Partition and Conquer

Martin Erwig & Markus Schneider

FernUniversität Hagen, Praktische Informatik IV
D-58084 Hagen, Germany
[martin.erwig | markus.schneider]@fernuni-hagen.de

Abstract. Although maps and partitions are ubiquitous in geographical information systems and spatial databases, there is only little work investigating their foundations. We give a rigorous definition for spatial partitions and propose partitions as a generic spatial data type that can be used to model arbitrary maps and to support spatial analysis. We identify a set of three powerful operations on partitions and show that the type of partitions is closed under them. These basic operators are sufficient to express all known application-specific operations. Moreover, many map operations will be considerably generalized in our framework. We also indicate that partitions can be effectively used as a meta-model to describe other spatial data types.

1 Introduction

In spatially-oriented disciplines like geography, cartography, and related areas as well as in computer-assisted systems like geographical information systems (GIS), spatial database systems, and image database systems, the most fundamental and well-known metaphor is that of a *map*. A map is a widely recognized geometric structure; it provides a powerful concept capable of carrying a large amount of information in visual form. It can be viewed from two different perspectives: the *space-based* approach models a map as a point set; each point in space represents a coordinate location and is associated with some properties or attributes. The *object-based* approach advocates the explicit and exact representation of spatial objects as single, self-contained entities. A map is regarded as a composition of a set of point objects, a set of line objects, and a set of region objects. Spatial data types for points, lines, regions, etc. have been intensively investigated in the literature and assessed as appropriate concepts for modeling spatial phenomena (for example, [CZ96, Eg89, EG94, Er94, GS95, Gü88, LN87, Sc95]). A main drawback of the object-based approach is the incapability of modeling topological relationships between elements of a set of spatial objects.

This paper focuses on a formal treatment of *partitions* as the central element of maps. The importance of partitions is in particular reflected by the fact that the notion "map" is frequently used as a synonym for "partition". A *partition* is a subdivision of the plane into pairwise disjoint *blocks* or *regions* where each region is associated with an attribute, which can have a simple or even complex structure. It implicitly models topological relationships between the participating regions. First of all, the neighborhood relationship is of particular interest here where different regions may have common boundaries. This property is immediately visible on a map. A related aspect is that different regions of a partition are always disjoint (if we neglect common boundaries), so that a visual representation of a partition has a very simple structure and is easy to grasp. Partitions by nature do not model overlap and containment relationships which can be considered for single and self-contained spatial objects [EF91].

There are numerous examples of partitions in the real world like subdivisions of the world map into countries, classifications of rural areas according to their agricultural use (maps arising from classifying space according to some aspect are frequently called *thematic maps* or *categorical coverages*), or sites in cadastral applications. In geographically-oriented applications and systems partitions are regarded as the primary tool for spatial analysis tasks [Be87, Fr87, HSH92, NW79, To90, Va91, FVM97, VE93]. Distinct features over the same space are to be combined and evaluated under different requirements. Partition-based spatial analysis functions include operations like overlay, generalization, and reclassification (see Section 2). They all produce a new partition as a result.

In the literature partitions have been identified as a central *spatial concept* [Fr90] to organize our perception and understanding of space. They correspond to the cognitive experience and knowledge the human has of areal phenomena in the real world. Humans cannot perceive overlapping regions but always disjoint decompositions of space. They can only guess and mentally complete the obscured part of a region (except for translucent regions). Also, if we consider the same space with respect to two different thematic or cognitive aspects (for example, districts and cereals) modeled as two partitions, their overlay is a partition again.

All the more it is surprising that a formal definition of *spatial* partitions and operations on them has been neglected. The work on categorical coverages done by Frank, Volta, and others [FVM97,VE93] focuses on partitions of attribute values alone – spatial operations are completely ignored, in particular, boundaries are not considered which play an important role in connection with geometric intersection. In categorical coverages, themes and attribute values are fixed, which means that dynamic extensions or combinations of different partitions are not possible.

In this paper we study partitions thoroughly and give a formal semantics to them and their operations. We show that partitions can serve as a fundamental and user-friendly data modeling tool offering a formal and powerful basis for coping with spatial analysis and cartographical tasks. All application-specific operations proposed so far are covered and even extended by our approach. Hence, our partition model is in particular interesting for GIS and spatial database systems. In our framework, from a data type and database point of view, a partition is a generic spatial data type which is formalized through a type constructor. From a general point of view, partitions model collections of objects (not single objects), or more precisely, collections of objects that are somehow related together. In particular, partitions are data types for collections of objects with integrated constraints like topological relationships.

On the one hand, partitions enable us to consider the attributes of single points (space-based view), on the other hand, they also provide access to collections of points having equal attributes (object-based view). Thus our model closes the gap between these two views of areal features. In this sense our framework yields a hybrid model. Unexpectedly, some very specific concepts are also covered by partitions; we are, for example, able to model *vague regions* [ES97] and even generalizations thereof on the basis of partitions.

Section 2 gives an overview of partitions and application-specific operations in the literature. In Section 3 we define a general model of spatial partitions. Section 4 identifies and formalizes three basic and very powerful operations on partitions. In Section 5 we demonstrate that these operations suffice to express and even generalize the application-specific operations introduced in Section 2. Section 6 concludes the paper.

2 Previous Work

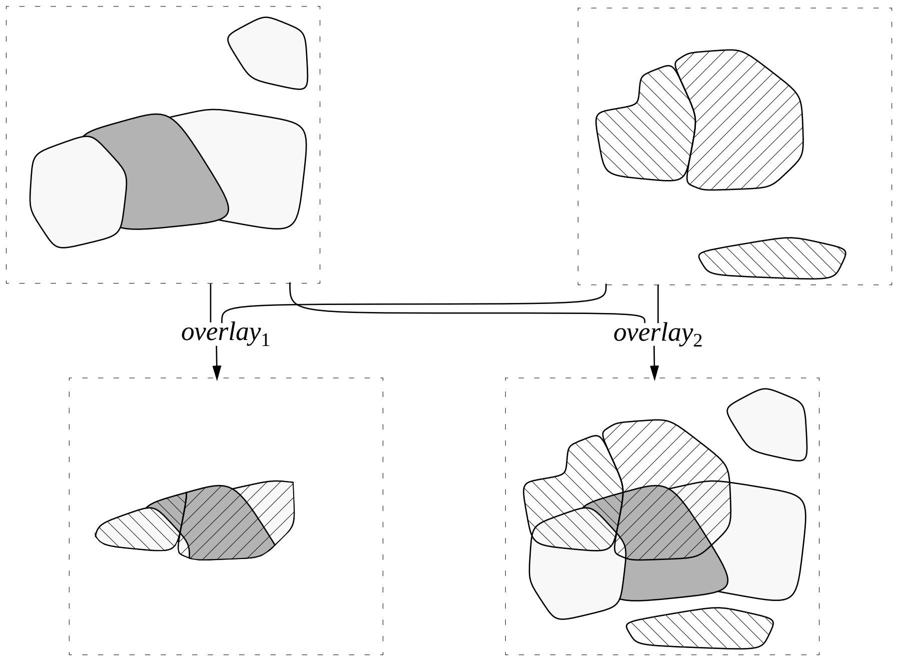
In this section we give an overview of previous work on partitions and pertaining application-specific operations as they have been identified in the geographically-oriented and computer science literature. The overview will reveal that, surprisingly, formal models for spatial partitions and formal descriptions of their semantics have not yet been studied satisfactorily. Expositions on this topic are predominantly based on an informal and intuitive level.

