_2p_3) = (x_3 - x_2)(y_1 - y_2) - (x_1 - x_2)(y_3 - y_2).$$

The latter quantity is positive if $p_1p_2p_3$ is a counter-clockwise triangle, is equal to zero if the three points are collinear, and is negative if $p_1p_2p_3$ is a clockwise triangle. Specifically:

$$\begin{aligned} \text{coll}(p_1, p_2, p_3) &\Leftrightarrow \text{area}(p_1p_2p_3) = 0, \\ \text{aside}(p_1, p_2, p_3) &\Leftrightarrow \text{area}(p_1p_2p_3) \neq 0, \\ \text{rs}(p_1, p_2, p_3) &\Leftrightarrow \text{area}(p_1p_2p_3) < 0, \\ \text{ls}(p_1, p_2, p_3) &\Leftrightarrow \text{area}(p_1p_2p_3) > 0. \end{aligned}$$

The relation *between* can be assessed by a first check on collinearity and a second check on conditions among

coordinates. Specifically, the relation $bt(p_1, p_2, p_3)$ is verified if and only if both the following conditions are true:

1. $\text{area}(p_1p_2p_3) = 0$.
2.
 - a. If the line $\overline{p_2p_3}$ is not vertical ($x_2 \neq x_3$):
 $x_2 \leq x_1 \leq x_3 \vee x_3 \leq x_1 \leq x_2$.
 - b. If the line $\overline{p_2p_3}$ is vertical ($x_2 = x_3$):

$$y_2 \leq y_1 \leq y_3 \vee y_3 \leq y_1 \leq y_2.$$

Similarly, the relation $bf(p_1, p_2, p_3)$ is verified if and only if both the following conditions are true:

1. $\text{area}(p_1p_2p_3) = 0$.
2.
 - a. If the line $\overline{p_2p_3}$ is not vertical ($x_2 \neq x_3$):
 $x_1 \leq x_2 \leq x_3 \vee x_3 \leq x_2 \leq x_1$.
 - b. If the line $\overline{p_2p_3}$ is vertical ($x_2 = x_3$):

$$y_1 \leq y_2 \leq y_3 \vee y_3 \leq y_2 \leq y_1.$$

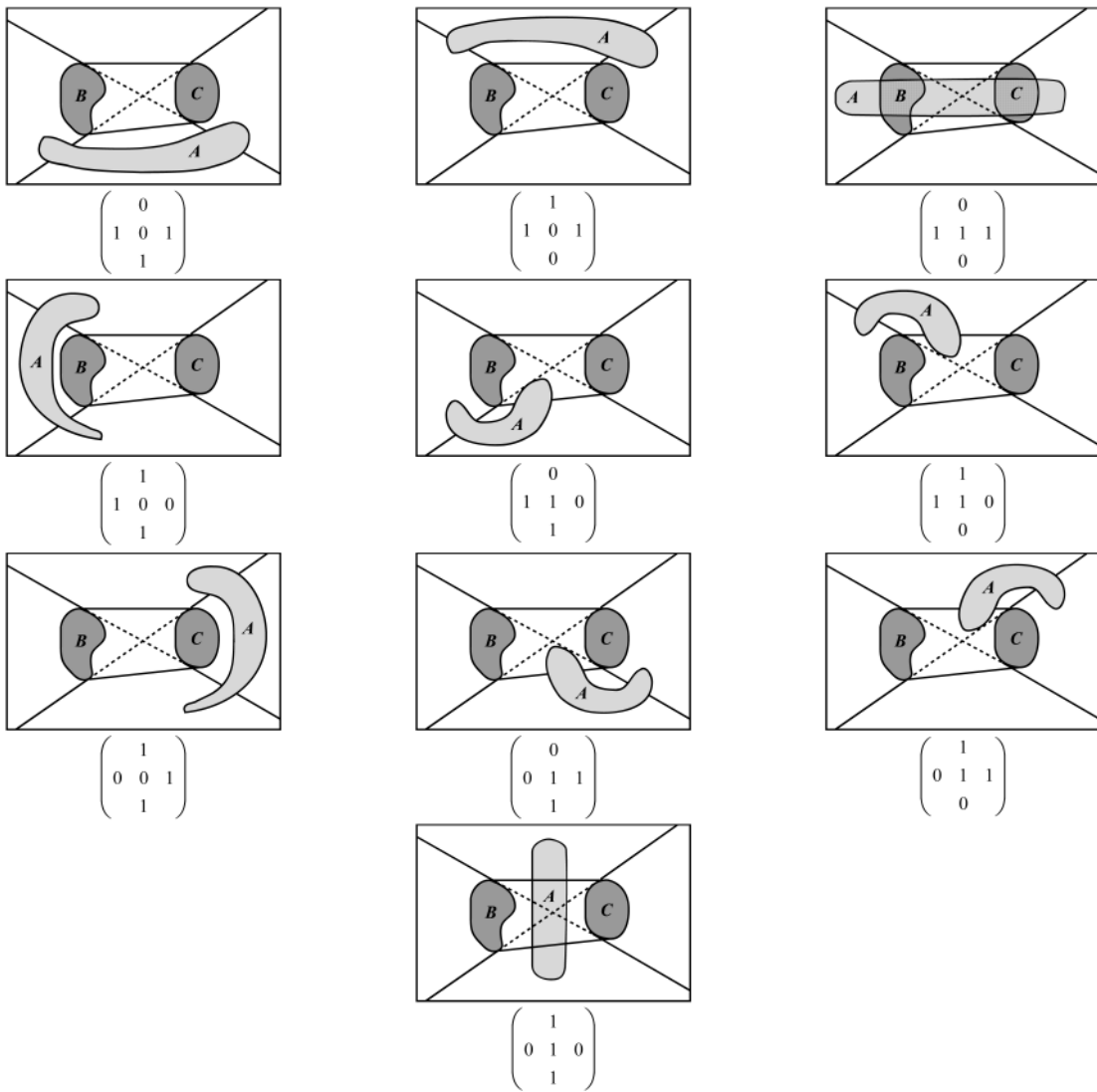


Fig. 9. The projective relations with object *A* intersecting three zones of the partition.

