Dimension-Refined Topological Predicates

Mark McKenney, Alejandro Pauly, Reasey Praing & Markus Schneider [*] [†]
University of Florida
Department of Computer Science & Engineering
Gainesville, FL 32611, USA
{mm7,apauly,rpraing,mschneid}@cise.ufl.edu

ABSTRACT

Topological predicates, as derived from the 9-intersection model, have been widely recognized in GIS, spatial database systems, and many other geo-related disciplines. They are based on the evaluation of nine Boolean predicates checking the intersections of the boundary, interior, and exterior of a spatial object with the respective parts of another spatial object for inequality to the empty set. In this paper, we replace each Boolean predicate, which is a topological invariant, by another topological invariant. This new invariant is given as a function yielding the dimension of the respective intersection in the 9-intersection matrix, resulting in a *dimension matrix*. The goal of this paper is to determine the definition and semantics of all predicates that can be derived from this matrix for all combinations of spatial data types. It turns out that these dimension-based predicates are special refinements of the aforementioned topological predicates; hence, we call them *dimension-refined topological predicates*. We show that these predicates allow us to pose a class of more fine-grained topological queries.

Categories and Subject Descriptors

H.2.8 [**Database Management**]: Spatial databases and GIS

General Terms

Design

Keywords

Dimension, topological predicates, spatial databases, GIS, SPAL2D

1. INTRODUCTION

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

GIS'05, November 4, 2005, Bremen, Germany.

In recent years, topological relationships between objects in space have been extensively investigated in a number of disciplines like artificial intelligence, cognitive science, GIS, image databases, spatial database systems, and spatial reasoning, to name only a few. From a database and GIS perspective, their development has been motivated by the need of formally defined topological predicates as filter conditions for spatial selections and spatial joins in spatial query languages and as a support for spatial data retrieval and analysis tasks.

Topological relationships like *overlap*, *inside*, and *meet* characterize the relative position of spatial objects towards each other and are preserved under affine transformations such as translation, scaling, and rotation. Besides the *RCC model* [4], which leverages spatial logic, the 9-*intersection model* [5], which rests on point set theory and point set topology, has established itself as a widely recognized concept for the specification of topological relationships. This model checks the nine intersections of the boundary, interior, and exterior of a spatial object with the respective components of another spatial object for the topologically invariant criteria of non-emptiness. The intersections are organized in a 9-*intersection matrix*, in which each entry is a Boolean predicate checking an intersection for inequality to the empty set.

In this paper, we replace the topological invariant of emptiness and non-emptiness of an intersection by the topological invariant of the *dimension* of an intersection. The goal of this paper is to explore which *dimension predicates* can be derived on the basis of this topological invariant for all combinations of the spatial data types *point*, *line*, and *region*. It turns out that these predicates are *refinements* of the aforementioned topological predicates on spatial data types so that we denote them as *dimension-refined topological predicates*. We show how these predicates can be leveraged in queries which are intended to be more fine-grained than purely topological queries.

Section 2 discusses related work about spatial data types, topological relationships, and the related dimension extended method of Clementini and Di Felice. Section 3 gives a formal definition of spatial data types and specifies their boundary, interior, and exterior parts. In Section 4 we introduce the concept of dimension, the dimension matrix, and our strategy to derive dimension-refined predicates automatically. Section 5 describes in detail the dimension-refined topological predicates for all type combinations. In Section 6 we demonstrate how this kind of predicate enables us to pose more fine-grained topological queries. Section 7 compares our method to the so-called dimension-extended approach. Finally, Section 8 draws some conclusions.

2. RELATED WORK

In the past, numerous data models and query languages for spa-

	simple point	simple line	simple region
simple point	2	3	3
simple line	3	33	19
simple region	3	19	8

	complex point	complex line	complex region
complex point	5	14	7
complex line	14	82	43
complex region	7	43	33

Table 1: Numbers of topological predicates between two simple spatial objects (top) and between two complex spatial objects (bottom).

tial data have been proposed with the aim of formulating and processing spatial queries in databases and GIS. *Spatial data types* (e.g., [2, 6]) like *point*, *line*, and *region* are the central concept of these approaches. They provide fundamental abstractions for modeling the structure of geometric entities, their relationships, properties, and operations. Since they are the operands of our dimension predicates, we give some definitions in Section 3.

The amount of literature on topological relationships is vast. We only mention here two references which are sufficient for our goals in this paper. Based on the 9-intersection model, the work in [5] identifies the topological relationships for all combinations of *simple* spatial data types. The work in [7] does the same for all combinations of *complex* spatial data types. Table 1 shows the number of predicates for each type combination. Note that all topological predicates for simple types are contained in those for complex types. We will later show the relationship between topological and dimension-refined predicates.

An approach which is comparable to, but in the end rather different from, ours is the *dimension-extended method* described in [1, 3]. Both works are based on simple spatial data types and make use of a dimension matrix, which is defined in a slightly different way than ours, and a predicate derivation mechanism, which is fundamentally different from ours. Since this is an important method, and the only dimension-based approach so far, we describe it in detail in Section 7 and compare it to our approach.

3. SPATIAL DATA TYPES

We distinguish two *generations* of spatial data types. The types of the first generation have a simple structure . A *simple point* describes an element of the Euclidean plane $\mathbb{R}^2$. A *simple line* is a one-dimensional, continuous geometric structure embedded in $\mathbb{R}^2$ with two end points. A *simple region* is a two-dimensional point set in $\mathbb{R}^2$ and topologically equivalent to a closed disk.

Additional requirements of applications as well as needed closure properties of operations led to the second generation of *complex spatial data types* illustrated in Figure 1 (see [6] for a survey). A *complex point* is a finite point collection (e.g., to gather the positions of all lighthouses in Florida). A *complex line* is an arbitrary, finite collection of one-dimensional curves, i.e., a spatially embedded network possibly consisting of several disjoint connected components (e.g., to model the ramifications of the Nile Delta). A *complex region* permits multiple areal components, called *faces*, and holes in faces (e.g., Italy with its mainland and offshore islands as components and with the Vatican as a hole).

In the following, we give a formal definition of the three complex spatial data types, which comprises simple spatial data types as a special case. Since the 9-intersection model makes extensive use of the *interior*, *boundary* and *exterior* of spatial objects, we formally define these topological notions for each complex spatial data type.

3.1 Complex Points

A *complex point* (Figure 1(a)) is defined as a disconnected and finite set of points in the plane. Formally:

$$point = \{P \subset \mathbb{R}^2 \,|\, P \text{ is finite}\}$$

The first generation *simple point* is defined as a complex point for which $|P| = 1$. For the purpose of completeness the definition admits the empty set ($\varnothing$) since it can result from a geometric operation. The interior P°, boundary ∂P, and exterior P^- of a point $P = \{p_1, ..., p_n\}$ are defined as: $P^\circ = \bigcup_{i=1}^n p_i$, $\partial P = \varnothing$ and $P^- = \mathbb{R}^2 - (\partial P \cup P^\circ)$.

