

Algebraic Formalism over Maps

João Pedro Cordeiro¹, Gilberto Câmara¹, Ubirajara F. Moura²,
Cláudio C. Barbosa¹, Felipe Almeida³

¹Divisão de Processamento de Imagens – Instituto Nacional de Pesquisas Espaciais
(DPI – INPE) – São José dos Campos, SP– Brasil

²FUNCATE – Fundação de Ciencia Aplicações e Tecnologias Espaciais, SP, Brasil

³Instituto Tecnológico da Aeronáutica (ITA) – São José dos Campos, SP, Brasil

{[jpedro](mailto:jpedro@dpi.inpe.br),[gilberto](mailto:gilberto@dpi.inpe.br),[claudio](mailto:claudio@dpi.inpe.br)}@dpi.inpe.br, bira@geo.funcate.org.br,
felal@ita.cta.br

***Abstract.** This paper describes features of a language approach for map algebra based on the use of algebraic expressions. To be consistent with formal approaches such as geoalgebra and image algebra, the proposed algebraic expressions are suitable for the usual modeling of layers and to represent neighborhoods and zones. A tight compromise between language and implementation issues based on the theory of automata is proposed as the needed support to define or extend coherently operators and grammar rules. This results in an efficient way of implementing map algebra for raster domains that can simplify its coupling to environmental and dynamic models without going too far from its well-known paradigm.*

1. Introduction

The main contribution towards an algebraic foundation to modeling operations in GIS came from the works of Tomlin and Berry at Yale University in the 1980's (see Tomlin and Berry, 1979; Tomlin, 1983 and Berry, 1987). It resulted in the compiled book "Geographic Information Systems and Cartographic Modeling" (Tomlin, 1990). They stated the foundations of map algebra, thus imposing a formal approach to accommodate modeling situations on spatial domains. Also a language approach in which a model is represented as a sequence of expressions given as textual sentences, or "scripts", that describes the operations and relations among locations and location data, usually represented as map layers. The language reflects features and properties of operations and relations, in a way similar to that of mathematical expressions in almost all branches of science.

Map algebra operations have been traditionally presented as functions of one, two or more variables representing map layers, lookup tables and constants, as well as a lot mathematical functions on such variables. Predicates given by verbs, prepositions and other constructions from the English language help adding meaning to specific parameter use. For example, consider the following map algebra expression from Tomlin's (1990) book:

- `windexposure = localrating of
Altitude and Vegetation`

```
with 0 for 290 ... on 0
With 1 for 290 ... on 1 ... 3
With 2 for ... 289 on 0 1 3
With 3 for ... 289 on 2
```

This expression represents a basic overlay operation to reclassifying locations by clearly assigning values based on criteria involving corresponding local values at different layers. The use of numbers to represent both quantitative and qualitative data such as heights and vegetation cover type may impose some limits to the semantic expressiveness of operations. For instance, in the expression above, it is not clear if the resulting integer values represent thematic values or integer weights. It will depend only on the role the resulting layer (windexposure) will play in a next step in the model.

Running a cartographic (or static) model is a matter of interpreting and parsing expressions based on syntax rules relating function names with their parameters. A sequence of intermediate map layers is usually generated at the model running time, some of which are incorporated to the model.

A model is an abstract and partial representation of some aspects of the world that can help deriving analysis, definitions and possibilities based on acquirable data (see Couclelis, 2000). Environmental models refer to any characteristic of the Earth's environment in a broad sense. Atmospheric, hydrological, biological and ecological systems, natural hazards, and many others are popular themes. As new techniques, computational resources and data, become widely available, the complexity of models also experiences a growing tendency and, of course, GIS technology has played a key role in the whole process.

Coupling GIS with more complex system involved in environment and dynamic modeling has been the object of intensive research. Map algebra plays a special role because of its spatial representation and descriptive characteristics which considers modeling based on cellular automata, adequate for raster GIS (White et al., 1994. Couclelis, 1997, 2000). However, problems arise about the interpretative approach commonly adopted in the implementation and the excess of intermediate data representation generated at model execution time (Dragosits, 1996).

