An Algebra for OpenGIS® Coverages based on Topological Predicates

Gilberto Camara, Danilo Palomo, Sérgio Souza Costa

Image Processing Division (DPI) – National Institute for Space Research (INPE)
Av dos Astronautas, 1758 – 12227-001 – São José dos Campos – SP – Brazil
{gilberto, danilo, sergio@dpi.inpe.br}

Abstract. Map algebra is a collection of functions for handling spatial datasets where each data contains a set of geometries of the same type which bound to a geographical reference. The current theory for Map Algebra uses *ad hoc* operators proposed by Dana Tomlin. His proposal has had great practical success, and most GIS implementations provide its operations. However, there is a lack of theoretical foundations for the operations proposed in Tomlin's map algebra. This is a limitation for the proposal of international standards for map algebra. Specifically, the Open Geospatial Consortium's proposal for handling a map (the coverage data type) lacks a set of functions to manipulate its content. To address this problem, our work proposes a specification for an algebra for Open GIS® coverages which uses a dimension-extended version of Egenhofer and Herring's 9-intersection predicates to express spatial operations. The proposed coverage algebra includes all Tomlin's functions, as well as operations that are not part of Tomlin's algebra, but are useful in practice. Our proposal could be the basis for setting up standard for operations on Open GIS coverages. The use of standards for operations in coverages would be a significant advance for increased interoperability of spatial data.

1 Introduction

In recent years, there has been a significant effort to standardize the technology of geographical information systems (GIS). This effort is motivated by the large diffusion of GIS worldwide, the need to share geographical data, and for long-term maintenance of geographical archives. Sharing geographical data requires that different institutions, using diverse technologies, have access to the same data sets. Long-term archive maintenance needs that data outlives both its original media support and the software used to build it. Thus, sharing and maintenance of geographical data need standardization. The Open Geospatial Consortium (OGC) is developing standards for modelling, accessing, storing, and sharing geographical data. The extent of the effort involved in OGC's mandate is significant. Therefore, despite the progress already achieved, there are still areas where OGC's task is not complete. One of the missing parts of OGC's specification is a set of functions for manipulation of coverages, a subject commonly referred to as 'map algebra'. The current version of OGC's specification for coverages [1] mentions unary and binary operations, and does not consider

spatial operations. Since there is a literature on map operations ([2] [3] [4]), it is important to consider how these operations can be used for handling OGC's coverages. This work will discuss how to extend OGC's coverage data type to create an algebra for coverages.

The main contribution to map algebra comes from the work of Tomlin [4]. Tomlin proposed a set of operations that has proven useful in practice. Extensions to Tomlin's map algebra include the GeoAlgebra of Takeyama and Couclelis [5] and MapScript, a language that includes dynamical models [6]. Extensions of map algebra for spatio-temporal data handling are discussed by Mennis et al. [7] and Frank [8]. All these works use Tomlin's algebra as a basis for their work. The main problem with Tomlin's algebra and its extensions is their *ad hoc* basis. There is no foundation for assessing the completeness of his algebra. Thus, there could be other types of operations that are missing in his algebra.

Therefore, one of the open challenges in spatial information science is to develop a theoretical foundation for a comprehensive set of operations on coverages. We need to find out if Tomlin's map algebra is part of a more general set. We state these questions as: *"What is the theoretical foundation for spatial operations on coverages?", "Could this theoretical foundation provide support for a comprehensive spatial algebra for coverages?"* To respond to these questions, we take the topological predicates of Egenhofer and Herring [9] as a basis for defining an algebra for coverages. Using these predicates, we develop a coverage algebra that includes Tomlin's map algebra as a subset.

In what follows, we briefly review the literature about coverages, map algebra and spatial predicates in chapter 2. In section 3, we present an algebraic specification of an algebra for coverages. In section 4, we present some examples of its use, including some problems that are not expressible in Tomlin's work. This paper is the third on our research on providing foundations for map algebra. In [10], we point out how to generalize Tomlin's map algebra by incorporating topological spatial predicates. That paper did not consider the OpenGIS® coverage type and lacked a rigorous algebraic specification. In [11], we discuss the use of functional programming for GIS application development, and provide an example of a map algebra implementation in the Haskell language. In the current paper, we use the findings of [10] and [11] to provide a specification for an algebra for OpenGIS® coverages, using a general-purpose specification language. We also provide examples of its use.

2 Literature Review

2.1 The OpenGIS® Coverage

OGC's definition of coverage provides a support for the concepts of 'fields' and 'maps'[1]. A coverage in a spatial representation that *covers* a geographical area and divides it in spatial partitions that may be either regular or irregular and assigns a value to each partition. The computer representation of a coverage consists of a coverage function over a discrete domain, called the *DiscreteC_Function*. Its domain is a

collection of geometries and its range is a set of vectors of attributes. The OGC specification describes the *DiscreteC_Function* as having four operations, shown in Figure 1 below:

- **Num**: finds the number of geometries in the coverage.
- **Values:** finds the possible values of the coverage function.
- **Domain:** finds the geometries in the domain of the coverage function.
- **Evaluate:** given a geometry, return a vector that represent the values of the coverage function for the associated location.