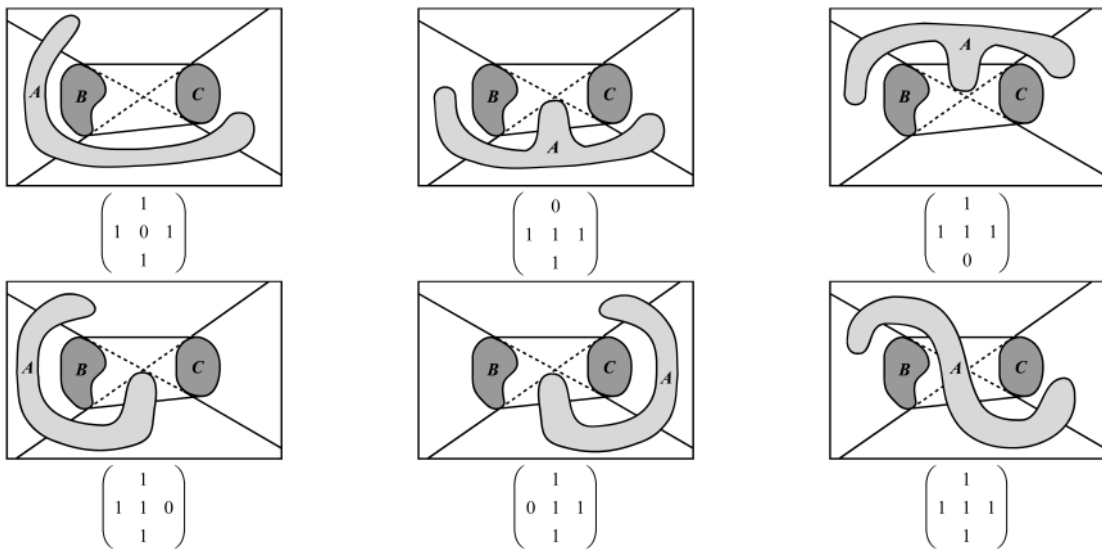


Fig. 10. The projective relations with object *A* intersecting four or five zones of the partition.

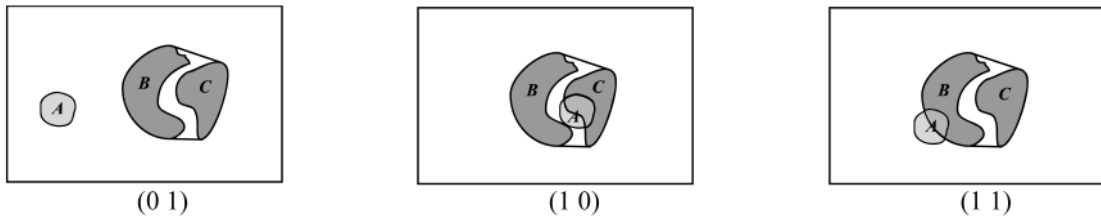


Fig. 11. The projective relations for reference regions with intersecting convex hulls.

Finally, the relation $af(p_1, p_2, p_3)$ is verified if and only if both the following conditions are true:

1. $area(p_1 p_2 p_3) = 0$.
2.
 - a. If the line $\overline{p_2 p_3}$ is not vertical ($x_2 \neq x_3$):

$$x_2 \leq x_3 \leq x_1 \vee x_1 \leq x_3 \leq x_2.$$
 - b. If the line $\overline{p_2 p_3}$ is vertical ($x_2 = x_3$):

$$y_2 \leq y_3 \leq y_1 \vee y_1 \leq y_3 \leq y_2.$$

4.2 Relations between Regions

In the case $CH(B) \cap CH(C) = \emptyset$, Algorithm 1 generates the partition of the plane into five zones for the two reference objects B and C and calculates the 5-intersection matrix expressing the projective relation between the primary object A and the reference objects B and C . In the case $CH(B) \cap CH(C) \neq \emptyset$, Algorithm 1 finds the 2-intersection matrix.

Algorithm 1.

1. Build the convex hull of regions B and C .
2. If $CH(B) \cap CH(C) \neq \emptyset$, then goto 7.
3. Find the four mutual tangents of $CH(B)$ and $CH(C)$. These are expressed by the four points of tangency of internal tangents, which we indicate with $t_{B1}^i, t_{B2}^i, t_{C1}^i, t_{C2}^i$, and the four points of tangency of external tangents, which we indicate with $t_{B1}^e, t_{B2}^e, t_{C1}^e, t_{C2}^e$ (see Fig. 12).
4. Find the points of intersection between the internal tangents and external tangents, which we indicate with r, s, u, v (see Fig. 12).

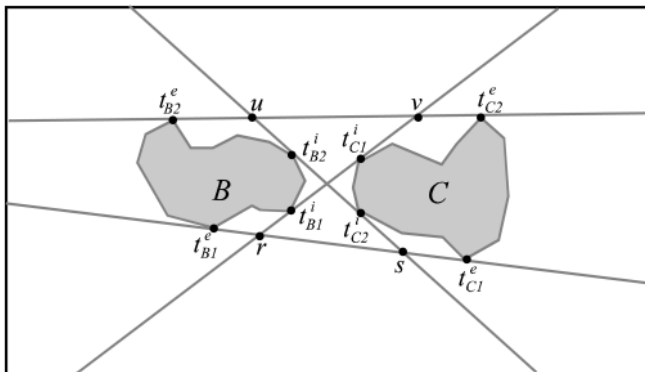


Fig. 12. The construction of the points of tangency and their intersections.

5. Find the convex hull of $B \cup C$.
6. Find the intersection of region A with internal tangents and with $CH(B \cup C)$. This is expressed by a 5-intersection matrix. Halt.
7. Find the convex hull of $B \cup C$.
8. Find the intersection of region A with $CH(B \cup C)$. This is expressed by a 2-intersection matrix.