3.2 Complex Lines

A *complex line* (Figure 1(b)) is defined as the union of the images of a finite number of continuous mappings [7]. For a function $f : X \to Y$ and a set $A \subseteq X$ let $f(A) = \{f(x) \,|\, x \in A\}$. Let further $\mathbb{N}_0 = \mathbb{N} \cup \{0\}$. Then the spatial data type *line* is defined as

$$\begin{aligned} line = \{L \subset \mathbb{R}^2 \,|\, & \text{(i)} \;\; L = \textstyle\bigcup_{i=1}^n f_i([0,1]) \text{ with } n \in \mathbb{N}_0 \\ & \text{(ii)} \;\; \forall 1 \le i \le n : f_i : [0,1] \to \mathbb{R}^2 \text{ is a} \\ & \qquad \text{continuous mapping} \\ & \text{(iii)} \; \forall 1 \le i < j \le n : \\ & \qquad f_i([0,1]) \cap f_j(]0,1[) = \varnothing \wedge \\ & \qquad f_i(]0,1[) \cap f_j([0,1]) = \varnothing \\ & \text{(iv)} \; \forall 1 \le i \le n : |f_i([0,1])| > 1\} \end{aligned}$$

Each continuous mapping f_i describes a *single-component* (simple) line with two end points $f(0)$ and $f(1)$. The first condition also allows a line object to be the empty set ($n = 0$). The third condition supports uniqueness of representation and prevents that a single-component line intersects the interior of another single-component line. The fourth condition avoids degenerate line objects consisting only of a single point.

Let $E(L) = \bigcup_{i=1}^n \{f_i(0), f_i(1)\}$ be the set of end points of all single-component lines. The set $E(L)$ forms the basis for the definition of the boundary of a complex line. But several single component lines may share a common end point. This shared end point belongs to the interior of a complex line and thus not to the boundary. We obtain:

$$\begin{aligned} \partial L = & E(L) - \\ & \{p \in E(L) \,|\, card(\{f_i \,|\, 1 \le i \le m \;\wedge\; f_i(0) = p\}) + \\ & \qquad card(\{f_i \,|\, 1 \le i \le m \;\wedge\; f_i(1) = p\}) \ne 1\} \end{aligned}$$

Let $L \ne \varnothing$. It is possible that ∂L is empty. The closure $\overline{L}$ of L is the set of all points of L including the end points. Therefore $\overline{L} = L$ holds. For the interior of L we obtain $L^\circ = \overline{L} - \partial L = L - \partial L \ne \varnothing$, and for the exterior we get $L^- = \mathbb{R}^2 - L$.

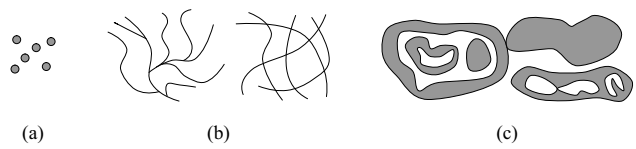

Figure 1: Examples of a complex point object (a), a complex line object (b), and a complex region object (c).

3.3 Complex Regions

An *complex region* (Figure 1(c)) is defined as a finite set of disjoint, connected areal components which we call *faces*. Each face can have zero or more disjoint holes. The spatial data type *region* is defined as

$$\begin{aligned} region = \{R \subset \mathbb{R}^2 \mid & \text{(i) } R \text{ is regular closed} \\ & \text{(ii) } R \text{ is bounded} \\ & \text{(iii) The number of connected sets of } R \\ & \quad \text{is finite}\} \end{aligned}$$

The definition of *regular closed* sets [8] ensures that a region object does not have dangling points and lines, does not have cuts and punctures, and has a complete boundary. Let $q = (x,y)$, a set $X \subseteq \mathbb{R}^2$ is said to be *bounded* if there exists a number $r \in \mathbb{R}^+$ such that $\sqrt{x^2+y^2} < r$ for every $q \in X$. Finally, a separation of X is a pair A, B satisfying the following three conditions: (1) $A \neq \emptyset$ and $B \neq \emptyset$, (2) $A \cup B = X$, and (3) $\overline{A} \cap B = \emptyset$ and $A \cap \overline{B} = \emptyset$. Hence, a set $X \subseteq \mathbb{R}^2$ is said to be *disconnected* if a separation of X exists, otherwise it is said to be *connected*.

For a region X, its *interior* X° is defined as the union of all interior points. A point q is an *interior point* of X if there exists a neighborhood N such that $N(q) \subseteq X$. In other words, all interior points of X are completely surrounded by points in X. A point q is defined as an exterior point of X if there exists a neighborhood N such that $N(q) \cap X = \varnothing$. The exterior of X denoted X^- is composed of the union of all exterior points of X. The boundary of X denoted ∂X is the union of all points that are not interior or exterior points of X.

4. THE DIMENSION MATRIX

Let us assume two spatial objects $A \in \alpha$ and $B \in \beta$ with $\alpha, \beta \in \{point, line, region\}$. Topological predicates, as they have been defined on the basis of the 9-*intersection topological matrix* (*9ITM*), test each of the nine intersections $A^\circ \cap B^\circ$, $A^\circ \cap \partial B$, $A^\circ \cap B^-$, $\partial A \cap B^\circ$, $\partial A \cap \partial B$, $\partial A \cap B^-$, $A^- \cap B^\circ$, $A^- \cap \partial B$, and $A^- \cap B^-$ with regard to the topologically invariant criterion of non-emptiness. An interesting research topic is to explore the effect of a replacement of this topological invariant by another one, such as the *dimension* of each intersection. A first issue is which new kind of predicate results from this replacement and what its features are. A second issue is which predicates can be derived for each combination of spatial data types. A third issue refers to the relationship between topological predicates and these new predicates. Finally, a fourth, and the most important, issue is what benefits we gain from the new predicates for applications and querying.

In Section 4.1, we give a definition of the concept of dimension, as it is appropriate for our purpose. Section 4.2 introduces the 9-*intersection dimension matrix* (*9IDM*) based on this definition and shows how it is leveraged as part of a general strategy for discovering *dimension predicates*.

4.1 Modeling A Concept of Dimension

Topology, a self-contained branch of mathematics, provides a concept for defining the dimension of a topological space. Let X be a set and $T \subseteq 2^X$ a subset of the power set of X. The pair (X,T) is called a *topological space*, if the following three axioms are satisfied: (i) $X \in T$, $\varnothing \in T$, (ii) $U,V \in T \Rightarrow U \cap V \in T$, and (iii) $S \subseteq T \Rightarrow \bigcup_{A \in S} A \in T$. T is called a *topology* on X. The elements in T are called *open sets*; their complements in X are called *closed sets*. Let further $C = \{U_i \mid i \in I, U_i \in T\}$ be an indexed family of open subsets of X for a given index set I. Then C is called an *open cover* of X if $\bigcup_{i \in I} U_i = X$. An open cover $D = \{V_j \mid j \in J, V_j \in T\}$ for some index set J is a *refinement* of the open cover $C = \{U_i \mid i \in I, U_i \in T\}$ if for any V_j there exists some U_i such that $V_j \subseteq U_i$. Finally, the *Lebesgue covering dimension* (*LCD*) (or *topological dimension*) of a topological space is defined to be the minimum value of n, such that any open cover has a refinement with no point included in more than $n+1$ elements.

To illustrate the concept, consider open covers of the unit circle, given by open circles. The circle has LCD 1, since any such cover can be refined to the stage where a given point p of the unit circle is contained in *at most* two open circles. That is, whatever open circles we begin with, enough can be discarded so that there are just simple overlaps.

