# Efficient Implementation Techniques for Topological Predicates on Complex Spatial Objects: The Evaluation Phase

Reasey Praing & Markus Schneider*

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

**Abstract**

Topological predicates like *overlap*, *inside*, *meet*, and *disjoint* uniquely characterize the relative position between objects in space. They have been the subject of extensive interdisciplinary research. Spatial database systems and geographical information systems have shown a special interest in them since they enable the support of suitable query languages for spatial data retrieval and analysis. A look into the literature reveals that the research efforts so far have mainly dealt with the conceptual design of and the reasoning with these predicates while the development of efficient and robust implementation methods for them has been largely neglected. Especially the recent design of topological predicates for different combinations of *complex* spatial data types has resulted in a large increase of their numbers and stressed the importance of their efficient implementation. The goal of this article is to develop efficient implementation techniques of topological predicates for all combinations of the complex spatial data types *point2D*, *line2D*, and *region2D* within the framework of the spatial algebra SPAL2D. Our solution consists of two phases. In the *exploration phase* described in previous work of the authors, for a given scene of two spatial objects, all *topological events* are registered in so-called *topological feature vectors*. These vectors serve as input for the *evaluation phase* which is the focus of this article and which analyzes the topological events and determines the Boolean result of a topological predicate or the kind of topological predicate by a formally defined method called 9-*intersection matrix characterization*. Besides this very general evaluation method, the article presents an optimized method for predicate verification, called *matrix thinning*, and an optimized method for predicate determination, called *minimum cost decision tree*. Our evaluation methods also turn out to be able to compute *dimension-refined* topological predicates.

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

# 1 Introduction

In several disciplines like artificial intelligence, linguistics, robotics, and cognitive science, the study of topological predicates has been a topic of extensive research. Topological predicates (like *overlap*, *meet*, *inside*) characterize the relative position of spatial objects (like points, lines, regions). They are of purely qualitative nature and are independent of any quantitative, metric measures like distance or direction measures. Instead, they are associated with notions like adjacency, coincidence, connectivity, inclusion, and continuity and are preserved under affine transformations such as translation, scaling, and rotation. Spatial database systems and geographical information systems have shown a special interest in these predicates since they enable the support of suitable query languages for spatial data retrieval and analysis. A look into the literature reveals that the research efforts so far have mainly dealt with the conceptual design of and the reasoning with these predicates as well as with strategies for avoiding the necessity of their computation at all or their repetitive computation at least. The two main conceptual approaches proposed so far are the 9-*intersection model* [16], which rests on point set theory and point set topology, and the *RCC model* [10], which leverages spatial logic. Their main, common features are the provision of a complete set of mutually exclusive topological predicates for each spatial data type combination and their restriction to *simple* spatial objects. Both models have produced very similar results and form the basis of most publications in this field. In this article, we are especially interested in topological predicates for *complex* spatial objects, as they have been recently specified in [38] on the basis of the 9-intersection model. In addition, we also contemplate *dimension-refined topological predicates* [30] which take into account the dimensions of the intersections between two spatial objects and enable the posing of more fine-grained than purely topological queries.

In contrast to the large amount of conceptual work, implementation issues for topological predicates have been widely neglected. Since topological predicates are *expensive predicates* that cannot be evaluated in constant time, the strategy in query plans has consequently been to avoid their computation. An important example are spatial index structures that are employed as filtering techniques in query processing. Their aim is to identify a hopefully small collection of candidate pairs of spatial objects that could possibly fulfil the predicate of interest and to exclude a large collection of pairs of spatial objects that definitely cannot satisfy the predicate. However, efficient implementation techniques for the topological predicates themselves cannot be found in the literature. But such techniques are indispensable due to the transition from simple to complex spatial objects, the consequential increase of the number of topological predicates, and the only, unacceptable alternative of error-prone and inappropriate ad hoc implementations.

The goal of this (and a previous [39]) article is to develop and present systematic, correct, robust, and efficient implementation strategies for topological predicates between all combinations of the three complex spatial data types *point2D*, *line2D*, and *region2D* within the framework of the spatial algebra SPAL2D. Given two objects $A$ and $B$ of any complex spatial data type *point2D*, *line2D*, or *region2D* [34], we can pose at least two kinds of topological queries that our implementation supports: (1) "Do $A$ and $B$ satisfy the topological predicate $p$?" and (2) "What is the topological predicate $p$ between $A$ and $B$?". Only query 1 yields a Boolean value, and we call it hence a *verification query*. Query 2 returns a predicate (name), and we call it hence a *determination query*. For both query types, we can even ask more fine-grained queries that include the dimension of an intersection. For example, the topological predicate *meet* between two region objects can be dimensionally refined into 0-*meet*, 1-*meet*, and 01-*meet* respectively. This states that the two region objects meet in a point object (0D), a line object (1D), and a point object and a line object (0D+1D) respectively. We distinguish two phases of predicate execution: In an *exploration phase* described in detail in [39], a plane sweep scans a given configuration of two spatial objects, detects all *topological events* (like intersections), and records them in so-called *topological feature vectors*. These vectors serve as input for the *evaluation phase* which analyzes these topological data and determines the Boolean result of a topological predicate (query 1) or the kind of topological predicate (query 2). This article reviews the exploration phase, which outputs the topological feature vectors, and puts an emphasis on sophisticated and efficient methods

for the evaluation phase, which takes the topological feature vectors as arguments. The two-phase approach provides a direct and sound interaction and synergy between conceptual work (9-intersection model) and implementation (algorithmic design).

The special goals of the evaluation phase are the correct and efficient interpretation and matching of the topological feature vectors from the exploration phase with characteristic properties of the topological predicates. For this purpose, we introduce a general method called 9-*intersection matrix characterization* that can be leveraged for both predicate verification and predicate determination, that depends on the spatial data type combination under consideration, and whose correctness is formally proved. A slight extension of this method can then be used for the matching of dimension-refined topological predicates. A fine-tuning of the 9-intersection matrix characterization leads to two optimized approaches called *matrix thinning* for predicate verification and *minimum cost decision trees* for predicate determination.

Section 2 discusses related work about simple and complex spatial data types as well as available design and implementation concepts for pure and dimension-refined topological predicates. In Section 3, we give an overview of the overall approach and review the exploration phase and a simple evaluation method called *direct predicate characterization* from [39]. Section 4 focuses on the evaluation phase and matches the acquired topological feature vectors with characteristic properties of the topological predicates by the 9-intersection matrix characterization. Section 5 presents the two fine-tuned and optimized approaches of matrix thinning for predicate verification and minimum cost decision trees for predicate determination. In Section 6, we give a brief overview of the implementation environment, present our testing strategy, and present an assessment of our overall approach. Finally, Section 7 draws some conclusions.

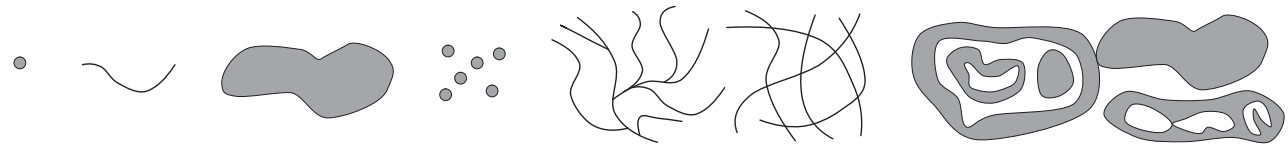
# 2 Related Work

In this section we present related work on spatial data types as the argument types of topological predicates (Section 2.1), give an overview of the essential conceptual models for topological relationships (Section 2.2) and dimension-refined topological relationships (Section 2.3), and discuss implementation aspects of topological predicates (Section 2.4).

## 2.1 Spatial Data Types

In the spatial database and GIS community, *spatial data types* (e.g., [14, 24, 34]) like *point*, *line*, or *region* have found wide acceptance as fundamental abstractions for modeling the structure of geometric entities, their relationships, properties, and operations. They form the basis of a number of data models and query languages for spatial data and have gained access into commercial software products. Topological predicates operate on instances of these data types, called *spatial objects*. The literature distinguishes *simple* and *complex* spatial data types, depending on the spatial complexity they are able to model. Simple spatial data types only provide simple object structures like single points, continuous lines, and simple regions (Figure 1(a)-(c)). Until recently, topological relationships have only been defined for them. However, from an application perspective, simple geometric structures have turned out to be inadequate abstractions for real spatial applications since they are insufficient to cope with the variety and complexity of geographic reality. From a formal perspective, simple spatial data types are not closed under the geometric set operations *intersection*, *union*, and *difference*. This means that these operations applied to two simple spatial objects as input can produce a spatial object that is *not* simple. Complex spatial data types solve both problems. They provide universal and versatile spatial objects and are closed under the geometric set operations. They allow objects with multiple components, region components that may have holes, and line components that may model ramified, connected networks(Figure 1(d)-(f)).

Our formal specification of complex spatial data types given in [38] rests on point set theory and point set topology [22] and forms the basis of our spatial data type implementation. We call this specification



(a) (b) (c) (d) (e) (f)

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

the *abstract model* of our spatial type system SPAL2D. Only a few formal approaches [5, 24, 40] have been developed for complex spatial objects; they all share a number of structural features with our type specifications. In their OGC Abstract Specification [31], the OpenGIS Consortium (OGC) has proposed some informally described geometric structures that are called *simple features* and that are similar to ours.

Implementations of spatial type systems are available in ESRI's Spatial Database Engine (ArcSDE) [21] and in a few spatial extension packages of commercial database management systems that have integrated ArcSDE functionality. Examples are the Informix Geodetic DataBlade [28], the Oracle Spatial Cartridge [32], and DB2's Spatial Extender [11]. Descriptions of data structures for spatial data types that are able to support the efficient implementation of topological predicates are rare. We have specified our own data structures in [39] and call this specification the *discrete model* of SPAL2D. They are an extension of the *realm-based* data structures of the *ROSE Algebra* [25] in the sense that they generalize these concepts and accommodate realm-based and general spatial objects.

## 2.2 Conceptual Models for Topological Relationships

Topological relationships characterize the relative position between spatial objects. The 9-*intersection model* [16], which is based on point set theory and point set topology, and the *RCC model* [10], which is based on spatial logic, have proved to be the fundamental approaches to deriving these relationships. Despite rather different foundations, both approaches come to very similar results. Our implementation strategy relies heavily on the 9-intersection model since it enables us to create a direct link between the concepts of this model and our implementation approach as well as to prove the correctness of our strategy.

Based on the 9-intersection model, a complete collection of mutually exclusive topological relationships can be determined for each combination of simple and recently also complex spatial data types. The model is based on the nine possible intersections of the boundary ($\partial A$), the interior ($A^\circ$), and the exterior ($A^-$) [22] of a spatial object $A$ with the corresponding components $\partial B$, $B^\circ$, and $B^-$ of another spatial object $B$. Each intersection is tested with regard to the topologically invariant criteria of non-emptiness. The topological relationship between two spatial objects $A$ and $B$ can be expressed by evaluating the $3 \times 3$-matrix in Figure 2(a). We call each matrix element a *matrix predicate*. A total number of $2^9 = 512$ different configurations are possible from which only a certain subset makes sense depending on the *definition* and *combination* of the spatial data types considered. For each combination of spatial types, this means that each of its predicates is associated with a unique *9-intersection matrix* so that all predicates are mutually exclusive and complete with regard to the topologically invariant criteria of non-emptiness and emptiness.

Topological relationships have been first investigated for two simple regions (*disjoint*, *meet*, *overlap*, *equal*, *inside*, *contains*, *covers*, *coveredBy*) [8, 12, 15, 16], for two simple lines [6, 13, 17], and for a simple line and a simple region [18]. Topological predicates involving simple points have been considered trivial. Two restricted attempts to a definition of topological relationships on more complex spatial objects are the TRCR (Topological Relationships for Composite Regions) model [7], which only allows sets of disjoint simple regions without holes and does not derive topological relationships systematically from the underly-

$$\begin{pmatrix} A^\circ \cap B^\circ \neq \varnothing & A^\circ \cap \partial B \neq \varnothing & A^\circ \cap B^- \neq \varnothing \\ \partial A \cap B^\circ \neq \varnothing & \partial A \cap \partial B \neq \varnothing & \partial A \cap B^- \neq \varnothing \\ A^- \cap B^\circ \neq \varnothing & A^- \cap \partial B \neq \varnothing & A^- \cap B^- \neq \varnothing \end{pmatrix}$$

(a)

| | point2D | line2D | region2D |
|---|---|---|---|
| point2D | 2/5 | 3/14 | 3/7 |
| line2D | 3/14 | 33/82 | 19/43 |
| region2D | 3/7 | 19/43 | 8/33 |

(b)

Figure 2: The 9-intersection matrix (a) and the numbers of topological predicates between two simple/complex spatial objects (b).

ing model, and the approach in [20], which only considers topological relationships of simple regions with holes, does not permit multi-part regions, and depends on the number of holes of the operand objects. In [1, 37, 38] we have given a thorough, systematic, and complete specification of topological relationships for all combinations of complex spatial data types. It is also based on the 9-intersection model and forms the basis of our implementation. Figure 2(b) shows the increase of topological predicates for complex objects compared to simple objects and underpins the need for efficient predicate execution techniques.

### 2.3 Conceptual Models for Dimension-Refined Topological Relationships

A replacement of the topological invariant of non-emptiness and emptiness of an intersection by the topological invariant of the *dimension* of an intersection leads from the 9-intersection matrix (Figure 2(a)) to the 9-*intersection dimension matrix* (Figure 3(a)). In [30] we have explored which predicates can be derived on the basis of the dimension matrix for all combinations of the spatial data types *point*, *line*, and *region*. It turns out that these predicates are *refinements* of the topological predicates on spatial data types. Hence, we denote them as *dimension-refined topological predicates*. Figure 3(b) shows the numbers of these predicates for all combinations of simple and complex spatial data types respectively. A comparison with Figure 2(b) reveals that all type combinations that include the type *point* cannot be dimension-refined. This is only possible if one-dimensional components of a spatial object are involved. But a point object is zero-dimensional. Therefore, a single dimension-refined topological predicate is derived from a corresponding "pure" topological predicate. This is different with the other type combinations that include the types *line* and *region*. The interior of a line object and the boundary of a region object are one-dimensional structures that can interact in several different ways. Either they are disjoint, or they only share a point object, or they only share a line object, or they share a point object and a line object. Consequently, the dimension of the intersection is undefined, only zero-dimensional, only one-dimensional, or zero- and one-dimensional. This leads to different refinements of the underlying topological predicates. In this article, we are interested in dimension-refined predicates since our approach also enables us to evaluate them.

Another approach that also includes a concept of dimension in predicates is the *dimension-extended model* described in [4, 8]. Our concept improves and generalizes this approach (for a comparison see [30]).

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

(a)

| | point2D | line2D | region2D |
|---|---|---|---|
| point2D | 2/5 | 3/14 | 3/7 |
| line2D | 3/14 | 61/146 | 43/75 |
| region2D | 3/7 | 43/75 | 16/53 |

(b)

Figure 3: The 9-intersection dimension matrix (a) and the numbers of dimension-refined topological predicates between two simple/complex spatial objects (b).

### 2.4 Implementation Aspects of Topological Predicates

In queries, topological predicates are usually employed as filter conditions in spatial selections and spatial joins. At least two main strategies can be distinguished for their processing. The first, optional but worthwhile strategy aims at avoiding the execution of topological predicates since they are expensive predicates that cannot be executed in constant time. At the algebraic level, topological reasoning techniques examine the topological relationships contained in a query for topological consistency [33]. Optimization techniques minimize the number of computations needed [9] (see also Section 5.1) and aim at eliminating topological relationships that are implied uniquely by composition [19]. At the physical level, methods like predicate migration [27], predicate placement [26], disjunctive query optimization [3], and approximation-based evaluation [2] pursue the concept of avoiding unnecessary or repetitive predicate evaluations in query access plans. In a filtering step on the basis of minimum bounding rectangles, spatial index structures [23] identify those candidate pairs of spatial objects that could possibly fulfil the topological predicate of interest.

The second, required strategy aims at designing algorithms for the efficient execution of topological predicates. This is the goal of this article together with a previous article in [39] that is reviewed in Section 3. This research goes far beyond our own initial attempts in [35, 36] that extend the concept of the eight topological predicates for two simple regions to a concept and implementation of these eight predicates for complex regions. We are not aware of any other published research on the implementation of topological predicates. Spatial extension packages like the ESRI ArcSDE, the Informix Geodetic DataBlade, the Oracle Spatial Cartridge, and DB2's Spatial Extender offer limited sets of named topological predicates for spatial objects. But their definitions are vague and their implementations unpublished. The open source JTS Topology Suite [29] implements topological predicates through *topology graphs*. Such a graph stores for each node (endpoint) and edge (segment) of a spatial object whether it is located in the interior, in the exterior, or on the boundary of another spatial object. Computing the topology graphs and deriving the 9-intersection matrix from them require quadratic time and quadratic space in terms of the nodes and edges of the two operand objects; our solution requires linearithmic (loglinear) time and linear space.

## 3 Review: Exploration Phase and Evaluation Phase

The motivation for this article and the companion article in [39] is our design and derivation of topological predicates ([38], Section 2.2) and dimension-refined topological relationships ([30], Section 2.3) for all combinations of the complex spatial data types *point2D*, *line2D*, and *region2D* (Section 2.1) and the resulting open issue of their efficient, correct, and robust implementation. Our proposed solution is a two-phase approach consisting of an *exploration phase* and a subsequent *evaluation phase*. It has been implemented within the framework of a new, sophisticated type system for two-dimensional spatial data, called *Spatial Algebra 2D* (*SPAL2D*), that arranges for its integration into extensible database systems. In this section, we review the exploration phase (Section 3.1) and also an *ad hoc* solution for the evaluation phase (Section 3.2) on the basis of our results in [39].

### 3.1 The Exploration Phase for Collecting Topological Information

The main task of the exploration phase is the detection and collecting of all relevant topological events as they are crucial for a predicate verification and predicate determination in the evaluation phase. For this, we have designed *exploration algorithms* (Figure 4). They take two general or realm-based spatial objects $F$ of type $\alpha$ and $G$ of type $\beta$ with $\alpha, \beta \in \{point2D, line2D, region2D\}$ as input, discover the topological events of this particular scene, and output the gained topological information in two *topological feature vectors* $v_F$ for object $F$ and $v_G$ for object $G$. The algorithms are all based on special and sophisticated instances of the plane sweep paradigm, which is a well known concept of computational geometry. We distinguish exploration

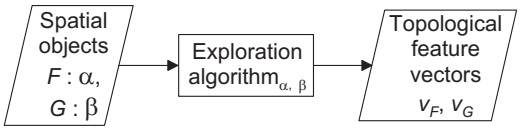



Figure 4: Exploration phase

algorithms for the type combinations *point2D*/*point2D*, *point2D*/*line2D*, *point2D*/*region2D*, *line2D*/*line2D*, *line2D*/*region2D*, and *region2D*/*region2D*. That is, the dimension of the first argument object is always lower than or equal to the dimension of the second argument object. Due to the rather different properties of the six type combinations, a single exploration algorithm covering all cases is unfavorable.