Step 1 of Algorithm 1 can be done with algorithms that build the convex hull of a polygon. For example, we consider the algorithm by Melkman [25], which runs in time $O(n)$ when n is the number of vertices. Step 2 can be carried out with an algorithm for checking the intersection of two convex polygons [29], even if for our purposes we do not need to calculate the intersection of the two polygons, but it suffices to know whether this intersection is void or not. This step has a time complexity of $O(m+n)$ for convex polygons of m and n vertices, respectively. To find the four mutual tangents (Step 3), we consider polygon-to-polygon tangency algorithms when both polygons are convex: These run in $O(m+n)$ or even $O(\log(m+n))$ time. The algorithm originally described by Preparata and Hong [30] as part of the convex hull divide-and-conquer algorithm runs in $O(m+n)$ time, while in [20], authors have described an $O(\log(m+n))$ algorithm that finds the common tangents between two convex polygons. Step 4 can be carried out with an algorithm for segment-line intersection, see, e.g., [28], in constant time. The convex hull of $B \cup C$ (Step 5) can be easily constructed by considering the already found $CH(B)$ and $CH(C)$ and their external tangents. With regard to Step 6, instead of taking into consideration general algorithms for the intersection between polygons, we prefer to build a customized algorithm to take advantage of the particular structure of the partition of the plane into five zones. This step is further expanded in Algorithm 2.

Algorithm 2: Build 5-intersection.

Input: region A ; $CH(B \cup C)$; internal tangents; intersections r, s, u, v ;

Output: 5-intersection matrix;

begin

```

i ← 1;
pos ← Check_Position(ai,  $CH(B \cup C)$ , internal tangents);
Update_5int(pos);
i ← i + 1;
while ai ≠ a1 do
  posnext ← Check_Position(ai,  $CH(B \cup C)$ ,
    internal tangents);
  Update_5int(posnext);
  Treat_Special_Cases(ai-1, ai, pos, posnext,

```

```

        CH(B ∪ C), r, s, u, v);
    pos ← posnext;
    i ← i + 1;
endwhile
end

```

Algorithm 2 for finding the 5-intersection matrix is structured with a main loop that considers iteratively all the vertices of the boundary of region A and for each vertex checks in which of the five zones of the partition the vertex belongs to. This is done by a call to the function `Check_Position`, which is discussed afterwards (Algorithm 3). The vertices of region A are indicated with $a_1 \dots a_n$ and the generic vertex with a_i ; the successor of vertex a_n is a_1 . The variables “pos” and “posnext” contain the positions of two consecutive vertices and can take the values *bf*, *bt*, *af*, *ls*, and *rs* that correspond to the five zones *Before*(B, C), *Between*(B, C), *After*(B, C), *Leftside*(B, C), and *Rightside*(B, C), respectively. The function `Update_5int` adds the newly discovered position to the 5-intersection matrix and halts the algorithm if at some point, the relation (1111|00) is found, since in this case, there would not be more intersections to be discovered. The function `Treat_Special_Cases` deals with cases when a side of region A connecting two vertices intersects zones of the partition that are different from the zones where the two vertices belong. This will be further discussed in Algorithm 4.

Algorithm 3: `Check_Position`.

Input: point p ; $CH(B \cup C)$; internal tangents;
Output: the zone value where p belongs;

```

begin
    if Inside_Convex_Polygon(p, CH(B ∪ C)) then return bt
    else if coll(p, t_{B1}^i, t_{C1}^i) or coll(p, t_{B2}^i, t_{C2}^i)
        then Check_Position(p - ε, CH(B ∪ C),
            internal tangents)
        else if ls(p, t_{B1}^i, t_{C1}^i) and ls(p, t_{B2}^i, t_{C2}^i) then
            return ls
        else if ls(p, t_{B1}^i, t_{C1}^i) and rs(p, t_{B2}^i, t_{C2}^i) then
            return bf
        else if rs(p, t_{B1}^i, t_{C1}^i) and ls(p, t_{B2}^i, t_{C2}^i) then
            return af
        else if rs(p, t_{B1}^i, t_{C1}^i) and rs(p, t_{B2}^i, t_{C2}^i) then
            return rs;
    end
end

```

Algorithm 3 `Check_Position` first checks whether the given point is inside the *Between*(B, C) zone: this is done by a call to a function that verifies if a point is internal to a convex polygon. If the point is not inside the *Between*(B, C) zone, the algorithm checks whether the given point is collinear to the internal tangents: this is a special case to be avoided since the zones of the partition are open sets. In this case, there is another call to `Check_Position` with a new point indicated with $p - \varepsilon$, which represents a small displacement of the original point towards the previous vertex of the polygon. If this case does not apply, then the position of the given point with respect to the internal tangents is checked with a series of calls to projective relations between points. Overall, Algorithm 3 can run in

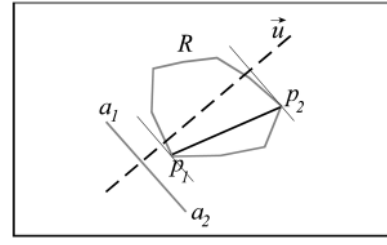


Fig. 13. An illustration of the function `Check_Intersect`.

$O(\log n)$ time if a logarithmic algorithm is used to perform the point in convex polygon test [28].

Algorithm 4: `Treat_Special_Cases`.

Input: two consecutive vertices a_{i-1}, a_i and their corresponding zone values pos and posnext;
 $CH(B \cup C)$; intersections r, s, u, v ;

Output: possible calls to `Update_5int`;

```

begin
    if pos = posnext then Treat_Same_Zone
    else if (posnext = bt) or (pos = bt) then
        Treat_Between_Zone
    else if Neighbor(pos, posnext) then
        Treat_Neighbor_Zone
    else Treat_Non_Neighbor_Zone
end;

```

Algorithm 4 treats special cases that may occur when the segment formed by two consecutive vertices a_{i-1}, a_i (with a zone value given by pos and posnext) crosses another zone of the partition, whose value is different from pos and posnext. Four subcases are taken into consideration: The first one corresponds to previous and current vertices of region A that are positioned in the same zone (function `Treat_Same_Zone`). The second subcase occurs when one of two vertices is within the zone *Between*(B, C) (function `Treat_Between_Zone`). The third subcase relates to positions of the previous and current vertices falling in neighboring zones (function `Treat_Neighbor_Zone`). The property of being neighboring zones is assessed by a function `Neighbor`, which returns true if the two zones are adjacent: The only pairs of zones that are not neighbors are *ls, rs*, and *bf, af*. The fourth and last subcase deals with previous and current vertices that are located in nonneighboring zones (function `Treat_Non_Neighbor_Zone`). These subcases use the function `Check_Intersect`, whose purpose is to determine if a segment does or does not intersect a convex polygon (see also Fig. 13). It can be implemented by finding the extreme points of the convex polygon in the direction perpendicular to the segment. This test can be performed in $O(\log n)$ [28]. The last test `Segment_Intersect` inside the function is done in constant time.