Optimization and the use of efficient algorithms are also important issues in accommodating the coupling problems. Wesseling's PCRaster (1996) is a good example of a successful language and implementation approach integrating GIS and a wide class of physical dynamic modeling applications in which optimization techniques plays an important role.

The growing complexity of modeling would benefit from more formal support to spatial analysis operations. The geoalgebra from Takeyama (1996) is a quite complete formal mathematical basis for extending the cartographic modeling of map algebra to deal with the dynamics of processes. Following close to Ritter's (1990) image algebra, an almost complete algebraic formalism for operations on images, Takeyama generalizes the idea of map to that of a function from a spatial domain to a given attribute domains. Allowing such functions to range over sets of functions instead of sets of attribute values, a more general "map" structure is derived that can help modeling the influence of sets of locations over locations. The interactions among maps

and such generalized map structures can model all classes of map algebra operations under a common framework.

This work proposes a revision of map algebra at the light of an algebraic structuring consistent with geocalgebra based essentially on introducing a binary operator to model the interaction of Boolean, with any other data usually represented on map layers. A new data type is also introduced in association to the idea of region, based on which both the concepts of zone and neighborhood of Tomlin's classes of nonlocal operations can be fully modeled.

We start by equating the ideas of zones and neighborhoods to logical or Boolean comparisons based on relations such as order, equality, proximity, accessibility and many others, that can be defined on the attribute or spatial domains of maps. For instance, the expression used earlier to illustrate Tomlin's map algebra operations uses comparisons based on order and equality relations. Putting those relations more clearly the same expression can be rewritten as follows:

- `windexposure =`
 - `0 : Alt >= 290 and Veg == 0`
 - `1 : Alt >= 290 and Veg != 0`
 - `2 : Alt < 290 and Veg != 2`
 - `3 : Alt < 290 and Veg == 2 ;`

The way comparisons are expressed here follows from grammar rules close to those of ordinary algebraic expressions, so that interpreting and parsing strategies can be easily derived. This work suggests that by adopting more formal compromise between languages and automata theories (Hopcroft, et al., 1969), regarding syntax and implementation, may avoid some problems that would demand for optimization in a traditional approach.

The notions of operation and expression are discussed in the initial sections, as well as their extensions to geo-spatial domains. The concept of region as algebraic expression is introduced in Section-2 and used in sections 3 and 4 as the basic paradigm to defining operations arguments; Section-5 is about summarizing values from specified regions. The consistency between concepts from the adopted approach and those from geo-algebra formalisms is evaluated in Section-6. An implementation strategy for map algebra based on languages and automata theories is pointed out in Section-7 as a natural transition from static to dynamic modeling functionality. As concluding remarks, some performance issues that may benefit from this approach are pointed out regarding its use with distributed and parallel architectures.

Expressions used as examples in this text will be based on the language LEGAL (Algebraic Geoprocessing Language). LEGAL is a map algebra implementation based on the Spring GIS data model (Camara et al., 1994) (Camara et al., 1996) (Cordeiro et al., 1996) that already follows some of the principles discussed in this paper. Spring GIS software is available free at www.dpi.inpe.br/spring. For most of the examples, partial expressions or subexpressions are important; only those expressions closed by the sign ';' (semicolon) may be considered complete regarding LEGAL syntax. Also new language forms are being introduced or suggested as extensions from the original syntax of LEGAL.

2. Extending Algebra to Maps

A map can be defined as a function on a spatial domain restriction usually referred to as study area, taking a specific set of values of qualitative or quantitative nature as attribute domain. Locations in a map consist of sets of georeferenced, elementary cells of fixed resolution whose union makes up a primary partitioning of the study area. From the algebraic structures available on these spatial and attribute domains much algebraic structuring can be stemmed from. Operations and relations are defined on maps based on operators, functions and relations, already defined on both the spatial and the attribute domains.