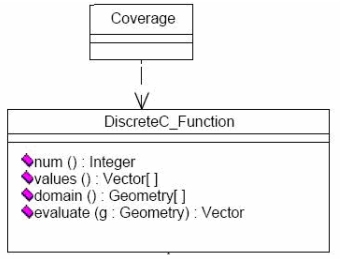


Fig. 1. The Open GIS discrete coverage function – source: [1]

An example is coverage whose domain is the geometries that describe the states of a country, and the range is each state's population in 2004. A second example would be a coverage whose domain is a set of regular cells and the range is a set of values representing the maximum and minimum yearly temperatures the region covered by the set of cells. OGC considers a set of coverage subtypes, where each subtype uses a different spatial data structure to build its domain. The subtypes include polygons, images, TINs, surfaces, and point sets. OGC describes how to evaluate the discrete coverage function for different coverage subtypes [1].

2.2 Topological operators for spatial relations

The OGC specification describes a set of topological predicates for spatial relations between geometries of simple features[12]. These predicates use a dimension-extended version of the 9-intersection model proposed by Egenhofer and Herring [9]. The 9-intersection model considers a geometrical object (A) as composed by a set of boundary points (∂A), a set of interior points (A^{o}), and a set of exterior points (A^{-}). For area-area relations, OGC proposes the set *{'disjoint', 'equal', 'touches', 'within', 'overlap', 'contains', 'intersects'}* (see Figure 2). For a rigorous definition for area-area relations to raster representations, see [13] and [14]. For line-area relations, the model proposes the set *{'disjoint', 'touch', 'within', 'cross', 'intersects'}*. For point-area relations, the model proposes the set *{'disjoint', 'touch', 'within', 'cross'}*. Other topological relations are described in [9]. In what follows, we define spatial operations in coverage algebra should using OGC's topological predicates.

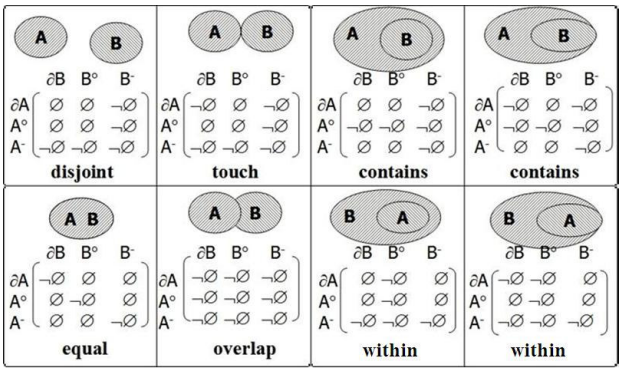
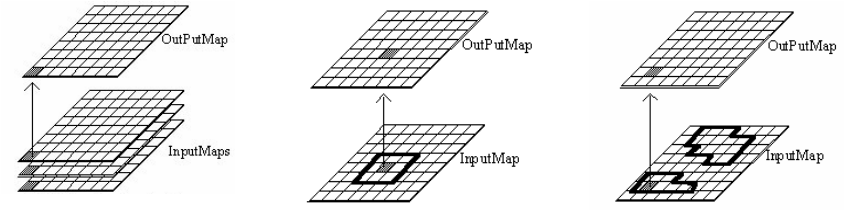


Fig. 2. Topological predicates for area-area relations based on the 9-intersection matrix. Adapted from [9].

2.3 Map Algebra

Tomlin's map algebra [4] includes first-order and higher-order functions for maps. *First order* functions take values as arguments (these are the functions associated to the map values). *Higher order functions* are functions that have other functions as arguments. Higher order functions are the basis for map algebra operations [15]. These functions apply a first-order function to all elements of map. Tomlin [4] proposes three higher-order functions:

- **Local function**: produce a new map, whose value in each location *p* depends only of the values in *p* in the input maps, as in "*classify as unsuitable for farming all areas with slope greater than 15%"*. A local operation is a mapping between the *ranges* of the input and output fields (Figure 3.a).
- **Focal function**: produce a new map, whose value in each location *p* depends only of the values of a neighbourhood around *p* in the input map, as in the expression "*for each county, calculate the average population of its neighbours*". (Fig. 3. b).
- **Zonal function**: produce a new map, whose value in each location *p* depends on the values of a region in an input map. This region is defined by a restriction on a third map, called the reference map. Example is "*given a map of cities and a digital terrain model, calculate the mean altitude for each city"* (Figure 3.c).



a. Local functions **b. Focal functions** **c. Zonal functions**

Fig. 3. Tomlin's functions for map algebra (source: Tomlin [4]).

We can express Tomlin's spatial operations using Egenhofer and Herring's [9] topological predicates. The focal operation uses the condition of *adjacency*, which matches the spatial predicate *'touch'*. The zonal operation uses the condition of containment, which matches the spatial predicate *'within'*. Since *'touch'* and *'within'* are part of a more general set of predicates, Tomlin's operations use only a subset of all possible topological relations between areas. Tomlin's algebra uses spatial predicates in a limited way. It applies the *'touch'* relation (focal function) only over the same input map. It also only applies the *'within'* relation (a zonal operation) over a reference map. If we remove the limits of Tomlin's algebra, we can have a coverage algebra based on the full set of topological predicates, whose operations don't restrict the way in which the predicates are used.