We refer the reader to [39] for a detailed description of the data structures of the three complex spatial data types as well as the exploration algorithms for the six type combinations. For our purposes here, we only need to know the following: A *point2D* object $F$ is represented as a *sequence* of lexicographically ordered points. Let $P(F)$ be the *set* of all points of $F$. A *line2D* object $F$ is represented as a *sequence* of ordered *halfsegments*. A halfsegment includes segment information and emphasizes one of its end points as the *dominating point*. Consequently, each segment corresponds to two halfsegments, a left halfsegment and a right halfsegment. Let $H(F)$ be the *set* of all halfsegments of $F$. For $f \in H(F)$, let $f.s$ denote its segment component and $dp(f)$ be a function yielding the dominating point of $f$. Let $B(F)$ be the set of all boundary points of $F$. These are all those points of $F$ from which exactly one segment emanates. A *region2D* object $F$ is represented as a *sequence* of ordered *attributed halfsegments*. In our case, the attribute of a halfsegment is a flag *ia* for "Interior Above" indicating whether the interior of a region is above or, for vertical segments, left of the segment. Let $H(F)$ be the *set* of all attributed halfsegments of $F$. For $f \in H(F)$, let $f.s$ denote its segment component and $f.ia$ its attribute component. Note that $H(F)$, where $F$ is a *line2D* or *region2D* object, only includes (half)segments that are either disjoint from, equal to, or meeting (in an end point) the (half)segments of $G$ due to our special splitting strategy during plane sweeps [39].

Topological feature vectors as the output of the exploration phase serve as input of the evaluation phase; i.e., they represent the interface between both phases. Therefore, we focus here on their definitions since they are later needed in this article. A topological feature vector $v_F$ of a spatial object $F$ is a Boolean vector consisting of a special set of "topological flags". These flags are specific for each type combination, and their values are unique for a given spatial configuration. The flags are all initialized to *false* at the beginning. Once certain topological information about a spatial argument object has been discovered, the corresponding flag of its topological feature vector is set to *true*. For example, if two *line2D* objects share a segment, the corresponding flag $v_F[seg\_shared]$ is set to *true* (see Definition 4(i) below). To minimize the number of topological flags, in symmetric cases, we do not specify a corresponding flag for $G$. In our example, we do not have a flag $v_G[seg\_shared]$. In the following, we give the definitions of the topological feature vectors for all type combinations and explain them as needed. Their detailed description as well as a discussion of their identification, uniqueness, and completeness can be found in [39].

For two *point2D* objects, we consider the cases whether both objects have a point in common (flag *poi_shared*) and whether $F$ ($G$) contains a point that is not part of $G$ ($F$) (flag *poi_disjoint* in $v_F$ and $v_G$).

**Definition 1** Let $F, G \in point2D$, and let $v_F$ and $v_G$ be their topological feature vectors. Then

$$\begin{array}{lll} \text{(i)} & v_F[poi\_shared] & :\Leftrightarrow \quad \exists\, f \in P(F)\ \exists g \in P(G) : f = g \\ \text{(ii)} & v_F[poi\_disjoint] & :\Leftrightarrow \quad \exists\, f \in P(F)\ \forall g \in P(G) : f \neq g \\ \text{(iii)} & v_G[poi\_disjoint] & :\Leftrightarrow \quad \exists g \in P(G)\ \forall\, f \in P(F) : f \neq g \end{array}$$

The predicates "$=$" and "$\neq$" check the equality and inequality of two single points respectively.

In case of a *point2D* object $F$ and a *line2D* object $G$, we distinguish the following cases: First, a point $f$ of $F$ can be disjoint from $G$ (flag *poi_disjoint*). Second, a point $f$ can lie in the interior of a segment of $G$

(flag *poi_on_interior*). Third, a point $f$ can be equal to a boundary point of $G$ (flag *poi_on_bound*). Fourth, $G$ can contain a boundary point that is unequal to all points in $F$ (flag *bound_poi_disjoint*).

**Definition 2** Let $F \in$ *point2D*, $G \in$ *line2D*, and $v_F$ and $v_G$ be their topological feature vectors. Then

$$\begin{array}{lll}
\text{(i)} & v_F[\textit{poi_disjoint}] & :\Leftrightarrow \ \exists\, f \in P(F)\ \forall g \in H(G) : \neg \textit{on}(f, g.s) \\
\text{(ii)} & v_F[\textit{poi_on_interior}] & :\Leftrightarrow \ \exists\, f \in P(F)\ \exists g \in H(G)\ \forall b \in B(G) : \textit{on}(f, g.s) \ \wedge \ f \neq b \\
\text{(iii)} & v_F[\textit{poi_on_bound}] & :\Leftrightarrow \ \exists\, f \in P(F)\ \exists g \in B(G) : f = g \\
\text{(iv)} & v_G[\textit{bound_poi_disjoint}] & :\Leftrightarrow \ \exists g \in B(G)\ \forall\, f \in P(F) : f \neq g
\end{array}$$

The predicate *on* is a robust geometric predicate and checks whether a point is located on a segment.

In case of a *point2D* object $F$ and a *region2D* object $G$, we are interested in the topological events whether a point of $F$ lies either inside $G$ (flag *poi_inside*), on the boundary of $G$ (flag *poi_on_bound*), or outside of $G$ (flag *poi_outside*).

**Definition 3** Let $F \in$ *point2D*, $G \in$ *region2D*, and $v_F$ and $v_G$ be their topological feature vectors. Then

$$\begin{array}{lll}
\text{(i)} & v_F[\textit{poi_inside}] & :\Leftrightarrow \ \exists\, f \in P(F) : \textit{poiInRegion}(f, G) \\
\text{(ii)} & v_F[\textit{poi_on_bound}] & :\Leftrightarrow \ \exists\, f \in P(F)\ \exists g \in H(G) : \textit{on}(f, g.s) \\
\text{(iii)} & v_F[\textit{poi_outside}] & :\Leftrightarrow \ \exists\, f \in P(F)\ \forall g \in H(G) : \neg \textit{poiInRegion}(f, G) \ \wedge \ \neg \textit{on}(f, g.s)
\end{array}$$

The predicate *poiInRegion* checks whether a point lies inside a *region2D* object.

For two *line2D* objects, we obtain the following cases: First, the interiors of two segments of $F$ and $G$ can partially or completely coincide (flag *seg_shared*). Second, if a segment of $F$ does not partially or completely coincide with any segment of $G$, we register this in the flag *seg_unshared*. Third, we set the flag *interior_poi_shared* if two segments intersect in a single point that does not belong to the boundaries of $F$ or $G$. Fourth, a boundary endpoint of a segment of $F$ can be located in the interior of a segment (including connector points) of $G$ (flag *bound_on_interior*). Fifth, both objects $F$ and $G$ can share a boundary point (flag *bound_shared*). Sixth, if a boundary endpoint of a segment of $F$ lies outside of all segments of $G$, we set the flag *bound_disjoint*. The last three cases introduce the analogous flags *seg_unshared*, *bound_on_interior*, and *bound_disjoint* for $G$.

**Definition 4** Let $F, G \in$ *line2D*, and let $v_F$ and $v_G$ be their topological feature vectors. Then

$$\begin{array}{lll}
\text{(i)} & v_F[\textit{seg_shared}] & :\Leftrightarrow \ \exists\, f \in H(F)\ \exists g \in H(G) : \textit{segIntersect}(f.s, g.s) \\
\text{(ii)} & v_F[\textit{interior_poi_shared}] & :\Leftrightarrow \ \exists\, f \in H(F)\ \exists g \in H(G)\ \forall p \in B(F) \cup B(G) : \\
 & & \textit{poiIntersect}(f.s, g.s) \ \wedge \ \textit{poiIntersection}(f.s, g.s) \neq p \\
\text{(iii)} & v_F[\textit{seg_unshared}] & :\Leftrightarrow \ \exists\, f \in H(F)\ \forall g \in H(G) : \neg \textit{segIntersect}(f.s, g.s) \\
\text{(iv)} & v_F[\textit{bound_on_interior}] & :\Leftrightarrow \ \exists\, f \in H(F)\ \exists g \in H(G)\ \exists p \in B(F) \setminus B(G) : \\
 & & \textit{poiIntersection}(f.s, g.s) = p \\
\text{(v)} & v_F[\textit{bound_shared}] & :\Leftrightarrow \ \exists\, p \in B(F)\ \exists q \in B(G) : p = q \\
\text{(vi)} & v_F[\textit{bound_disjoint}] & :\Leftrightarrow \ \exists\, p \in B(F)\ \forall g \in H(G) : \neg \textit{on}(p, g.s) \\
\text{(vii)} & v_G[\textit{seg_unshared}] & :\Leftrightarrow \ \exists g \in H(G)\ \forall\, f \in H(F) : \neg \textit{segIntersect}(f.s, g.s) \\
\text{(viii)} & v_G[\textit{bound_on_interior}] & :\Leftrightarrow \ \exists\, f \in H(F)\ \exists g \in H(G)\ \exists p \in B(G) \setminus B(F) : \\
 & & \textit{poiIntersection}(f.s, g.s) = p \\
\text{(ix)} & v_G[\textit{bound_disjoint}] & :\Leftrightarrow \ \exists q \in B(G)\ \forall\, f \in H(F) : \neg \textit{on}(q, f.s)
\end{array}$$

The predicates *poiIntersect* and *segIntersect* test whether two segments intersect in a point or segment respectively. The operation *poiIntersection* returns the intersection point of two segments.

In case of a *line2D* object and a *region2D* object, we differentiate the following cases: First, the intersection of the interiors of $F$ and $G$ means that a segment of $F$ lies in $G$ (flag *seg_inside*). Second and third, the

interior of a segment of $F$ intersects with a boundary segment of $G$ if either both segments partially or fully coincide (flag *seg_shared*), or if they properly intersect in a single point (flag *poi_shared*). Fourth, the interior of a segment of $F$ intersects with the exterior of $G$ if the segment is disjoint from $G$ (flag *seg_outside*). Fifth, a boundary point of $F$ intersects the interior of $G$ if the boundary point lies inside of $G$ (flag *bound_inside*). Sixth, if it lies on the boundary of $G$, we set the flag *bound_shared*. Seventh, if it lies outside of $G$, we set the flag *bound_disjoint*. Eighth, if the boundary of $G$ intersects the exterior of $F$, a boundary segment of $G$ must be disjoint from $F$ (flag *seg_unshared*).

**Definition 5** Let $F \in$ *line2D*, $G \in$ *region2D*, and $v_F$ and $v_G$ be their topological feature vectors. Then

$$
\begin{array}{rlll}
\text{(i)} & v_F[\textit{seg\_inside}] & :\Leftrightarrow & \exists\, f \in H(F)\ \forall g \in H(G): \\
 & & & \neg \textit{segIntersect}(f.s, g.s)\ \wedge\ \textit{segInRegion}(f.s, G) \\
\text{(ii)} & v_F[\textit{seg\_shared}] & :\Leftrightarrow & \exists\, f \in H(F)\ \exists g \in H(G): \textit{segIntersect}(f.s, g.s) \\
\text{(iii)} & v_F[\textit{seg\_outside}] & :\Leftrightarrow & \exists\, f \in H(F)\ \forall g \in H(G): \\
 & & & \neg \textit{segIntersect}(f.s, g.s)\ \wedge\ \neg \textit{segInRegion}(f.s, G) \\
\text{(iv)} & v_F[\textit{poi\_shared}] & :\Leftrightarrow & \exists\, f \in H(F)\ \exists g \in H(G): \textit{poiIntersect}(f.s, g.s)\ \wedge \\
 & & & \textit{poiIntersection}(f.s, g.s) \notin B(F) \\
\text{(v)} & v_F[\textit{bound\_inside}] & :\Leftrightarrow & \exists\, f \in H(F): \textit{poiInRegion}(dp(f), G)\ \wedge\ dp(f) \in B(F) \\
\text{(vi)} & v_F[\textit{bound\_shared}] & :\Leftrightarrow & \exists\, f \in H(F)\ \exists g \in H(G): \textit{poiIntersect}(f.s, g.s)\ \wedge \\
 & & & \textit{poiIntersection}(f.s, g.s) \in B(F) \\
\text{(vii)} & v_F[\textit{bound\_disjoint}] & :\Leftrightarrow & \exists\, f \in H(F)\ \forall g \in H(G): \neg \textit{poiInRegion}(dp(f), G)\ \wedge \\
 & & & dp(f) \in B(F)\ \wedge\ \neg \textit{on}(dp(f), g.s) \\
\text{(viii)} & v_G[\textit{seg\_unshared}] & :\Leftrightarrow & \exists g \in H(G)\ \forall\, f \in H(F): \neg \textit{segIntersect}(f.s, g.s)
\end{array}
$$

The operation *segInRegion* checks whether a segment is located inside a *region2D* object.

Finally, we consider the possible topological events between two *region2D* objects. The main goal is here to determine the existing *segment classes* in both objects. The segment class of a segment $s$ is a pair $(m/n)$ of *overlap numbers* indicating the number of overlapping *region2D* objects below/above $s$ in a given scene. In our case, $0 \leq m,n \leq 2$ holds. The topological feature vector for each object is therefore a *segment classification vector*. Each vector contains a field for the segment classes $(0/1)$, $(1/0)$, $(0/2)$, $(2/0)$, $(1/2)$, $(2/1)$, and $(1/1)$. We know from [39] that the pairs $(0/0)$ and $(2/2)$ are invalid. Another flag *bound_poi_shared* indicates whether any two unequal boundary segments of both objects share a common point. The following definition makes a later needed connection between representational and point set topological concepts. For a segment $s = (p,q) \in$ *seg2D*, the function *pts* yields the infinite point set of $s$ as $pts(s) = \{r \in \mathbb{R}^2 \,|\, r = p + \lambda(q-p), \lambda \in \mathbb{R}, 0 \leq \lambda \leq 1\}$. Further, for $F \in$ *region2D*, we define $\partial F = \bigcup_{f \in H(F)} pts(f.s)$, $F^\circ = \{p \in \mathbb{R}^2 \,|\, \textit{poiInRegion}(p, F)\}$, and $F^- = \mathbb{R}^2 - \partial F - F^\circ$.

**Definition 6** Let $F, G \in$ *region2D* and $v_F$ be the segment classification vector of $F$. Then

$$
\begin{array}{rlll}
\text{(i)} & v_F[(0/1)] & :\Leftrightarrow & \exists\, f \in H(F): f.ia\ \wedge\ pts(f.s) \subset G^- \\
\text{(ii)} & v_F[(1/0)] & :\Leftrightarrow & \exists\, f \in H(F): \neg f.ia\ \wedge\ pts(f.s) \subset G^- \\
\text{(iii)} & v_F[(1/2)] & :\Leftrightarrow & \exists\, f \in H(F): f.ia\ \wedge\ pts(f.s) \subset G^\circ \\
\text{(iv)} & v_F[(2/1)] & :\Leftrightarrow & \exists\, f \in H(F): \neg f.ia\ \wedge\ pts(f.s) \subset G^\circ \\
\text{(v)} & v_F[(0/2)] & :\Leftrightarrow & \exists\, f \in H(F)\ \exists g \in H(G): f.s = g.s\ \wedge\ f.ia\ \wedge\ g.ia \\
\text{(vi)} & v_F[(2/0)] & :\Leftrightarrow & \exists\, f \in H(F)\ \exists g \in H(G): f.s = g.s\ \wedge\ \neg f.ia\ \wedge\ \neg g.ia \\
\text{(vii)} & v_F[(1/1)] & :\Leftrightarrow & \exists\, f \in H(F)\ \exists g \in H(G): f.s = g.s\ \wedge \\
 & & & ((f.ia\ \wedge\ \neg g.ia)\ \vee\ (\neg f.ia\ \wedge\ g.ia)) \\
\text{(viii)} & v_F[\textit{bound\_poi\_shared}] & :\Leftrightarrow & \exists\, f \in H(F)\ \exists g \in H(G): f.s \neq g.s\ \wedge\ dp(f) = dp(g)
\end{array}
$$

The segment classification vector $v_G$ of $G$ includes the cases (i) to (iv) with $F$ and $G$ swapped; we omit the flags for the cases (v) to (viii) due to their symmetry (or equivalence) to flags of $F$.

### 3.2 Direct Predicate Characterization as a Simple Evaluation Method

The goal of the evaluation phase is to leverage the output of the exploration phase, i.e., the topological feature vectors $v_F$ and $v_G$ of two given spatial argument objects $F$ and $G$, for either verifying a given (proper or dimension-refined) topological predicate or determining such a predicate. A first method already presented in [39] is the *direct predicate characterization* of all $n$ topological predicates of each type combination (see Figure 2(b) for the different type combinations and values of $n$). For example, for the *line2D*/*line2D* case, we have to determine which topological flags of $v_F$ and $v_G$ must be turned on and off so that a given topological predicate (verification query) or a predicate to be found (determination query) is fulfilled. The direct predicate characterization gives an answer for each individual predicate of each individual type combination. This means that we obtain 184 individual predicate characterizations without converse predicates and 248 individual predicate characterizations with converse predicates. In general, each characterization is a Boolean expression in conjunctive normal form and expressed in terms of $v_F$ and $v_G$.

As an example, we consider the topological predicate number 8 (*meet*) between two *line2D* objects $F$ and $G$ (Figure 5 and [38]) and see how the flags of the topological feature vectors (Definition 4) are used.

$$\begin{aligned} p_8(F,G) \quad :\Leftrightarrow \quad & \neg v_F[\mathit{seg\_shared}] \wedge \neg v_F[\mathit{interior\_poi\_shared}] \wedge v_F[\mathit{seg\_unshared}] \wedge \\ & \neg v_F[\mathit{bound\_on\_interior}] \wedge v_F[\mathit{bound\_shared}] \wedge v_F[\mathit{bound\_disjoint}] \wedge \\ & v_G[\mathit{seg\_unshared}] \wedge \neg v_G[\mathit{bound\_on\_interior}] \wedge v_G[\mathit{bound\_disjoint}] \end{aligned}$$

If we take into account the semantics of the topological feature flags, the right side of the equivalence means that both objects may only and must share boundary parts. More precisely and by considering the matrix in Figure 5, intersections between both interiors ($\neg v_F[\mathit{seg\_shared}]$, $\neg v_F[\mathit{interior\_poi\_shared}]$) as well as between the boundary of one object and the interior of the other object ($\neg v_F[\mathit{bound\_on\_interior}]$, $\neg v_G[\mathit{bound\_on\_interior}]$) are not allowed; besides intersections between both boundaries ($v_F[\mathit{bound\_shared}]$), each component of one object must interact with the exterior of the other object ($v_F[\mathit{seg\_unshared}]$, $v_G[\mathit{seg\_unshared}]$, $v_F[\mathit{bound\_disjoint}]$, $v_G[\mathit{bound\_disjoint}]$).