In our case, where $X = \mathbb{R}^2$ is the two-dimensional Euclidean plane, we determine the Lebesgue covering dimension by a function $LCD : 2^{\mathbb{R}^2} \rightarrow \{0,1,2\}$. That is, for each subset $U \in 2^{\mathbb{R}^2}$, *LCD* yields its dimension $k \in \{0,1,2\}$. Note that U can contain *maximal connected components* of smaller dimension. This cannot be directly detected by the function *LCD*. As an example, assume that U consists of a circle and some isolated points outside the circle. The LCD of U is 1, but the LCD of each isolated point as a maximal connected component of U is 0.

In our approach, we would like to be able to differentiate the different dimensions of maximal connected components in a given point set. Thus, we define a more detailed "dimension type" as follows:

$$dimType = \{\perp, 0D, 1D, 2D, 01D, 02D, 12D, 012D\}$$

This type allows us to represent the dimension of a given point set as the "union" of the dimensions of its maximal connected components. If a point set consists only of single points, only of lines, or only of regions, the corresponding value of type *dimType* is $0D$, $1D$, or $2D$ respectively. If a point set contains maximal connected components of different dimension, the corresponding value is $01D$, $02D$, $12D$, and $012D$ respectively. The $\perp$ symbol is used to represent the undefined dimension of an empty point set.

Since the function *LCD* applied to a point set cannot produce a value of type *dimType*, we define another function $dim : 2^{\mathbb{R}^2} \rightarrow dimType$, which is based on the function *LCD* and returns the corresponding value of type *dimType* for this point set. Let $P \in 2^{\mathbb{R}^2}$, and let $\{P_i \mid i \in I, P_i \subseteq P\}$ be the indexed family of maximal connected components of P for a given index set I. This means that (i) $P = \bigcup_{i \in I} P_i$ and (ii) $\forall i, j \in I, i \neq j : P_i \cap P_j = \varnothing$. We are now able to define the semantics of the function *dim*.

$$dim(P) = \begin{cases} \perp & \text{if } P = \varnothing \\ 0D & \text{if } LCD(P) = 0 \\ 1D & \text{if } LCD(P) = 1 \wedge \nexists i \in I : LCD(P_i) = 0 \\ 2D & \text{if } LCD(P) = 2 \wedge \nexists i \in I : LCD(P_i) = 1 \\ & \wedge \nexists j \in I : LCD(P_j) = 0 \\ 01D & \text{if } LCD(P) = 1 \wedge \exists i \in I : LCD(P_i) = 0 \\ 02D & \text{if } LCD(P) = 2 \wedge \nexists i \in I : LCD(P_i) = 1 \\ & \wedge \exists j \in I : LCD(P_j) = 0 \\ 12D & \text{if } LCD(P) = 2 \wedge \exists i \in I : LCD(P_i) = 1 \\ & \wedge \nexists j \in I : LCD(P_j) = 0 \\ 012D & \text{if } LCD(P) = 2 \wedge \exists i \in I : LCD(P_i) = 1 \\ & \wedge \exists j \in I : LCD(P_j) = 0 \end{cases}$$

This function allows us in a precise way to determine all dimensions of maximal connected components in a given point set.

4.2 Discovering Dimension Predicates

Based on the function *dim*, we introduce the 9-*intersection dimension matrix* (*9IDM*) in Table 2. Each entry of this matrix (a value of type *dimType*) represents the dimension of one of the nine component intersections of two spatial objects A and B.

$$\begin{pmatrix} dim(A^\circ \cap B^\circ) & dim(A^\circ \cap \partial B) & dim(A^\circ \cap B^-) \\ dim(\partial A \cap B^\circ) & dim(\partial A \cap \partial B) & dim(\partial A \cap B^-) \\ dim(A^- \cap B^\circ) & dim(A^- \cap \partial B) & dim(A^- \cap B^-) \end{pmatrix}$$

Table 2: The 9-intersection dimension matrix

The maximum number of dimension matrices is $|dimType|^9 = 134,217,728$. However, many of these matrices do not represent valid dimension predicates. This results from three main observations that we will use as a basis for developing a general strategy to systematically determine valid dimension predicates. This strategy holds for any two objects of equal or different spatial data type but has to be executed for each type combination individually; this is performed in Section 5.

The first observation establishes an important and fundamental relationship between dimension predicates and topological predicates. It states that a dimension matrix and thus a dimension predicate is only valid if, by using an appropriate mapping, a corresponding and already known topological matrix and thus topological predicate is valid. Let $T = [b_{ij}]$ be a 9ITM, and let $D = [d_{ij}]$ be a 9IDM. We know that $b_{ij} \in \{true, false\}$ and $d_{ij} \in dimType$. The following correspondence holds between b_{ij} and d_{ij} (for b_{ij} the Boolean value *true* (*false*) is represented by t (f)):

$$\begin{array}{lcl} b_{ij} = f & \Leftrightarrow & d_{ij} = \perp \\ b_{ij} = t & \Leftrightarrow & d_{ij} \in \{0D, 1D, 2D, 01D, 02D, 12D, 012D\} \end{array}$$

This correspondence allows us to define a function $r : dimType \rightarrow bool$ (but not vice versa), which maps (reduces) each dimension value uniquely to a Boolean value. This leads us to a necessary but insufficient condition for the validity of a dimension matrix and thus predicate:

$$D = [d_{ij}] \text{ is valid} \quad \Rightarrow \quad T = [r(d_{ij})] \text{ is valid}$$

That is, if T is invalid (for a particular type combination), then D cannot be a valid dimension matrix. This excludes a large number of invalid dimension matrices but does on the other hand not yield us all valid ones. The reason is that it is still possible that an invalid dimension matrix corresponds to a valid topological matrix. To illustrate these concepts, we consider the following 9IDMs D_1, D_2, and D_3 as well as the 9ITMs T_1 and T_2 for two simple region objects:

$$\underset{D_1}{\begin{pmatrix} 2D & \perp & \perp \\ \perp & 1D & \perp \\ \perp & \perp & 2D \end{pmatrix}} \quad \underset{D_2}{\begin{pmatrix} 2D & \perp & \perp \\ \perp & 2D & \perp \\ \perp & \perp & 2D \end{pmatrix}} \quad \underset{D_3}{\begin{pmatrix} 2D & \perp & \perp \\ \perp & 12D & \perp \\ \perp & \perp & \perp \end{pmatrix}}$$

$$\underset{T_1}{\begin{pmatrix} t & f & f \\ f & t & f \\ f & f & t \end{pmatrix}} \quad \underset{T_2}{\begin{pmatrix} t & f & f \\ f & t & f \\ f & f & f \end{pmatrix}}$$

T_1 represents the well known topological predicate *equal*. We know that T_2 does not represent *any* topological predicate, since the intersection of the exterior of any two spatial objects is always non-empty. Hence, T_1 is valid, and T_2 is invalid. Both D_1 and D_2 map to T_1, but only D_1 is a valid dimension matrix. D_1 is valid, since in the case of two equal simple region objects the dimension of the intersection of their interiors is $2D$ only, the dimension of the intersection of their boundaries is $1D$ only, the dimension of the intersection of their exteriors is $2D$ only, and all other dimensions are undefined. The difference between D_1 and D_2 only refers to the dimension of the regions' boundaries. But this dimension cannot be $2D$, since the boundaries themselves are only one-dimensional. Hence, D_2 is invalid, although it maps to T_1. D_3 maps to T_2. But since T_2 is invalid, D_3 cannot be valid. If D_3 was valid, T_2 would be valid too.