3 A Generalized Algebra for Coverages

This section presents the design of an algebra for coverages. The proposed algebra extends the *coverage* data type defined by the Open GIS® consortium [1] and has *nonspatial* and *spatial* higher-order functions. The nonspatial operations are Tomlin's local operations. The spatial operations perform operations on coverages using topological predicates.

We define the coverage operations using an algebraic specification of data types. In our definitions, we use CASL, the Common Algebraic Specification Language [16]. CASL is a general-purpose language for both requirements and design specifications. A basic specification in CASL consists of a set of declarations of symbols, and a set of axioms and constraints, which restrict the interpretations of the declared symbols. For a detailed syntax for CASL specifications, see [16]. We provide examples from the CASL user manual [16] to allow the reader to better follow our proposal. The first example has a unique sort and a predicate:

spec STRICT_PARTIAL_ORDER =

sort *Elem*

pred __ < __: *Elem* × *Elem*

axioms

forall x,y,z: Elem

- $x \leq x$ *%(reflexive)%*
- $x=y$ *if* $x \leq y \wedge y \leq x$ *%(antisymmetric)%*
- $x \leq z$ *if* $x \leq y \wedge y \leq z$ *%(transitive)%*

The STRICT_PARTIAL_ORDER specification uses a sort *Elem* and a binary infix predicate symbol '<'. Argument sorts are separated by the sign '×'. CASL uses '__ ' (pairs of underscores) as place-holders for arguments. The predicate '<' is associated to three axioms: reflectivity, antisymmetry and transitiveness. CASL provides the keyword **type** to shorten declarations of sorts and constructors, as in the example:

spec CONTAINER [**sort** *Elem*] =

type *Container::= empty | insert (Elem: Container)*

pred *__inside __: Elem × Container*

axioms

$\forall e, e' : Elem; C: Container$

- $\neg(e \text{ inside } empty)$
- $e \text{ inside } (insert\ (e', C)) \Leftrightarrow (e = e' \vee e \text{ inside } C)$

end

3.1 The Coverage Data Type

This section presents an algebraic specification for an extended version of Open GIS coverage. A coverage is a *c_function:: G→ V* over a finite collection of geometries G and a set of attribute values V. Without loss of generality, we will discuss the case of coverages where each geometry has only one value. The generalization to a coverage that returns a vector of values is simple, but would need a slightly more complicated notation. We will assume that *Geometry* and *Value* sorts are those used by OGC. The reader should refer to [1] for details. We begin by defining a set of auxiliary specifications: *list* (a list of elements), *single and multiargument functions*, *comparison predicates* and *topological predicates*. For brevity's sake, we provide a limited list of single and multiargument functions and of selection predicates. We can extend these lists of functions if needed.

spec SINGLEFUNCTION =

sort *Value*

ops

log:	$Value \rightarrow Value$
exp:	$Value \rightarrow Value$
sin:	$Value \rightarrow Value$
sqrt:	$Value \rightarrow Value$

end

spec MULTIFUNCTION =
sort *Value*
ops
sum: *Value* × *Value* → *Value*
product: *Value* × *Value* → *Value*
maximum: *Value* × *Value* → *Value*
mean: *Value* × *Value* → *Value*
minimum: *Value* × *Value* → *Value*
end

spec LIST [**sort** *Elem*]
type *List::=* *empty* | *cons (Elem*; *List*)
ops
length: List → *Integer*
end

spec COMPPRED =
sort *Value*
pred
__ **==** *__:* *Value* × *Value*
__ **<** *__ :* *Value* × *Value*
__ **>** *__ :* *Value* × *Value*
__ **!=** *__ :* *Value* × *Value*
end

spec TOPOPRED =
sort *Geometry*
pred
__within__: *Geometry* × *Geometry*
__overlap__: *Geometry* × *Geometry*
__disjoint__: *Geometry* × *Geometry*
__equal __: *Geometry* × *Geometry*
__touch__: *Geometry* × *Geometry*
__contains__: *Geometry* × *Geometry*
__cross__: *Geometry* × *Geometry*
__intersects__: *Geometry* × *Geometry*
end

Using these definitions, we provide an abstract specification the *Coverage* data type. It uses the operations defined by OGC (see Figure 1 above) and includes three constructors, a new predicate and a new operation.