function `Check_Intersect` (point a_1, a_2 ; polygon R): Boolean
 /* This function checks whether the segment $[a_1a_2]$ intersects the region R . First, it finds the direction \vec{u} perpendicular to the segment; then it finds the extreme points of polygon R along direction \vec{u} , which are indicated with p_1, p_2 ; hence, it checks the intersection of the two segments $[a_1a_2]$ and $[p_1p_2]$ */

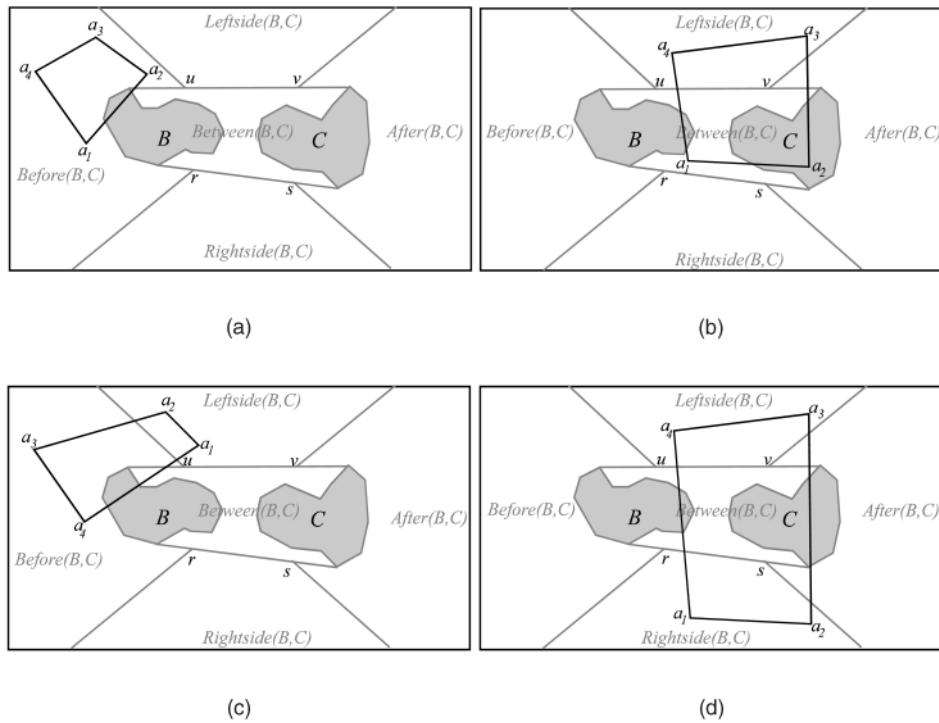


Fig. 14. An illustration of the algorithm `Treat_Special_Cases`.

begin

```

 $\vec{u}$  = Perpendicular_Direction( $a_1, a_2$ );
Find_Extreme_Points( $R, \vec{u}, p_1, p_2$ );
if Segment_Intersect( $a_1, a_2, p_1, p_2$ ) then return true;
end;

```

Regarding the function `Treat_Same_Zone`, it may discover that there is an intersection of the segment $[a_{i-1}a_i]$ with the $Between(B, C)$ zone (see Fig. 14a). The check itself is not done if the value bt is already present in the 5-intersection matrix: This is performed by a function `Check_Matrix` that returns true if the values that are passed to it are already in the matrix. The check for intersection of a segment with a convex polygon is performed by the function `Check_Intersect`. This function is only applied if the two vertices lie in the $Before(B, C)$ or $After(B, C)$ zones. It runs in $O(\log n)$ due to the possible calls to `Check_Intersect`.

function `Treat_Same_Zone`

begin

```

if posnext =  $bf$  or  $af$  then
  if not Check_Matrix( $bt$ ) then
    if Check_Intersect( $a_{i-1}, a_i, CH(B \cup C)$ ) then
      Update_5int( $bt$ );

```

end;

When one of the vertices lies in the $Between(B, C)$ zone, the segment $[a_{i-1}a_i]$ could intersect with one of the neighboring zones of the zone where the other vertex lies (see Fig. 14b). The function `Treat_Between_Zone` allows the detection of such cases and is carried out in constant time. As it was previously explained, the various checks are performed only if the corresponding values in the 5-intersection matrix are not already present. The checks are different whether we consider the cases where the other vertex lies in the

$Before(B, C)$ or $After(B, C)$ zones (function `BT_Case_Before_After`) or the cases where the other vertex lies in the $Leftside(B, C)$ or $Rightside(B, C)$ zones (function `BT_Case_Leftside_Rightside`).

function `Treat_Between_Zone`

begin

```

if (pos =  $bf$  or  $af$ ) or (posnext =  $bf$  or  $af$ ) then
  if not Check_Matrix( $ls, rs$ ) then
    BT_Case_Before_After else;
else/* (pos =  $ls$  or  $rs$ ) or (posnext =  $ls$  or  $rs$ ) */
  if not Check_Matrix( $bf, af$ ) then
    BT_Case_Leftside_Rightside

```

end;

function `BT_Case_Before_After`

begin

```

if pos =  $bf$  then {firstvertex =  $a_{i-1}$ ; secondvertex =  $a_i$ }
else/* pos =  $af$  */
  {firstvertex =  $a_i$ ; secondvertex =  $a_{i-1}$ };
if  $ls(r, \text{firstvertex}, \text{secondvertex})$  or
   $ls(s, \text{firstvertex}, \text{secondvertex})$ 
then Update_5int( $rs$ )
else if  $rs(u, \text{firstvertex}, \text{secondvertex})$  or
   $rs(v, \text{firstvertex}, \text{secondvertex})$ 
then Update_5int( $ls$ )

```

end;

function `BT_Case_Leftside_Rightside`

begin

```

if (pos =  $ls$ ) or (posnext =  $rs$ ) then
  {firstvertex =  $a_{i-1}$ ; secondvertex =  $a_i$ }
else/* (pos =  $rs$ ) or (posnext =  $ls$ ) */