$dim(X \cap Y) \in$		$LCD(Y) =$ 0	1	2
$LCD(X) =$	0	$\{\perp, 0D\}$	$\{\perp, 0D\}$	$\{\perp, 0D\}$
	1	$\{\perp, 0D\}$	$\{\perp, 0D, 01D, 1D\}$	$\{\perp, 1D\}$
	2	$\{\perp, 0D\}$	$\{\perp, 1D\}$	$\{\perp, 2D\}$

Table 3: Determination of $dim(X \cap Y)$.

If we consider the inverse of the function r, we obtain the interesting result of a 1-to-many relation from the topological predicates to the dimension predicates. That is, each topological predicate can be associated with one or more dimension predicates. Consequently, dimension predicates turn out to be *refinements* of topological matrices. Therefore, we refer to dimension predicates as *dimension-refined topological predicates* (*DRTP*).

Regarding our strategy to discover all dimension predicates this means that we do not have to reduce the large number of possible dimension matrices by "cancellation rules". Instead, we can start with the known topological predicates for each type combination and design rules for deriving the valid dimension predicates from them. This is one of the basic differences to the approaches in [1, 3] (see the discussion in Section 7). This strategy is also independent of the involvement of *simple* or *complex* spatial data types.

The second observation is that the dimension of each intersection cannot take all values of type *dimType* but depends on the dimensions of its argument objects. Topology [9] helps us determine upper and lower bounds for the dimension of the intersection of two subsets $X, Y \subset \mathbb{R}^2$. An upper bound is given as

$$LCD(X \cap Y) \leq \min(LCD(X), LCD(Y))$$

and a lower bound is given as

$$LCD(X \cap Y) \geq LCD(X) + LCD(Y) - n$$

where n is the dimension of the embedding space. In our case, $n = 2$ holds, since the embedding space is the Euclidean plane $\mathbb{R}^2$. If we assume that both $LCD(X)$ and $LCD(Y)$ traverse the Lebesgue dimension values 0, 1, and 2, we obtain nine possible combinations of dimension values. Table 3 determines the upper and lower dimension bounds of $X \cap Y$ for each combination of dimensions of X and Y and expresses the result in terms of values of type *dimType*. A special case is $X \cap Y = \varnothing$ so that $dim(X \cap Y) = \perp$ has to be added to each dimension result set. As Table 3 indicates, for $X \cap Y \neq \varnothing$ we only obtain more than one dimension (in addition to the $\perp$ dimension) if $LCD(X) = 1$ and $LCD(Y) = 1$. That is, only linear components can provide for dimension alternatives. For all other combinations of Lebesgue dimensions, the resulting dimension is unique.

The third observation is that the possible set of dimensions at a position in the dimension matrix can be restricted by other entries of the same matrix due to interdependencies. Due to Table 3, we know that we only have to look for situations that restrict the dimension of the intersection of two one-dimensional object parts.

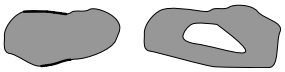

Figure 2: Example of the containment rule: The one-dimensional part (interior) of a complex line (bold) is a subset of the one-dimensional part (boundary) of a complex region.

Those restrictions that are of interest limit the number of possible dimensions at a position in the dimension matrix to less than 3. As an example, consider two simple regions A and B which are equal. The fact that $\partial A \subseteq \partial B$ implies that the dimension of their intersection must be equal to $1D$. We have identified two restrictions. The first restriction is referred to as the *containment rule* (see example above): Let us assume a valid topological predicate between two objects A and B. Let C (D) be a one-dimensional object part (boundary of a region, interior of a line) of object A (B). Then

$$C \subseteq D \Rightarrow C \cap D = C \Rightarrow dim(C \cap D) = dim(C) = 1D$$

The open question is now how such a containment relationship can be detected in a 9ITM. Given a spatial object A, we know that $\mathbb{R}^2 = \partial A \cup A^\circ \cup A^-$, $\partial A \cap A^\circ = \varnothing$, $\partial A \cap A^- = \varnothing$, and $A^\circ \cap A^- = \varnothing$ (correspondingly for a spatial object B). Let $A^{1D}, A^i, A^j \in \{A^\circ, \partial A, A^-\}$ with $A^{1D} \neq A^i \neq A^j$ and $B^{1D}, B^k, B^l \in \{B^\circ, \partial B, B^-\}$ with $B^{1D} \neq B^k \neq B^l$. A^{1D} and B^{1D} shall denote the one-dimensional parts of A and B respectively. That is, $A^{1D} = \partial A$ if $A \in$ *region*, and $A^{1D} = A^\circ$ if $A \in$ *line* (correspondingly for B). Hence,

$$A^{1D} \subseteq B^{1D} \Leftrightarrow A^{1D} \cap B^{1D} \neq \varnothing \wedge A^{1D} \cap B^k = \varnothing \wedge A^{1D} \cap B^l = \varnothing$$

The second restriction forces two $1D$ components to have either a $01D$ or a $1D$ intersection. This restriction applies to two objects that share an area (either interior or exterior) which is enclosed by a boundary that exists completely in the intersection of the $1D$ components of the objects. In this case the boundary enclosing the shared area will always have dimension value $1D$, therefore an exclusive $0D$ refinement of this predicate is not possible.

We refer to this second restriction as the *"Shared-Boundary Containment"* (SBC) restriction. The SBC restriction only applies to spatial objects that consist of a one-dimensional component which separates two two-dimensional components. In our current spatial object models, only the region type is applicable. Given two spatial objects A and B as defined earlier for the first restriction, let A^{1D} and B^{1D} denote the one-dimensional parts of A and B respectively. A^i, A^j and B^k, B^l denote the two-dimensional components of A and B respectively. That is, $A^{1D} = \partial A$, $A^i = A^\circ$, $A^j = A^-$ if $A \in$ *region* (correspondingly for B). Hence, if there is an intersection Q between any of the two-dimensional components, say A^i and B^k, but no intersection between A^{1D} and B^k or B^{1D} and A^i, then Q must be completely enclosed (in the case of interiors) or limited (exteriors) by the intersection of A^{1D} and B^{1D}. Formally,

$$A^i \cap B^k = Q \wedge Q \neq \varnothing \wedge A^{1D} \cap B^k = \varnothing \wedge B^{1D} \cap A^i = \varnothing \Rightarrow$$
$$A^{1D} \cap B^{1D} \neq \varnothing \wedge dim(A^{1D} \cap B^{1D}) = \{01D, 1D\}$$

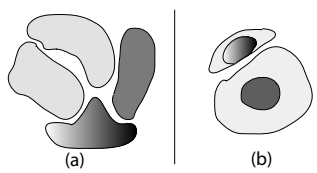

Figure 3: Examples of the shared-boundary containment: In each figure, lightly shaded objects are part of one complex region while darker objects belong to the second complex region. The gradient filled object is an overlapping area of both complex regions.

Two examples of object configurations that satisfy this condition are shown in Figure 3. Configuration (a) of this figure shows two complex regions that each contain an identical area. The boundary of that common area is completely contained in the intersection of the boundaries of both complex objects. This boundary encloses exclusively a single, two-dimensional interior area from each object. Configuration (b) is less intuitive. The boundary of the circular dark area and the boundary of the hole in the larger light area are identical, and therefore exist in the intersection of the boundaries of both objects. This boundary encloses a two-dimensional area that includes exclusively a part of the interior of one object and a part of the exterior of the other object. There is no topologically equivalent configuration of either of these complex object configurations that allows the boundaries in question to intersect at only points. Thus, the topological predicate representing these configurations can only be refined to $01D$ and $1D$ versions.