spec COVERAGE [**sort** *Geometry,* **sort** *Value,* **sort** *List*] =
type *Coverage::=* *empty* |
new (List [(Geometry, Value)])
subset (Coverage, List [Geometry])

ops

insert:	*Coverage* × *(Geometry, Value)* → *Coverage*
evaluate:	*Coverage* × *Geometry* → *Value*
domain:	*Coverage* → *List [Geometry]*
num:	*Coverage* → *Integer*
values:	*Coverage* → *List [Value]*
pred	
__contains __:	*Coverage* × *Geometry*

axioms

∀*g, g'*: *Geometry*; *v: Value; C: Container*

- *contains (C, g)* ⇔ *g* ∈ *domain (C)*
- *evaluate (C, g) == error* ⇔ *contains (C, g) == false*
- *evaluate (insert (C, (g, v)), g) = v*
- *v = evaluate (C, g)* ⇔ *v* ∈ *values (C)*
- *num (C) = length (domain (C))*
- *values (C) = List [evaluate (C, g),* ∀ *g* ∈ *domain (C)]*
- *subset (C, List[g])* ⇔ *(*C_1 *= empty ())*∧ *insert (*C_1*,(g, v)),* ∀ *g* ∈ *List [g]* ∧ *v = evaluate (C, g)*

end

The first constructor (*empty*) builds an empty *Coverage*. The second constructor (*new*) builds a new *Coverage* by providing a list of *(Geometry, Value)* pairs. The third constructor (*subset*) builds a new *Coverage* by extracting a subset of the original locations. The predicate *contains* verifies if a certain instance of *Geometry* is in the *Coverage*. *Insert* includes a new *(geometry, value)* pair in the coverage. *Evaluate* takes a coverage and a geometry and produces an output value ("*give me the value of the coverage at location g*"). *Domain* returns the geometries of the coverage's domain. *Num* returns the number of geometries on the coverage's domain. *Values* returns the list of values of the coverage's range. The axioms point to the various restrictions on the *Coverage* specification. The last axiom shows that creating a *Coverage* from a subset of locations of an existing one is the same as building an empty *Coverage* and then inserting a list of *(Geometry, Value)* pairs.

3.2 Coverage operations

Operations on coverages are operations that produce a new coverage, and include nonspatial and spatial ones. For *nonspatial operations*, the value of a location in the output map depends on the values of the same location in one or more input maps. They include logical expressions such as "*classify as high-risk all areas without vegetation with slope greater than 15%" and "find the average of deforestation in the last two years"*. We consider three types of nonspatial operations: *single* (single argument

operations), *multiple* (multiple argument operations) and *select* (nonspatial selection using a comparison predicate).

Spatial operations are higher-order functions that use a topological predicate and generalize Tomlin's focal and zonal operations. They include expressions such as *"calculate the local mean of the map values"* and *"given a map of cities and a digital terrain model, calculate the mean altitude for each city"*. These operations take two coverages (the *reference coverage* and the *input coverage*) and produce a new coverage, which has the same geometries as the reference coverage. A spatial operation has two parts: *spatial query* and *composition*. The *spatial query* operation starts by finding the matching geometry in the *reference* coverage for each location p in the new coverage. Then it applies a topological predicate between that geometry and all geometries of the *input* coverage. The result of the spatial query is a new (temporary) coverage containing the input geometries that match the predicate. The *composition operation* then applies a function to the values on the new coverage to produce the result (Figure 4).

Take the expression *"given a coverage of cities and a coverage of altitudes, calculate the mean altitude for each city"*. In this expression, the *input coverage* is the altitude one and the *reference coverage* is one with cities. To evaluate the expression, we first select the terrain values within each city. This uses the selection operator with the *within* predicate. Then, we calculate the average of each set of these values. This uses the composition operator with the *mean* function.

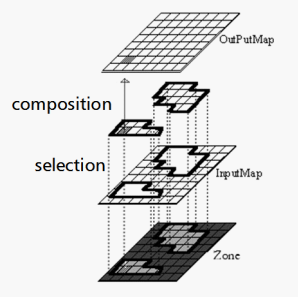


Fig. 4. Spatial operations (selection + composition). Adapted from [4]

To provide the abstract specification for the coverage operations, we distinguish between nonspatial and spatial operations. For spatial operations, we define two auxiliary functions (*query* and *compose*). We use the CASL the **'local … within'** construct for such needs. This construct distinguishes between operations which are visible outside the specification and auxiliary functions.

spec COVERAGE_OPERATIONS

[**sort** *Geometry*, **sort** *Value*, sort *Coverage*, **sort** *SingleFunction*, sort *MultiFunction,* **sort** *List,* **sort** *CompPred* **sort** *TopoPred*]

ops

single: $SingleFunction \times Coverage \rightarrow Coverage$

multiple: $MultiFunction \times List[Coverage] \rightarrow Coverage$

select: $Coverage \times CompPred \times Value \rightarrow Coverage$

local ops

sp_query: $Coverage \times TopoPred \times Geometry \rightarrow Coverage$

compose : $MultiFunction \times Coverage \rightarrow Integer$

within op

spatial: $MultiFunction \times Coverage \times TopoPred \times Coverage \rightarrow Coverage$

axioms

$\forall g, g_1$: *Geometry*; v, v_1: *Value*; C, C_1, C_2: *Container*, *topo: TopoPred, comp: CompPred, fs: SingleFunction, fm: MultiFunction*