```

```

    {firstvertex =  $a_i$ ; secondvertex =  $a_{i-1}$ };
if  $ls(u, \text{firstvertex}, \text{secondvertex})$  or
     $ls(r, \text{firstvertex}, \text{secondvertex})$ 
then Update_5int( $bf$ )
else if  $rs(v, \text{firstvertex}, \text{secondvertex})$  or
     $rs(s, \text{firstvertex}, \text{secondvertex})$ 
    then Update_5int( $af$ )
end;

```

The function `Treat_Neighbor_Zone` allows detecting the possible intersection of the segment $[a_{i-1}a_i]$ with the *Between*(B, C) zone (see Fig. 14c). It works similar to the function `Treat_Same_Zone`. Its complexity is $O(\log n)$ due to the possible calls to `Check_Intersect`.

```

function Treat_Neighbor_Zone
/* this function applies when the two vertices lie in
neighboring zones but none of them is a  $bt$  */
begin
    if not Check_Matrix( $bt$ ) then
        if Check_Intersect( $a_{i-1}, a_i, CH(B \cup C)$ )
            then Update_5int( $bt$ );
end;

```

The function `Treat_Non_Neighbor_Zone` allows detecting which zones are intersected by the segment $[a_{i-1}a_i]$ (see Fig. 14d). It holds some similarity with the function `Treat_Between_Zone`, like the call to two different functions depending on the position of the vertices (functions `NN_Case_Before_After` and `NN_Case_Leftside_Rightside`). It runs in $O(\log n)$ due to the possible calls to `Check_Intersect` inside the function `NN_Case_Leftside_Rightside`.

```

function Treat_Non_Neighbor_Zone
/* this function applies when the two vertices lie in
nonneighboring zones:  $bf$  and  $af$  or  $ls$  and  $rs$  */
begin
    if  $pos_{next} = bf$  or  $af$  then
        if not Check_Matrix( $ls, bt, rs$ ) then
            NN_Case_Before_After
        else /*  $pos_{next} = ls$  or  $rs$  */
            if not Check_Matrix( $bf, bt, af$ ) then
                NN_Case_Leftside_Rightside
end;

```

```

function NN_Case_Before_After
begin
    if  $pos = bf$  then {firstvertex =  $a_{i-1}$ ; secondvertex =  $a_i$ }
    else /*  $pos = af$  */
        {firstvertex =  $a_i$ ; secondvertex =  $a_{i-1}$ };
    if  $rs(r, \text{firstvertex}, \text{secondvertex})$  and
         $rs(s, \text{firstvertex}, \text{secondvertex})$  and
         $ls(u, \text{firstvertex}, \text{secondvertex})$  and
         $ls(v, \text{firstvertex}, \text{secondvertex})$ 
    then Update_5int( $bt$ )
    else if  $ls(r, \text{firstvertex}, \text{secondvertex})$  and
         $ls(s, \text{firstvertex}, \text{secondvertex})$ 
        then Update_5int( $rs$ )
    else if  $rs(u, \text{firstvertex}, \text{secondvertex})$  and
         $rs(v, \text{firstvertex}, \text{secondvertex})$ 
        then Update_5int( $ls$ )
    else if  $ls(r, \text{firstvertex}, \text{secondvertex})$  or

```

```

     $ls(s, \text{firstvertex}, \text{secondvertex})$ 
    then Update_5int( $rs$ ); Update_5int( $bt$ )
    else if  $rs(u, \text{firstvertex}, \text{secondvertex})$ 
        or  $rs(v, \text{firstvertex}, \text{secondvertex})$ 
        then Update_5int( $ls$ ); Update_5int( $bt$ )

```

```

end;
function NN_Case_Leftside_Rightside
begin
    if  $pos = ls$  then {firstvertex =  $a_{i-1}$ ; secondvertex =  $a_i$ }
        else /*  $pos = rs$  */
            {firstvertex =  $a_i$ ; secondvertex =  $a_{i-1}$ };
    if Check_Intersect(firstvertex, secondvertex,  $CH(B \cup C)$ )
        then Update_5int( $bt$ );
    if  $ls(u, \text{firstvertex}, \text{secondvertex})$  or
         $ls(r, \text{firstvertex}, \text{secondvertex})$ 
        then Update_5int( $bf$ )
    else if  $rs(v, \text{firstvertex}, \text{secondvertex})$  or
         $rs(s, \text{firstvertex}, \text{secondvertex})$ 
        then Update_5int( $af$ )
end;

```

Overall, Algorithm 2 has a worst-case time complexity $O(n \log n)$ since it has a loop that runs in time proportional to the number of vertices of A and the cost of the algorithms `Check_Position` and `Treat_Special_Cases` is at most logarithmic. Algorithm 2 checks the intersection of the zones of the partition with every vertex of polygon A and every edge of its boundary. The only case the algorithm fails to find an intersection with the zone *Between*(B, C) is when the latter is properly contained inside the polygon A . In this case, the algorithm would end producing a 5-intersection matrix (11011|00) instead of the correct one (11111|00). To correct this case, it is sufficient to check whether an arbitrary point of $CH(B \cup C)$ is inside A , by using a point-in-polygon algorithm, such as the fast winding number algorithm [36]. This test is performed in $O(n)$ time.

Steps 7 and 8 of Algorithm 1 refer to the case $CH(B) \cap CH(C) \neq \emptyset$. We do not further expand the comment to these steps because they are a variation of the general case discussed so far. While, for the sake of clarity, we restricted ourselves to simple polygons throughout Section 4, an extension to composite regions does not imply major changes to the algorithms: The implications are on the construction of $CH(B)$ and $CH(C)$ in Algorithm 1 and on the fact that Algorithm 2 needs to be repeated on every component of region A .

5 SPATIAL QUERIES

Spatial queries making use of binary relations fall in one of the following categories:

- Which are the objects B that have the relation r with a given object A ?
- Which is the relation r between two given objects A and B ?

Both kinds of query have been studied (e.g., [7]). Usually, in implementations of topological relations based on OGC specifications, the second kind of query is the one that corresponds to geometric calculations, while the first kind is obtained by a repeated application of the second kind.