By the language side, expressions describing operations will involve symbols and names, for operators, functions, variables, constants etc. Also properties and priorities must be reflected by expressions in the same way ordinary mathematical expressions do. The language should stimulate the use of direct expressions to representing data whenever possible, instead of physically creating partial results in a model. Types should be associated not only to variables representing layers, but also to the expressions describing operations that could eventually be used to generate a new layer. Only existing map layers and meaningful result layers should need to be represented as variables.

Mathematical operations and functions are induced by locally applying their one-dimensional versions defined on quantitative attribute domains associated to the locations in a study area. For instance, consider the idea of “vegetation index” defined by the normalized differences of radiometric values from two different bands of a multispectral image. It can be described as follows:

- $(b4 - b3)/(b4 + b3)$

In above expression the variables b3 and b4 represent the image bands involved.

Relations such as order and equality can also be extended to spatial domains in a similar way by comparing local data through relations already defined on the attribute domain of maps. Expressions describing relations can be identified to the sets of locations satisfying the relations as showed by expressions below:

- `veg == "forest"`
- `slope >= 30`
- $(b4-b3)/(b4+b3) > 0.5$

The first describes the set of locations with “forest” coverage based on a map layer associated to the variable “veg”. The second describes the set of locations at not less than 30% slope based on a grid layer represented by variable “slope”. The last one represents the set of locations with vegetation indexes higher than 0.5, based on direct evaluating the indexes at each location before comparing

Results from locally evaluating order and equality relations such as ‘<’, ‘>’, ‘<=’, ‘>=’, ‘==’, ‘!=’, can be identified to binary values such as ‘true’ or ‘false’, ‘0’ or ‘1’. Therefore, Boolean algebra can be naturally extended to map domains by inducing operators like ‘and’, ‘or’ and ‘not’ from their original versions.

In this work the expressiveness of combining comparisons by Boolean expressions is the basis for describing sets of locations in the study area to be involved in operations, considering language. For example, the three Boolean expressions in the previous example can be combined into a single one as follows:

- `veg == "forest" and
(b4-b3)/(b4+b3) > 0.5 or
slope >= 30`

To explore the interactions among Boolean and any other type, a binary operator is now introduced, such that at least one of its arguments is of type Boolean, while the other (and so the result) may assume any valid type as stated (informally) by the table below:

*	Value	Null
True	Value	Null
False	Null	Null

The term ‘Value’ above represents an arbitrary value of some specified data type, while ‘Null’ is associated to the absence of data. The symbol ‘*’ is adopted here to represent the operator itself. This interacting operator corresponds to the selection of a set of locations and their corresponding associated values at possibly different layers, so we adopt the term ‘selecting’ to referring to it by now.

For image processing purposes this operator can be induced from number multiplication, provided the integers ‘1’ and ‘0’ plays the role of ‘true’ and ‘false’. Applying the selecting operator in this case will return a new image in which some locations keep the original image values, while the others become 0-valued. For example:

- `(ndvi > 0.5) * img`

The evaluation of this expression will result in selecting all values from the image layer associated to the variable “img”, at locations with vegetation index values, given by the grid layer represented by variable “ndvi”, that are greater than 0.5. Remaining locations become ‘0’ valued.

With images the notion of a ‘null’ value can be well represented by the integer ‘0’, but it is not always the case. For instance, consider the expression:

- `(ndvi > 0.5) * slope`

Here the notion of ‘null’ value cannot be represented locally by the integer ‘0’ because this is a meaningful value for a numeric grid layer representing slopes. Besides, it would be also desirable to have the selecting operator working for qualitative data as well, so that one could write expressions like:

- `(ndvi > 0.5) * soils`

In above expression, variable “soils” represents a soil types thematic (qualitative) layer.

Selecting operator has properties similar to those of ordinary multiplication so one can adopt the same symbol '*' to represent both operators. Different interpretations are implied for the same symbol but without any syntactic or semantic ambiguity, as showed in example bellow.

- (veg == "crop" AND slope < 30) * heights * distances
- heights * (veg == "crop" AND slope < 30) * distances
- heights * distances * (veg == "crop" AND slope < 30)

Each equivalent expression above involves two occurrences of the symbol '*', one for selecting and the other for usual multiplication.