- $single\ (fs, C_{)} = new\ (List[g, v]),$
  $\forall (g, v) \mid g \in domain\ (C) \ \wedge\ v = fs\ (evaluate\ (C, g))$
- $multiple\ (fm, C_1, C_{2, \ldots,} C_n) = new\ (List[g, v]),$
  $\forall (g, v) \mid g \in (domain\ (C_1) \cap domain\ (C_2) \ldots) \cap domain\ (C_n)) \wedge$
  $v = fm\ (evaluate\ (C_1, g), evaluate\ (C_2, g)))$
- $select\ (C, comp, v_1) = subset\ (C, S),\ where$
  $S = List\ [g \mid g \in domain\ (C) \wedge comp\ (evaluate\ (C, g), v_1)]$
- $sp\_query\ (C_{,} topo, g_1) = subset\ (C, S),\ where$
  $S = List\ [g \mid g \in domain\ (C) \wedge topo\ (g, g_1)]$
- $compose\ (fm, C) = fm\ (values\ (C))$
- $spatial\ (f, C_1, topo, C_{2)} = new\ (List[g, v]),$
  $\forall (g,v) \mid g \in domain\ (C_{2)} \wedge v = compose\ (fm, sp\_query(C_1, topo, g))$

end

The first axiom describes the *single* operation, which applies a function to all values of the input. The second axiom describes the *multiple* operation, which applies a multivalued function to all values of the input. The third axiom describes the nonspatial selection operation. The fourth axiom shows that the *spatial query* selects a subset of the original coverage whose geometries satisfy a topological predicate ("*select all deforested areas within the state of Amazonas*"). The fifth axiom describes the *composition* operation. The last axiom describes the *spatial* function. It builds a new coverage by providing a list of *(Geometry, Value)* pairs. The geometries of the *new coverage* are the same as those of the *reference coverage*. We get the values of the new coverage by combining a *spatial query* and a *composition*. In the next section, we show how these operations for coverages are enough for a comprehensive algebra.

3.3 Expressiveness of Topological Operators for Coverage Algebra

All coverage subtypes proposed by OGC are implemented as discrete geometrical spatial data structures. Thus it is important to consider what extent the topological operators can cover the spatial relations between these coverage subtypes. As discussed above, spatial operations have two parts: a spatial query and a composition on the values selected by the query. The expressive power of the spatial query is limited by the capacity of computing them it in each spatial data structures. Based on OGC's proposal for coverage subtypes [1], we can distinguish different types of discrete data structures for coverages:

- *2,5D structures:* TIN coverages.
- *2D Area-based structures:* grid coverages, images, polygon coverages, surfaces.
- *1D Line-based structures*: segmented line coverages, line string coverages.
- *0D point-based structures*: discrete point coverages, nearest neighbour coverages, lost area interpolation.

Since the dimension-extended version of the 9-intersection model only handles 2D, 1D and 0D data structures, the proposed coverage algebra cannot be used for TIN coverages. Also, consider that a spatial query operation involves two coverages: the reference and the input coverages (see Section 3.2). Thus, application of topological operators depends on the geometries of the reference and the input coverages, as outlined in Table 1.

Table 1. Topological operators applicable to spatial operations on coverages

Input coverage type	**Reference coverage type**	**Operators**
area	area	*{'disjoint', 'equal', 'touch', 'within', 'overlap', 'contains', 'intersects'}*
area	line	*{'disjoint', 'touch', 'intersects', 'contains'}*
area	point	*{'disjoint', 'touch', 'contains'}*
line	area	*{'disjoint', 'cross', 'touch', 'within', 'intersects'}*
line	line	*{'disjoint', 'equal', 'touch', 'within', 'overlap', 'intersects'}*
line	point	*{'disjoint', 'touch', 'contains'}*
point	area	*{'disjoint', 'touch', 'within'}*
point	line	*{'disjoint', 'touch', 'within'}*
point	point	*{'disjoint', 'equal'}*

3.4 Examples of Coverage Algebra

To provide examples of the proposed coverage algebra, we propose a shorthand notation for the operations, as follows:

Table 2. Convenience shorthand for non-spatial operators

```
new_cov:= singlefun in_cov;   % single value functions%
new_cov:= multifun [in_cov]; % multivalued functions%
new_cov:= in_cov comp_pred value; % selection%
new_cov:= multifun in_cov topo_pred ref_cov;
%for spatial operations%
```

The parameters for the operations are:

- `new_cov` is the coverage with the new values.
- `singlefun` is a single argument function as given in section 3.1.
- `multifun` is a multiargument function as given in section 3.1.
- `in_cov` is the input coverage.
- `[in_cov]` is a list of input coverages.
- `comp_pred` is a comparison predicate.
- `value` is the comparison value.
- `topo_pred` is a topological predicate *(section 3.1)*
- `ref_cov` is the reference coverage used as a basis for applying the topological predicate.

The examples of the Table 3 show the use of non-spatial operators applied to coverages. Table 4 shows examples of spatial operations.