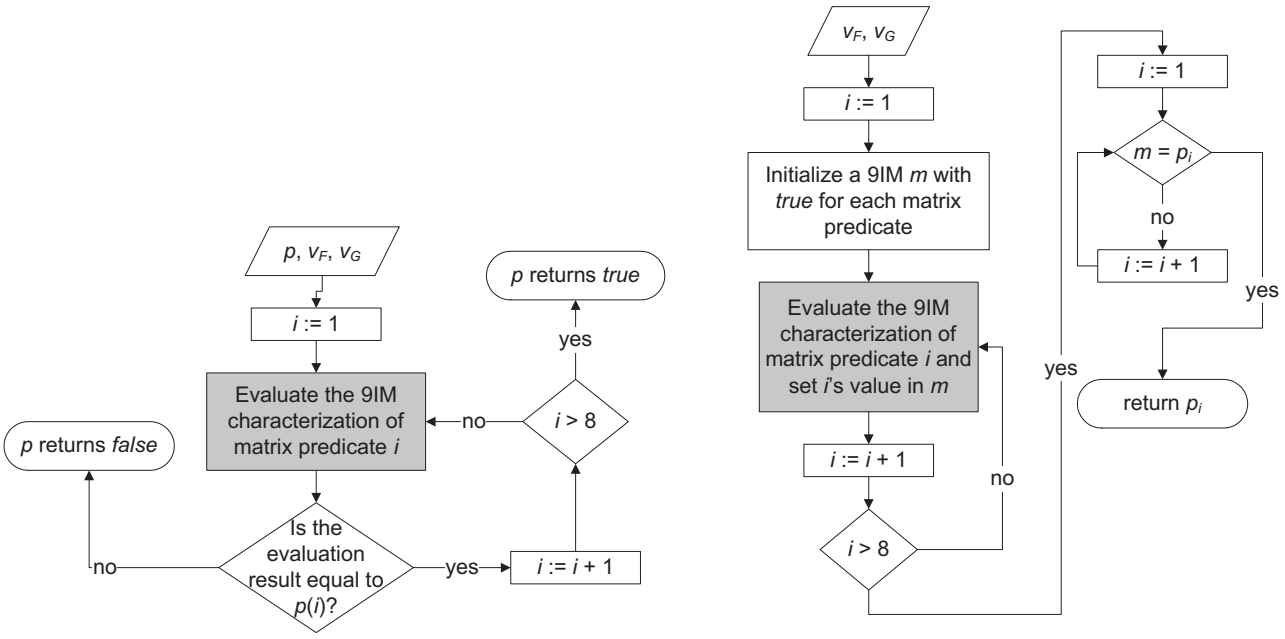
$$\begin{pmatrix} 0 & 0 & 1 \\ 0 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix}$$

Figure 5: The 9-intersection matrix number 8 for the predicate *meet* between two *line2D* objects

The predicate characterizations can be read in both directions. For predicate verification, i.e., for evaluating a specific topological predicate, we look from left to right and check the respective right side of the predicate's direct characterization. For predicate determination, i.e., for deriving the topological relationship from a given spatial configuration of two spatial objects, we have to look from right to left. That is, consecutively we evaluate the right sides of the predicate characterizations by applying them to the given topological feature vectors $v_F$ and $v_G$. For the characterization that matches we look on its left side to obtain the name or number of the pertaining predicate.

## 4 The Evaluation Phase: 9-Intersection Matrix Characterization for Matching Topological Relationships

The motivation for this article consists in the fact that the direct predicate characterization method suffers from three main shortcomings. First, this evaluation method depends on the number of topological predicates. That is, each of the 184 (248) topological predicates between complex spatial objects requires an own specification. Second, in the worst case, all direct predicate characterizations with respect to a particular




(a) (b)

Figure 6: Predicate verification (a) and predicate determination (b) based on the 9-intersection matrix characterization method

type combination have to be checked for predicate determination. Third, the direct predicate characterization method is error-prone. It is difficult to ensure that each characterization is correct and unique and that all characterizations together are mutually exclusive and cover all topological predicates. From this standpoint, this solution is an *ad hoc* approach.

In the following, we present a sophisticated approach, called *9-intersection matrix characterization* (*9IMC*), that avoids these shortcomings and has a formal foundation. Besides a higher efficiency, this approach has the additional feature that its correctness can be formally proved. Section 4.1 introduces the general method. The next two subsections elaborate on a particular step of the general method that is dependent on the type combination under consideration. Section 4.3 deals with the special *region2D*/*region2D* case while Section 4.2 handles the cases of all other type combinations. Section 4.4 applies and extends our approach to the evaluation of dimension-refined topological predicates.

## 4.1 General Method

Instead of characterizing each topological predicate directly, the central idea of our second approach is to uniquely characterize each element of the $3 \times 3$-matrix of the 9-intersection model (Figure 2(a)) by means of the topological feature vectors $v_F$ and $v_G$. As we know, each matrix element is a predicate called *matrix predicate* that checks one of the nine intersections between the boundary $\partial F$, interior $F^\circ$, or exterior $F^-$ of a spatial object $F$ with the boundary $\partial G$, interior $G^\circ$, or exterior $G^-$ of another spatial object $G$ for inequality to the empty set. For each topological predicate, its specification is then given as the logical conjunction of the characterizations of the nine matrix predicates. Since the topological feature vectors are different for each type combination, the characterization of each matrix predicate is different for each type combination too. The characterizations themselves are the themes of the next subsections.

Figure 6(a) shows the general method for predicate verification. Based on the topological predicate $p$ to be verified as well as $v_F$ and $v_G$ as input, we evaluate in a loop the characterizations of all matrix predicates

numbered from left to right and from top to bottom. The nineth matrix predicate $F^- \cap G^- \neq \varnothing$ always yields *true* [38]; hence, we do not have to check it. After the computation (dark shaded action element) of the value of the matrix predicate *i*, we compare it to the corresponding value of the matrix predicate $p(i)$ of *p*. If the values are equal, we proceed with the next matrix predicate $i+1$. Otherwise, we stop, and *p* yields *false*. If there is a coincidence between the computed values of all matrix predicates with the corresponding values of *p*'s matrix, *p* yields *true*. The benefit of this approach is that it only requires eight predicate characterizations and that these characterizations are the same for each of the *n* topological predicates of the same type combination. In particular, an individual characterization of all *n* topological predicates is not needed here. In Section 5.1 we show that this method can be even further improved.

Figure 6(b) shows the general method for predicate determination. Based on $v_F$ and $v_G$ as input, we evaluate the 9IM characterizations (dark shaded action element) of all eight matrix predicates and insert the Boolean values into an intersection matrix *m* initialized with *true* for each matrix predicate. Matrix *m* is then compared against the matrices $p_i$ of all *n* topological predicates. We know that one of them must match *m*. The merit of this approach is that only eight characterizations are needed to determine the intersection matrix of the topological predicate. But unfortunately we need *n* matrix comparisons to determine the pertaining topological predicate in the worst case. In Section 5.2 we introduce a method that eliminates this problem to a large extent. But the method here is already a significant improvement compared with the necessity to compute all *n* direct predicate characterizations.

We apply the general method and the characterizations to both proper and dimension-refined topological predicates. However, the dimension-refined predicates require some additional characterizations.

## 4.2 Type Combination Dependent 9-Intersection Matrix Characterization

The last, missing step refers to the characterizations of the eight matrix predicates of the 9-intersection matrix for all spatial data type combinations. A 9IMC means that each matrix predicate, which takes abstract, infinite point sets *F* and *G* representing spatial objects as arguments, is uniquely characterized by the topological feature vectors $v_F$ and $v_G$, which are discrete implementation concepts. For this purpose, for each discrete spatial object $F \in \alpha \in \{point2D, line2D, region2D\}$, we determine the corresponding abstract point sets of its boundary, interior, and exterior. For the *region2D* data type, we have already done this for Definition 6. For $F \in point2D$, we define $\partial F = \varnothing$, $F^\circ = P(F)$, and $F^- = \mathbb{R}^2 - P(F)$. For $F \in line2D$, we define $\partial F = \{p \in \mathbb{R}^2 \mid \forall f,g \in H(F), f \neq q : p = dp(f) \Rightarrow p \neq dp(g)\}$, $F^\circ = \bigcup_{f \in H(F)} pts(f.s) - \partial F$, and $F^- = \mathbb{R}^2 - \partial F - F^\circ$. As we will see, each characterization can be performed in constant time, and its correctness can be shown by a simple proof. In this subsection, we present the characterizations for all type combinations except for the more complicated case of two *region2D* objects; this case is dealt with in the next subsection. The central idea in the proofs of the lemmas below is to accomplish a correspondence between a matrix predicate based on the point sets $\partial F$, $F^\circ$, $F^-$, $\partial G$, $G^\circ$, and $G^-$ and an equivalent expression based on finite representations like $P(F), H(F)$, $B(F)$, $P(G), H(G)$, and $B(G)$.

In case of two *point2D* objects, the $3 \times 3$-matrix is reduced to a $2 \times 2$-matrix since the boundary of a *point2D* object is defined to be empty [38]. We obtain the following statement:

**Lemma 1** Let $F, G \in point2D$. Then the characterization of the matrix predicates of the (reduced) 9-intersection matrix is as follows:

(i) $F^\circ \cap G^\circ \neq \varnothing \quad \Leftrightarrow \quad v_F[poi\_shared]$
(ii) $F^\circ \cap G^- \neq \varnothing \quad \Leftrightarrow \quad v_F[poi\_disjoint]$
(iii) $F^- \cap G^\circ \neq \varnothing \quad \Leftrightarrow \quad v_G[poi\_disjoint]$
(iv) $F^- \cap G^- \neq \varnothing \quad \Leftrightarrow \quad true$

*Proof.* In (i), the intersection of the interiors of $F$ and $G$ is non-empty if, and only if, both objects share a point. That is, $\exists\, f \in P(F)\ \exists g \in P(G) : \mathit{equal}(f,g)$. This matches directly the definition of $v_F[\mathit{poi\_shared}]$ in Definition 1(i). In (ii), a point of $F$ can only be part of the exterior of $G$ if it does not belong to $G$. That is, $\exists\, f \in P(F)\ \forall g \in P(G) : \neg \mathit{equal}(f,g)$. This fits directly to the definition of $v_F[\mathit{poi\_disjoint}]$ in Definition 1(ii). Case (iii) is symmetric to (ii). Case (iv) follows from Lemma 5.1.2 in [38]. □

In case of a *point2D* object and a *line2D* object, the $3 \times 3$-matrix is reduced to a $2 \times 3$-matrix since the boundary of a *point2D* object is defined to be empty. We obtain the following statement:

**Lemma 2** Let $F \in \mathit{point2D}$ and $G \in \mathit{line2D}$. Then the characterization of the matrix predicates of the (reduced) 9-intersection matrix is as follows:

| | | | |
|---|---|---|---|
| (i) | $F^\circ \cap G^\circ \neq \varnothing$ | $\Leftrightarrow$ | $v_F[\mathit{poi\_on\_interior}]$ |
| (ii) | $F^\circ \cap \partial G \neq \varnothing$ | $\Leftrightarrow$ | $v_F[\mathit{poi\_on\_bound}]$ |
| (iii) | $F^\circ \cap G^- \neq \varnothing$ | $\Leftrightarrow$ | $v_F[\mathit{poi\_disjoint}]$ |
| (iv) | $F^- \cap G^\circ \neq \varnothing$ | $\Leftrightarrow$ | *true* |
| (v) | $F^- \cap \partial G \neq \varnothing$ | $\Leftrightarrow$ | $v_G[\mathit{bound\_poi\_disjoint}]$ |
| (vi) | $F^- \cap G^- \neq \varnothing$ | $\Leftrightarrow$ | *true* |

*Proof.* In (i), the intersection of the interiors of $F$ and $G$ is non-empty if, and only if, a point of $F$ is located on $G$ but is not a boundary point of $G$. That is, $\exists\, f \in P(F)\ \exists g \in H(G)\ \forall b \in B(G) : \mathit{on}(f, g.s) \wedge f \neq b$. This corresponds directly to the definition of $v_F[\mathit{poi\_on\_interior}]$ in Definition 2(ii). In (ii), the intersection of the interior of $F$ and the boundary of $G$ is non-empty if, and only if, a point of $F$ coincides with a boundary point of $G$. That is, $\exists\, f \in P(F)\ \exists g \in B(G) : f = g$. But this matches the definition of $v_F[\mathit{poi\_on\_bound}]$ in Definition 2(iii). Statement (iii) is satisfied if, and only if, a point of $F$ is outside of $G$. That is, $\exists\, f \in P(F)\ \forall g \in H(G) : \neg \mathit{on}(f, g.s)$. But this is just the definition of $v_F[\mathit{poi\_disjoint}]$ in Definition 2(i). Statement (iv) always holds according to Lemma 6.1.2 in [38]. To be fulfilled, statement (v) requires that a boundary point of $G$ lies outside of $F$. That is, $\exists g \in B(G)\ \forall\, f \in P(F) : f \neq g$. This corresponds to the definition of $v_G[\mathit{bound\_poi\_disjoint}]$ in Definition 2(iv). The last statement follows from Lemma 6.1.3 in [38]. □

In case of a *point2D* object and a *region2D* object, we also obtain a reduction of the $3 \times 3$-matrix to a $2 \times 3$-matrix. We obtain the following statement:

**Lemma 3** Let $F \in \mathit{point2D}$ and $G \in \mathit{region2D}$. Then the characterization of the matrix predicates of the (reduced) 9-intersection matrix is as follows:

| | | | |
|---|---|---|---|
| (i) | $F^\circ \cap G^\circ \neq \varnothing$ | $\Leftrightarrow$ | $v_F[\mathit{poi\_inside}]$ |
| (ii) | $F^\circ \cap \partial G \neq \varnothing$ | $\Leftrightarrow$ | $v_F[\mathit{poi\_on\_bound}]$ |
| (iii) | $F^\circ \cap G^- \neq \varnothing$ | $\Leftrightarrow$ | $v_F[\mathit{poi\_outside}]$ |
| (iv) | $F^- \cap G^\circ \neq \varnothing$ | $\Leftrightarrow$ | *true* |
| (v) | $F^- \cap \partial G \neq \varnothing$ | $\Leftrightarrow$ | *true* |
| (vi) | $F^- \cap G^- \neq \varnothing$ | $\Leftrightarrow$ | *true* |

*Proof.* Statement (i) requires that a point of $F$ is located inside $G$ but not on the boundary of $G$. That is, $\exists\, f \in P(F) : \mathit{poiInRegion}(f, G)$ (where *poiInRegion* is the predicate which checks whether a single point lies inside a *region2D* object). This corresponds directly to the definition of $v_F[\mathit{poi\_inside}]$ in Definition 3(i). In (ii), the intersection of $F$ and the boundary of $G$ is non-empty if, and only if, a point of $F$ lies on one of the boundary segments of $G$. That is, $\exists\, f \in P(F)\ \exists (g, ia) \in H(G) : \mathit{on}(f, g.s)$. This matches the definition of $v_F[\mathit{poi\_on\_bound}]$ in Definition 3(ii). Statement (iii) is satisfied if, and only if, a point of $F$ is outside of $G$. That is, $\exists\, f \in P(F)\ \forall (g, ia) \in H(G) : \neg \mathit{poiInRegion}(f, G) \wedge \neg \mathit{on}(f, g.s)$. This corresponds to the definition of $v_F[\mathit{poi\_outside}]$ in Definition 3(iii). Statements (iv) and (v) follow from Lemma 6.2.3 in [38]. The last statement follows from Lemma 6.2.1 in [38]. □

In case of two *line2D* objects, we obtain the following statement:

**Lemma 4** Let $F, G \in$ *line2D*. Then the characterization of the matrix predicates of the 9-intersection matrix is as follows:

$$
\begin{array}{rlcl}
\text{(i)} & F^\circ \cap G^\circ \neq \varnothing & \Leftrightarrow & v_F[\mathit{seg\_shared}] \vee v_F[\mathit{interior\_poi\_shared}] \\
\text{(ii)} & F^\circ \cap \partial G \neq \varnothing & \Leftrightarrow & v_G[\mathit{bound\_on\_interior}] \\
\text{(iii)} & F^\circ \cap G^- \neq \varnothing & \Leftrightarrow & v_F[\mathit{seg\_unshared}] \\
\text{(iv)} & \partial F \cap G^\circ \neq \varnothing & \Leftrightarrow & v_F[\mathit{bound\_on\_interior}] \\
\text{(v)} & \partial F \cap \partial G \neq \varnothing & \Leftrightarrow & v_F[\mathit{bound\_shared}] \\
\text{(vi)} & \partial F \cap G^- \neq \varnothing & \Leftrightarrow & v_F[\mathit{bound\_disjoint}] \\
\text{(vii)} & F^- \cap G^\circ \neq \varnothing & \Leftrightarrow & v_G[\mathit{seg\_unshared}] \\
\text{(viii)} & F^- \cap \partial G \neq \varnothing & \Leftrightarrow & v_G[\mathit{bound\_disjoint}] \\
\text{(ix)} & F^- \cap G^- \neq \varnothing & \Leftrightarrow & \mathit{true}
\end{array}
$$

*Proof.* In (i), the interiors of two *line2D* objects intersect if, and only if, any two segments partially or completely coincide or if two segments share a single point that does not belong to the boundaries of $F$ and $G$. That is, $\exists f \in H(F)\ \exists g \in H(G) : \mathit{segIntersect}(f.s, g.s) \vee \exists f \in H(F)\ \exists g \in H(G)\ \forall p \in B(F) \cup B(G) : \mathit{poiIntersect}(f.s, g.s) \wedge \mathit{poiIntersection}(f.s, g.s) \neq p$. The first expression corresponds to the definition of $v_F[\mathit{seg\_shared}]$ in Definition 4(i). The second expression is the definition of $v_F[\mathit{interior\_poi\_shared}]$ in Definition 4(ii). Statement (ii) requires that an intersection point $p$ of $F$ and $G$ exists such that $p$ is a boundary point of $G$ but not a boundary point of $F$. That is, $\exists f \in H(F)\ \exists g \in H(G)\ \exists p \in B(G) \setminus B(F) : \mathit{poiIntersection}(f.s, g.s) = p$. This matches the definition of $v_G[\mathit{bound\_on\_interior}]$ in Definition 4(viii). Statement (iii) is satisfied if, and only if, there is a segment of $F$ that is outside of $G$. That is, $\exists f \in H(F) \forall g \in H(G) : \neg \mathit{segIntersect}(f.s, g.s)$. This corresponds to the definition of $v_F[\mathit{seg\_unshared}]$ in Definition 4(iii). Statement (iv) is symmetric to statement (ii) and based on Definition 4(iv). In (v), the boundaries of $F$ and $G$ intersect if, and only if, they share a boundary point. That is, $\exists p \in B(F)\ \exists q \in B(G) : p = q$. This matches the definition of $v_F[\mathit{bound\_shared}]$ in Definition 4(v). Statement (vi) requires the existence of a boundary point of $F$ that is not located on any segment of $G$. That is, $\exists p \in B(F)\ \forall g \in H(G) : \neg \mathit{on}(p, g.s)$. This corresponds to the definition of $v_F[\mathit{bound\_disjoint}]$ in Definition 4(vi). Statement (vii) is symmetric to statement (iii) and based on Definition 4(vii). Statement (viii) is symmetric to statement (vi) and based on Definition 4(ix). The last statement follows from Lemma 5.2.1 in [38]. □

In case of a *line2D* object and a *region2D* object, we obtain the following statement:

**Lemma 5** Let $F \in$ *line2D* and $G \in$ *region2D*. Then the characterization of the matrix predicates of the 9-intersection matrix is as follows:

$$
\begin{array}{rlcl}
\text{(i)} & F^\circ \cap G^\circ \neq \varnothing & \Leftrightarrow & v_F[\mathit{seg\_inside}] \\
\text{(ii)} & F^\circ \cap \partial G \neq \varnothing & \Leftrightarrow & v_F[\mathit{seg\_shared}] \vee v_F[\mathit{poi\_shared}] \\
\text{(iii)} & F^\circ \cap G^- \neq \varnothing & \Leftrightarrow & v_F[\mathit{seg\_outside}] \\
\text{(iv)} & \partial F \cap G^\circ \neq \varnothing & \Leftrightarrow & v_F[\mathit{bound\_inside}] \\
\text{(v)} & \partial F \cap \partial G \neq \varnothing & \Leftrightarrow & v_F[\mathit{bound\_shared}] \\
\text{(vi)} & \partial F \cap G^- \neq \varnothing & \Leftrightarrow & v_F[\mathit{bound\_disjoint}] \\
\text{(vii)} & F^- \cap G^\circ \neq \varnothing & \Leftrightarrow & \mathit{true} \\
\text{(viii)} & F^- \cap \partial G \neq \varnothing & \Leftrightarrow & v_G[\mathit{seg\_unshared}] \\
\text{(ix)} & F^- \cap G^- \neq \varnothing & \Leftrightarrow & \mathit{true}
\end{array}
$$