In practice, this restriction holds for a very small set of topological predicates. Simple spatial objects do not meet the criteria for this restriction, and from the complex objects, there are only four topological matrices for complex regions that match it.

Based on these three observations, algorithm *Discover9IDMS* presents an algorithm for discovering dimension-refined topological predicates. First, the dimensions of the interior, boundary, and exterior are determined for both argument data types by employing non-empty example objects. Then a new dimension matrix is constructed. Afterwards, we traverse all topological matrices. For each matrix we check each entry. If the entry has the value *false*, the corresponding dimension value is $\perp$. Otherwise, we know the value is *true*, and we test whether the two participating object parts are both one-dimensional. For each combination of spatial data types, there is only one combination of object parts such that both parts are one-dimensional. The combination of object parts is saved for later until the other entries of the dimension matrix have been determined. If at least one participating object has a dimension unequal to 1, a look-up in Table 3 reveals the correct dimension of the intersection. Finally, the special case of two one-dimensional object parts is dealt with. If the matrix matches the containment restriction, only a single matrix is constructed. If the matrix does not match the containment restriction, but matches the shared-boundary containment restriction, only two matrices are constructed ($01D$ and $1D$). Finally, if neither restriction holds, three matrices are constructed, and the dimensions $0D$, $01D$, and $1D$ are assigned to them.

Algorithm *Discover9IDMs*
input: (i) Spatial data types α and β,
(ii) complete and sound collection S of 9ITMs for α and β
output: Complete and sound set D of valid 9IDMs for α and β
begin
  $D := \varnothing$; let $A \in \alpha, A \neq \varnothing$; let $B \in \beta, B \neq \varnothing$;
  $a_0 := LCD(A^\circ)$; $a_1 := LCD(\partial A)$; $a_2 := LCD(A^-)$;
  $b_0 := LCD(B^\circ)$; $b_1 := LCD(\partial B)$; $b_2 := LCD(B^-)$;
  for each $p \in S$ **do**
    $r := 9IDM()$; /* Construct new 9IDM */
    numDM := 1; /* The number of DMs that have to be created */
    for each $p_{i,j} \in p$ **do** /* $0 \leq i, j \leq 2$ */

```
        if p_{i,j} = f then r_{i,j} := ⊥;
        else if a_i = 1 and b_j = 1 then
            if (p_{i,(j+1)%3} = f and p_{i,(j+2)%3} = f) or
                (p_{(i+1)%3,j} = f and p_{(i+2)%3,j} = f) then
                r_{i,j} := 1D; /* Containment rule */
            else if a_0 = 2 and /* Shared-boundary containment */
                b_0 = 2 and
                ((p_{0,0} = t and p_{0,1} = f and p_{1,0} = f) or
                (p_{0,2} = t and p_{0,1} = f and p_{1,2} = f) or
                (p_{2,2} = t and p_{1,2} = f and p_{2,1} = f) or
                (p_{2,0} = t and p_{1,0} = f and p_{2,1} = f)) then
                numDM := 2; k := i; l := j; /* Special case */
            else numDM := 3; k := i; l := j; /* Special case */
            endif;
        else r_{i,j} := getDim(a_i, b_j);   /* Look-up in Table 3 */
        endif;
    endfor;
    if numDM ≥ 2 then
        s := 9IDM(); s := r; r_{k,l} := 1D; s_{k,l} := 01D;
        insert(D, s); /* insert element s into set D */
    endif;
    if numDM = 3 then
        t := 9IDM(); t := r; t_{k,l} := 0D; insert(D, t);
    endif;
    insert(D, r);
  endfor;
end Discover9IDMs
```

5. DIMENSION-REFINED TOPOLOGICAL PREDICATES

The dimension refinement strategy is independent of the complexity of the spatial object model as well as the type combination. Therefore, we can apply our strategy to all type combinations of both simple and complex spatial objects. In Section 5.1, we discover all dimension refined topological predicates (DRTPs) between simple spatial objects by applying Algorithm *Discover9IDMs* to the set of topological matrices for each simple spatial type combination specified in [5]. In Section 5.2, the set of topological matrices for each complex spatial type combination specified in [7] is used to discover all DRTPs between complex spatial objects.

5.1 Dimension Refined Topological Predicates between Simple Spatial Data Types

The number of predicates for each simple spatial type combination is shown in Table 4. Executing Algorithm *Discover9IDMs* on each set of matrices for *point* × *point*, *point* × *line*, and *point* × *region*, produces a new set of DRTMs which has the same number of matrices as the original set of 9ITMs. This is because point objects have no one-dimensional component, therefore the algorithm only generates a single DRTM from a single 9ITM. On the other hand, this is not the case for *line* × *line*, *line* × *region*, and *region* × *region* combinations. For *line* × *line*, the one-dimensional intersection is between the interiors of the lines. For *line* × *region*, the interior of the line and the boundary of the region form the one-dimensional intersection. The one-dimensional intersection occurs between boundaries of regions for the *region* × *region* combination. Table 4 shows the number of single predicate refinements, multi-predicate refinements, and the total number of DRTMs and thus DRTPs discovered by the algorithm. Tables 6, 7 and 8 illustrate the DRTMs and geometric interpretations resulting from multi-predicate refinements for *line* × *line*, *line* × *region*, and *region* × *region* combinations respectively. For the purpose of simplicity, converse cases, e.g. coveredBy, are not shown. The DRTMs which result from single predicate refinements can be trivially constructed by simple look-ups in Table 3. Their corresponding geometric interpretations are exactly the same as those of the original topological predicates in [5]. Hence, due to space constraints, these DRTMs are not shown.

Type	TP	SP	MP	Total DRTPs
point × *point*	2	2	0	$2+(0\cdot 3)=2$
point × *line*	3	3	0	$3+(0\cdot 3)=3$
point × *region*	3	3	0	$3+(0\cdot 3)=3$
line × *line*	33	19	14	$19+(14\cdot 3)=61$
line × *region*	19	7	12	$7+(12\cdot 3)=43$
region × *region*	8	4	4	$4+(4\cdot 3)=16$

Table 4: The number of DRTPs for each simple spatial data type combination. The single predicate column (SP) indicates the number of topological predicates (TP) that refine to a single DRTP. The multi-predicate column (MP) indicates the number of topological predicates that refine to three DRTPs.

Type	TP	SP	MP	Total DRTPs
point × *point*	5	5	0	$5+(0\cdot 3)=5$
point × *line*	14	14	0	$14+(0\cdot 3)=14$
point × *region*	7	7	0	$7+(0\cdot 3)=7$
line × *line*	82	50	32	$50+(32\cdot 3)=146$
line × *region*	43	27	16	$27+(16\cdot 3)=75$
region × *region*	33	21	12	$21+(8\cdot 3+4\cdot 2)=53$

Table 5: The number of DRTPs for each complex spatial data type combination.

5.2 Dimension Refined Topological Predicates between Complex Spatial Data Types

The number of topological predicates in each type combination for complex spatial objects is shown in Table 5. Applying the refinement algorithm to each of these type combinations allows us to identify single and multiple refinements, thus producing a new set of DRTMs. Each of the multi-predicate refinements produces 3 DRTMs except 4 of the 12 multi-predicate refinements for complex regions, which produce only 2 DRTMs. These 4 refinements do not satisfy the first containment restriction rule. However, they satisfy the second shared-boundary containment rule which limits their results to only 2 DRTMs for each refinement. Due to space constraints, we omit the matrices and geometric representations of these predicates.