On (spatial) partitions two kinds of partial ordering can be defined, either induced by refinement or by inclusion [Fr87]. A partition π is said to be a *refinement* of a partition σ if each region in π is contained in a region of σ. With respect to refinement and inclusion partitions form not only a partially ordered set but a lattice. Both lattice structures are related but distinct: the lattice induced by refinement is one in which the elements are the partitions, whereas the lattice induced by inclusion consists of single regions ordered by the spatial subdivision in regions.

From a data type perspective there have been some unsatisfactory attempts to model partitions. In [Gü88] a spatial data type *area* is introduced to guarantee the disjointedness constraint. Within the framework of an extended relational data model, called the geo-relational algebra, the set of polygons occurring in a relation as a column of an attribute of type *area* has to fulfill the integrity constraint that all polygons are disjoint. Unfortunately, the maintenance of this property is not supported by the data model, but it is up to the user's responsibility. A generic data type (type constructor) for partitions, called *tessellation*, is informally introduced by [HSH92] as a specialized type for sets of polygons; this type can be parametrized with an attribute of a yet unspecified type. In [GS95] so-called *restriction types* have been proposed. For a data type *d* and a binary predicate *p* a special set d^p of types (called "kind") is defined; each type *d'* in d^p describes a set of values such that for any two distinct elements of *d'* the predicate *p* holds. Furthermore, any such type *d'* is defined to be a *subtype* of *d* which means that all operations defined for type *d* are also applicable to instances of type *d'*. If, for example, *d* is a type for regions and *p* is equal to a predicate "disjoint", we obtain a kind for partitions. In [MC80] partitions (maps) have been recognized as a fundamental geometric data type which is realized in files. Their whole system GADS (Geographic Analysis and Display System) is based on this single structure.

Previous approaches to define the semantics of operations on partitions have been based on a relational [Gü88, SV89] or object-oriented [HSH92] setting. Unfortunately, they do not formalize the concept of a partition. In [SV89] spatial join, spatial selection, spatial product, and fusion are introduced as spatial auxiliary operators which are then used to express the application-specific operations on partitions.

For spatial analysis tasks several application-specific operations on partitions have been identified. The most important operation is the *overlay* operation [Be87, Da90, Fr87, GS95, Gü88, HSH92, KBS91, Sc95, SV89, To90, Va91] which allows to lay two partitions with different attribute categories on top of each other and to combine them through intersection into a new partition of disjoint regions.[1] The attributes from the input partitions are then either distributed to each block in the refined partition or appropriately combined to form a new attribute. Two different interpretations of the overlay

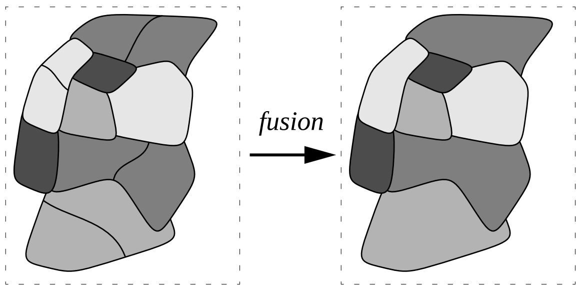


1. The dashed boxes in the following figures represent equal reference frames of partitions.

mechanism are given in the literature. In the first interpretation (*overlay*$_1$) [GS95, Gü88, KBS91, Sc95, SV89] only those regions are part of the new partition that result from intersecting a region of the first partition with a region of the second partition. Since the plane need not be completely covered by regions, it is possible that a region of one partition does not intersect any region of the other partition. In this case it will not be part of the new partition. In the second interpretation (*overlay*$_2$) [HSH92] also those regions are taken into account that do not intersect any region of the other partition.

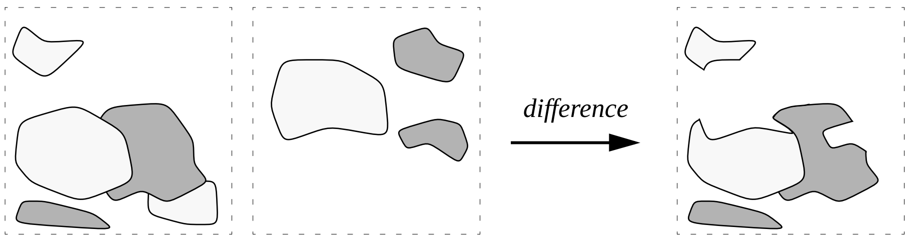
The *reclassify* operation [Be87, Da90, HSH92] retains the geometric structure of the partition and performs a transformation on the partition attributes. It can be thought of as a “recoloring” process; each or only a few regions are assigned with a new or modified attribute. An example is to map the population number of each region to its population density.

The *fusion* operation [CZ96, GS95, HSH92, KBS91, Sc95, SV89], called *merge* in [KBS91] and *generalization* in [HSH92], is a kind of grouping operation with subsequent geometric union. It merges adjacent regions of a partition with respect to partially identical attributes. An example is a partition depicting districts together with their land use (left side of the image). The task is to compute the regions with the same land use. Neighboring districts with the same land use are replaced by a single region, that is, their common boundary line is erased. District boundaries are not distinguished any more.



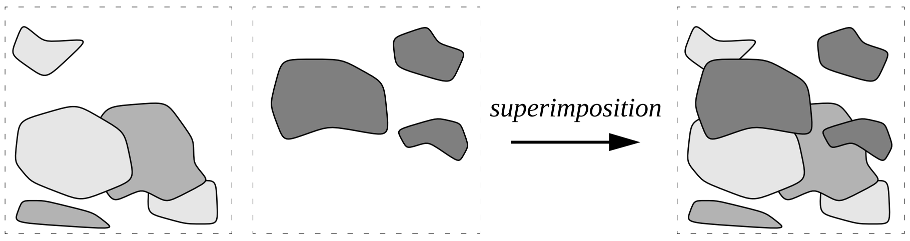
The *cover* operation proposed by [SV89] forms the geometric union of all regions of a partition and yields a partition consisting of one region. Since polygons with holes are not allowed in [SV89], the input partition must be complete, that is, it must not have holes. We will drop this restriction. The *clipping* operation [SV89] is a special case of the overlay operation; it computes the intersection of a partition and a given rectangular window. We will generalize this operation and will allow so-called “unit partitions” as clipping windows. The regions of a unit partition are associated with a “neutral” attribute which leaves an attribute of the first partition unchanged during an intersection.