*Proof.* In (i), the interiors of $F$ and $G$ intersect if, and only if, a segment of $F$ is located in $G$ but does not coincide with a boundary segment of $G$. That is, $\exists f \in H(F)\ \forall g \in H(G) : \neg \mathit{segIntersect}(f.s, g.s) \wedge \mathit{segInRegion}(f.s, G)$. This corresponds to the definition of $v_F[\mathit{seg\_inside}]$ in Definition 5(i). Statement (ii) requires that either $F$ and $G$ share a segment, or they share an intersection point that is not a boundary point of $F$. That is, $\exists f \in H(F)\ \exists g \in H(G) : \mathit{segIntersect}(f.s, g.s) \vee \exists f \in H(F)\ \exists g \in H(G) :$

*poiIntersect*$(f.s, g.s) \wedge$ *poiIntersection*$(f.s, g.s) \notin B(F)$. The first argument of the disjunction matches the definition of $v_F[$*seg_shared*$]$ in Definition 5(ii). The second argument matches the definition of $v_F[$*poi_shared*$]$ in Definition 5(iv). Statement (iii) is satisfied if, and only if, a segment of $F$ is located outside of $G$. That is, $\exists f \in H(F)\ \forall g \in H(G) : \neg$*segIntersect*$(f.s, g.s) \wedge \neg$*segInRegion*$(f.s, G)$. This corresponds to the definition of $v_F[$*seg_outside*$]$ in Definition 5(iii). Statement (iv) holds if, and only if, a segment of $F$ lies inside $G$ and one of the end points of the segment is a boundary point. That is, $\exists f \in H(F)$ : *poiInRegion*$(dp(f), G) \wedge dp(f) \in B(F)$. This corresponds to the definition of $v_F[$*bound_inside*$]$ in Definition 5(v). In (v), we must find a segment of $F$ and a segment of $G$ which intersect in a point that is a boundary point of $F$. That is, $\exists f \in H(F)\ \exists g \in H(G) :$ *poiIntersect*$(f.s, g.s) \wedge$ *poiIntersection*$(f.s, g.s) \in B(F)$. This matches the definition of $v_F[$*bound_shared*$]$ in Definition 5(vi). Statement (vi) requires the existence of an endpoint of a segment of $F$ that is a boundary point and not located inside or on any segment of $G$. That is, $\exists f \in H(F)\ \forall g \in H(G) : \neg$*poiInRegion*$(dp(f), G) \wedge dp(f) \in B(F) \wedge \neg$*on*$(dp(f), g.s)$. This corresponds to the definition of $v_F[$*bound_disjoint*$]$ in Definition 5(vii). Statement (vii) always holds according to Lemma 6.3.2 in [38]. Statement (viii) is satisfied if, and only if, a segment of $G$ does not coincide with any segment of $F$. That is, $\exists g \in H(G)\ \forall f \in H(F) : \neg$*segIntersect*$(f.s, g.s)$. This fits to the definition of $v_F[$*seg_unshared*$]$ in Definition 5(viii). The last statement follows from Lemma 6.3.1 in [38]. □

### 4.3 9-Intersection Matrix Characterization of the *region2D*/*region2D* Case

As indicated in [39], the exploration algorithm for the *region2D*/*region2D* case is quite different from the algorithms of the other type combinations. It has to take into account the areal extent of both objects and has resulted in the concepts of overlap number, segment classes, and segment classification vector. In this subsection, we deal with the 9IMC based on two segment classification vectors. The goal of the following lemmas is to prepare the unique characterization of all matrix predicates by means of segment classes. The first lemma provides a translation of each segment class into a matrix predicate expression.

**Lemma 6** Let $F, G \in$ *region2D* and $v_F$ and $v_G$ be their segment classification vectors. Then we can infer the following implications and equivalences between segment classes and matrix predicates:

| | | | |
|---|---|---|---|
| (i) | $v_F[(0/1)] \vee v_F[(1/0)]$ | $\Leftrightarrow$ | $\partial F \cap G^- \neq \varnothing$ |
| (ii) | $v_G[(0/1)] \vee v_G[(1/0)]$ | $\Leftrightarrow$ | $F^- \cap \partial G \neq \varnothing$ |
| (iii) | $v_F[(1/2)] \vee v_F[(2/1)]$ | $\Leftrightarrow$ | $\partial F \cap G^\circ \neq \varnothing$ |
| (iv) | $v_G[(1/2)] \vee v_G[(2/1)]$ | $\Leftrightarrow$ | $F^\circ \cap \partial G \neq \varnothing$ |
| (v) | $v_F[(0/2)] \vee v_F[(2/0)]$ | $\Rightarrow$ | $\partial F \cap \partial G \neq \varnothing \wedge F^\circ \cap G^\circ \neq \varnothing$ |
| (vi) | $v_F[(1/1)]$ | $\Rightarrow$ | $\partial F \cap \partial G \neq \varnothing \wedge F^\circ \cap G^- \neq \varnothing \wedge F^- \cap G^\circ \neq \varnothing$ |

*Proof.* According to Definition 6(i) and (ii), the left side of (i) is equivalent to the expression $\exists f \in H(F)$ : $pts(f.s) \subset G^-$. This is equivalent to $\partial F \cap G^- \neq \varnothing$. The proof of (iii) is similar and based on Definition 6(iii) and (iv); only the term $G^-$ has to be replaced by $G^\circ$. The proof of (ii) can be obtained by swapping the roles of $F$ and $G$ in (i). Similarly, the proof of (iv) requires a swapping of $F$ and $G$ in (iii). According to Definition 6(v) and (vi), the left side of (v) is equivalent to the expression $\exists f \in H(F)\ \exists g \in H(G)$ : $f.s = g.s \wedge ((f.ia \wedge g.ia) \vee (\neg f.ia \wedge \neg g.ia))$. From the first element of the conjunction, we can (only) conclude that $\partial F \cap \partial G \neq \varnothing$. Equivalence does not hold since two boundaries can also intersect if they only share single intersection or meeting points but not (half)segments. The second element of the conjunction requires that the interiors of both *region2D* objects are located on the same side. Hence, $F^\circ \cap G^\circ \neq \varnothing$ must hold. Also this is only an implication since an intersection of both interiors is possible without having any $(0/2)$- or $(2/0)$-segments. According to Definition 6(vii), the left side of (vi) is equivalent to the expression $\exists f \in H(F)\ \exists g \in H(G) : f.s = g.s \wedge ((f.ia \wedge \neg g.ia) \vee (\neg f.ia \wedge g.ia))$. The first element of the conjunction implies that $\partial F \cap \partial G \neq \varnothing$. The second element of the conjunction requires that the interiors of both *region2D*

objects are located on different sides. Since the definition of type *region2D* disallows $(1/1)$-segments for single objects, the interior of $F$ must intersect the exterior of $G$, and vice versa. This is only an implication since an intersection of the interior of one *region2D* object with the exterior of another *region2D* object is possible without having $(1/1)$-segments. □

The second lemma provides a translation of some matrix predicates into segment classes.

**Lemma 7** Let $F, G \in$ *region2D* and $v_F$ and $v_G$ be their segment classification vectors. Then we can infer the following implications between matrix predicates and segment classes:

$$\begin{array}{lll}
\text{(i)} & F^\circ \cap G^\circ \neq \varnothing & \Rightarrow \; v_F[(0/2)] \vee v_F[(2/0)] \vee v_F[(1/2)] \vee v_F[(2/1)] \vee \\
& & \quad\; v_G[(1/2)] \vee v_G[(2/1)] \\
\text{(ii)} & F^\circ \cap G^- \neq \varnothing & \Rightarrow \; v_F[(0/1)] \vee v_F[(1/0)] \vee v_F[(1/1)] \vee \\
& & \quad\; v_G[(1/2)] \vee v_G[(2/1)] \\
\text{(iii)} & F^- \cap G^\circ \neq \varnothing & \Rightarrow \; v_F[(1/2)] \vee v_F[(2/1) \vee v_F[(1/1)] \vee \\
& & \quad\; v_G[(0/1)] \vee v_G[(1/0)]
\end{array}$$

*Proof.* In (i), the intersection of the interiors of $F$ and $G$ implies that both objects share a common area. Consequently, this area must have overlap number 2 so that at least one of the two objects must have a $(a/2)$- or $(2/a)$-segment with $a \in \{0,1\}$. In (ii), the fact that $F^\circ$ intersects $G^-$ means that $F$ contains an area which it does not share with $G$. That is, the overlap number of this area is 1, and $F$ must have a $(a/1)$- or a $(1/a)$-segment with $a \in \{0,1\}$. The fact that a part of the interior of $F$ is located outside of $G$ implies two possible topological situations for $G$: either both objects share a common segment and their interiors are on different sides, i.e., $G$ has a $(1/1)$-segment (covered by $v_F[(1/1)]$), or the interior of $F$ is intersected by the boundary and the interior of $G$ so that $G$ has a $(1/2)$- or $(2/1)$-segment. We prove (iii) by swapping $F$ and $G$ in (ii). □

The third lemma states some implications between matrix predicates.

**Lemma 8** Let $F, G \in$ *region2D*. Then we can infer the following implications between matrix predicates:

$$\begin{array}{lll}
\text{(i)} & v_F[\textit{bound_poi_shared}] & \Rightarrow \; \partial F \cap \partial G \neq \varnothing \\
\text{(ii)} & \partial F \cap G^- \neq \varnothing & \Rightarrow \; F^\circ \cap G^- \neq \varnothing \wedge F^- \cap G^- \neq \varnothing \\
\text{(iii)} & F^- \cap \partial G \neq \varnothing & \Rightarrow \; F^- \cap G^\circ \neq \varnothing \wedge F^- \cap G^- \neq \varnothing \\
\text{(iv)} & \partial F \cap G^\circ \neq \varnothing & \Rightarrow \; F^\circ \cap G^\circ \neq \varnothing \wedge F^- \cap G^\circ \neq \varnothing \\
\text{(v)} & F^\circ \cap \partial G \neq \varnothing & \Rightarrow \; F^\circ \cap G^\circ \neq \varnothing \wedge F^\circ \cap G^- \neq \varnothing
\end{array}$$

*Proof.* Statement (i) can be shown by considering the definition of *bound_poi_shared*. This flag is *true* if any two halfsegments of $F$ and $G$ share a single meeting or intersection point. Hence, the intersection of both boundaries is non-empty. The proofs for (ii) to (v) require point set topological concepts. Statements (ii) and (iii) follow from Lemma 5.3.6 in [38]. Statements (iv) and (v) result from Lemma 5.3.5 in [38]. □

The following theorem collects the results we have already obtained so far and proves the lacking parts of the nine matrix predicate characterizations.

**Theorem 1** Let $F, G \in$ *region2D* and $v_F$ and $v_G$ be their segment classification vectors. Then the matrix predicates of the 9-intersection matrix are equivalent to the following segment class characterizations:

$$\begin{array}{lll}
\text{(i)} & F^\circ \cap G^\circ \neq \varnothing & \Leftrightarrow \; v_F[(0/2)] \vee v_F[(2/0)] \vee v_F[(1/2)] \vee v_F[(2/1)] \vee \\
& & \quad\; v_G[(1/2)] \vee v_G[(2/1)] \\
\text{(ii)} & F^\circ \cap \partial G \neq \varnothing & \Leftrightarrow \; v_G[(1/2)] \vee v_G[(2/1)] \\
\text{(iii)} & F^\circ \cap G^- \neq \varnothing & \Leftrightarrow \; v_F[(0/1)] \vee v_F[(1/0)] \vee v_F[(1/1)] \vee v_G[(1/2)] \vee v_G[(2/1)] \\
\text{(iv)} & \partial F \cap G^\circ \neq \varnothing & \Leftrightarrow \; v_F[(1/2)] \vee v_F[(2/1)]
\end{array}$$

$$
\begin{array}{rlcl}
\text{(v)} & \partial F \cap \partial G \neq \varnothing & \Leftrightarrow & v_F[(0/2)] \vee v_F[(2/0)] \vee v_F[(1/1)] \vee v_F[\textit{bound\_poi\_shared}] \\
\text{(vi)} & \partial F \cap G^- \neq \varnothing & \Leftrightarrow & v_F[(0/1)] \vee v_F[(1/0)] \\
\text{(vii)} & F^- \cap G^\circ \neq \varnothing & \Leftrightarrow & v_F[(1/2)] \vee v_F[(2/1)] \vee v_F[(1/1)] \vee v_G[(0/1)] \vee v_G[(1/0)] \\
\text{(viii)} & F^- \cap \partial G \neq \varnothing & \Leftrightarrow & v_G[(0/1)] \vee v_G[(1/0)] \\
\text{(ix)} & F^- \cap G^- \neq \varnothing & \Leftrightarrow & \textit{true}
\end{array}
$$

*Proof.* For (i), the forward implication corresponds to Lemma 7(i). The backward implication can be derived from Lemma 6(v) for $(0/2)$- and $(2/0)$-segments of $F$ (and $G$). For $(1/2)$- and $(2/1)$-segments, Lemma 6(iii) and 6(iv) imply $\partial F \cap G^\circ \neq \varnothing$ and $F^\circ \cap \partial G \neq \varnothing$, respectively. From these two implications, by using Lemma 8(iv) and 8(v), we can derive in both cases $F^\circ \cap G^\circ \neq \varnothing$. Statements (ii) and (iv) correspond to Lemma 6(iv) and 6(iii), respectively. For (iii) [(vii)], the forward implication corresponds to Lemma 7(ii) [7(iii)]. The backward implication for (iii) [(vii)] requires Lemma 6(i) [6(ii)] and Lemma 8(ii) [8(iii)] for the $(0/1)$- and $(1/0)$-segments of $F$ [$G$], Lemma 6(vi) [6(vi)] for the $(1/1)$-segments of $F$ (and hence $G$), as well as Lemma 6(iv) [6(iii)] and Lemma 8(v) [8(iv)] for the $(1/2)$- and $(2/1)$-segments of $G$ [$F$]. For (v), the forward implication can be shown as follows: if the boundaries of $F$ and $G$ intersect, then either they share a common meeting or intersection point, i.e., the flag $v_F[\textit{bound\_poi\_shared}]$ is set, or there are two halfsegments of $F$ and $G$ whose segment components are equal. No other alternative is possible due to our splitting strategy for halfsegments during the plane sweep. As we know, equal segments of $F$ and $G$ must have the segment classes $(0/2)$, $(2/0)$, or $(1/1)$. The backward implication requires Lemma 6(v) for $(0/2)$- and $(2/0)$-segments of $F$ (and hence $G$), Lemma 6(vi) for $(1/1)$-segments of $F$ (and hence $G$), and Lemma 8(i) for single meeting and intersection points. Statement (vi) [(viii)] corresponds to Lemma 6(i) [6(ii)]. Statement (ix) turns out to be always true since our assumption in an implementation is that our universe of discourse $U$ is always properly larger than the union of spatial objects contained in it. This means for $F$ and $G$ that always $F \cup G \subset U$ holds. We can conclude that $U - (F \cup G) \neq \varnothing$. According to DeMorgan's Laws, this is equivalent to $(U - F) \cap (U - G) \neq \varnothing$. But this leads us to the statement that $F^- \cap G^- \neq \varnothing$. □

Summarizing our results from the last two subsections, we see that Lemmas 1, 2, 3, 4, 5, and Theorem 1 provide us with a unique characterization of each individual matrix predicate of the 9-intersection matrix for each type combination. This approach has several benefits. First, it is a systematically developed and not an *ad hoc* approach. Second, it has a formal and sound foundation. Hence, we can be sure about the correctness of topological flags and segment classes assigned to matrix predicates, and vice versa. Third, this evaluation method is independent of the number of topological predicates and only requires a constant number of evaluations for matrix predicate characterizations. Instead of nine, even only eight matrix predicates have to be checked since the predicate $F^- \cap G^- \neq \varnothing$ yields *true* for all type combinations. Fourth, we have proved the correctness of our provided implementation.

Based on this result, we accomplish the *predicate verification* of a topological predicate $p$ with respect to a particular spatial data type combination on the basis of $p$'s 9-intersection matrix (see the complete matrices in Figures 7 to 9 and 11 to 13) and the topological feature vectors $v_F$ and $v_G$ as follows: Depending on the spatial data type combination, we evaluate the logical expression (given in terms of $v_F$ and $v_G$) on the right side of the first 9IMC according to Lemma 1, 2, 3, 4, 5, or Theorem 1, respectively. We then match the Boolean result with the Boolean value at the respective position in $p$'s intersection matrix. If both Boolean values are equal, we proceed with the next matrix predicate in the 9-intersection matrix; otherwise $p$ is *false*, and the algorithm terminates. Predicate $p$ yields *true* if the Boolean results of the evaluated logical expressions of *all* 9IMCs coincide with the corresponding Boolean values in $p$'s intersection matrix. This requires constant time.

*Predicate determination* also depends on a particular spatial data type combination and leverages 9-intersection matrices and topological feature vectors. In a first step, depending on the spatial data type combination and by means of $v_F$ and $v_G$, we evaluate the logical expressions on all right sides of the 9IMCs

according to Lemma 1, 2, 3, 4, 5, or Theorem 1, respectively. This yields a Boolean 9-intersection matrix. In a second step, this Boolean matrix is checked consecutively for equality against all 9-intersection matrices of the topological predicates of the particular type combination. If $n_{\alpha,\beta}$ with $\alpha, \beta \in \{point2D, line2D, region2D\}$ is the number of topological predicates between the types $\alpha$ and $\beta$, this requires $n_{\alpha,\beta}$ tests in the worst case.

### 4.4 Evaluation of Dimension-Refined Topological Predicates

The 9IMC described so far is applicable to the evaluation of proper topological predicates. In this subsection, we show that, with a few extensions, the 9IMC can also evaluate *dimension-refined* topological predicates (Section 2.3) for which the dimension of an intersection of two spatial objects plays an essential role. That is, the exploration algorithms provide us with enough topological information to perform this task. In this sense, dimension-refined topological predicates are a refinement of proper topological predicates. For example, in a meeting situation, we could be interested in a query whether two *line2D* objects meet in a zero-dimensional object (i.e., a *point2D* object), or in a one-dimensional object (i.e., a *line2D* object), or in both. Dimension refinement is only possible for the type combinations *line2D*/*line2D*, *line2D*/*region2D*, and *region2D*/*region2D*. We distinguish the different dimensions of the maximal connected components of a given point set in the two-dimensional space and define a "dimension type" as follows [30]:

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

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 $\bot$ symbol is used to represent the undefined dimension of an empty point set.

In [30], we have shown that the dimension of the intersection of a zero-, one-, or two-dimensional spatial component with another zero-, one-, or two-dimensional spatial component can only have the value $\bot$, $0D$, $1D$, $2D$, or $01D$. Further, only in case of two one-dimensional components, their intersection may lead to components of different dimensions. This refers exclusively to the intersection between the interiors of two *line2D* objects, between the interior of a *line2D* object and the boundary of a *region2D* object, or between the boundaries of two *region2D* objects. The possible dimension values for these three cases are $\bot$, $0D$, $1D$, or $01D$. They are evaluated or determined by our extended 9IMC method described now.