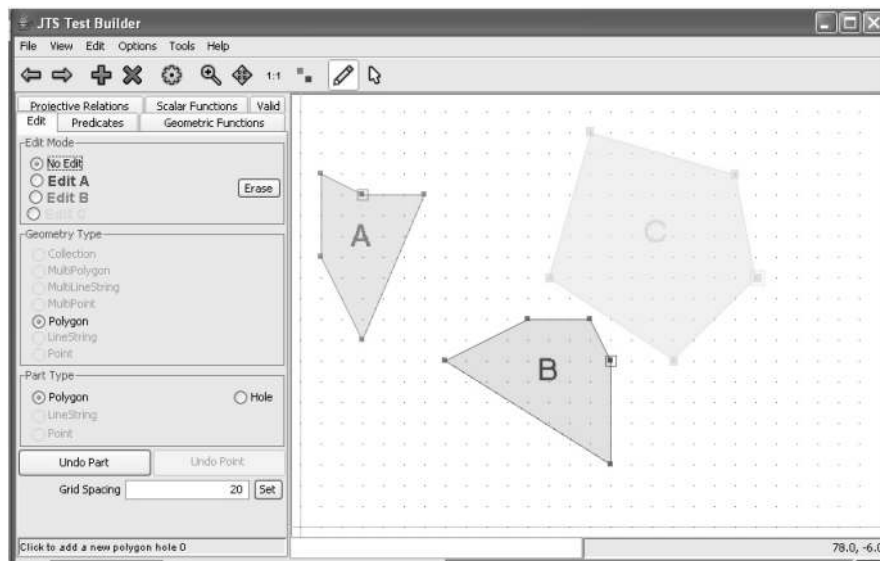


Fig. 15. The extension to the JTS Test Builder for drawing three objects.

Analogously, for ternary relations, we can distinguish the following categories of queries:

1. Which is the relation r among three given objects A , B , and C ?
2. Which are the objects A that have the relation r with the given objects B and C ? In this query, we need to build the partition of zones from the two reference regions. Then, we concentrate on the matrix given by the relation r and find the objects that satisfy this matrix.
3. Which are the objects B that have the relation r with the given objects A and C ? In this query, for each object B , we can define a different partition. We have to select the partitions that allow realizing the relation r .
4. Which are the objects C that have the relation r with the given objects A and B ? This query has no substantial difference with respect to previous one.

The first kind of queries can be calculated by a direct application of the algorithms. It corresponds to a second-order logic. Nonetheless, it is recognized to be of equal importance with respect to other queries [7]. It is implemented in our prototype. The other types of queries can be computed reusing this first type.

Examples of queries of the second kind are the following:

- What are the lakes that are *between* lake B and lake C ?
- What are the towns that are in the relation $ls : bt : af : rs$ with towns B and C ?

Similarly to the current practice in spatial databases, this query could be implemented by repeating the computation of relation for each possible region A . A more effective computation could be carried out with approximation strategies: It is not necessary to calculate the complete relation among three objects, but it is enough to calculate some key properties for each relation to identify a set of candidates [7].

An example of queries of the third kind is the following:

- What are the municipalities such that a municipality A is *before* those municipalities and municipality C ?

The third and fourth kind of queries can be dealt with either by direct methods or the applications of reasoning rules that allow to express the same queries in equivalent queries of the second kind. These rules allow exchanging the arguments in a relation applying rotation and converse properties [1]. The inconvenience is that often the equivalent queries are expressed as a disjunction of relations and, therefore, the result can be less accurate.

6 IMPLEMENTATION

Our implementation focuses on the first kind of query, which corresponds to a direct application of the algorithms of Section 4. It is based on an extension of the Java Topology Suite (JTS), a free package downloadable from the Internet [37]. We chose JTS after analyzing it and discovering that all the geometric definitions conform to the Simple Features Specification for SQL by the OGC. Many of the topological functions needed for our work were already implemented in this package, such as the convex hull or the area calculation.

We also chose Java as programming language for its portability and for its strong technology in a client-server application. In this context, every application is developed by splitting the application into layers: A typical layer partition is made up of a presentation layer, a business logic layer, and a data layer. Our prototype can be thought of as a 2-tier architecture, where the business logic layer is merged with the presentation layer. For the data layer, due to the lack of projective functions in any existing spatial DBMS, we chose not to use any DBMS and we instead decided to use shape files which contain the data for our tests.

Another advantage of using JTS is that it is possible to integrate our implementation into a 3-tier architecture, such that of the free software project "deegree" [9]: The JTS is supported by this software as a component to be added to the Web Feature Service (WFS), which allows Web-based access to vector data.

The current implementation has been useful to experiment queries of the first kind with test data. We extended the "Test-Builder" JTS tool with a third arbitrary object "C"

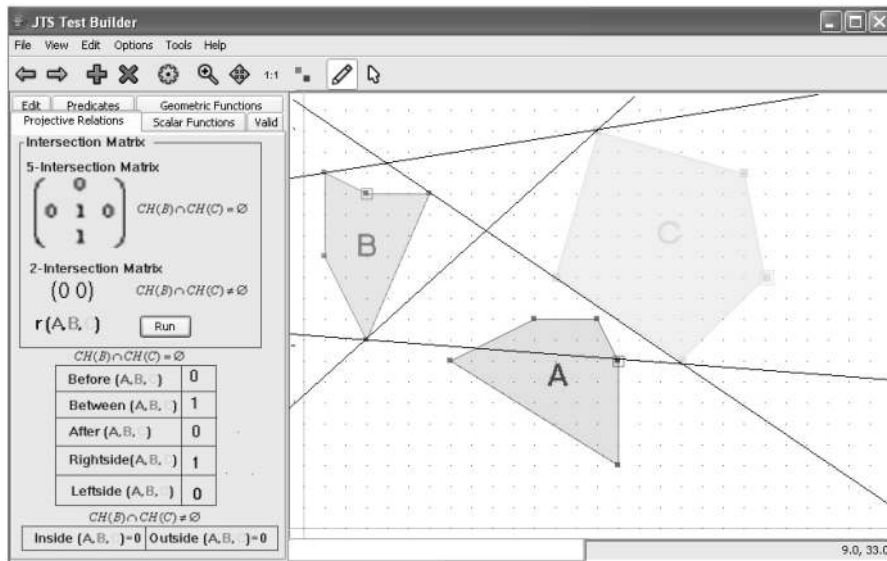


Fig. 16. The interface to run a query with projective relations.

besides the already available objects "A" and "B," as shown in Fig. 15. According to the algorithms in Section 4, we calculated the convex hulls of regions B and C. If they were disjoint convex hulls, then the 5-intersection matrix is calculated; otherwise, the 2-intersection matrix is calculated (see Fig. 16). In the interface, besides the matrices, the corresponding basic relations among the three objects are also shown.