The *difference* operation [HSH92] takes two partitions defined over the same attribute domain and computes the geometric difference of their point sets. All the regions of the first partition are maintained in the result partition except for those parts that have the same attributes in both partitions. We will generalize the difference operation in two

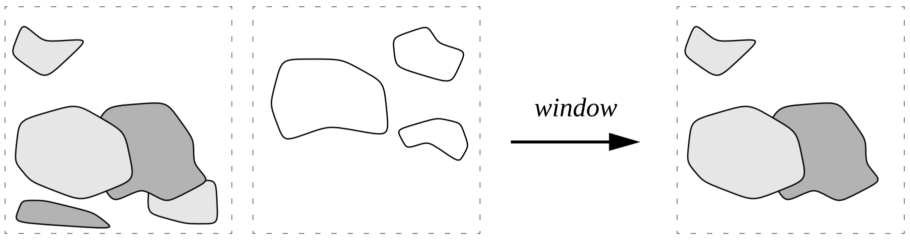


ways. First, we will allow partitions as operands with different attribute domains. Second, we can observe that giving priority to the first partition leads to a kind of “weak” difference. The difference operation gets a stronger interpretation if we allow the second partition to dominate. This means that the area of the second partition (precisely: its cover) is subtracted from the area of the first partition. Consider a partition of mineral resources and a partition of inhabitable areas. Subtracting the second partition from the first one yields the partition of currently exploitable areas. We will take both interpretations into account.

The *superimposition* operation [CZ96, SV89] allows to superimpose the regions of a partition onto another partition and to cover and erase parts of the other partition.



The *window* operation [SV89] allows to retrieve those (complete) regions of a partition whose intersection with a given (rectangular) window is not empty. Windowing is also applied in queries where the window is defined as a circle with center p and radius r, for instance, if we ask for all regions whose distance from a given point p is less than r. We generalize this operation and allow unit partitions instead of rectangles or circles.



The *divide* operation [CZ96] takes a partition, which consists of one region, and a spatially embedded planar graph as operands and yields a new partition which results from the decomposition or splitting of the input partition through the graph structure, as far as this is possible.

3 A Formal Model of Spatial Partitions

In naive set theory a partition is a complete decomposition of a set S into non-empty, disjoint subsets $\{S_i \mid i \in I\}$, called *blocks*, that is,

(i) $\forall i \in I: S_i \neq \emptyset$,
(ii) $\cup_{i \in I} S_i = S$, and
(iii) $\forall i, j \in I, i \neq j: S_i \cap S_j \neq \emptyset$.

Here, I is just an uninterpreted index set used to name different blocks, that is, I has no semantically relevant content. Equivalently, a partition can be regarded as a total and surjective function $\pi : S \to I$. Accordingly, one could try to define a *spatial partition*[2] simply as a set-theoretic partition of the plane, that is, as a partition of the set of points $\mathbb{R}^2$ or as a function $\pi : \mathbb{R}^2 \to I$. There are two observations, however, that motivate a slightly different definition.

First, from an application point of view, different blocks (or regions) of a spatial partition are often colored or marked differently. Colors and marks are only examples of rather arbitrary values, also called *labels*, that can be assigned to regions. A partition model should take this into account and should therefore regard point sets together with the associated values. The set of values that are actually used in a specific partition, say, A, determines in a certain way the type of the partition. Note that A might be a simple type, such as $\mathbb{N}$ or {red, green, blue}, or a cartesian product of two or more sets where labels are actually given by n-tuples of values. This means that spatial partitions of type A are actually functions $\pi : \mathbb{R}^2 \to A$ where A, in contrast to I, contains semantically relevant values. In most cases partitions are defined only partially, that is, there are regions which have no explicitly assigned labels. (These regions are sometimes considered the *outside* of the partition.) To ensure that π is a total function, we assume that each label type A contains an element $\perp_A$ (called *undefined* or *unknown*) and that the outside area of a partition is labeled by $\perp_A$. For the cartesian product of two types A and B we let $\perp_{A \times B} = (\perp_A, \perp_B)$ (this is like the identification of bottom elements in domain theory), and for the union of A and B we identify $\perp_A$, $\perp_B$, and $\perp_{A \cup B}$ (that is, we take the coalesced sum). If no ambiguities can arise, we sometimes omit the type index and simply use $\perp$.

Second, regions that actually do appear in applications are in most cases not just point sets, but point sets that are in a certain sense *regular*. This means that regions typically do not have isolated points or lines or cuts or punctures. This idea is well modeled by the topological concept of regular sets as shown by Tilove [Ti80].[3] So we would like to

2. In the following we will mostly use the term partition in place of spatial partition.

have regularity for partitions, too; in particular, we require interior regions (that is, regions without their boundary) to be regular open sets. Since points on the boundary cannot be uniquely assigned to either adjacent region, we cannot simply map them to single A-values. Instead we map boundary points to the set of values given by the labels of all adjacent regions. Thus spatial partitions are defined as functions $\pi : \mathbb{R}^2 \to A \cup 2^A$ (with additional constraints). We give the definition in several steps.

Definition 1. A *spatial mapping* of type A is a total mapping $\pi : \mathbb{R}^2 \to A \cup 2^A$.

The *range* of a spatial mapping π gives the set of labels actually used in π and is denoted by $range(\pi)$.The *blocks* of a spatial mapping π are point sets that are mapped to the same value. Formally, the blocks are given by the quotient of $\mathbb{R}^2$ with respect to the equivalence relation $\mathbf{ker}(\pi)$, the kernel of π.

The application of a function f to a set of values A is defined as: $f(A) := \{f(a) \mid a \in A\}$, and for convenience we use the following definition of function inverse: for $f : X \to A$ and $\forall a \in A$: $f^{-1}(a) := \{x \in X \mid f(x) = a\}$. Note that f^{-1} applied to a set yields a set of sets. Now the block for a single label a (or a set s of labels) is simply given by $\pi^{-1}(a)$ ($\pi^{-1}(s)$). The common label of a block b of π is denoted by $\pi[b]$, that is, $\pi(b) = \{l\} \Rightarrow \pi[b] = l$.

The cardinality of block labels identifies different parts of a partition. In the first place, we can distinguish between the interior of a partition and its boundary. A *region* of π is any block of π that is mapped to a single element of A, and a *border* of π is given by a block that is mapped to a set of A-values. The *interior* of π is the union of all of its regions, and the *boundary* is given by the union of all of its border blocks:

Definition 2. Let π be a spatial mapping of type A.