Let $F, G \in \alpha \in \{line2D, region2D\}$, $p$ be a topological predicate between $F$ and $G$, and $p_d$ be a dimension-refined predicate that is a refinement of $p$. We leverage the following relationship between $p$ and $p_d$:

$$(p_d(F,G) \Rightarrow p(F,G)) \Leftrightarrow (\neg p(F,G) \Rightarrow \neg p_d(F,G))$$

For the predicate verification of $p_d$, we first apply the 9IMC for the predicate verification of $p$. From the right side of the equivalence, we conclude that, if $p(F,G)$ yields *false*, $p_d(F,G)$ must yield *false* too. Otherwise, if $p(F,G)$ yields *true*, we cannot make a statement about $p_d$ because the satisfaction of $p$ is only a necessary but not sufficient condition for the satisfaction of $p_d$. In this case, we need additional *dimension characterizations* that are given in Table 1. For each type combination, the table shows the relevant intersection between the one-dimensional components of these types, the possible dimensions such an intersection can have, and for each dimension its characterization on the basis of the topological feature vectors of this type combination. Since the dimension characterizations are different for each type combination, they are unique. Therefore, we only have to look up the type combination and the dimension of interest, evaluate the corresponding dimension characterization, and obtain its Boolean value as the result for the dimension-refined predicate.

| Type combination | Intersection | Dimension | Dimension characterization |
|---|---|---|---|
| $line \times line$ | $F^\circ \cap G^\circ = \varnothing$ | $\bot$ | $\neg v_F[seg\_shared] \wedge \neg v_F[interior\_poi\_shared]$ |
| | $F^\circ \cap G^\circ \neq \varnothing$ | $0D$ | $\neg v_F[seg\_shared] \wedge v_F[interior\_poi\_shared]$ |
| | | $01D$ | $v_F[seg\_shared] \wedge v_F[interior\_poi\_shared]$ |
| | | $1D$ | $v_F[seg\_shared] \wedge \neg v_F[interior\_poi\_shared]$ |
| $line \times region$ | $F^\circ \cap \partial G = \varnothing$ | $\bot$ | $\neg v_F[seg\_shared] \wedge \neg v_F[poi\_shared]$ |
| | $F^\circ \cap \partial G \neq \varnothing$ | $0D$ | $\neg v_F[seg\_shared] \wedge v_F[poi\_shared]$ |
| | | $01D$ | $v_F[seg\_shared] \wedge v_F[poi\_shared]$ |
| | | $1D$ | $v_F[seg\_shared] \wedge \neg v_F[poi\_shared]$ |
| $region \times region$ | $\partial F \cap \partial G = \varnothing$ | $\bot$ | $\neg(v_F[(0/2)] \vee v_F[(2/0)] \vee v_F[(1/1)] \vee v_F[(bound\_poi\_shared)])$ |
| | $\partial F \cap \partial G \neq \varnothing$ | $0D$ | $\neg(v_F[(0/2)] \vee v_F[(2/0)] \vee v_F[(1/1)]) \wedge v_F[(bound\_poi\_shared)]$ |
| | | $01D$ | $(v_F[(0/2)] \vee v_F[(2/0)] \vee v_F[(1/1)]) \wedge v_F[(bound\_poi\_shared)]$ |
| | | $1D$ | $(v_F[(0/2)] \vee v_F[(2/0)] \vee v_F[(1/1)]) \wedge \neg v_F[(bound\_poi\_shared)]$ |

Table 1: Dimension characterizations for dimension-refined topological predicates

For the predicate determination of the matching dimension-refined topological predicate $p_d$ between $F$ and $G$, we first apply the 9IMC for the predicate determination of the matching topological predicate $p$ between $F$ and $G$. Afterwards, we consecutively evaluate the four dimension characterizations of the type combination under consideration from Table 1. If the resulting dimension is the $\bot$ element, the dimension-refined predicate $p_d$ coincides with $p$. Otherwise, we obtain either $p_d = 0D\text{-}p$, $p_d = 01D\text{-}p$, or $p_d = 1D\text{-}p$.

# 5 Optimized Evaluation Methods

Based on the exploration algorithms and leveraging the 9-intersection matrix characterization, we have found a universal, correct, complete, and effective method for both predicate verification and predicate determination of proper and dimension-refined topological predicates. So far, we have focused on the general applicability and universality of our overall approach. In this section, we show that it is even possible to fine-tune and thus improve our 9IMC approach with respect to efficiency if we look at predicate verification and predicate determination separately. Section 5.1 delineates a sophisticated method called *matrix thinning* for speeding up predicate verification. Section 5.2 describes a fine-tuned method called *minimum cost decision tree* for accelerating predicate determination.

## 5.1 Matrix Thinning for Predicate Verification

The approach of *matrix thinning* (*MT*) described in this subsection is based on the observation that for predicate verification only a subset of the nine matrix predicates has to be evaluated in order to determine the validity of a given topological relationship between two spatial objects $F$ and $G$. For example, for the aforementioned predicate 1 (*disjoint*) of the *region2D*/*region2D* case, the combination that $F^\circ \cap G^\circ = \varnothing \wedge \partial F \cap \partial G = \varnothing$ holds (indicated by two 0's) is unique among the 33 predicates. Consequently, only these two matrix predicates have to be tested in order to decide about *true* or *false* of this topological predicate.

The question arises how the 9-intersection matrices can be systematically "thinned out" and nevertheless remain unique among the $n_{\alpha,\beta}$ topological predicates between two spatial data types $\alpha$ and $\beta$. We use a brute-force algorithm (Figure 10) that is applicable to all type combinations and that determines the thinned

$$1:\begin{pmatrix} 0|0 & 0|* & 1|* \\ 0|* & 0|* & 0|* \\ 1|* & 0|* & 1|* \end{pmatrix} \quad 2:\begin{pmatrix} 1|* & 0|* & 0|0 \\ 0|* & 0|* & 0|* \\ 0|0 & 0|* & 1|* \end{pmatrix} \quad 3:\begin{pmatrix} 1|* & 0|* & 0|0 \\ 0|* & 0|* & 0|* \\ 1|1 & 0|* & 1|* \end{pmatrix} \quad 4:\begin{pmatrix} 1|* & 0|* & 1|1 \\ 0|* & 0|* & 0|* \\ 0|0 & 0|* & 1|* \end{pmatrix} \quad 5:\begin{pmatrix} 1|1 & 0|* & 1|1 \\ 0|* & 0|* & 0|* \\ 1|1 & 0|* & 1|* \end{pmatrix}$$

Figure 7: Complete and thinned out matrices for the 5 topological predicates of the *point2D*/*point2D* case.

$$1:\begin{pmatrix} 0|0 & 0|0 & 1|* \\ 0|* & 0|* & 0|* \\ 1|* & 0|0 & 1|* \end{pmatrix} \quad 2:\begin{pmatrix} 0|0 & 0|0 & 1|* \\ 0|* & 0|* & 0|* \\ 1|* & 1|1 & 1|* \end{pmatrix} \quad 3:\begin{pmatrix} 0|0 & 1|* & 0|0 \\ 0|* & 0|* & 0|* \\ 1|* & 0|0 & 1|* \end{pmatrix} \quad 4:\begin{pmatrix} 0|0 & 1|* & 0|0 \\ 0|* & 0|* & 0|* \\ 1|* & 1|1 & 1|* \end{pmatrix} \quad 5:\begin{pmatrix} 0|0 & 1|1 & 1|1 \\ 0|* & 0|* & 0|* \\ 1|* & 0|0 & 1|* \end{pmatrix}$$

$$6:\begin{pmatrix} 0|0 & 1|1 & 1|1 \\ 0|* & 0|* & 0|* \\ 1|* & 1|1 & 1|* \end{pmatrix} \quad 7:\begin{pmatrix} 1|* & 0|0 & 0|0 \\ 0|* & 0|* & 0|* \\ 1|* & 0|0 & 1|* \end{pmatrix} \quad 8:\begin{pmatrix} 1|* & 0|0 & 0|0 \\ 0|* & 0|* & 0|* \\ 1|* & 1|1 & 1|* \end{pmatrix} \quad 9:\begin{pmatrix} 1|1 & 0|0 & 1|1 \\ 0|* & 0|* & 0|* \\ 1|* & 0|0 & 1|* \end{pmatrix} \quad 10:\begin{pmatrix} 1|1 & 0|0 & 1|1 \\ 0|* & 0|* & 0|* \\ 1|* & 1|1 & 1|* \end{pmatrix}$$

$$11:\begin{pmatrix} 1|1 & 1|1 & 0|0 \\ 0|* & 0|* & 0|* \\ 1|* & 0|0 & 1|* \end{pmatrix} \quad 12:\begin{pmatrix} 1|1 & 1|1 & 0|0 \\ 0|* & 0|* & 0|* \\ 1|* & 1|1 & 1|* \end{pmatrix} \quad 13:\begin{pmatrix} 1|1 & 1|1 & 1|1 \\ 0|* & 0|* & 0|* \\ 1|* & 0|0 & 1|* \end{pmatrix} \quad 14:\begin{pmatrix} 1|1 & 1|1 & 1|1 \\ 0|* & 0|* & 0|* \\ 1|* & 1|1 & 1|* \end{pmatrix}$$

Figure 8: Complete and thinned out matrices for the 14 topological predicates of the *point2D*/*line2D* case.

out version of each intersection matrix associated with one of the $n_{\alpha,\beta}$ topological predicates. Since this algorithm only has to be executed once for each type combination, runtime performance and space efficiency are not important here.

In a first step (lines 8 to 10), we create a matrix *pos* of so-called *position matrices* corresponding to all possible 9-intersection matrices, i.e., to the binary versions of the decimal numbers 1 to 511 if we read the 9-intersection matrix (9IM) entries row by row. Each "1" in a position matrix indicates a position or entry that is later used for checking two intersection matrices against each other. A "0" in a position matrix means that the corresponding entries in two compared intersection matrices are not compared and hence ignored.

Because our goal is to minimize the number of matrix predicates that have to be evaluated, in a second step, we sort the position matrices with respect to the number of ones in increasing order (lines 11 to 12). That is, the list of position matrices will first contain all matrices with a single "1", then the matrices with two ones, etc., until we reach the matrix with nine ones. At the latest here, it is guaranteed that an intersection matrix is different to all the other $n_{\alpha,\beta} - 1$ intersection matrices. Hence, our algorithm terminates.

In a third step, we initialize the entries of all $n_{\alpha,\beta}$ thinned out intersection matrices with the "don't care" symbol "$*$".

The fourth and final step computes the thinned out matrices (lines 15 to 33). The idea is to find for each intersection matrix (line 15) a minimal number of entries that together uniquely differ from the corresponding entries of all the other $n_{\alpha,\beta} - 1$ intersection matrices. Therefore, we start traversing the 511 position matrices (line 17). For all "1"-positions of a position matrix we find out whether for the intersection matrix under consideration another intersection matrix exists that has the same matrix values at these positions (lines 20 to 21). As long as no equality has been found, the intersection matrix under consideration is compared to the next intersection matrix (lines 19 to 23). If an equality is found, the next position matrix is taken (line 30). Otherwise, we have found a minimal number of matrix predicates that are sufficient and unique for evaluation (line 24). It remains to copy the corresponding values of the 9-intersection matrix into the thinned out matrix (lines 25 to 28).

$$1:\begin{pmatrix} 0|0 & 0|0 & 1|* \\ 0|* & 0|* & 0|* \\ 1|* & 1|* & 1|* \end{pmatrix} \quad 2:\begin{pmatrix} 0|0 & 1|* & 0|0 \\ 0|* & 0|* & 0|* \\ 1|* & 1|* & 1|* \end{pmatrix} \quad 3:\begin{pmatrix} 0|0 & 1|1 & 1|1 \\ 0|* & 0|* & 0|* \\ 1|* & 1|* & 1|* \end{pmatrix} \quad 4:\begin{pmatrix} 1|* & 0|0 & 0|0 \\ 0|* & 0|* & 0|* \\ 1|* & 1|* & 1|* \end{pmatrix} \quad 5:\begin{pmatrix} 1|1 & 0|0 & 1|1 \\ 0|* & 0|* & 0|* \\ 1|* & 1|* & 1|* \end{pmatrix}$$

$$6:\begin{pmatrix} 1|1 & 1|1 & 0|0 \\ 0|* & 0|* & 0|* \\ 1|* & 1|* & 1|* \end{pmatrix} \quad 7:\begin{pmatrix} 1|1 & 1|1 & 1|1 \\ 0|* & 0|* & 0|* \\ 1|* & 1|* & 1|* \end{pmatrix}$$

Figure 9: Complete and thinned out matrices for the 7 topological predicates of the *point2D*/*region2D* case.

```
01 algorithm MatrixThinning
02 input:  Three-dimensional 9IM im. im[i,l,m] ∈ {0,1}
03         denotes entry (l,m) (1 ≤ l,m ≤ 3) of the ith
04         9IM (1 ≤ i ≤ n_{α,β}).
05 output: Three-dimensional thinned out 9IM tim.
06         tim[i,l,m] ∈ {0,1,∗}. '∗' is 'don't care' symbol.
07 begin
08   Create three-dimensional matrix pos of 'position'
09     matrices where pos[j,l,m] ∈ {0,1} denotes entry
10     (l,m) of the jth possible 9IM (1 ≤ j ≤ 511);
11   Sort pos increasingly with respect to the number of
12     ones in a matrix;
13   Initialize all entries of matrices of tim with '∗'; r := 1;
14   // Compute thinned out matrices
15   for each i in 1...n_{α,β} do
16     j := 1; stop := false;
17     while j ≤ 511 and not stop do
18       k := 1; unequal := true;
19       while 1 ≤ k ≤ n_{α,β} and i ≠ k and unequal do
20         equal := im[i] and im[k] have the same values
21           at all positions (l,m) where pos[j,l,m] = 1;
22         unequal := unequal and not equal; inc(k);
23       endwhile;
24       if unequal then // Thin out im[i] by pos[j].
25         for each l,m in 1...3 do
26           if pos[j,l,m] = 1
27           then tim[r,l,m] := im[i,l,m] endif
28         endfor;
29         inc(r); stop := true;
30       else inc(j);
31       endif
32     endwhile
33   endfor
34 end MatrixThinning.
```

Figure 10: Algorithm for computing the thinned out versions of the $n_{\alpha,\beta}$ intersection matrices associated with the topological predicates between two spatial data types $\alpha$ and $\beta$

Note that for the same intersection matrix it may be possible to find several thinned out matrices with the same number of matrix predicates to be checked such that each of them represents the intersection matrix uniquely among the $n_{\alpha,\beta}$ intersection matrices. Our algorithm always computes the thinned out matrix with the "lowest numerical value". Figures 7 to 9 and 11 to 13 show the results for the different spatial data type combinations. Definition 7 defines the measures we use to summarize and interpret these results.

**Definition 7** Let $IM^{MT}$ be a thinned out 9IM, and *cnt* be a function that counts the number of relevant matrix predicates of $IM^{MT}$. Let $n_{\alpha,\beta}$ with $\alpha,\beta \in \{point2D, line2D, region2D\}$ be the number of (thinned out) 9IMs of the topological predicates between the types $\alpha$ and $\beta$, and $n^k_{\alpha,\beta}$ be the number of thinned out 9IMs for

$$1:\begin{pmatrix} 0|0 & 0|* & 1|* \\ 0|* & 0|0 & 1|* \\ 1|* & 1|* & 1|* \end{pmatrix} \quad 2:\begin{pmatrix} 0|0 & 0|* & 1|* \\ 0|* & 1|* & 0|0 \\ 1|* & 1|* & 1|* \end{pmatrix} \quad 3:\begin{pmatrix} 0|0 & 0|* & 1|* \\ 0|* & 1|* & 1|* \\ 1|* & 0|0 & 1|* \end{pmatrix} \quad 4:\begin{pmatrix} 0|0 & 0|* & 1|* \\ 0|* & 1|1 & 1|1 \\ 1|* & 1|1 & 1|* \end{pmatrix} \quad 5:\begin{pmatrix} 1|* & 0|* & 0|0 \\ 0|* & 1|* & 0|* \\ 0|0 & 0|* & 1|* \end{pmatrix}$$

$$6:\begin{pmatrix} 1|* & 0|* & 0|0 \\ 0|0 & 1|* & 0|* \\ 1|* & 1|1 & 1|* \end{pmatrix} \quad 7:\begin{pmatrix} 1|* & 0|* & 0|0 \\ 1|* & 0|0 & 0|* \\ 1|* & 1|* & 1|* \end{pmatrix} \quad 8:\begin{pmatrix} 1|* & 0|* & 0|0 \\ 1|* & 1|* & 0|* \\ 1|1 & 0|0 & 1|* \end{pmatrix} \quad 9:\begin{pmatrix} 1|* & 0|* & 0|0 \\ 1|1 & 1|1 & 0|* \\ 1|* & 1|1 & 1|* \end{pmatrix} \quad 10:\begin{pmatrix} 1|1 & 0|0 & 1|1 \\ 0|0 & 1|* & 0|0 \\ 1|* & 1|* & 1|* \end{pmatrix}$$

$$11:\begin{pmatrix} 1|* & 0|0 & 1|* \\ 0|* & 1|* & 1|1 \\ 0|0 & 0|* & 1|* \end{pmatrix} \quad 12:\begin{pmatrix} 1|1 & 0|0 & 1|* \\ 0|0 & 1|* & 1|* \\ 1|1 & 0|0 & 1|* \end{pmatrix} \quad 13:\begin{pmatrix} 1|1 & 0|0 & 1|* \\ 0|0 & 1|* & 1|1 \\ 1|* & 1|1 & 1|* \end{pmatrix} \quad 14:\begin{pmatrix} 1|* & 0|0 & 1|* \\ 1|1 & 0|0 & 1|1 \\ 1|* & 1|* & 1|* \end{pmatrix} \quad 15:\begin{pmatrix} 1|* & 0|0 & 1|1 \\ 1|* & 1|* & 0|0 \\ 1|* & 0|0 & 1|* \end{pmatrix}$$

$$16:\begin{pmatrix} 1|* & 0|0 & 1|1 \\ 1|1 & 1|* & 0|0 \\ 1|* & 1|1 & 1|* \end{pmatrix} \quad 17:\begin{pmatrix} 1|* & 0|0 & 1|* \\ 1|1 & 1|* & 1|1 \\ 1|* & 0|0 & 1|* \end{pmatrix} \quad 18:\begin{pmatrix} 1|* & 0|0 & 1|* \\ 1|1 & 1|1 & 1|1 \\ 1|* & 1|1 & 1|* \end{pmatrix} \quad 19:\begin{pmatrix} 1|* & 1|* & 1|* \\ 0|* & 0|0 & 1|* \\ 0|0 & 0|* & 1|* \end{pmatrix} \quad 20:\begin{pmatrix} 1|* & 1|1 & 1|* \\ 0|0 & 0|0 & 1|* \\ 1|* & 1|1 & 1|* \end{pmatrix}$$

$$21:\begin{pmatrix} 1|* & 1|* & 1|1 \\ 0|* & 1|* & 0|0 \\ 0|0 & 0|* & 1|* \end{pmatrix} \quad 22:\begin{pmatrix} 1|* & 1|* & 1|* \\ 0|0 & 1|* & 0|0 \\ 1|1 & 0|0 & 1|* \end{pmatrix} \quad 23:\begin{pmatrix} 1|* & 1|1 & 1|* \\ 0|0 & 1|* & 0|0 \\ 1|* & 1|1 & 1|* \end{pmatrix} \quad 24:\begin{pmatrix} 1|* & 1|1 & 1|* \\ 0|* & 1|1 & 1|1 \\ 0|0 & 0|* & 1|* \end{pmatrix} \quad 25:\begin{pmatrix} 1|* & 1|1 & 1|* \\ 0|0 & 1|* & 1|1 \\ 1|1 & 0|0 & 1|* \end{pmatrix}$$

$$26:\begin{pmatrix} 1|* & 1|1 & 1|* \\ 0|0 & 1|1 & 1|1 \\ 1|* & 1|1 & 1|* \end{pmatrix} \quad 27:\begin{pmatrix} 1|* & 1|* & 1|1 \\ 1|* & 0|0 & 0|0 \\ 1|* & 1|* & 1|* \end{pmatrix} \quad 28:\begin{pmatrix} 1|* & 1|* & 1|* \\ 1|* & 0|0 & 1|* \\ 1|1 & 0|0 & 1|* \end{pmatrix} \quad 29:\begin{pmatrix} 1|* & 1|1 & 1|* \\ 1|1 & 0|0 & 1|1 \\ 1|* & 1|1 & 1|* \end{pmatrix} \quad 30:\begin{pmatrix} 1|* & 1|1 & 1|* \\ 1|1 & 1|* & 0|0 \\ 1|* & 0|0 & 1|* \end{pmatrix}$$

$$31:\begin{pmatrix} 1|* & 1|1 & 1|* \\ 1|1 & 1|1 & 0|0 \\ 1|* & 1|1 & 1|* \end{pmatrix} \quad 32:\begin{pmatrix} 1|* & 1|1 & 1|* \\ 1|1 & 1|1 & 1|1 \\ 1|* & 0|0 & 1|* \end{pmatrix} \quad 33:\begin{pmatrix} 1|* & 1|1 & 1|* \\ 1|1 & 1|1 & 1|1 \\ 1|* & 1|1 & 1|* \end{pmatrix}$$

Figure 11: Complete and thinned out matrices for the 33 topological predicates of the *region2D*/*region2D* case.