6. QUERYING

In this section we demonstrate the significance and the applicability of the dimension-refined topological predicates by its use in a database query language. When posing a query to the database, the user must be able to be as specific as possible in terms of the desired dimension of the results. Ideally, the query shall be written in a SQL-like language, and the questions about dimensionally aware relations shall be expressed by predicates.

Before we can pose queries containing dimension-refined topological predicates, we need a naming convention for them. Common topological predicates such as *overlap* or *meet* have well known

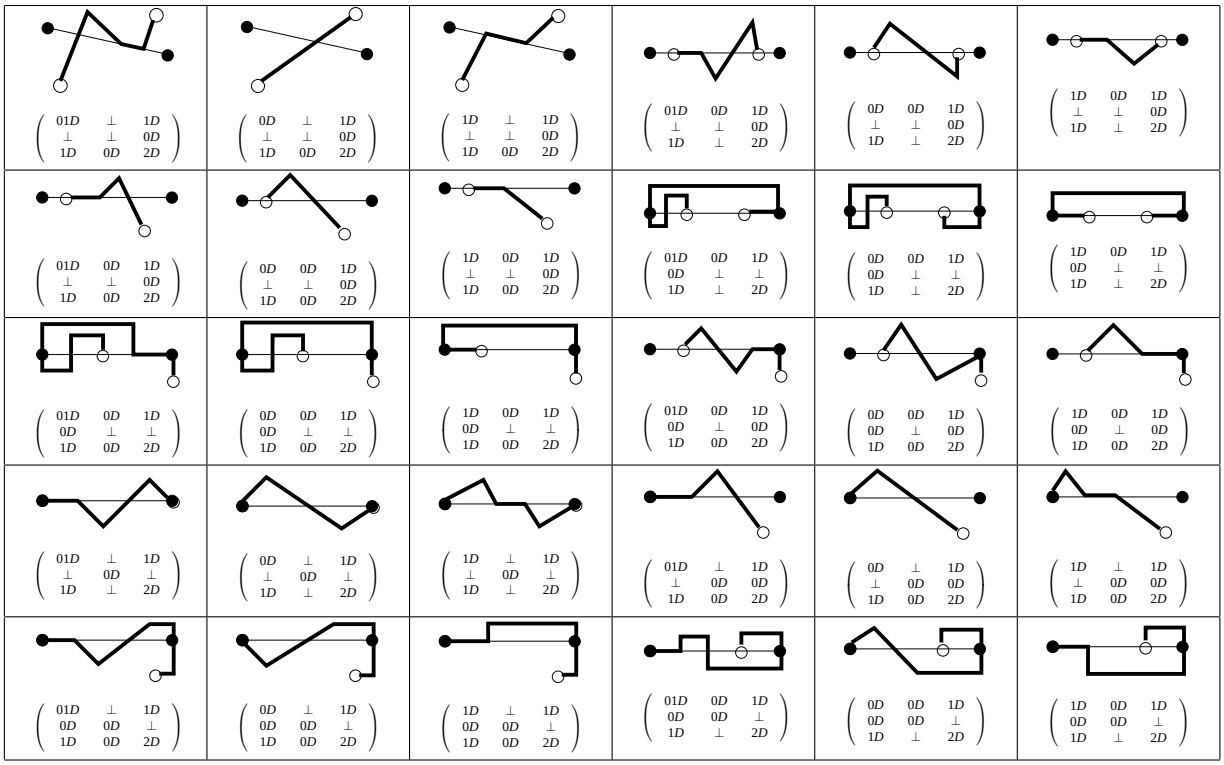

Table 6: Geometric representations of 30 DRTPs resulted from 10 multi-predicate refinements on $line \times line$. The converse cases which consist of 12 DRTPs are not shown. The thin and thick line objects correspond to the row and column of the matrices respectively.

semantics and are generally understood and widely accepted by users. Our naming convention takes advantage of what the user has already learned. We name dimension-refined topological predicates based on their corresponding topological predicate but differentiate them from each other by adding their dimension specific information at the beginning of the name. For example, *0D-meet* between two meeting regions denotes the case in which the boundaries of the regions intersect at a single point or at multiple, disconnected points (dimension 0D). A *1D-meet* between two meeting regions represents the case where the boundaries share a common line or multiple common lines (dimension 1D). In Section 4, we have shown how, for refining purposes, only the non-empty intersection between two $1D$ objects might be of interest. For all cases in which $1D$ parts do not intersect, and for all cases for which the restriction rules allow only one possible dimension as the result from the intersection of two $1D$ parts, the dimension-refined predicate is named exactly the same as its corresponding topological predicate. In such cases, no actual refinement occurs and the relationship between the dimension-refined predicate and the topological predicate is one-to-one. For all the other cases, we apply a general dimension-refined naming for each predicate generated by the algorithm such that for a topological predicate P, the dimension-refined predicates *0D-P*, *1D-P* and *01D-P* are generated. The predicate P itself can be regarded as a generalization that does not include an explicit dimension aspect. Thus, it can be expressed by the disjunction of its dimension-refined topological predicates. For a topological predicate P between spatial objects A and B, we obtain

$$P(A,B) = 0D\text{-}P(A,B) \ \vee \ 1D\text{-}P(A,B) \ \vee \ 01D\text{-}P(A,B).$$

We proceed by showing, by way of example queries in sample scenarios, how the user can benefit from using dimension-refined topological predicates as opposed to using only topological predicates. The first sample scenario (see Figure 4) involves rivers and some ideas on how flooding might occur when rain increases. Studies are able to distinguish the flooding risk two rivers might pose when they *meet* each other. Such flooding risks increase when the two rivers share a common part of their path and should not be as important when the meeting situation only involves single points. If resources are scarce and we can only take care of increased risk flooding areas, we can query the database to return those pairs of rivers for which an increased risk exists. In this way, people in the surrounding areas can be kept safe. The query for this can be written as:

```
SELECT a.name, b.name
FROM   rivers a, rivers b
WHERE  1D-meet(a.path,b.path) OR
       01D-meet(a.path,b.path);
```

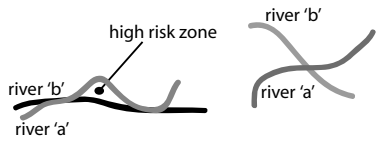

Figure 4: Two river scenarios, one with increased flood risk (left) and another with no increased flood risk (right).