(i) $\rho(\pi) := \pi^{-1}(range(\pi) \cap A)$ (*regions*)
(ii) $\beta(\pi) := \pi^{-1}(range(\pi) \cap 2^A)$ (*borders*)
(iii) $\iota(\pi) := \cup_{r \in \rho(\pi)}\, r$ (*interior*)
(iv) $\partial(\pi) := \cup_{b \in \beta(\pi)}\, b$ (*boundary*)

Now we can define a spatial partition by topologically constraining regions to regular open sets and by semantically constraining boundary labels to those of adjacent regions.

Definition 3. A *spatial partition* of type A is a spatial mapping π of type A with

(i) $\forall r \in \rho(\pi)$: r is a regular open set, and
(ii) $\forall b \in \beta(\pi)$: $\pi[b] = \{\pi[r] \mid r \in \rho(\pi) \wedge b \subseteq \bar{r}\}$ [3]

The set of all spatial partitions of type A is denoted by $[A]$, that is, $[A] \subseteq \mathbb{R}^2 \to A \cup 2^A$.

The partition boundary can be viewed as an undirected planar graph. From this point of view, we can discriminate the cardinality of border labels further: an *edge block* is

3. For the standard notions of open/closed set and interior/closure of a set A (denoted by Int(A)/$\overline{A}$), see any textbook on topology, such as [Du66]. An open set A is called *regular* if $A = \mathrm{Int}(\overline{A})$. An important property of regular open sets is that they are closed under intersection.

mapped to a two-element A-set and defines border curves between two regions. A *vertex block* is mapped to an A-set containing three or more elements; a vertex block is always a singleton point set and describes locations where three or more regions of a partition meet.

Definition 4. Let $\pi : [A]$. Then

(i) $\varepsilon(\pi) := \{b \in \beta(\pi) : |\pi[b]| = 2\}$ (*edge blocks*)

(ii) $\nu(\pi) := \{b \in \beta(\pi) : |\pi[b]| > 2\}$ (*vertex blocks*)

The distinction between edge and vertex blocks is helpful when describing the behavior of some of the following operations.

4 Basic Operations on Partitions

As it turns out, there are three basic, yet very powerful, operators on partitions which suffice to express most of the conceivable application-specific operations.

4.1 Intersection

Given two partitions of types A and B, we consider the partition that results from actually computing the intersections of all regions: each resulting region is labeled with a pair of values from $A \times B$, and the values on the boundary are derived from these. Thus we first determine the regions of the intersection. This can be done by a simple set-intersection of all regions, since $\cap$ is closed on regular open sets.

$$\rho_\cap(\pi, \sigma) := \{r \cap s \mid r \in \rho(\pi), s \in \rho(\sigma)\}$$

Now the interior of the intersection is the union of all these regions:

$$\iota_\cap(\pi, \sigma) := \cup_{r \in \rho_\cap(\pi, \sigma)}\, r$$

Next we define the spatial mapping restricted just to the interior: the label of each interior point is simply given by the pair of labels of the argument partitions.

$$\iota\text{-}\mathit{intersection}(\pi, \sigma) := \{(p, (\pi(p), \sigma(p))) \mid p \in \iota_\cap(\pi, \sigma)\}$$

Finally, the boundary labels are derived from the labels of all touching regions. We say that a region r *touches* a point p if $p \in \bar{r}$. Now let $R := \rho_\cap(\pi, \sigma)$, $I := \iota_\cap(\pi, \sigma)$, and $\tau := \iota\text{-}\mathit{intersection}(\pi, \sigma)$. Then we have:

$$\mathit{intersection} : [A] \times [B] \to [A \times B]$$
$$\mathit{intersection}(\pi, \sigma) := \tau \cup \{(p, \{\tau[r] \mid r \in R \wedge p \in \bar{r}\}) \mid p \in \mathbb{R}^2 - I\}$$

Figure 1 show two partitions π and σ of type C and R modeling two countries c and d and mineral resources oil (o) and gas (g). Overlaying these two partitions is actually equivalent to computing their intersection. In Figure 2 regions colored ▨ (▩) denote oil fields in country c (d) and are labeled by (c, o), respectively, (d, o). The region colored ▤ is labeled (d, g) and denotes the gas field in country d (there is no gas field in country c). Region colors ▨ and ▤ with labels ($\perp_C$, o) and ($\perp_C$, g) show mineral re-

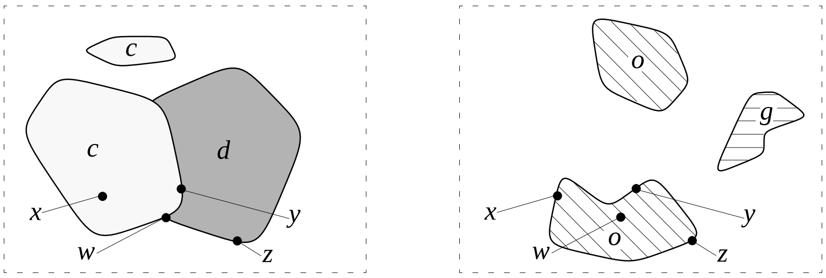


Figure 1. Two partitions for countries and mineral resources.

sources in no man's land, whereas regions ⬭ and ⬬ with labels $(c, \perp_R)$ and $(d, \perp_R)$ show country parts with unknown mineral resources. Finally, the outside of the intersection, that is, the region colored ⬭ and labeled $(\perp_C, \perp_R) = \perp_{C \times R}$ denotes the part of the no man's land with unknown mineral resources. In a similar way, we can distinguish the labels on the resulting partition boundary: for example, points x, y, and z have all the same label $\{o, \perp_R\}$ under σ and the labels c, $\{c, d\}$, and $\{d, \perp_C\}$ under π. In contrast, $\tau = \mathit{intersection}(\pi, \sigma)$ maps them to $\{(c, o), (c, \perp_R)\}$, $\{(c, o), (c, \perp_R), (d, o), (d, \perp_R)\}$, and $\{(d, o), (d, \perp_R), (\perp_C, o), \perp_{C \times R}\}$, respectively. For w the original partitions give $\pi(w) = \{c, d, \perp_C\}$ and $\sigma(w) = o$ whereas $\tau(w) = \{(c, o), (d, o), (\perp_C, o)\}$.

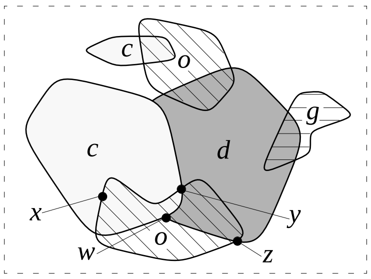


Figure 2. Intersection partition for countries and mineral resources.

We have to show that the definition of *intersection* is sound, that is, partitions are closed under *intersection*:

Lemma 1. If $\pi : [A]$ and $\sigma : [B]$, then $\mathit{intersection}(\pi, \sigma) : [A \times B]$.