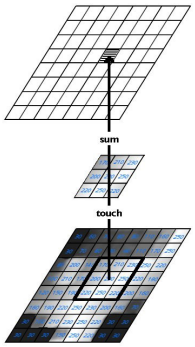
Table 3. – Examples of non-spatial operators

Informal description	***Coverage Algebra Expression***
"Find the square root of the topography"	`topoSqrt:= sqrt topography;`
"Find the square root of the cities' population"	`popSqrt := sqrt  cityPop;`
"Find the average of deforestation in the last two years"	`defAve := mean (defor2004, defor2003);`
"Select areas higher than 500 meters"	`highM := topography > 500;`
"Select the cities with the population higher than 50.000"	`highPop := cityPop > 50000;`

Table 4. Examples of Spatial Operations

Informal description	***Coverage Algebra Expression***
"Given a coverage of cities and one of states, find the total population for each state"	`statePop := sum cityPop within statePop;`
"For each cell, calculate the average deforestation of its neighbours"	`fsum := sum defor touch fsum;`
"Given a coverage of rivers and one of cities with population, find the number of people that live along each river".	`riversPop:= sum cityPop intersects rivers;`
"Given a coverage of cities and one of deforestation, find the average deforestation of each city".	`ave_city_defor:= average defor within city;`
"Given a coverage containing roads and a one of deforestation, calculate the mean deforestation along each road".	`defRoads:= mean defor intersects roads;`

Consider the operation: *"For each cell, calculate the average deforestation of its neighbours*". In Tomlin's algebra, this is a focal operation (Figure4.b). Considering the deforestation map (`defor`) as input and the local sum map (`lsum`) as output, we state the operation as shown in Figure 5.




```
fsum := sum defor
touch fsum;
```

Fig. 5. Focal sum of deforestation

Note one interesting feature: the result (`fsum`) is also the reference coverage for the spatial predicate (`defor touch fsum).` This syntax may seem odd at first sight, but it follows from the generality of the proposal. By taking the reference map and the new map to be the same, we ensure the outcome satisfies the condition ("local mean"). Had we used a third map as a reference, the result would be different if this map would not have the same spatial partitioning as the output map.

4 Examples and comparison with Tomlin's map algebra

In this section, we give some examples and compare our proposal to Tomlin's map algebra. The examples use INPE's database of deforestation of the Brazilian Amazonia. We selected a data set from the central area of the Pará state, composed of three coverages: deforestation (grid cells of 25 x 25 km^2), roads (lines), and protected areas (polygons), as shown in Figure 6.

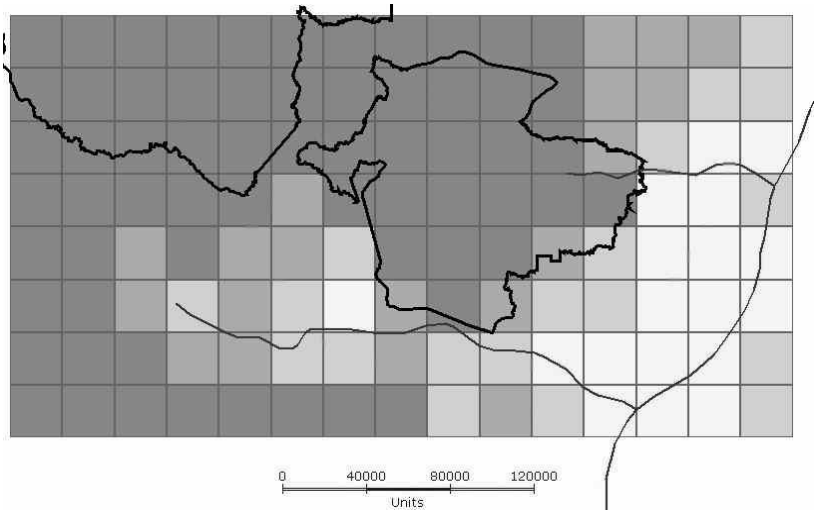


Fig. 6. Deforestation, protected areas and roads (Pará State, Brazil)

Our first example considers the expression: "*Given a coverage of deforestation and a classification function, return the classified coverage*". The inputs is the deforestation coverage (`def_cov`) and the output is a classified coverage (`def_cov`) The classification function defines four classes: (1) dense forest; (2) mixed forest with agriculture; (3) agriculture with forest fragments; (4) agricultural area. This function is:

```
classify :: Value → Value
classify v
     | v < 0.2                    = "1"
     | ((v > 0.2) && (v < 0.5))   = "2"
     | (v > 0.5) && (v < 0.8)     = "3"
     | v > 0.8                    = "4"
```

We get the classified map (Figure 7) using the expression

```
def_class = classify def_cov
```