It is possible that the existence of both a shared path and a single meeting point might create a special situation that reflects maxi-

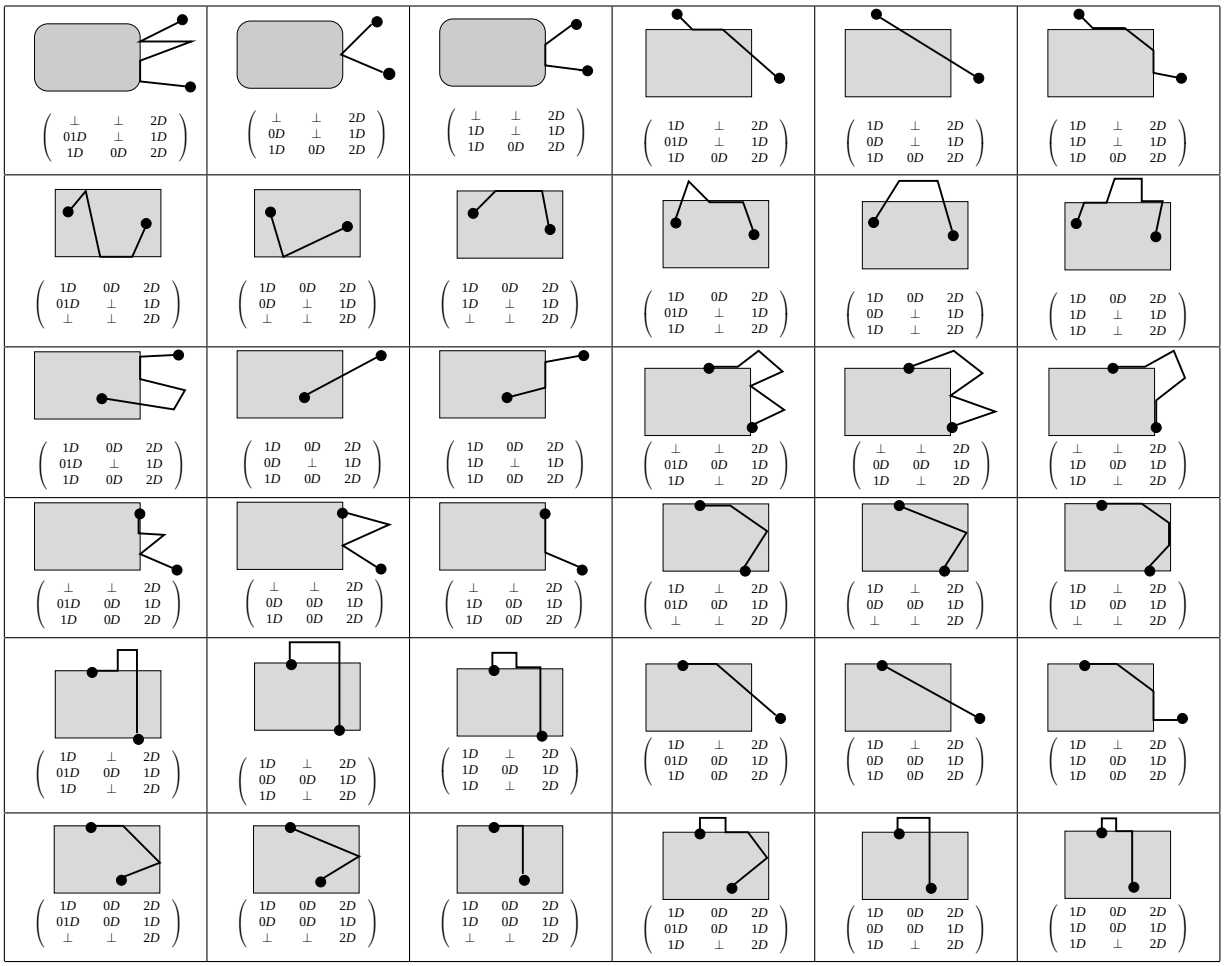

Table 7: Geometric representations of 36 DRTPs resulted from 12 multi-predicate refinements on $line \times region$**. The region and the line objects correspond to the row and column of the matrices respectively.**

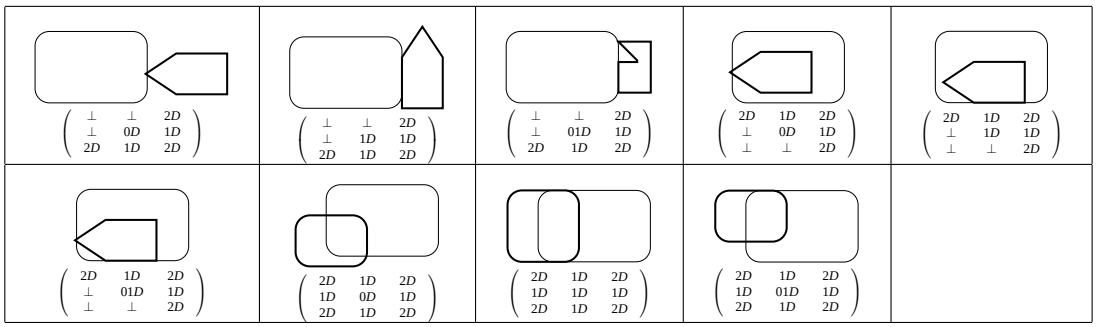

Table 8: Geometric representation of 9 DRTPs resulted from 3 multi-predicate refinements on $region \times region$**. Converse case (coveredBy) is not shown. From left to right and top to bottom in groups of three, each group depicts DRTPs for meet, covers, and overlap respectively. The thin and thick stroke objects correspond to the row and column of the matrices respectively.**

mum flooding risk for the area that is enclosed by both rivers in between the shared line and the meeting point. Such cases can be uniquely identified by asking only for *01D-meet* and not including the exclusive *1D-meet* in the query. A second possible scenario

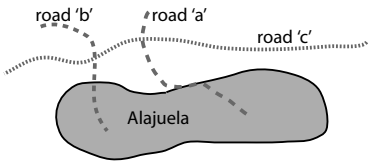

Figure 5: Border patrol query scenario

and sample query involves the currently hot topic of border patrol and immigration (see Figure 5). A road network overlaps in several places with the region of *Alajuela* which the government intends to protect. The government would like to allocate single officers in a patrol station to those roads that cross the boundary of the region at a single point. For those roads that share a common line with the boundary of the region, the government must allocate more than one officer plus at least one vehicle. An estimate needs to be done and a query is posed in order to know which roads need cars and which roads only need officers in patrol stations. The roads that need cars are found by asking:

```
SELECT r.name
FROM   roads r, provinces p
WHERE  p.name="ALAJUELA" AND
       (1D-overlap(r.path,p.area) OR
        01D-overlap(r.path,p.area));
```

Another interesting query involves the planning and upkeep of intracoastal waterways. These are essentially canals dug near the ocean that are used by smaller ships and barges that need limited access to the ocean but are not designed to operate in large waves. The canals provide a protected channel for these boats to travel. Because these waterways need to be protected, they can meet the ocean at a point to allow access, but not along a line, which would expose the boats to the rough ocean water (see Figure 6). Therefore, to discover invalid paths for planned waterways, and to determine if a waterway needs repair, the following query can be posed:

```
SELECT iw.name
FROM   intracoastal_waterways iw, ocean o
WHERE  NOT(0D-meet(iw.path, o.boundary));
```

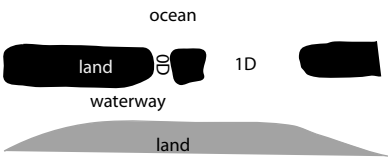

Figure 6: Intracoastal waterway scenario with one allowed $0D$ opening and an opening that requires maintenance ($1D$).

DRTPs also prove to be useful in highway planning. Typically, the planned routes for highways are through areas that are currently not well accessible by existing roadways. Because of this, the planned highway route cannot easily be physically examined by people. The problem is that it can potentially intersect with rivers in unexpected ways. If the route crosses a river at a point, this is acceptable because a small bridge can be built. However, if the route intersects a river along a line, then the route must be modified so that it only intersects the river at a point, or not at all (see Figure 7). A query that identifies such intersections is:

```
SELECT h.name
FROM   highway_routes h, rivers r
WHERE  NOT(0D-overlap(h.path,r.path));
```

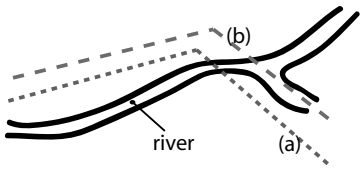

Figure 7: Two road plans, one invalid (b) and one valid (a).