Proof. Let $\tau = \mathit{intersection}(\pi, \sigma)$ and $\tau' := \iota\text{-}\mathit{intersection}(\pi, \sigma)$. First, it is obvious from the definition of *intersection* that τ is a total map on $\mathbb{R}^2$ and that $\mathit{range}(\tau) \subseteq (A \times B) \cup 2^{(A \times B)}$. It remains to be shown that (i) $\forall r \in \rho(\tau)$: r is a regular open set, and that (ii) $\forall b \in \beta(\tau)$: $\tau[b] = \{\tau[r] \mid r \in \rho(\tau) \wedge b \subseteq \bar{r}\}$. Concerning (i) it is clear that $\rho(\tau) = \rho_\cap(\pi, \sigma)$ (which we know to be a set of regular open sets) since only τ' maps to single values and since the domain of τ' is exactly $\rho_\cap(\pi, \sigma)$. Concerning (ii) we have for all border points $p \in b$: $\tau(p) = \{\tau'[r] \mid r \in \rho_\cap(\pi, \sigma) \wedge p \in \bar{r}\} = \{\tau[r] \mid r \in \rho(\tau) \wedge p \in \bar{r}\}$ since $\rho_\cap(\pi, \sigma) =$

$\rho(\tau)$ and since τ' equals τ on $\rho(\tau)$. Since $\beta(\tau) = \mathbb{R}^2 - I$, we thus have $\forall b \in \beta(\tau)$: $\tau[b] = \{\tau[r] \mid r \in \rho(\tau) \wedge b \subseteq \bar{r}\}$. □

4.2 Relabel

Relabeling a partition π of type A essentially means to apply a function $f : A \rightarrow B$ to each region of π. In the simplest case, when f is injective, the regions of π remain the same, and only the labels change as demanded by f. In general, however, f might map two or more regions of π to the same B-value, and if some of these regions are adjacent in π, the border between them disappears and the regions are fused in the result partition. We would like to express relabeling simply by applying f to the value each point has under π. Then, however, border points might be mapped to singleton sets where we would expect a single value. Consider, for example, two adjacent regions r and r' with $\pi[r] = x$ and $\pi[r'] = y$. The border b between r and r' is labeled $\pi[b] = \{x, y\}$. Now if $f(x) = f(y) = z$, we have $\pi[b] = \{z\}$ where we would like to have $\pi[b] = z$. We can adjust cases like this by simply applying a function $flat : A \cup 2^A \rightarrow A \cup 2^A$ that leaves single values and sets with two or more elements unchanged and extracts elements from singleton sets:

$$flat(l) = \begin{cases} a & \text{if } l = \{a\} \\ l & \text{otherwise} \end{cases}$$

Now we can easily define relabeling of partitions:

$$relabel : [A] \times (A \rightarrow B) \rightarrow [B]$$
$$relabel(\pi, f) := \{(p, flat(f(\pi(p)))) \mid p \in \mathbb{R}^2\}$$

Whenever a border label is mapped by *flat* to a single value, the border and its adjacent regions are identified and make up a new region containing the union of the border points and the points of all adjacent regions. This means that border points can become interior points through relabeling:

$$\iota(relabel(\pi, f)) \supseteq \iota(\pi) \qquad \text{and} \qquad \partial(relabel(\pi, f)) \subseteq \partial(\pi)$$

As an application consider the task of building a map showing regions of oil fields that can be exploited by either country c or d showing a possible conflict between c and d. Such a partition can be defined by applying *relabel* to the intersection of π and σ from above using a function mapping $(C \times R)$-tuples to $E = \{e, \bot_E\}$ for just coloring exploitable oil fields. This function is defined by:

$$f(x, y) = \begin{cases} e & \text{if } x \in \{c, d\} \text{ and } y = o \\ \bot_E & \text{otherwise} \end{cases}$$

Thus, the required map is given by the expression *relabel*(*intersection*(π, σ), f).The result is shown in Figure 3 as regions colored ▨; for clarity we have included the boundaries of the original partitions.

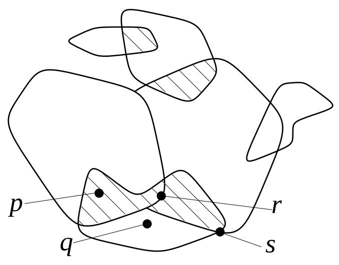


Figure 3. Reclassification of country/oil map.

The points *p*, *q*, *r*, and *s* are mapped by π, σ, and $\tau = relabel(intersection(\pi, \sigma), f)$ to the following values:

	π	σ	τ
p	c	o	e
q	$\perp_C$	o	$\perp_E$
r	$\{c, d\}$	o	e
s	$\{d, \perp_C\}$	$\{o, \perp_R\}$	$\{e, \perp_E\}$

Again, we have to show that *relabel* indeed yields proper partitions. We use the following results which makes the above mentioned relationships between interior and boundary of a partition and its relabeled version more precise. First, all borders whose labels are mapped by *f* to a singleton set are "moved" by *flat* into the interior. Thus we know:

Lemma 2. Let $\pi : [A]$ and $f : A \rightarrow B$. Then $\beta(relabel(\pi, f)) = \{b \in \beta(\pi) \colon |f(\pi[b])| > 1\}$

Establishing a similar relationship for regions gets a bit more involved. The application of *f* induces an equivalence on blocks. Let:

$$\forall r, s \in (\rho(\pi) \cup \beta(\pi)) \colon r{\sim}s :\Leftrightarrow \{f(\pi[r])\} = f(\pi[s]) \vee f(\pi[r]) = \{f(\pi[s])\}$$

The relationship ~ collects pairs of regions and borders that are mapped to one identical value. The equivalence relation $r \equiv s$ is then defined as the reflexive, symmetric, and transitive closure of ~ (that is, $r \equiv s :\Leftrightarrow (r{\sim}s) \vee (s{\sim}r) \vee (\exists t\colon s \equiv t \wedge t \equiv r)$). The equivalence classes of ≡ are maximal sets of adjacent blocks that have the same label. The regions of *relabel* are now obtained by taking the unions of blocks of all equivalence classes that have a single label.

Lemma 3. Let $\pi : [A]$ and $f : A \rightarrow B$. Then
$$\rho(relabel(\pi, f)) = \{\cup_{r \in s}\, r \mid s \in (\rho(\pi) \cup \beta(\pi))/{\equiv} \colon flat(f(\pi[s])) \in B\}$$

Now we can show that partitions are closed under *relabel*:

Lemma 4. If $\pi : [A]$ and $f : A \rightarrow B$, then $relabel(\pi, f) : [B]$.

Proof. Let $\sigma = relabel(\pi, f)$. The fact that σ is a spatial mapping of type *B* follows directly from the definition.