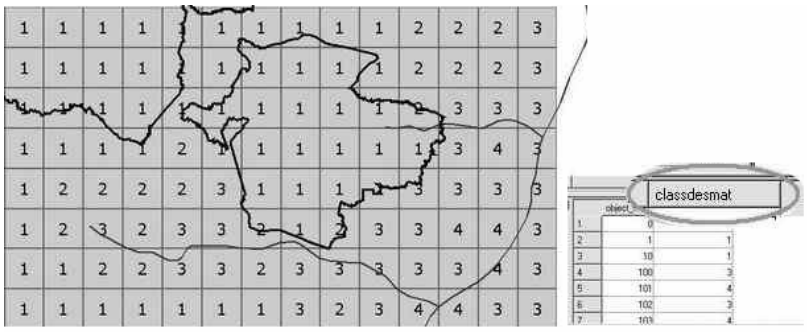


Fig. 7. Resulting coverage with classified deforestation

As a second example, take the expression: *"Calculate the mean deforestation for each protection area"*. The inputs are: the deforestation coverage (`def_cov`), a spatial predicate (`within`), a multivalued function (`mean`) and the coverage of protected areas (`prot_areas`). The output is a coverage of the protected areas (`def_prot`) with the same objects as the reference coverage (`prot_areas`) and the deforestation for each area .

```
def_prot = mean def_cov within prot_areas
```

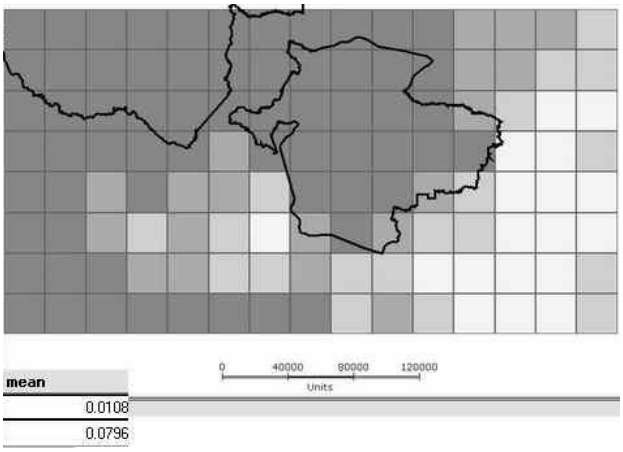


Fig. 8. Deforestation mean by protection area

Our third example is the expression: "*Given a coverage containing roads and one with deforestation, calculate the mean of the deforestation along each roads*". We have as inputs: the deforestation coverage (`def_cov`), a spatial predicate (`intersect`), a multivalued function (`mean`) and a road map (`roads`). The product is a roads coverage with one value for each road. This value is the mean of the cells that intercept this road.

```
road_def =  mean def_cov intersect roads
```

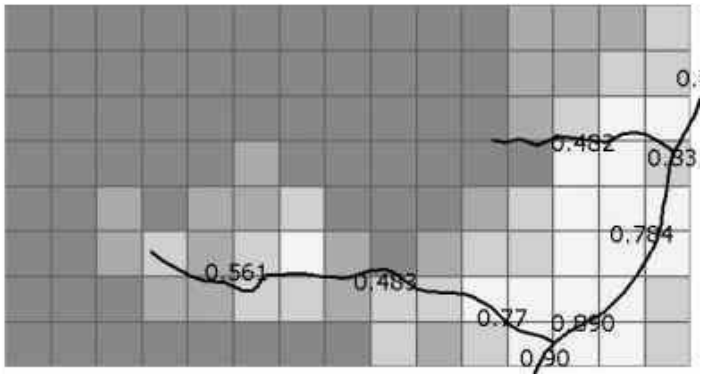


Fig. 9. Deforestation mean along the roads

Table 4 presents a comparison between the spatial operators as expressed in our proposal and in Tomlin's map algebra. The examples show that the proposed map algebra expresses the focal and zonal functions of Tomlin's map algebra, using the *'touch'* and *'within'* topological predicates. Operations involving '*overlap', 'contains'* and *'intersects'* predicates are part of our proposed coverage algebra and are not directly expressible by Tomlin's algebra. This shows that the proposed algebra is richer than Tomlin's, as well as having a solid conceptual basis.

Table 5. Comparison of spatial operators with Tomlin's map algebra

Informal Description	*Generalized Map Algebra*	*Tomlin*
"Focal mean of topography"	`fmean:= mean topo` **`touch`** `fmean`	`FOCALMEAN OF TOPOGRAPHY`
"Given a coverage of cities and one of topography, find the mean altitude for each city."	`altcit:= mean topo` **`within`** `city`	`ZONALMEAN OF TOPOGRAPHY WITHIN CITIES`
"Given a coverage of national forests, get the deforestation at the edges of each forest"	`defBord:= sum defor` **`overlaps`** `forests`	**(no equivalent)**
"Calculate the mean of the deforestation along the road"	`defRoad:= mean defor` **`intersects`** `road`	**(no equivalent)**

5 Conclusions

Map algebra is a fundamental class of operations for spatial data sets. Most of the current implementations of map algebra use Tomlin's [4] proposal for local, focal and zonal operations. However, Tomlin's proposal uses ad hoc concepts and lacks a sound theoretical basis. This work addresses this problem, by proposing a new foundation for operations involving coverages. We have designed a coverage algebra that uses topological predicates to express spatial operations and that includes Tomlin's algebra as a subset.