Many military applications also exist for DRTPs. Unmanned Aerial Vehicles (UAVs), e.g., typically fly along pre-specified flight paths. When these paths are planned, the location and effective range (defining a *danger zone*) of many enemy anti-aircraft installations are known. The UAV is typically safe if it stays out of the danger zones, or it is only in a danger zone for a very short time. Therefore, flight paths that do not intersect and 0D-meet danger zones are considered safe. Flight paths that intersect the danger zones or meet them along a line are unsafe for the UAV because they provide time for the installation to shoot down the vehicle (see Figure 8). Valid UAV flight paths are determined by the following query:

```
SELECT fp.name
FROM   UAV_flight_paths fp,
       anti-aircraft-danger-zones d
WHERE  NOT(overlap(fp.path,d.area)) AND
       NOT(01D-meet(fp.path,d.boundary)) AND
       NOT(1D-meet(fp.path,d.boundary));
```

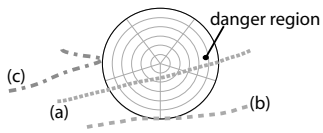

Figure 8: Danger zone (radar) and possible flights paths with only (c) representing a safe path.

These scenarios and queries illustrate the power and utility of dimension-refined topological predicates. More complex applications in areas ranging from environmental conservation to military can take advantage of the added precision that dimension provides to the commonly used topological predicates. Such applications can utilize the dimension-refined predicates without having to relinquish the pure topological predicates that are still available under the dimension-refined model.

7. METHOD COMPARISON

The *dimension-extended model* (*DEM*) described in [1, 3] is another approach to include the concept of dimension in predicates between spatial objects. However, our approach of dimensional refinement differs significantly from the DEM. The notable discrepancies in the two models deal with the method of discovering the valid dimension predicates, as well as basic assumptions such as the definition of dimension.

The DEM centers around the concept of "cancellation rules", whereas our dimension refinement method does not. The cancellation concept essentially assumes that all possible matrices for dimension-extended predicates are initially valid. Then, a series of cancellation steps eliminates any matrices that are invalid based on the dimension of point set intersections at each matrix entry. Further analysis eliminates more matrices based on the interdependence of entries in particular matrices. One problem with this approach is that it is not data type independent. Different cancellation rules must be defined for both simple and complex objects, as well as for each specific combination of simple or complex spatial data types. Furthermore, the DEM as described in [1, 3] does not provide any cancellation rules and does not explicitly deal with complex spatial objects. The DEM method is analogous to a top down approach of discovering DE predicates.

Our approach, on the other hand, is a novel method of discovering DRTPs in a bottom up fashion. Instead of considering all possible matrices, we begin by examining only those dimension matrices that have a correspondence relationship to a topologically valid 9ITM. For a given valid 9ITM, if there is no non-empty intersection between one-dimensional components, the dimension predicate is unique and equivalent to the topological predicate. If there is a non-empty intersection between one-dimensional components and the containment condition holds between the one-dimensional components, then only a single 9IDM is constructed due to the impossibility of further refinement. If the containment condition does not hold, but the shared-boundary containment condition holds, then two 9IDMs are constructed from the 9ITM. If neither of those conditions hold, then three 9IDMs are constructed based on the upper and lower bounds of the dimension of the intersection of two point sets. Therefore, the set of valid 9IDMs is initially empty and grows as valid 9IDMs are built from valid 9ITMs. In contrast, the DEM begins with the set of all possible dimension-extended matrices and reduces the set to a final set of valid matrices. Furthermore, because our dimension refinement method is based purely on topology and rules that are based on the dimension of point sets, it is type independent in the sense that the same method is applicable to both simple and complex spatial objects, as well as all combinations of spatial data types.

The second point that differentiates the dimension refinement model from the DEM is the definition of dimension. The DEM relies on an intuitive definition of dimension. No formal definition of dimension is given, and a *dim* function is defined to return the dimension of a point set as empty, zero-, one-, or two-dimensional. The dimension refinement method, on the other hand, defines a very precise notion of dimension for the purpose of representing dimension in DRTPs. This definition is based on the Lebesgue covering dimension as it applies to point sets in two dimensional space. Following from this precise definition of dimension, the dimension refinement approach is able to define the notion of multiple dimension point sets. For example, point sets containing both a point and a line are considered to be $01D$ and are distinguished from those containing only $1D$ or only $0D$. This allows the dimension refinement method to produce predicates that are mutually exclusive (i.e., only a single dimension refined predicate is true for a given pair of objects). This concept of mutual exclusion does not hold for predicates as they are defined by the DEM. Based on these differing concepts of dimension, the resulting number of predicates and the usage of the predicates defined by each model differ significantly.

8. CONCLUSION

We have formally defined dimension predicates and presented a method for discovering all the valid dimension predicates between two spatial objects of any type and complexity. We make use of the well defined theory and previous work on dimensions and topological predicates to achieve a generic yet robust model that results in dimension-refined topological predicates. The main contributions of our work are presented as part of the comparison of methods in Section 7.

This model was developed as a necessity that arose during the design and development of a two-dimensional spatial algebra. The concept of dimension predicates turns out to be useful not only in high-level applications (as shown in Section 6) but also as part of a development hierarchy that includes *robust geometric primitives* (*RGP*). RGPs represent building blocks of the well known complex spatial data types *point*, *line* and *region*. Applying dimension predicates to RGPs during development may benefit correctness and robustness of the implemented data type model. Future work includes the complete implementation of the dimension-refined topological predicates and the analysis of the impact of such predicates on query optimization. This is also interesting as the DRTPs provide new and more refined information than previously available through the purely topological predicates.

9. REFERENCES

[1] E. Clementini and P. Di Felice. A Comparison of Methods for Representing Topological Relationships. *Information Sciences Applications*, 3(3):149–178, 1995.

[2] E. Clementini and P. Di Felice. A Model for Representing Topological Relationships between Complex Geometric Features in Spatial Databases. *Information Systems*, 90(1-4):121–136, 1996.

[3] E. Clementini, P. Di Felice, and P. van Oosterom. A Small Set of Formal Topological Relationships Suitable for End-User Interaction. *3rd Int. Symp. on Advances in Spatial Databases*, LNCS 692, pp. 277–295, 1993.

[4] 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.

[5] M. J. Egenhofer and J. Herring. Categorizing Binary Topological Relations Between Regions, Lines, and Points in Geographic Databases. Technical Report 90-12, National Center for Geographic Information and Analysis, University of California, Santa Barbara, 1990.

[6] M. Schneider. *Spatial Data Types for Database Systems - Finite Resolution Geometry for Geographic Information Systems*, volume LNCS 1288. Springer-Verlag, Berlin Heidelberg, 1997.

[7] M. Schneider and T. Behr. Topological Relationships between Complex Spatial Objects. Technical Report 011, University of Florida, Department of Computer & Information Science & Engineering, 2004.

[8] R. B. Tilove. Set Membership Classification: A Unified Approach to Geometric Intersection Problems. *IEEE Trans. on Computers*, C-29:874–883, 1980.

[9] E. W. Weisstein, editor. *CRC Concise Encyclopedia of Mathematics*. Chapman & Hall/CRC, Boca Raton, 2nd edition, 2003.