We must now show that $\forall r \in \rho(\sigma)$: r is a regular open set. We consider two cases: first, $s = \{r\} \in (\rho(\pi) \cup \beta(\pi))/\equiv$, that is, r results from a singleton set and thus $\cup_{r \in s}\, r = r$. This can only be the case if $r \in \rho(\pi)$, since for any border $b \in \beta(\pi)$ that is mapped to a singleton set, say $\{c\}$, at least the adjacent regions will also be mapped to c, and thus the equivalence class contains more than one element. Then we know from the partition property of π that r is a regular open set. On the other hand, if r is a union of two or more sets, the definitions of ~ and ≡ ensure that these are borders and regions of π that are completely connected. This means that (i) for each border there are at least two adjacent regions, (ii) for each pair of regions there is an adjacent border, and (iii) for each set of regions that are adjacent to a point p, $\{p\}$ is also in r. This implies that the union $\cup_{r \in s}\, r$ contains no dangling points or lines and no punctures and cuts which just means that r is a regular open set.

Next we must show that $\forall b \in \beta(\sigma)$: $\sigma[b] = \{\sigma[r] \mid r \in \rho(\sigma) \wedge b \subseteq \bar{r}\}$. If f is injective, σ is identical to π up to a "renaming" of labels which means that the above property follows directly from the same property of π. If f is not injective, we consider two cases: (i) $b \in \varepsilon(\pi)$. Let $\pi[b] = \{x, y\}$, and assume $\{r, r'\}$ are the adjacent regions. W.l.o.g. let $\pi(r) = x$ and $\pi(r') = y$. Now if $f(x) = z \neq z' = f(y)$, b remains an edge border with adjacent regions r and r', that is, $b \in \varepsilon(\sigma)$, and we have: $\sigma[b] = \{z, z'\} = \{\sigma[r] \mid r \in \rho(\sigma) \wedge b \subseteq \bar{r}\}$. If, on the other hand, $f(x) = z = f(y)$, the points in b are mapped to the single value z (caused by the application of *flat*), and thus b is merged with r and r' (which are also mapped to z) into a single region. (ii) $b = \{p\} \in \nu(\pi)$. Consider any subset of labels $l \subseteq \pi[b]$ that is mapped by f to the same value z. It is clear that each region r that touches p with $\pi[r] \in l$ is also mapped to z. Since the edge borders between any two such regions are also mapped to z, all regions are merged into a single new region s of σ which, of course, also touches p. So the one-to-one correspondence between the single labels of regions and the set of labels of adjacent points is preserved by mapping with f. □

At this point we can describe overlays of partitions by *intersection*, and by choosing an appropriate function f partitions can be arbitrarily generalized or reclassified through *relabel*. In particular, if f is not injective, a *coarsening* of partitions is achieved. However, we cannot really refine a single partition, for example, into its connected components. Therefore we need an additional operation *refine* that extends labels of a partition to distinguish different connected components of a region.

4.3 Refine

It is often the case that a region of a partition consists of two or more components. This might result from a preceding *intersection* or *relabel* operation, or it might be simply given by the application. In any case, it is sometimes interesting to distinguish different components. For example, if we want to find out mineral resources that can be exploited by a specific country c, we would like to perform a kind of *window* operation with window c on the partition of mineral resources. The result contains only some components of regions that intersect c, and for a specific resource, say oil, this also is, in general, only a subset of the corresponding components.

Refining a partition means to add tags (for example, numbers) to the components of a region so that they can be distinguished. Intuitively, a connected component of an open set s is a maximum subset $c \subseteq s$ such that any two points of c can be connected by a curve lying completely inside c (for a formal account, see [Du66]). We denote the set of connected components of a region $r \in \rho(\pi)$ by $\gamma(r) = \{c_1, \dots, c_{n_r}\}$. First, we define refinement on the interior:

$$\iota\text{-}\mathit{refine}(\pi) := \{(c_i, (\pi(c_i), i)) \mid r \in \rho(\pi) \wedge \gamma(r) = \{c_1, \dots, c_{n_r}\} \wedge i \in \{1, \dots, n_r\}\}$$

The set of all components are the regions of the refined partition, and the interior is just the union of all components:

$$\rho_\gamma(\pi) := \cup_{r \in \rho(\pi)} \gamma(r)$$
$$\iota_\gamma(\pi) := \cup_{r \in \rho_\gamma(\pi)} r$$

This means that the set of interior points (and thus the boundary points, too) is not changed by *refine*. As we have done in the definition for intersection, we derive the labels of the boundary from the interior. Let $\sigma := \iota\text{-}\mathit{refine}(\pi)$.

$$\mathit{refine} : [A] \to [A \times \mathbb{N}]$$
$$\mathit{refine}(\pi) := \sigma \cup \{(p, \{\sigma[r] \mid r \in \rho_\gamma(\pi) \wedge p \in \bar{r}\}) \mid p \in \mathbb{R}^2 - \iota_\gamma(\pi)\}$$

An immediate use of the *refine* operation is to explicate different components of specific partition regions. For example, we might be interested in the number of oil fields of the mineral resources partition σ. Since *refine*(σ) labels all components of a region consecutively, we can get the result as the maximum number n of any (o, n) label.

Partitions are closed under *refine*:

Lemma 5. If $\pi : [A]$, then *refine*(π) : $[A \times \mathbb{N}]$.

The proof is very similar to that for Lemma 1, and we omit it for brevity. It is clear that all regions of a refined partition are connected:

Lemma 6. $\rho(\mathit{refine}(\pi)) = \rho_\gamma(\mathit{refine}(\pi))$.

Finally, as a corollary of lemmas 1, 4, and 5 we obtain the following closure properties:

Theorem 1. Spatial partitions are closed under *intersection*, *relabel*, and *refine*.

5 Applications

Applications of partitions can be found almost everywhere in cartography, spatial analysis, etc. Operations that arise in those applications have been already reported in Section 2. Next we show how these (and some additional operations) can be realized and generalized by the basic operators introduced in the previous section. We thus come to the “conquering” step by demonstrating that our set of operations is complete from an application point of view. We shall also reveal interesting relationships among the different application operators (for example, *clipping* is just a special case of *overlay*$_1$, and *cover* is just a special case of *fusion*). If not stated otherwise, we assume that $\pi : [A]$ and $\sigma : [B]$. It is interesting to note that only two of the basic operators have been identified, more or less directly, in applications: *overlay* corresponds to *intersection*, and *reclassify*, *fusion*, and *cover* are all special cases of *relabel*, however, *refine* has no direct counterpart.