7 CONCLUSIONS AND FURTHER WORK

This paper introduces a set of jointly exhaustive and pairwise disjoint projective relations between three regions of the plane. The number of relations is 34 and is obtained by considering the collinearity of three points as the basic projective invariant. Our approach can be compared to existing models for cardinal and orientation relations, e.g., [13], [16] and their subsequent developments: These models consider binary relations that are calculated with respect to a particular frame of reference and consider point features or minimum bounding rectangles. In contrast, our model applies to all kinds of spatial features and is based on ternary relations, thus avoiding to consider a specific frame of reference; it is capable of taking the actual size and shape of objects into account for defining the relations: Specific frames of reference can be seen as special cases of our model.

Further, we discuss the algorithms to compute the relations over a vector data structure, where a region is represented as a polygon. We show that the operators can be computed in time $O(n \log n)$, when n is the total number of vertices of the three polygons involved in a ternary projective relation. Eventually, we describe an implementation of the algorithms, which corresponds to a simple kind of query, namely, "What is the projective relation among the three objects A, B, and C?"

The work presented in this paper is the basis for many other developments on projective relations. We can envisage several theoretical extensions to deal with other kinds of projective relations, such as n-ary relations and point-line relations. Relations such as "surrounded by" involve an arrangement of three or more reference objects and can be modelled as an extension of our model. Relations such as in

the example "shops in the right side of the road" are point-line relations and can also be modelled as an extension. Beyond traditional point, line, and region objects embedded in a 2D space, volumetric objects within a 3D space of reference can be envisaged too; the result would be a new set of projective relationships based on a segmentation of the space. The study of the formal properties of relations and the development of a reasoning system will be another major issue to expand: First results on composition tables for basic relations have been reported in [1]. The study of properties will be important for optimizing the computation of more complex kinds of queries and for integrating the projective relations as operators in a spatial database system.

ACKNOWLEDGMENTS

This work was supported by Italian M.I.U.R. under project "Representation and management of spatial and geographic data on the Web" and by the International Exchange Programme of the Royal Society of Edinburgh.

REFERENCES

- [1] R. Billen and E. Clementini, "Introducing a Reasoning System Based on Ternary Projective Relations," *Developments in Spatial Data Handling, Proc. 11th Int'l Symp. Spatial Data Handling*, pp. 381-394, 2004.
- [2] R. Billen and E. Clementini, "A Model for Ternary Projective Relations between Regions," *Proc. Ninth Int'l Conf. Extending DataBase Technology*, pp. 310-328, 2004.
- [3] R. Billen and E. Clementini, "Semantics of Collinearity among Regions," *Proc. First Int'l Workshop Semantic-Based Geographical Information Systems (SeBGIS '05)*, pp. 1066-1076, 2005.
- [4] E. Clementini and P. DiFelice, "Spatial Operators," *ACM SIGMOD Record*, vol. 29, pp. 31-38, 2000.
- [5] E. Clementini, P. DiFelice, and D. Hernández, "Qualitative Representation of Positional Information," *Artificial Intelligence*, vol. 95, pp. 317-356, 1997.
- [6] E. Clementini, P. DiFelice, and P. vanOosterom, "A Small Set of Formal Topological Relationships Suitable for End-User Interaction," *Proc. Third Int'l Symp. Advances in Spatial Databases (SSD '93)*, pp. 277-295, 1993.
- [7] E. Clementini, J. Sharma, and M.J. Egenhofer, "Modelling Topological Spatial Relations: Strategies for Query Processing," *Computers and Graphics*, vol. 18, pp. 815-822, 1994.