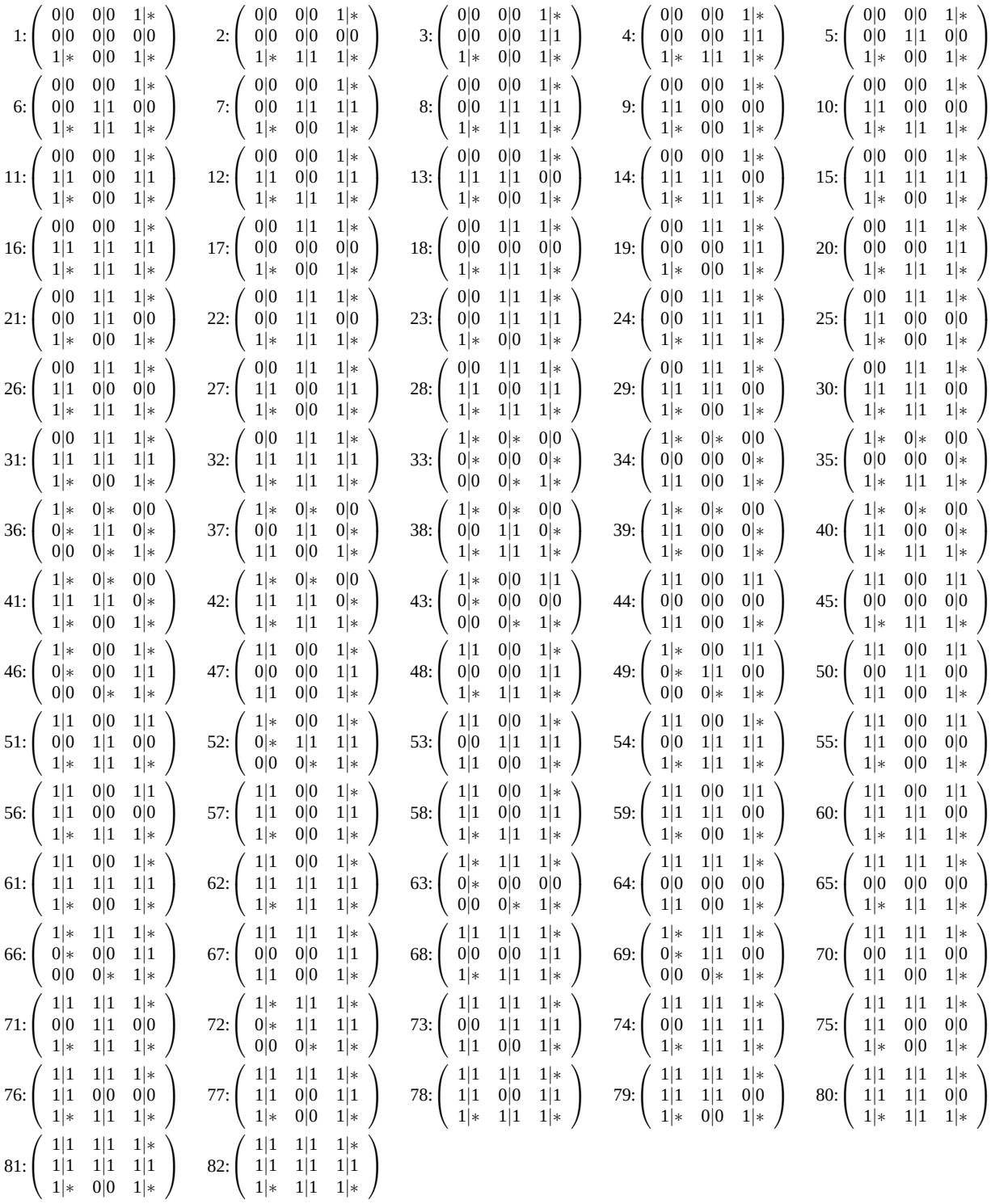



Figure 12: Complete and thinned out matrices for the 82 topological predicates of the *line2D*/*line2D* case.

which $k$ (with $1 \leq k \leq 9$) matrix predicates have to be evaluated. Let the cost, i.e., the total number of matrix predicates to be evaluated for $\alpha$ and $\beta$, be $C_{\alpha,\beta}$ without matrix thinning and $C^{MT}_{\alpha,\beta}$ with matrix thinning. We then denote with $RAC^{MT}_{\alpha,\beta}$ the reduced average cost in percent when using matrix thinning. We obtain:

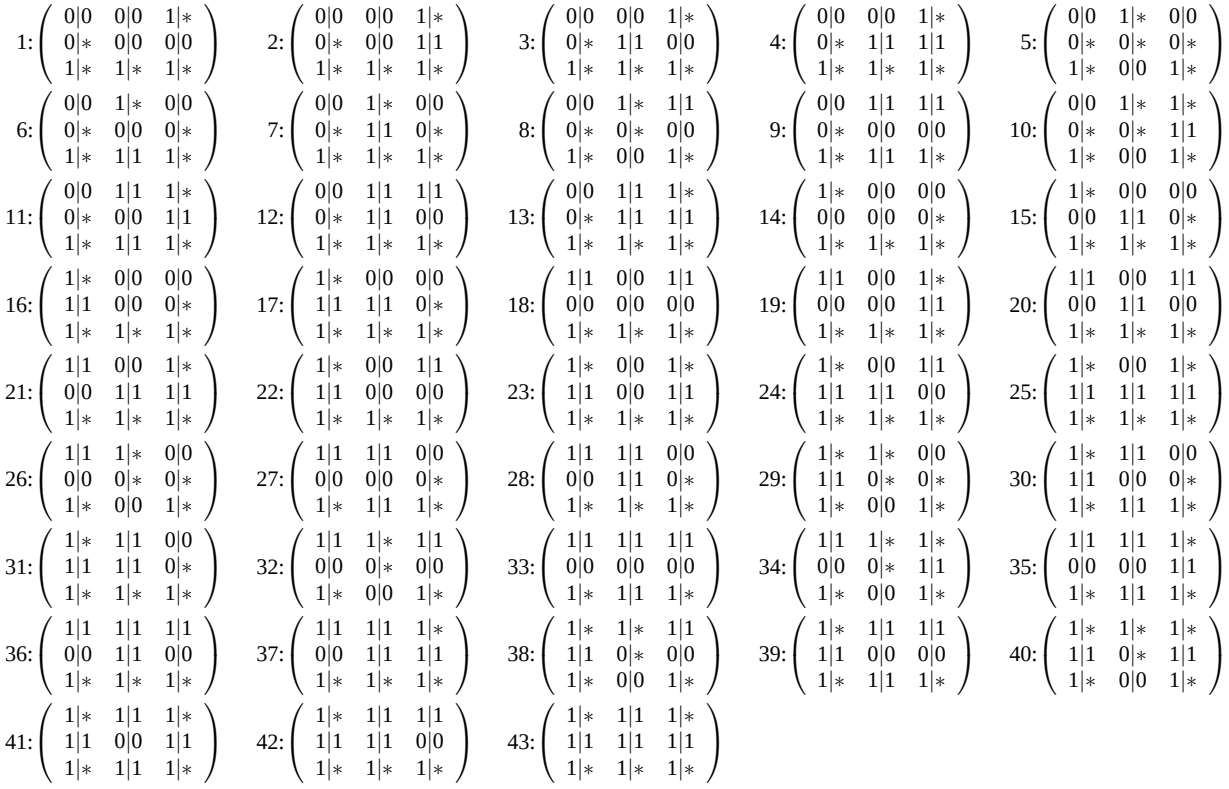



Figure 13: Complete and thinned out matrices for the 43 topological predicates of the *line2D*/*region* case.

$$
\begin{array}{rrcl}
\text{(i)} & cnt(IM^{MT}) & = & |\{(l,m)\,|\,1 \le l,m \le 3, IM^{MT}[l,m] \in \{0,1\}\}| \\
\text{(ii)} & n^k_{\alpha,\beta} & = & |\{IM^{MT}_i\,|\,1 \le i \le n_{\alpha,\beta}, 1 \le k \le 9, cnt(IM^{MT}_i) = k\}| \\
\text{(iii)} & n_{\alpha,\beta} & = & \sum_{k=1}^{9} n^k_{\alpha,\beta} \\
\text{(iv)} & C_{\alpha,\beta} & = & 8 \cdot n_{\alpha,\beta} \\
\text{(v)} & AC_{\alpha,\beta} & = & C_{\alpha,\beta}/n_{\alpha,\beta} = 8 \\
\text{(vi)} & C^{MT}_{\alpha,\beta} & = & \sum_{k=1}^{9} k \cdot n^k_{\alpha,\beta} \\
\text{(vii)} & AC^{MT}_{\alpha,\beta} & = & C^{MT}_{\alpha,\beta}/n_{\alpha,\beta} \\
\text{(viii)} & RAC^{MT}_{\alpha,\beta} & = & 100 \cdot AC^{MT}_{\alpha,\beta}/AC_{\alpha,\beta} = 100 \cdot C^{MT}_{\alpha,\beta}/C_{\alpha,\beta}
\end{array}
$$

$AC^{MT}_{\alpha,\beta}$ denotes the average number of matrix predicates to be evaluated. Table 2 shows a summary of the results and in the last two columns the considerable performance enhancement of matrix thinning. The reduction of matrix predicate computations ranges from 27 percent for the *line2D* / *line2D* case to 75 percent for the *point2D* / *point2D* case.

## 5.2 The Minimum Cost Decision Tree for Predicate Determination

In Section 4, we have seen that, in the worst case, $n_{\alpha,\beta}$ matching tests are needed to determine the topological relationship between any two spatial objects. For each test, Boolean expressions have to be evaluated that are equivalent to the eight matrix predicates and based on topological feature vectors. We propose two methods to improve the performance. The first method reduces the number of matrix predicates to be evaluated. This goal can be directly achieved by applying the method of matrix thinning described in Section 5.1. That is, the number $n_{\alpha,\beta}$ of tests remains the same but for each test we can reduce the number of matrix predicates that have to be evaluated by taking the thinned out instead of the complete 9-intersection matrices.

| Type combination | $n_{\alpha,\beta}$ | $n^k_{\alpha,\beta}$ with $k=$ | | | | | | | | | $C_{\alpha,\beta}$ | $AC_{\alpha,\beta}$ | $C^{MT}_{\alpha,\beta}$ | $AC^{MT}_{\alpha,\beta}$ | $RAC^{MT}_{\alpha,\beta}$ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | | | | | |
| *point2D* / *point2D* | 5 | 1 | 3 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 40 | 8 | 10 | 2.00 | 25.00 |
| *line2D* / *line2D* | 82 | 0 | 0 | 2 | 12 | 4 | 50 | 12 | 2 | 0 | 656 | 8 | 474 | 5.78 | 72.26 |
| *region2D* / *region2D* | 33 | 0 | 6 | 6 | 10 | 11 | 0 | 0 | 0 | 0 | 264 | 8 | 125 | 3.79 | 47.35 |
| *point2D* / *line2D* | 14 | 0 | 0 | 6 | 8 | 0 | 0 | 0 | 0 | 0 | 112 | 8 | 50 | 3.57 | 44.64 |
| *point2D* / *region2D* | 7 | 0 | 3 | 4 | 0 | 0 | 0 | 0 | 0 | 0 | 56 | 8 | 18 | 2.57 | 32.14 |
| *line2D* / *region2D* | 43 | 0 | 0 | 5 | 18 | 12 | 7 | 1 | 0 | 0 | 344 | 8 | 196 | 4.56 | 56.98 |

Table 2: Summary of complete and thinned out 9IMs for the topological predicates of all type combinations.

The second method, which will be our focus in this subsection, aims at reducing the number $n_{\alpha,\beta}$ of tests. This method is based on the complete 9-intersection matrices (Figures 7 to 9 and 11 to 13) but also manages to reduce the number of matrix predicates that have to be evaluated. We propose a *global* concept called *minimum cost decision tree* (*MCDT*) for this purpose. The term "global" means that we do not look at each intersection matrix individually but consider all $n_{\alpha,\beta}$ intersection matrices together. The idea is to construct a *full binary decision tree* whose inner nodes represent all matrix predicates, whose edges represent the Boolean values *true* or *false*, and whose leaf nodes are the $n_{\alpha,\beta}$ topological predicates. Note that, in a full binary tree, each node has exactly zero or two children. For searching, we employ a depth-first search procedure that starts at the root of the tree and proceeds down to one of the leaves which represents the matching topological predicate. The performance gain through the use of a decision tree is significant since the tree partitions the search space at each node and gradually excludes more and more topological predicates. In the best case, at each node of the decision tree, the search space, which comprises the remaining topological predicates to be assigned to the remaining leaves of the node's subtree, is partitioned into two halves so that we obtain a perfectly balanced tree. This would guarantee a search time of $O(\log n_{\alpha,\beta})$. But in general, we cannot expect to obtain a bisection of topological predicates at each node since the number of topological predicates yielding *true* for the node's matrix predicate will be different from the number of topological predicates yielding *false* for that matrix predicate. An upper bound is the number 8, since at most eight matrix predicates have to be checked to identify a topological predicate uniquely; the nineth matrix predicate yields always *true*. Hence, our goal is to determine a nearly balanced, cost-optimal, full binary decision tree for each collection of $n_{\alpha,\beta}$ intersection matrices.

If we do not have specific knowledge about the probability distribution of topological predicates in an application (area), we can only assume that they occur with equal distribution. But sometimes we have more detailed information. For example, in cadastral map applications, an adequate estimate is that 95 percent of all topological relationships between regions are *disjoint* and the remaining 5 percent are *meet*. Our algorithm for constructing MCDTs considers these frequency distributions. It is based on the following cost model:

**Definition 8** Let $M_{\alpha,\beta}$ be an MCDT for the spatial data types $\alpha,\beta \in \{point2D, line2D, region2D\}$, $w_i$ be the weight of the topological predicate $p_i$ with $1 \le i \le n_{\alpha,\beta}$ and $0 < w_i < 1$, and $d_i$ with $1 \le d_i \le 8$ be the depth of a node in $M_{\alpha,\beta}$ at which $p_i$ is identified. We define the total cost $C^{MCDT}_{\alpha,\beta}$ of $M_{\alpha,\beta}$ as

$$C^{MCDT}_{\alpha,\beta} = \sum_{i=1}^{n_{\alpha,\beta}} w_i \cdot d_i \qquad \text{with} \sum_{i=1}^{n_{\alpha,\beta}} w_i = 1$$

That is, our cost model is to sum up all the weighted path lengths from each leaf node representing a topological predicate to the root of the MCDT. If all topological predicates occur with equal probability, we set $w_i = \frac{1}{n_{\alpha,\beta}}$. The issue now is how to find and build an optimal MCDT with minimal total cost $C^{MCDT}_{\alpha,\beta}$ on the basis of a given probability distribution (weighting) for the topological predicates. If all topological

```
01 algorithm MCDT
02 input:   list im = ⟨(im_1, w_1), ..., (im_{n_{α,β}}, w_{n_{α,β}})⟩ of 9IMs
03          with weights, list mp of the eight matrix predicates
04 output:  MCDT M_{α,β}
05 begin
06    best_node := new_node(); stop := false;
07    discriminator := select_first(mp);
08    while not eol(mp) and not stop do
09       node := new_node();
10       node.discr := discriminator; node.im := im;
11       if no_of_elem(im) = 1 then /* leaf node */
12          best_node := node; best_node.cost := 0;
13          stop := true;
14       else
15          /* Let im = ⟨(im_{k_1}, w_{k_1}), ..., (im_{k_n}, w_{k_n})⟩
16             with 1 ≤ k_1 ≤ ... ≤ k_n ≤ n_{α,β}. */
17          partition(im, discriminator, im_l, im_r);
18          if no_of_elem(im_l) ≠ 0 and
19             no_of_elem(im_r) ≠ 0 then
20             copy(mp, new_mp); del(new_mp, discriminator);
21             node.lchild := MCDT(im_l, new_mp);
22             node.rchild := MCDT(im_r, new_mp);
23             node.cost := node.lchild.cost + node.rchild.cost
24                          + Σ_{i=k_1}^{k_n} w_i;
25             if node.cost < best_node.cost
26             then best_node := node; endif;
27          endif;
28          discriminator := select_next(mp);
29       endif
30    endwhile;
31    return best_node;
32 end MCDT.
```

Figure 14: Minimum cost decision tree algorithm

predicates occur with equal probability, this problem corresponds to finding an optimal MCDT that requires the minimal number of matrix predicate evaluations to arrive at an answer.