To begin with *overlay*, it is obvious that

$$\mathit{overlay}_2(\pi, \sigma) = \mathit{intersection}(\pi, \sigma)$$

In contrast, *overlay*$_1$ forgets all parts of the intersection that are undefined (that is, labeled with $\perp$) in either argument partition. We can recover this behavior by relabeling the intersection with a function mapping partially undefined values to undefined:

$$\mathit{both}(x, y) = \begin{cases} \perp_{A \times B} & \text{if } x = \perp_A \text{ or } y = \perp_B \\ (x, y) & \text{otherwise} \end{cases}$$

Then we simply have:

$$\mathit{overlay}_1(\pi, \sigma) = \mathit{relabel}(\mathit{intersection}(\pi, \sigma), \mathit{both})$$

Reclassification preserves the spatial structure of a partition and thus corresponds to relabeling with an injective function:

$$\mathit{reclassify}(\pi, f) = \mathit{relabel}(\pi, f) \qquad (\mathit{reclassify} \text{ is only defined if } f \text{ is injective})$$

In contrast, *fusion* does not perform any real computations on labels, it essentially defines an equivalence relationship on labels which causes adjacent regions with equivalent labels to be merged. Technically, *fusion* applies to partitions of type $\times_{i \in I} A_i$ (which is $A_1 \times \ldots \times A_k$ for $I = \{1, \ldots, k\}$ and models k attributes). As an additional argument *fusion* takes a set $S = \{i_1, \ldots, i_n\} \subseteq I$ specifying which attributes are to be kept. To realize this by *relabel* we need a function $\Pi_S : \times_{i \in I} A_i \to \times_{i \in S} A_i$ that projects onto the attributes given by S as follows: $\Pi_S(a_1, \ldots, a_k) = (a_{i_1}, \ldots, a_{i_n})$. So we have:

$$\mathit{fusion}(\pi, S) = \mathit{relabel}(\pi, \Pi_S)$$

Since *cover* is a special case of *fusion*, it can also be realized by *relabel*: *cover* ignores all attributes of a partition and only distinguishes between inside and outside (with label $\perp$). This amounts to fusion with an empty subset, that is,

$$\mathit{cover}(\pi) = \mathit{fusion}(\pi, \emptyset)$$

Although we are able to express *cover* by *fusion*, a more direct realization of *cover* is to relabel the partition with a special constant function that maps any (defined) value to a unit value "•" of type $U = \{\bullet\}$. (U denotes the unit type that consists of just one value.)

$$\mathit{unit}(x) = \begin{cases} \perp & \text{if } x = \perp \\ \bullet & \text{otherwise} \end{cases}$$

This also makes the type of *cover* more explicit, namely, *cover* : $[A] \rightarrow [U]$. (We call partitions of type $[U]$ *unit partitions*.) Now *cover* is defined by:

$$cover(\pi) = relabel(\pi, unit)$$

The *clipping* operation is essentially a special case of $overlay_1$. The restriction is that the overlaid partition consists of exactly one (unlabeled) rectangle. In particular, this means that the second argument is a unit partition and that the result partition should be of the same type as the first argument. Hence *clipping* has the restricted type $[A] \times [U] \rightarrow [A]$ so that the correspondence to $overlay_1$ only holds up to isomorphism of the two types $[A]$ and $[A \times U]$.

$$clipping(\pi, \sigma) \cong overlay_1(\pi, \sigma)$$

The type-correct realization of *clipping* by intersection and relabeling thus differs slightly from that of $overlay_1$. With

$$inside(x, y) = \begin{cases} x & \text{if } y = \bullet \\ \bot_A & \text{otherwise} \end{cases}$$

we thus obtain:

$$clipping(\pi, \sigma) = relabel(intersection(\pi, \sigma), inside)$$

Compared with the operations considered so far, *window* has a remarkably complex definition. It is also the only operation that really requires the use of *refine*. In applications, the window is assumed to be a rectangle, again of type U. We give a definition which allows windowing with respect to an arbitrary unit partition. This facilitates many more applications: for instance, we can window the country partition

(i) with a specific oil field,
(ii) with all oil fields, or even
(iii) with the (cover of the) whole mineral resources partition

to determine

(i) all countries that are possible competitors for that specific oil field,
(ii) all countries that can exploit oil, or
(iii) all countries that have access to mineral resources at all.

For the definition of the *window* operation, we assume we are given a partition $\pi : [A]$ and a window $\omega : [U]$. First, we refine π and construct the intersection of the components with ω. We then determine the set L of all labels that contain • which means to obtain the labels of all components that intersect ω. Finally, we relabel the components with a function that keeps labels contained in L and maps all other labels to $\bot$.

$$\begin{aligned} &window(\pi, \omega) = relabel(refine(\pi), covered) \\ &\quad \text{where} \quad L = \{(x, i) \mid ((x, i), \bullet) \in range(intersection(refine(\pi), \omega))\} \\ &\qquad\qquad covered(x, i) = \begin{cases} x & \text{if } (x, i) \in L \\ \bot_A & \text{otherwise} \end{cases} \end{aligned}$$

What makes the definition of *window* so complex compared with all other operations? The fundamental difference is that usually in an operator definition the label of any region is determined locally, that is, independent from other regions. However, this is not the case for *window*: to compute the label of a region it is necessary to look at all adjacent regions since the final label of a region labeled $((x, i), \perp)$ cannot be predicted in isolation: the result is x if there exists an adjacent region with label $((x, i), \bullet)$, and it is $\perp$ if there is no such region. So what we express here is essentially a kind of *aggregation* of partitions. We have deliberately omitted such an operator to keep the partition model simple and short.[4]

We can explain the meaning of *difference*(π, σ) by specifying the label for each region of the intersection of π and σ. Regions that are undefined in any argument partition, remain unchanged. It is also clear that regions that are labeled equally should be mapped to $\perp$. It remains to define the label of intersection regions r for which $\pi[r] = x$ and $\sigma[r] = y$ (with $x \neq y$). Now this can be done in two different ways: we can give priority to either argument partition π or σ. In the first case, this means to simply ignore the subtraction of y since it does not match, that is, r will remain labeled x. This is, in a certain sense, a "conservative" view that allows subtraction only for matching region labels. In contrast, the "aggressive" point of view subtracts everything (except $\perp$) so that r will get labeled $\perp$. We capture this behavior by the following two relabeling functions:

$$\mathit{left}(x, y) = \begin{cases} \perp & \text{if } x = y \\ x & \text{otherwise} \end{cases}$$

$$\mathit{right}(x, y) = \begin{cases} \perp & \text{if } y \neq \perp \\ x & \text{otherwise} \end{cases}$$

Now we can define *left* and *right difference* simply by:

$$\mathit{l\text{-}difference}(\pi, \sigma) = \mathit{relabel}(\mathit{intersection}(\pi, \sigma), \mathit{left})$$
$$\mathit{r\text{-}difference}(\pi, \sigma) = \mathit{relabel}(\mathit{intersection}(\pi, \sigma), \mathit{right})$$

Finally, we can define the superimposition of two partitions π and σ very similarly to *overlay*$_1$: we build the intersection of π and σ and perform a relabeling giving priority to σ. The function *second* takes the label given by σ as long as it is defined:

$$\mathit{second}(x, y) = \begin{cases} x & \text{if } y = \perp_B \\ y & \text{otherwise} \end{cases}$$

Then we have (note that *superimposition* has type $[A] \times [B] \to [A \cup B]$):

4. Actually, aggregations of partitions are closely related to the aggregation of (their dual) graphs, and as shown in [Er97] there are quite different reasonable possible definitions for that. Moreover, we can actually express all presented application operations without an aggregation operator. Nevertheless, aggregation has quite interesting applications, for example, testing whether a map is colored consistently, that is, checking for each region whether it is colored differently from all neighbors. We will deal with aggregations of partitions in a subsequent paper.