- [8] H.S.M. Coxeter, *Projective Geometry*, second ed. Springer-Verlag, 1987.
- [9] Deegree: A Free Software Project by the GIS and Remote Sensing Unit of the Dept. of Geography, Univ. of Bonn, <http://deegree.sourceforge.net/>, 2003.
- [10] V. Dugat, P. Garbarotto, and Y. Larvor, "Qualitative Theory of Shape and Orientation," *Proc. 16th Int'l Joint Conf. Artificial Intelligence (IJCAI '99)*, pp. 45-53, 1999.
- [11] M.J. Egenhofer, "Deriving the Composition of Binary Topological Relations," *J. Visual Languages and Computing*, vol. 5, pp. 133-149, 1994.
- [12] M.J. Egenhofer and R.D. Franzosa, "Point-Set Topological Spatial Relations," *Int'l J. Geographical Information Systems*, vol. 5, pp. 161-174, 1991.
- [13] C. Freksa, "Using Orientation Information for Qualitative Spatial Reasoning," *Theories and Models of Spatio-Temporal Reasoning in Geographic Space*, pp. 162-178, 1992.
- [14] K.-P. Gapp, "From Vision to Language: A Cognitive Approach to the Computation of Spatial Relations in 3D Space," *Proc. First European Conf. Cognitive Science in Industry*, pp. 339-357, 1994.
- [15] K.-P. Gapp, "Angle, Distance, Shape, and Their Relationship to Projective Relations," *Proc. 17th Conf. Cognitive Science Soc.*, pp. 112-117, 1995.
- [16] R. Goyal and M.J. Egenhofer, "The Direction-Relation Matrix: A Representation of Direction Relations for Extended Spatial Objects," *UCGIS Ann. Assembly and Summer Retreat*, 1997.
- [17] D. Hernández, *Qualitative Representation of Spatial Knowledge*. Springer-Verlag, 1994.
- [18] A. Isli, "Combining Cardinal Direction Relations and Other Orientation Relations in QSR," *Proc. Eighth Int'l Symp. Artificial Intelligence and Math.*, Jan. 2004.
- [19] A. Isli and A.G. Cohn, "A New Approach to Cyclic Ordering of 2D Orientations Using Ternary Relation Algebras," *Artificial Intelligence*, vol. 122, pp. 137-187, 2000.
- [20] D. Kirkpatrick and J. Snoeyink, "Computing Common Tangents without a Separating Line," *Proc. Workshop Algorithms and Data Structures*, pp. 183-193, 1995.
- [21] F. Klein, "Vergleicheinde Betrachtungen über Neuere Geometrischen Forschungen," *Bull. of the New York Math. Soc.*, vol. 2, pp. 215-249, 1893.
- [22] C. Kray and A. Blocher, "Modeling the Basic Meanings of Path Relations," *Proc. 16th Int'l Joint Conf. Artificial Intelligence (IJCAI-99)*, pp. 384-389, 1999.
- [23] L. Kulik, C. Eschenbach, C. Habel, and H.R. Schmidtke, "A Graded Approach to Directions between Extended Objects," *Proc. Second Int'l Conf. Geographic Information Science*, pp. 119-131, 2002.
- [24] L. Kulik and A. Klippel, "Reasoning about Cardinal Directions Using Grids as Qualitative Geographic Coordinates," *Proc. Int'l Conf. Spatial Information Theory. Cognitive and Computational Foundations of Geographic Information Science (COSIT '99)*, pp. 205-220, 1999.
- [25] A. Melkman, "On-Line Construction of the Convex Hull of a Simple Polygon," *Information Processing Letters*, vol. 25, pp. 11-12, 1987.
- [26] R. Moratz and K. Fischer, "Cognitively Adequate Modelling of Spatial Reference in Human-Robot Interaction," *Proc. 12th IEEE Int'l Conf. Tools with Artificial Intelligence (ICTAI '00)*, pp. 222-228, 2000.
- [27] OpenGIS Consortium, *OpenGIS Simple Features Specification for SQL*, 1998.
- [28] J. O'Rourke, *Computational Geometry in C*, second ed. Cambridge Univ. Press, 1998.
- [29] J. O'Rourke, C.-B. Chien, T. Olson, and T. Naddor, "A New Linear Algorithm for Intersecting Convex Polygons," *Computer Graphics and Image Processing*, vol. 19, pp. 384-391, 1982.
- [30] F. Preparata and S.J. Hong, "Convex Hulls of Finite Sets of Points in Two and Three Dimensions," *Comm. ACM*, vol. 20, pp. 87-93, 1977.
- [31] G. Retz-Schmidt, "Various Views on Spatial Prepositions," *AI Magazine*, vol. 9, pp. 95-105, 1988.
- [32] C. Schlieder, "Reasoning about Ordering," *Proc. Int'l Conf. Spatial Information Theory: A Theoretical Basis for GIS (COSIT '95)*, pp. 341-349, 1995.
- [33] H.R. Schmidtke, "The House Is North of the River: Relative Localization of Extended Objects," *Proc. Int'l Conf. Spatial Information Theory. Foundations of Geographic Information Science (COSIT '01)*, pp. 415-430, 2001.
- [34] A. Scivos and B. Nebel, "Double-Crossing: Decidability and Computational Complexity of a Qualitative Calculus for Navigation," *Proc. Int'l Conf. Spatial Information Theory. Foundations of Geographic Information Science (COSIT '01)*, pp. 431-446, 2001.
- [35] S. Skiadopoulos, C. Giannoukos, P. Vassiliadis, T. Sellis, and M. Koubarakis, "Computing and Handling Cardinal Direction Information," *Proc. Ninth Int'l Conf. Extending DataBase Technology*, pp. 329-347, 2004.
- [36] D. Sunday, "Fast Winding Number Inclusion of a Point in a Polygon," http://softsurfer.com/algorithm_archive.htm, 2004.
- [37] Vivid Solutions Inc., "JTS Topology Suite," <http://www.vivid-solutions.com/jts/JTSHome.htm>, 2004.
- [38] C. Vorwerg, G. Socher, T. Fuhr, G. Sagerer, and G. Rickheit, "Projective Relations for 3D Space: Computational Model, Application, and Psychological Evaluation," *Proc. 14th Nat'l Conf. Artificial Intelligence and Ninth Innovative Applications of Artificial Intelligence Conf. (AAAI '97, IAAI '97)*, pp. 159-164, 1997.
- [39] D. Waller, J.M. Loomis, R.G. Golledge, and A.C. Beall, "Place Learning in Humans: the Role of Distance and Direction Information," *Spatial Cognition and Computation*, vol. 2, pp. 333-354, 2000.



Eliseo Clementini received the DrIng degree in electronics engineering from the University of L'Aquila in 1990. Currently, he is an associate professor of computer science in the Electrical and Information Engineering Department at the University of L'Aquila. His research activity has been directed mainly to the field of spatial databases, where he has given original contributions toward conceptual data models for the representation of geographic information in object-oriented databases, spatial models for topological relations between geometric objects, optimization techniques for spatial query languages, consistency assessment among multiple representations of spatial data, qualitative models for spatial reasoning related to shape, distance and orientation, and models for the treatment of uncertainty in spatial data. He participated in several Italian research projects and collaborated with many international experts in the field of spatial databases. He was a visiting researcher in the Department of Geography and Geomatics at the University of Glasgow, UK, in the National Center for Geographic Information and Analysis at the University of Maine, in the Faculty of Informatics at the Technical University of Munich, Germany, in the Information Systems Center at the Hiroshima University, Japan, and in the Center for Systems Science of the Simon Fraser University, Canada. He has been a member of the program committee of several conferences in the fields of spatial databases and spatial reasoning.



Roland Billen received the DrSc degree in geomatics from the University of Liege in 2002 for his work on 3D spatial relationships and 3D data structures. He has been an associate professor of geomatics in the Department of Geography at the University of Liege since 2005. From 1998, he was a PhD candidate of the Belgian National Scientific Foundation (FNRS). He was appointed lecturer in the Department of Geographical and Earth Sciences (formerly Geography and Geomatics) of the University of Glasgow (Scotland, UK) in January 2003. He is active in several fields of spatial information sciences such as data acquisition techniques (photogrammetry, surveying, GNSS—GPS), GIS and spatial databases design, and qualitative spatial reasoning. He has been involved in several projects on spatial database design and has initiated research projects on 3D urban GIS. He was a visiting researcher at the GIS technology section of the TU Delft (The Netherlands) and in the Electrical Engineering Department at the University of L'Aquila (Italy). He has been guest editor of journal special issues on 3D GIS and spatial databases and has taken part in the organization of related conferences and workshops.