Figure 14 shows our recursive algorithm *MCDT* for computing a minimum cost decision tree for a set *im* of $n_{\alpha,\beta}$ 9-intersection matrices that are annotated with a weight representing the corresponding predicates's probability of occurrence, as it is characteristic in a particular application (line 2). Later these matrices become the leaves of the decision tree. In addition, the algorithm takes as an input the list *mp* of eight matrix predicates (we remember that the exterior/exterior intersection yields always *true*) that serve as discriminators and are attached to the inner nodes (line 3). This list is necessary to ensure that a matrix predicate is not used more than once as a discriminator in a search path. During the recursion, the while-loop (lines 8 to 30) terminates if either the list *mp* of matrix predicates to be processed is empty or the list *im* of 9-intersection matrices contains only a single element. For each matrix predicate used as a discriminator, the operation *new_node* creates a new tree node *node* (line 9). The matrix predicate *discriminator* as well as the list *im* annotate the tree node *node* (line 10). If *im* has only one element (line 11), we know that *node* is a leaf node representing the topological predicate pertaining to the single element in *im*. The cost for this leaf node is 0 since its current depth is 0 (line 12). Otherwise, if *im* consists of more than one element, we partition it into two lists $im_l$ and $im_r$ (line 17). The partitioning is based on the values of each 9-intersection matrix in *im* with respect to the matrix predicate serving as the discriminator. If such a value is 0 (*false*), the corresponding 9-intersection matrix is added to the list $im_l$; otherwise, it is added to the list $im_r$. A special case now is that *im* has not been partitioned so that either $im_l$ or $im_r$ is empty (condition in lines 18 to 19 yields *false*). In this case, the discriminator does not contribute to a decision and is skipped; the next discriminator is selected (line 28). If both lists $im_l$ and $im_r$ are nonempty (lines 18 to 19), we remove the discriminator from a new copy *new_mp* of the list *mp* (line 20) and recursively find the minimum cost decision trees for the 9-intersection matrices in $im_l$ (line 21) and in $im_r$ (line 22). Eventually, all recursions will reach all leaf nodes and begin returning while recursively calculating the cost of each subtree found. The cost of a leaf node is 0. The cost of an inner node *node* can be expressed in terms of the cost of its two nonempty subtrees *node.lchild* and *node.rchild* processing the lists $im_l$ and $im_r$ respectively. The depth of each leaf node with respect to *node* is exactly one larger than the depth of the same leaf node with respect to either *node.lchild* or *node.rchild*. Therefore, besides the costs of these two subtrees, for each leaf node of the subtree with root *node*, we have to add the leaf node's cost (weight) one time (lines 23 to 24). These weights are stored in *node.im*. The cost of *node* is then compared with the best cost determined so far, and the minimum will be the new best option (lines 25 to 26). Eventually, when all the matrix predicates have been considered, we

| Type combination | MCDT preorder representation |
|---|---|
| *point2D* / *point2D* | $^{\circ-}$ $^{-\circ}$ 2 3 $^{\circ\circ}$ 1 $^{-\circ}$ 4 5 |
| *line2D* / *line2D* | $^{\circ}\partial$ $\partial^{\circ}$ $\partial\partial$ $^{\circ-}$ $^{-\circ}$ $^{-\circ}$ 33 $^{-}\partial$ 34 35 $^{\circ\circ}$ $^{-}\partial$ 1 2 $^{-\circ}$ 43 $^{-}\partial$ 44 45 $^{\circ\circ}$<br>$^{-}\partial$ 3 4 $^{-\circ}$ 46 $^{-}\partial$ 47 48 $^{\circ-}$ $^{\circ-}$ $^{-\circ}$ 36 $^{-}\partial$ 37 38 $^{\circ\circ}$ $^{-}\partial$ 5 6 $^{-\circ}$<br>49 $^{-}\partial$ 50 51 $^{\circ\circ}$ $^{-}\partial$ 7 8 $^{-\circ}$ 52 $^{-}\partial$ 53 54 $^{\circ\circ}$ $\partial\partial$ $^{\circ-}$ $^{-}\partial$ 9 10 $^{-}\partial$<br>11 12 $^{\circ-}$ $^{-}\partial$ 13 14 $^{-}\partial$ 15 16 $^{\circ-}$ $\partial\partial$ $^{-}\partial$ 39 40 $^{-}\partial$ 41 42 $\partial\partial$ $^{\circ-}$ $^{-}\partial$<br>55 56 $^{-}\partial$ 57 58 $^{\circ-}$ $^{-}\partial$ 59 60 $^{-}\partial$ 61 62 $^{\circ\circ}$ $\partial^{\circ}$ $\partial\partial$ $^{\circ-}$ $^{-}\partial$ 17 18 $^{-}\partial$<br>19 20 $^{\circ-}$ $^{-}\partial$ 21 22 $^{-}\partial$ 23 24 $\partial\partial$ $^{\circ-}$ $^{-}\partial$ 25 26 $^{-}\partial$ 27 28 $^{\circ-}$ $^{-}\partial$ 29<br>30 $^{-}\partial$ 31 32 $\partial^{\circ}$ $\partial\partial$ $^{\circ-}$ $^{-\circ}$ 63 $^{-}\partial$ 64 65 $^{-\circ}$ 66 $^{-}\partial$ 67 68 $^{\circ-}$ $^{-\circ}$ 69<br>$^{-}\partial$ 70 71 $^{-\circ}$ 72 $^{-}\partial$ 73 74 $\partial\partial$ $^{\circ-}$ $^{-}\partial$ 75 76 $^{-}\partial$ 77 78 $^{\circ-}$ $^{-}\partial$ 79 80<br>$^{-}\partial$ 81 82 |
| *region2D* / *region2D* | $^{\circ}\partial$ $\partial^{\circ}$ $^{\circ-}$ $^{\circ-}$ $^{-\circ}$ 5 6 $^{\circ\circ}$ 2 10 $^{\circ\circ}$ $\partial\partial$ 1 $^{-}\partial$ 3 4 $^{-\circ}$ 11 $^{-}\partial$ 12<br>13 $^{\circ-}$ $\partial\partial$ 7 $^{-}\partial$ 8 9 $^{\circ-}$ $^{-}\partial$ 15 16 $\partial\partial$ 14 $^{-}\partial$ 17 18 $\partial^{\circ}$ $^{\circ-}$ $^{-\circ}$ 21<br>$^{-}\partial$ 22 23 $\partial\partial$ $^{-\circ}$ 19 20 $^{-\circ}$ 24 $^{-}\partial$ 25 26 $\partial\partial$ $^{\circ-}$ 27 $^{-}\partial$ 28 29 $^{\circ-}$ $^{-}\partial$<br>30 31 $^{-}\partial$ 32 33 |
| *point2D* / *line2D* | $^{\circ\circ}$ $^{\circ}\partial$ $^{-}\partial$ 1 2 $^{\circ-}$ $^{-}\partial$ 3 4 $^{-}\partial$ 5 6 $^{\circ}\partial$ $^{\circ-}$ $^{-}\partial$ 7 8 $^{-}\partial$ 9 10<br>$^{\circ-}$ $^{-}\partial$ 11 12 $^{-}\partial$ 13 14 |
| *point2D* / *region2D* | $^{\circ\circ}$ $^{\circ}\partial$ 1 $^{\circ-}$ 2 3 $^{\circ}\partial$ $^{\circ-}$ 4 5 $^{\circ-}$ 6 7 |
| *line2D* / *region2D* | $\partial^{\circ}$ $^{\circ\circ}$ $\partial\partial$ $^{\circ-}$ $^{\circ-}$ $^{-}\partial$ 5 6 $^{\circ}\partial$ 1 $^{-}\partial$ 8 9 $^{\circ}\partial$ 2 $^{-}\partial$ 10 11 $^{\circ}\partial$ $^{\circ-}$<br>3 4 $^{\circ-}$ 7 $^{\circ-}$ 12 13 $^{\circ}\partial$ $^{\circ-}$ $\partial\partial$ 14 15 $\partial\partial$ $^{\circ-}$ 18 19 $^{\circ-}$ 20 21 $^{\circ-}$<br>$\partial\partial$ $^{-}\partial$ 26 27 28 $\partial\partial$ $^{\circ-}$ $^{-}\partial$ 32 33 $^{-}\partial$ 34 35 $^{\circ-}$ 36 37 $^{\circ}\partial$ $^{\circ-}$ $\partial\partial$ 16<br>17 $\partial\partial$ $^{\circ-}$ 22 23 $^{\circ-}$ 24 25 $^{\circ-}$ $\partial\partial$ $^{-}\partial$ 29 30 31 $\partial\partial$ $^{\circ-}$ $^{-}\partial$ 38 39 $^{-}\partial$<br>40 41 $^{\circ-}$ 42 43 |

Table 3: MCDT pre-order representations for all type combinations on the basis of equal probability of occurrence of all topological predicates.

obtain the best choice and return the corresponding minimum cost decision tree (line 31).

Table 3 shows the results of this algorithm by giving a textual pre-order (depth-first search) encoding of all MCDTs for all type combinations on the basis of equal probability of occurrence of all topological predicates. The encodings allow an easy construction of the MCDTs. Since MCDTs are full (or proper) binary trees, each node has exactly zero or two child nodes. We leverage this feature by representing an MCDT as the result of a pre-order tree traversal. The pre-order sequence of nodes is unique in this case for constructing the decision tree since inner nodes with only one child node cannot occur. Each inner node in the pre-order sequence is described as a term $XY$ where $X,Y \in \{^{\circ},\partial,^{-}\}$. Such a term represents a matrix predicate $AX \cap BY \neq \varnothing$ serving as a discriminator. For example, the term $XY =^{\circ} \partial$ denotes the matrix predicate $A^{\circ} \cap \partial B \neq \varnothing$ (prefix notation for boundary). Each leaf node represents the 9-intersection matrix number of a topological predicate. The matrix numbers are specified in the Figures 7 to 9 and 11 to 13.

Figures 15 shows a visualization of the MCDTs of three spatial data type combinations on the assumption that all topological predicates occur with equal probability. The MCDTs for the other type combinations have been omitted due to their very large size. Each inner node is annotated with a label $XY$ where $X \in \{A^{\circ},\partial A,A^{-}\}$ and $Y \in \{B^{\circ},\partial B,B^{-}\}$. A label represents a matrix predicate $X \cap Y \neq \varnothing$ serving as a discriminator. For example, the label $XY = A^{\circ}\partial B$ denotes the matrix predicate $A^{\circ} \cap \partial B \neq \varnothing$. If the evaluation of a matrix predicate yields *false*, we move to the left child; otherwise, we move to the right child. Each leaf node represents the 9-intersection matrix number of a topological predicate.

The following Definition 9 defines measures that we use to summarize and interpret these results. We are especially interested in the average number of matrix predicates to be evaluated.

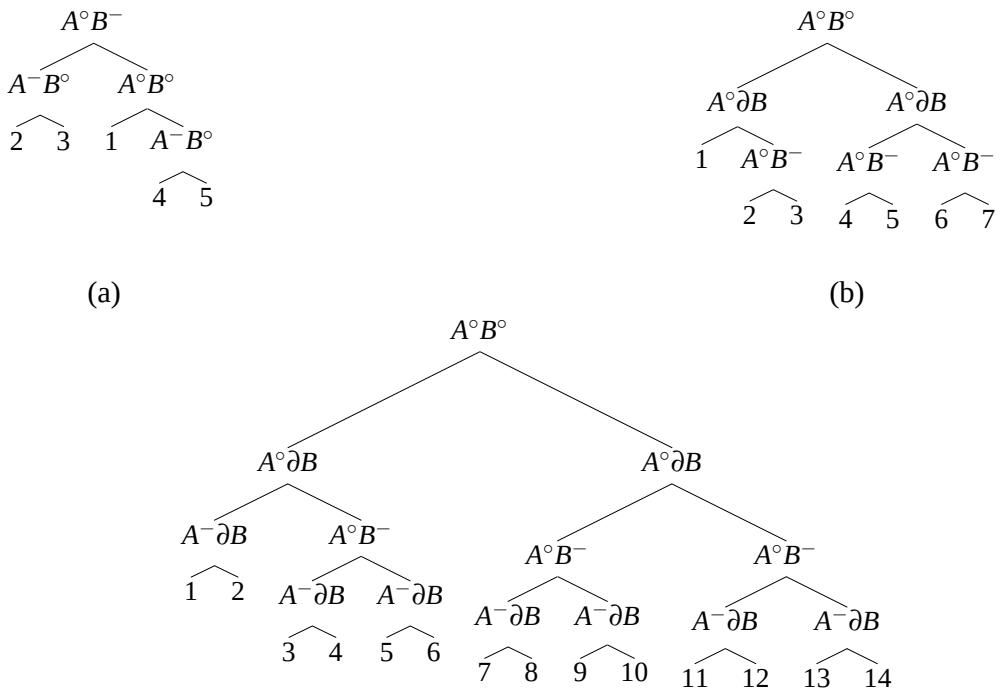



Figure 15: Minimum cost decision trees for the 5 topological predicates of the point/point case (a), the 7 topological predicates of the point/region case (b), and the 14 topological predicates of the point/line case (c) under the assumption that all topological predicates occur with equal probability.

**Definition 9** Let $C^{MCDT}_{\alpha,\beta}$ denote the total cost of an MCDT $M_{\alpha,\beta}$ according to Definition 8. Let $n_{\alpha,\beta}$ with $\alpha,\beta \in \{point2D, line2D, region2D\}$ be the number of 9IMs of the topological predicates between the types $\alpha$ and $\beta$, $IM_i$ with $1 \leq i \leq n_{\alpha,\beta}$ be a 9IM, and $d^k_{\alpha,\beta}$ be the number of topological predicates associated with leaf nodes in $M_{\alpha,\beta}$ of depth $k$ (with $1 \leq k \leq 9$). Further, let $C_{\alpha,\beta}$ be the cost without using an MCDT, $AC_{\alpha,\beta}$ be the average cost without using an MCDT, $AC^{MCDT}_{\alpha,\beta}$ be the average cost when using an MCDT, and $RAC^{MCDT}_{\alpha,\beta}$ be the reduced average cost in percent when using an MCDT. The measures are defined as:

$$
\begin{array}{lrcl}
\text{(i)} & d^k_{\alpha,\beta} & = & |\{IM_i \,|\, 1 \leq i \leq n_{\alpha,\beta}, 1 \leq k \leq 9, depth(IM_i, M_{\alpha,\beta}) = k\}| \\
\text{(ii)} & n_{\alpha,\beta} & = & \sum_{k=1}^{9} d^k_{\alpha,\beta} \\
\text{(iii)} & C_{\alpha,\beta} & = & 8 \cdot n_{\alpha,\beta} \\
\text{(iv)} & AC_{\alpha,\beta} & = & 4 \cdot (n_{\alpha,\beta} + 1) \\
\text{(v)} & AC^{MCDT}_{\alpha,\beta} & = & C^{MCDT}_{\alpha,\beta} / n_{\alpha,\beta} \\
\text{(vi)} & RAC^{MCDT}_{\alpha,\beta} & = & 100 \cdot AC^{MCDT}_{\alpha,\beta} / AC_{\alpha,\beta}
\end{array}
$$

To determine the average cost $AC_{\alpha,\beta}$ without using an MCDT in (iv), we observe that the best case is to check 8 matrix predicates, the second best case is to check 16 matrix predicates, etc., and the worst case is to check all $8 \cdot n_{\alpha,\beta}$ matrix predicates. The average number of matrix predicates that has to be checked without using an MCDT is therefore $8 \cdot (1 + 2 + \ldots + n_{\alpha,\beta}) / n_{\alpha,\beta} = 4 \cdot (n_{\alpha,\beta} + 1)$. $AC^{MCDT}_{\alpha,\beta}$ in (v) yields the average number of matrix predicates to be evaluated. Table 4 shows a summary of the results and in the last two columns the considerable performance enhancement of minimum cost decision trees. The reduction of matrix predicate computations ranges from 90 percent for the *point2D* / *point2D* case to 98 percent for the *line2D* / *line2D* case.

| Type combination | $n_{\alpha,\beta}$ | $d^k_{\alpha,\beta}$ with $k=$ | | | | | | | | | $C_{\alpha,\beta}$ | $AC_{\alpha,\beta}$ | $C^{MCDT}_{\alpha,\beta}$ | $AC^{MCDT}_{\alpha,\beta}$ | $RAC^{MCDT}_{\alpha,\beta}$ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | | | | | |
| *point2D* / *point2D* | 5 | 0 | 3 | 2 | 0 | 0 | 0 | 0 | 0 | 0 | 40 | 24 | 12 | 2.40 | 10.00 |
| *line2D* / *line2D* | 82 | 0 | 0 | 0 | 0 | 0 | 48 | 30 | 4 | 0 | 656 | 332 | 530 | 6.46 | 1.95 |
| *region2D* / *region2D* | 33 | 0 | 0 | 0 | 3 | 22 | 8 | 0 | 0 | 0 | 264 | 136 | 170 | 5.15 | 3.79 |
| *point2D* / *line2D* | 14 | 0 | 0 | 2 | 12 | 0 | 0 | 0 | 0 | 0 | 112 | 60 | 54 | 3.86 | 6.43 |
| *point2D* / *region2D* | 7 | 0 | 1 | 6 | 0 | 0 | 0 | 0 | 0 | 0 | 56 | 32 | 20 | 2.86 | 8.94 |
| *line2D* / *region2D* | 43 | 0 | 0 | 0 | 3 | 15 | 19 | 6 | 0 | 0 | 344 | 176 | 243 | 5.65 | 3.21 |

Table 4: Summary of the MCDTs for all type combinations on the basis of equal probability of occurrence of all topological predicates.

The MCDT approach is similar to a technique introduced in [9] for topological predicates between simple regions. However, their method of determining a binary decision tree rests on the thinned out 9-intersection matrices and results in a near optimal algorithm and solution. The reason why optimality is not achieved is that a topological predicate can have multiple, equipollent thinned out matrices, i.e., thinned out matrices are not unique. Therefore, using a specific set of thinned out matrices as the basis for partitioning the search space can only lead to an optimal decision tree for this set of thinned out matrices and may not be optimal in the general case. Our algorithm rests on the complete 9-intersection matrices. It produces an optimal decision tree (several optimal trees with the same total cost may exist) for the specified set of 9-intersection matrices and the given probability distribution. One can verify this by applying our algorithm to the eight 9-intersection matrices for two simple regions and the same probability distribution as specified in [9]. Our algorithm produces an optimal tree with the total cost of 2.13 while the so-called "refined cost method" in [9], which uses thinned out matrices, produces a tree with the total cost of 2.16.

We can observe the following relationship between MCDTs and thinned out matrices:

**Lemma 9** For each combination of spatial data types $\alpha$ and $\beta$, the total cost of its minimum cost decision tree (given in Table 4) is greater than or equal to the total cost of all its thinned out matrices (given in Table 2), i.e.,

$$C^{MCDT}_{\alpha,\beta} \geq C^{MT}_{\alpha,\beta}$$

*Proof.* The proof is given by contradiction. Assume that for a spatial data type combination the total cost of its MCDT is less than the total cost of all its thinned out matrices. Consequently, there must be at least one path from the root to a leaf in the MCDT that contains a smaller number of matrix predicates than the number of matrix predicates in the thinned out matrix for the topological predicate associated with that leaf. This implies that we can identify this topological predicate with a smaller number of matrix predicate decisions than the number of matrix predicates in its thinned out matrix. But this contradicts the definition of a thinned out matrix. □

# 6 Implementation, Testing, Approach Assessment, and Performance Study

Section 6.1 describes our implementation approach and environment for the concepts presented in this article. Section 6.2 delineates various tests based on this implementation in order to verify the correctness of the concepts. We perform an overall assessment of our approach from two perspectives. From a qualitative perspective, in Section 6.3, we briefly summarize the benefits of our design decision and compare our approach with an alternative *ad hoc* approach. From a quantitative perspective, in Section 6.4, we conduct a performance study and analyze the results.

## 6.1 Implementation

To verify the feasibility, practicality, and correctness of the concepts presented, we have implemented and tested our approach. The implementation of the algebra package SPAL2D for handling two-dimensional spatial data includes the implementation of all six exploration algorithms from [39], the general evaluation method of 9-intersection matrix characterization from Section 4 as well as the optimized evaluation techniques of matrix thinning and minimum cost decision trees from Section 5. The implementation makes use of the complex spatial data types *point2D*, *line2D*, and *region2D*, as they have been described in Sections 2.1 and 3.1 and specified in [39]. Since performance is one of the goals of this implementation, C++ is our employed programming language.

Two universal interface methods *TopPredVerification* and *TopPredDetermination* are provided to handle predicate verification and predicate determination queries respectively. Both interfaces are overloaded and take two spatial objects of any and possibly different type as input. The interface method *TopPredVerification* takes a predicate identification number as an additional input parameter. It corresponds to the matrix number (specified in [38] and used in Figures 7 to 9 and 11 to 13) of the topological predicate to be evaluated. The output is the Boolean value *true* if the topological relationship between the two spatial objects corresponds to the specified predicate; otherwise, the value is *false*. The interface method *TopPredDetermination* outputs the matrix number of the topological predicate corresponding to the topological relationship between the two spatial objects. Each of these interface methods involves three consecutive execution steps: (i) the execution of an appropriate exploration algorithm which produces two topological feature vectors (this functionality is provided by the interface method *TopPredExploration* from [39]), (ii) the computation of the 9-intersection characterization for the respective spatial data type combination, and (iii) the execution of either the predicate verification using thinned out matrices or the predicate determination using a minimum cost decision tree.