$superimposition(\pi, \sigma) = relabel(intersection(\pi, \sigma), second)$

Since we have concentrated on operations on partitions alone, we cannot currently define an operation *divide*(π, *G*) operator which requires a planar graph argument *G*. However, when we clip *G* with *cover*(π) and add *cover*(π), this results in a partition Γ which is actually the remainder of *G* with respect to the relevant area, that is, the area to which *divide* actually applies. Then we obtain the same result as *divide* by simple intersection:

$divide(\pi, G) = intersection(\pi, \Gamma)$

6 Conclusions

We have presented a very general model of partitions which, to our knowledge, is the first rigorous formal approach to defining a spatial partition type. We have defined three powerful operators that can express any application operation that has been mentioned so far – actually, for most operators we even achieve a much more general definition. The presented model serves as a specification for general spatial analysis and map-manipulation systems which can be the bases for many GIS applications. Of course, it remains to investigate the relationships of partitions to points and lines and to define corresponding types and operations.

The partition model can also serve as a meta-model for (some) spatial data types: for instance, an element of a region data type can be viewed as a partition with only one region or as a unit partition. Another example is the model of vague regions presented in [ES97]: a *vague region* is given by a pair of disjoint regions $v = (r, s)$ where r denotes the part which definitely belongs to v and s gives the uncertain part. It is obvious that a vague region can be viewed as a partition of type {*certain*, *uncertain*} with exactly two regions. All operations on vague regions can then be defined in terms of *intersection* and *relabel*.[5] With this representation we can immediately obtain a generalization of vague regions: a vague region can be simply regarded as a partition without a restriction on the number of regions, and labels are drawn from a type containing values for different degrees of vagueness, for example, real values from the interval [0..1].

References

[Be87] J.K. Berry. Fundamental Operations in Computer-Assisted Map Analysis. *Int. Journal of Geographical Information Systems*, vol. 1, no. 2, pp. 119-136, 1987.

[CZ96] E.P.F. Chan & R. Zhu. QL/G: A Query Language for Geometric Databases. *1st Int. Conf. on GIS in Urban and Environmental Planning*, pp. 271-286, 1996.

[Da90] J. Dangermond. A Classification of Software Components Commonly Used in Geographic Information Systems. *Introductory Readings in Geographic Information Systems*, Taylor & Francis, pp. 30-51, 1990.

[Du66] J. Dugundji. *Topology.* Allyn and Bacon, Boston, 1966.

5. Actually the explanations of the vague region operations have been given in just that way: namely, define the result label of any possible intersection region.

[EF91] M.J. Egenhofer & R.D. Franzosa. Point-Set Topological Spatial Relations. *Int. Journal of Geographical Information Systems*, vol. 5, no. 2, pp. 161-174, 1991.

[Eg89] M.J. Egenhofer. Spatial SQL: A Spatial Query Language. Report 103, Dept. of Surveying Engineering, University of Maine, 1989.

[EG94] M. Erwig & R.H. Güting. Explicit Graphs in a Functional Model for Spatial Databases. *IEEE Transactions on Knowledge and Data Engineering*, vol. 6, no. 5, pp. 787-804, 1994.

[Er94] M. Erwig. *Graphs in Spatial Databases*. Doctoral Thesis, FernUniversität Hagen, 1994.

[Er97] M. Erwig. Functional Programming with Graphs. *2nd ACM SIGPLAN Int. Conf. on Functional Programming*, pp. 52-65, 1997.

[ES97] M. Erwig & M. Schneider. Vague Regions. *5th Int. Symp. on Spatial Databases (SSD'97)*, 1997. To appear.

[Fr87] A.U. Frank. Overlay Processing in Spatial Information Systems. *Proc. of the 8th Int. Symp. on Computer-Assisted Cartography, AUTOCARTO 8*, pp. 16-31, 1987.

[Fr90] A.U. Frank. Spatial Concepts, Geometric Data Models and Data Structures. *Computer and Geosciences*, 1990.

[FVM97] A.U. Frank, G.S. Volta & M. MacGranaghan. Formalization of Families of Categorical Coverages. *Int. Journal of Geographical Information Science*, vol. 11, no. 3, pp. 215-231, 1997.

[GS95] R.H. Güting & M. Schneider. Realm-Based Spatial Data Types: The ROSE Algebra. *VLDB Journal*, vol. 4, pp. 100-143, 1995.

[Gü88] R.H. Güting. Geo-Relational Algebra: A Model and Query Language for Geometric Database Systems. *Int. Conf. on Extending Database Technology*, LNCS 303, pp. 506-527, 1988.

[HSH92] Z. Huang, P. Svensson & H. Hauska. Solving Spatial Analysis Problems with GeoSAL, A Spatial Query Language. *6th Int. Working Conf. on Scientific and Statistical Database Management*, 1992.

[KBS91] H.-P. Kriegel, T. Brinkhoff & R. Schneider. The Combination of Spatial Access Methods and Computational Geometry in Geographic Database Systems. *2nd Symp. on Advances in Spatial Databases* (*SSD'91*), LNCS 525, pp. 5-21, 1991.

[LN87] U. Lipeck & K. Neumann. Modelling and Manipulating Objects in Geoscientific Databases. *5th Int. Conf. on the Entity-Relationship Approach*, pp. 67-86, 1987.

[MC80] P.E. Mantey & E.D. Carlson. Integrated Geographic Data Bases: The GADS Experience. *Data Base Techniques for Pictorial Applications*, Springer, pp. 173-190, 1980.

[NW79] G. Nagy & S. Wagle. Geographic Data Processing. *ACM Computing Surveys*, vol. 11, no. 2, pp. 139-181, 1979.

[Sc95] M. Schneider. *Spatial Data Types for Database Systems*. Doctoral Thesis, FernUniversität Hagen, 1995.

[SV89] M. Scholl & A. Voisard. Thematic Map Modeling. *1st Int. Symp. on Large Spatial Databases* (*SSD'89*), pp. 167-190, 1989.

[Ti80] R.B. Tilove. Set Membership Classification: A Unified Approach to Geometric Intersection Problems. *IEEE Transactions on Computers*, vol. C-29, pp. 874-883, 1980.

[To90] C.D. Tomlin. *Geographic Information Systems and Cartographic Modeling*. Prentice Hall, 1990.

[Va91] C.R. Valenzuela. Data Analysis and Modeling. *Remote Sensing and Geographical Information Systems for Resource Management in Developing Countries*, pp. 335-348, 1991.

[VE93] G.S. Volta & M.J. Egenhofer. Interaction with Attribute Data Based on Categorical Coverages. *Conf. on Spatial Information Theory (COSIT'93)*, LNCS 716, pp. 215-233, 1993.