There is one important set of operations on coverages that is not part of our proposal nor of Tomlin's: *convolution operators.* A convolution operation requires two coverages C_1 and C_2 produces a third coverage C_3. The value of each point p of C_3 is the integral of the product of C_1 and C_2, when C_2 is shifted so that its central point is coincident with p. From a conceptual point of view, convolutions are not part of map algebra, since the geometrical support for the second coverage C_2 (also called a *mask*) changes for each point of the output coverage. Convolution does not involve topological relations, but rather the definition of an integral function.

Our proposal points to a situation where all modeling of topological relations in two-dimensional spatial datasets can be handled by the 9-intersection model (dimension-extended), both for simple features and for coverages. Spatial data sets of higher dimensions (e.g., TIN coverages) need a different foundation. The foundation for handling spatial relation of higher dimensions requires topological operators that operate on 3D surfaces [17]. Convolution operations are a special case and need to be handled separately. A possible extension to our algebra would be to consider directional relations [18], which would be useful to express operations such as *"find the population of all cities north of the river"*.

Even considering these limitations, the expressiveness of the proposed coverage algebra is considerable. Given that it is based on a solid foundation, it could be considered as the basis for setting up standard for operations on Open GIS coverages. The use of standards for operations in coverages would be a significant advance for increased interoperability of spatial data.

Acknowledgments

Gilberto Câmara's work is partially funded by CNPq (grants PQ - 300557/1996-5 and 550250/2005-0) and FAPESP (grant 04/11012-0). Danilo Palomo's and Sérgio Costa's work is funded by CAPES. Gilberto Câmara would like to thank Andrew Frank and Max Egenhofer for many stimulating discussions on the issues covered by this paper.

References

1. Kottman, C., Roswell, C.: The OpenGIS Abstract Specification Topic 6: The Coverage Type and Its Subtypes (Version 4). Open Geospatial Consortium, Wayland, MA (2000)
2. Berry, J.K.: Fundamental Operations in Computer-Assisted Map Analysis. International Journal of Geographical Information Systems **1** (1987) 119–136
3. Frank, A.: Overlay Processing in Spatial Information Systems. In: Chrisman, N.R. (ed.): AUTO-CARTO 8, Eighth International Symposium on Computer-Assisted Cartography, Baltimore, MD (1987) 16-31
4. Tomlin, C.D.: Geographic Information Systems and Cartographic Modeling. Prentice-Hall, Englewood Cliffs, NJ (1990)
5. Takeyama, M., Couclelis, H.: Map Dynamics: Integrating Cellular Automata and GIS through Geo-Algebra. International Journal of Geographical Information Systems **11** (1997) 73-91
6. Pullar, D.: MapScript: A Map Algebra Programming Language Incorporating Neighborhood Analysis. GeoInformatica **5** (2001) 145-163
7. Mennis, J., Viger, R., Tomlin, D.: Cubic Map Algebra Functions for Spatio-Temporal Analysis. Cartography and Geographic Information Science **32** (2005) 17-32
8. Frank, A.: Map Algebra Extended with Functors for Temporal Data. In: Akoka, J. (ed.): Perspectives in Conceptual Modeling: ER 2005 Workshops. Springer, Klagenfurt, Austria (2005) 194-207
9. Egenhofer, M., Herring, J.: Categorizing Binary Topological Relationships Between Regions, Lines, and Points in Geographic Databases. Department of Surveying Engineering, University of Maine, Orono, ME (1991)
10. Câmara, G., Palomo, D., Souza, R.C.M., Oliveira, O.R.F.: Towards a generalized map algebra: principles and data types. In: Fonseca, F. (ed.): VII Brazilian Symposium on Geoinformatics (GEOINFO 2005). SBC, Campos do Jordao, Brazil (2005)
11. Costa, S., Camara, G., Palomo, D.: TerraHS: Integration of Functional Programming and Spatial Databases for GIS Application Development. In: Davis, C. (ed.): VIII Brazilian Symposium in Geoinformatics, GeoInfo 2006. SBC, Campos do Jordão, Brazil (2006)
12. Herring, J.: OpenGIS® Implementation Specification for Geographic information - Simple feature access - Part 1: Common architecture (version 1.2.0). Open Geospatial Consortium, Wayland, MA (2006)
13. Winter, S.: Topological Relations between Discrete Regions. In: Egenhofer, M., Herring, J. (eds.): Advances in Spatial Databases—4th International Symposium, SSD '95, Portland, ME, Vol. 951. Springer-Verlag, Berlin (1995) 310-327
14. Winter, S., Frank, A.: Topology in Raster and Vector Representation. GeoInformatica **4** (2000) 35-65
15. Frank, A.: Higher order functions necessary for spatial theory development. Auto-Carto 13, Vol. 5. ACSM/ASPRS, Seattle, WA (1997) 11-22
16. Bidoit, M., Mosses, P.D.: CASL User Manual. Lecture Notes in Computer Science 2900 (IFIP Series). Springer, Heidelberg (2004)
17. Egenhofer, M.: Spherical Topological Relations. Journal on Data Semantics **III** (2005) 25-49
18. Frank, A.: Qualitative Spatial Reasoning about Distances and Directions in Geographic Space. Journal of Visual Languages and Computing **3** (1992) 343-371