## 6.2 Testing

The testing of the concepts presented in this article is an extension of the testing environment and procedure in [39] checking the functionality of the exploration algorithms and the correctness of the resulting values of the topological feature vectors. We take the topological feature vectors as input for the 9-intersection matrix characterization method and the optimization methods of matrix thinning and minimum cost decision trees. The correctness of all methods has been checked by a technique known as *gray-box testing* (see [39] for more details), which combines the advantages of two other techniques called *black-box testing* and *white-box testing*. The black-box testing technique arranges for well defined input and output objects. In our case, the input consists of two correct topological feature vectors as well as a matrix number of the topological predicate to be verified in case of predicate verification. This enables us to test the functional behavior of the three method implementations. The output is guaranteed to be either a Boolean value (predicate verification) or a valid matrix number of a topological relationship predefined for the type combination under consideration (predicate determination). The white-box testing technique considers every single execution path and guarantees that each statement is executed at least once. This ensures that all cases that are specified and handled by the algorithms are properly tested. All cases have been successfully tested and indicate the correctness of our concepts and the ability of our algorithms to correctly verify or determine a topological predicate.

## 6.3 Qualitative Assessment

Although it is usually accepted that some kind of plane-sweep algorithm is sufficient for implementing topological predicates, the article in [39] and this article have demonstrated that the decision on an appropriate

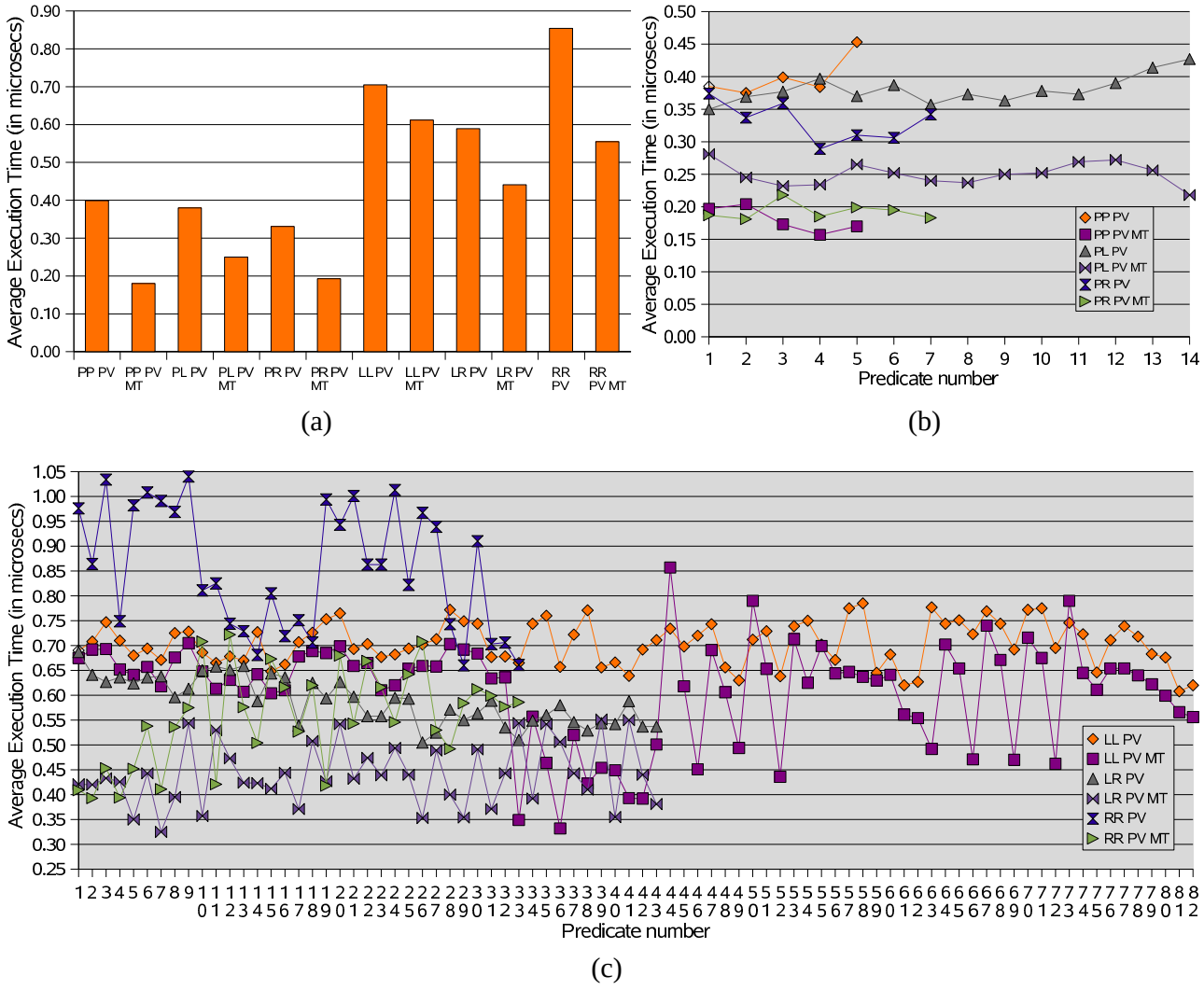



Figure 16: Predicate verification without and with matrix thinning

and sophisticated implementation strategy is of crucial importance. The possible *ad hoc* approach of implementing a separate algorithm for each of the topological predicates results in a large number of algorithms possibly up to the total number of topological predicates of a type combination. Even though this approach is relatively straightforward, it suffers from many problems including large system implementation, non-guaranteed correctness of the algorithms, error-proneness, redundancy, testing and evaluation difficulties, and performance degradation. An essential problem of the *ad hoc* approach is the difficulty in handling predicate determination queries. No particular algorithm is suitable for this task, thus requiring a linear iteration through the large number of algorithms for all topological predicates.

Unlike the *ad hoc* approach, our approach does not suffer from these problems. In our implementation, only a single plane-sweep algorithm is employed for all exploration algorithms. Only a single exploration algorithm is implemented for all topological predicates of each type combination. This implementation strategy allows us to take advantage of significantly smaller system implementation, widespread code reusability and sharing, manageable system testing, and efficient handling of both predicate determination and verification queries. This centralized approach is possible because, instead of considering each topological predicate individually, we look deeper into their common definition blocks which are the nine matrix predicates of the 9-intersection matrix. This leads us to a systematic method. By creating a bidirectional link between the ma-

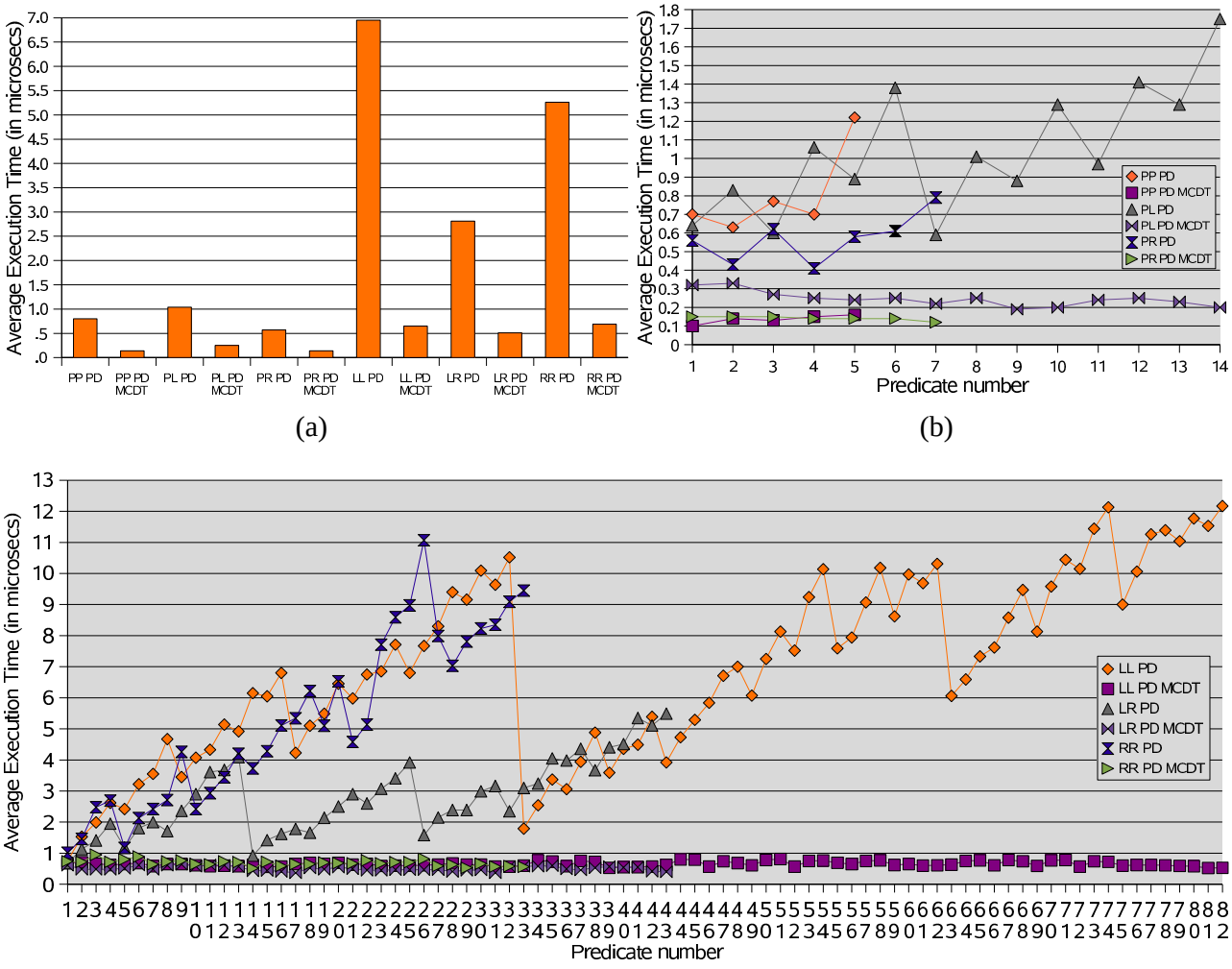



Figure 17: Predicate determination without and with MCDT

trix predicates and topological feature vectors, we are able to give a unique characterization for each matrix predicate. This unique characterization frees us from providing algorithms for each topological predicate in case of predicate verification and from evaluating all topological predicates in case of predicate determination. Furthermore, the correctness of the method is formally proven. Last but not least, based on the concept of topological feature vectors, predicate matching techniques such as matrix thinning and minimum cost decision trees can be used to increase the efficiency of answering predicate verification and predicate determination queries respectively.

## 6.4 Performance Study and Analysis

We have performed a performance study that underpins the strengths of our approach by quantitatively comparing the performance of our non-optimized alternative (only 9IMC) with our optimized evaluation techniques (9IMC plus matrix thinning, 9IMC plus minimum cost decision tree). Our study shows that our approach does not only provide qualitative but also quantitative benefits by applying optimization methods for the evaluation of topological predicates. For each type combination, we measure and calculate the average execution time for verifying and determining each predicate both without and with optimization.

For predicate verification (PV), Figures 16(b) and 16(c) illustrate the average execution time for each predicate of each type combination without and with matrix thinning (MT). The overall average for each type combination is shown in Figure 16(a). The performance improvements from using matrix thinning are quite noticeable and range from 13 percent execution time reduction for the *line2D*/*line2D* case up to 55 percent for the *point2D*/*point2D* case.

Similarly, for predicate determination (PD), Figures 17(b) and 17(c) show the average execution time for each predicate of each type combination without and with the use of minimum cost decision trees. The overall average for each type combination is shown in Figure 17(a). The results indicate significant performance improvements from using minimum cost decision trees. The improvements range from 75 percent execution time reduction for the the *point2D*/*region2D* case up to 91 percent for the *line2D*/*line2D* case.

Although the execution time reductions are remarkable for both predicate verification and especially predicate determination and clearly reflect the trend, the empirical results shown in Figures 16 and 17 are not as optimistic as the computational results given in the Tables 2 and 4. The reason that we cannot reach these lower bounds in practice consists in programming and runtime overheads such as extra conditional checks, construction of thinned out matrices and minimum cost decision trees, and their traversals. Even with these overheads, it is evident that our approach permits and offers considerable performance optimizations with respect to predicate evaluations.

# 7 Conclusions

Conceptual models of topological predicates have been investigated in a large number of publications. However, their efficient implementation, which is of essential importance for the processing of queries that contain spatial joins and spatial selections, has been widely neglected. Especially, the introduction of *complex* spatial data types and the resulting increase of topological predicates between all their combinations require efficient and correct implementation concepts. We propose a two-phase approach that consists of an *exploration phase* and an *evaluation phase* and that can be applied to both *predicate verification* and *predicate determination*. The goal of the exploration phase is to traverse a given scene of two objects in space, collect any topological information of importance, and store it in *topological feature vectors*. The goal of the evaluation phase is to interpret the gained topological information and match it against the topological predicates.

In this article, we have focused on the evaluation phase and in detail developed a general evaluation method called 9-*intersection matrix characterization*. This method generates a formally proven connection between theoretical concepts (9-intersection matrix, matrix predicate) and implementation concepts (topological feature vector, topological flag, segment class). Two sophisticated optimization techniques, called *matrix thinning* and *minimum cost decision tree*, are introduced which help to speed up predicate determination and predicate verification respectively. Besides a quantitative analysis, we have also provided an experimental performance study which documents both the correctness, robustness, and efficiency of our approach. Our approach has been implemented in the SPAL2D software library which is currently under development and determined for an integration into extensible databases.

# References


[1] T. Behr and M. Schneider. Topological Relationships of Complex Points and Complex Regions. *Int. Conf. on Conceptual Modeling*, pp. 56–69, 2001.

[2] A. Brodsky and X. S. Wang. On Approximation-based Query Evaluation, Expensive Predicates and Constraint Objects. *Int. Workshop on Constraints, Databases, and Logic Programming*, 1995.

[3] J. Claussen, A. Kemper, G. Moerkotte, K. Peithner, and M. Steinbrunn. Optimization and Evaluation of Disjunctive Queries. *IEEE Trans. on Knowledge and Data Engineering*, 12(2):238–260, 2000.

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

[5] 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.

[6] E. Clementini and P. Di Felice. Topological Invariants for Lines. *IEEE Trans. on Knowledge and Data Engineering*, 10(1), 1998.

[7] E. Clementini, P. Di Felice, and G. Califano. Composite Regions in Topological Queries. *Information Systems*, 20(7):579–594, 1995.

[8] 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.

[9] E. Clementini, J. Sharma, and M.J. Egenhofer. Modeling Topological Spatial Relations: Strategies for Query Processing. *Computers and Graphics*, 18(6):815–822, 1994.

[10] 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.

[11] J.R. Davis. IBM's DB2 Spatial Extender: Managing Geo-Spatial Information within the DBMS. Technical report, IBM Corporation, 1998.

[12] M. J. Egenhofer. A Formal Definition of Binary Topological Relationships. *3rd Int. Conf. on Foundations of Data Organization and Algorithms*, LNCS 367, pp. 457–472. Springer-Verlag, 1989.

[13] M. J. Egenhofer. Definitions of Line-Line Relations for Geographic Databases. *16th Int. Conf. on Data Engineering*, pp. 40–46, 1993.

[14] M. J. Egenhofer. Spatial SQL: A Query and Presentation Language. *IEEE Trans. on Knowledge and Data Engineering*, 6(1):86–94, 1994.

[15] M. J. Egenhofer and R. D. Franzosa. Point-Set Topological Spatial Relations. *Int. Journal of Geographical Information Systems*, 5(2):161–174, 1991.

[16] M. J. Egenhofer and J. Herring. A Mathematical Framework for the Definition of Topological Relationships. *4th Int. Symp. on Spatial Data Handling*, pp. 803–813, 1990.

[17] 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.

[18] M. J. Egenhofer and D. Mark. Modeling Conceptual Neighborhoods of Topological Line-Region Relations. *Int. Journal of Geographical Information Systems*, 9(5):555–565, 1995.

[19] M.J. Egenhofer. Deriving the Composition of Binary Topological Relations. *Journal of Visual Languages and Computing*, 2(2):133–149, 1994.

[20] M.J. Egenhofer, E. Clementini, and P. Di Felice. Topological Relations between Regions with Holes. *Int. Journal of Geographical Information Systems*, 8(2):128–142, 1994.

[21] ESRI Spatial Database Engine (SDE). Environmental Systems Research Institute, Inc., 1995.

[22] S. Gaal. *Point Set Topology*. Academic Press, 1964.

[23] V. Gaede and O. Günther. Multidimensional Access Methods. *ACM Computing Surveys*, 30(2):170–231, 1998.

[24] R. H. Güting and M. Schneider. Realm-Based Spatial Data Types: The ROSE Algebra. *VLDB Journal*, 4:100–143, 1995.

[25] R.H. Güting, T. de Ridder, and M. Schneider. Implementation of the ROSE Algebra: Efficient Algorithms for Realm-Based Spatial Data Types. *Int. Symp. on Advances in Spatial Databases*, 1995.

[26] J. M. Hellerstein. Practical Predicate Placement. *ACM SIGMOD Int. Conf. on Management of Data*, pp. 325–335.

[27] J. M. Hellerstein and M. Stonebraker. Predicate Migration: Optimizing Queries with Expensive Predicates. *ACM SIGMOD Int. Conf. on Management of Data*, pp. 267–276, 1993.

[28] Informix Geodetic DataBlade Module: User's Guide. Informix Press, 1997.

[29] JTS Topology Suite. Vivid Solutions. URL: http://www.vividsolutions.com/JTS/JTSHome.htm.

[30] M. McKenney, A. Pauly, R. Praing, and M. Schneider. Dimension-Refined Topological Predicates. *13th ACM Symp. on Geographic Information Systems*, pp. 240–249, 2005.

[31] OGC Abstract Specification. OpenGIS Consortium (OGC), 1999. URL: http://www.opengis.org/techno/specs.htm.

[32] Oracle8: Spatial Cartridge. An Oracle Technical White Paper. Oracle Corporation, 1997.

[33] M. A. Rodriguez, M. J. Egenhofer, and A. D.Blaser. Query Pre-Processing of Topological Constraints: Comparing a Composition-Based with Neighborhood-Based Approach. *Int. Symp. on Spatial and Temporal Databases*, LNCS 2750, pp. 362–379. Springer-Verlag, 2003.

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

[35] M. Schneider. Implementing Topological Predicates for Complex Regions. *Int. Symp. on Spatial Data Handling*, pp. 313–328, 2002.

[36] M. Schneider. Computing the Topological Relationship of Complex Regions. *15th Int. Conf. on Database and Expert Systems Applications*, pp. 844–853, 2004.

[37] M. Schneider and T. Behr. Topological Relationships between Complex Lines and Complex Regions. *Int. Conf. on Conceptual Modeling*, 2005.

[38] M. Schneider and T. Behr. Topological Relationships between Complex Spatial Objects. *ACM Trans. on Database Systems*, 31(1):39–81, 2006.

[39] M. Schneider and R. Praing. Efficient Implementation Techniques for Topological Predicates on Complex Spatial Objects: The Exploration Phase. Technical report, University of Florida, Department of Computer & Information Science & Engineering, 2006.

[40] M.F. Worboys and P. Bofakos. A Canonical Model for a Class of Areal Spatial Objects. *3rd Int. Symp. on Advances in Spatial Databases*, LNCS 692, pp. 36–52. Springer-Verlag, 1993.