At the implementation level, interpreting expressions involving the selecting operator follows the rules of a context-free grammar similar to those for arithmetic and Boolean expressions. Then, parsing and execution can follow the same pushdown automata strategy (Hopcroft et al., 1969) used for other algebraic expressions in map algebra.

By now, the discussion has concerned essentially the class of local operations of Tomlin's taxonomy. The other classes of zonal, focal and incremental operations, can model situations to which locations in the study area are characterized by sets of influencing locations whose associated attribute values must considered while settling new values to it. Selecting and evaluating such influencing sets is needed before summarizing single values to characterize each location in the study area.

In implementing Tomlin's classes of nonlocal operations three basic steps are essentially of concern:

1. sets of locations are selected
2. sets of values at selected locations are recorded;
3. values are summarized for each set recorded above.

The selecting operator just defined can be used to model step-2 above. It also can be implemented as a local operator so that all other classes of map algebra operations can be founded on the same framework used for local operations. The recording of values at selected locations is usually driven by the characteristics of the summary functions to be applied on then.

3. Regions and zones

The term "region" will be adopted here as a generalization for sets of locations. The type "Region" can be introduced at this point as a synonym for Boolean, just to ease the task of describing regions at the language level, as in the following example:

- Region reg : veg == "forest" AND slope < 30 ;

The variable "reg" above describes a single region obtained by intercepting two regions described by equality and order relations. Although the involved variables "veg" and "slope" above may be associated to map layers, the resulting definition for "reg" will never need to be associated to such physical representations; it is only a description for a set.

Usually sets of regions instead of a single one are involved in operations so introducing a type for “sets of regions” is suggestive at this point to characterize the sets of expressions describing regions at the language level. The type “Regions” is then adopted here, to characterize lists of comma separated regions given either by Boolean expressions or by already defined variables representing regions, as illustrated by the following example:

- Regions regs:


```
reg,
veg == "crop" AND distr == "d1",
height > 1000 OR rain == "low",
(b4-b3)/(b4+b3) > 0.5 ;
```

The interaction between regions and location data can be naturally induced from the selection operator defined in Section-2 so that one can write expressions like:

- reg*ndvi
- reg*(soils == "pdz")
- regs*(b4-b3)/(b4+b3)
- (distance(<)<3)*img

If a set of regions consists of a collection of disjoint regions, whose union (possibly implicitly including a background region) is the whole study area, the term “zone” can replace “region”. A type “Zones” can then be derived from the type “Regions” just to emphasize the nonoverlapping criteria.

Zones are used to aggregating common properties of its locations based on relations defined both on the spatial and/or the attribute domains represented in maps. Concepts such as “states in a country”, “soil types”, “parcels”, “ranges of height”, “buffers” etc, can be modeled as zones. Simplified syntax rules can be introduced so that repetitive lists of Boolean expressions can be avoided. For example:

- Zones districts : distr == "d1", "d2", "d3";
- Zones vegetation : veg == All;

Also distance and direction measures can be applied in zone specifications as in example bellow:

- Zones buffers : distance (rivers=="main")


```
<= 10,
> 10 and <= 20,
> 20 and <= 30,
> 30;
```

The reference region to which distance must be evaluated and ranked is given by a Boolean expression describing a class called “main”, represented in a layer associated to the variable rivers.

As for regions in general, Boolean algebra can also be used to refine a zonal partitioning, for instance the zones defined by each vegetation type contained in each district can be expressed as:

- districts AND vegetation

The 'selecting' operator can also work well with zones, so that the values at locations belonging to each zone can be unambiguously selected from data described by any other expression type. For example:

- $(b4 - b3)/(b4 + b3) * buffers$

There is nothing special to say about zones that wasn't said before for general regions, however, in practice, except for neighborhood analysis operations, there is no useful applications for regions that do not constitute a set of zones in cartographic modeling. However in the former case a region is intended to characterize a single location, instead of each location, inside the region, as is the case for zones.

4. Regions and Neighborhoods

The notion of neighborhood is usually applied to characterize specified reference (or focus) locations, by the influence exerted on them by specified sets of locations grouped together based on some proximity notion induced from relations and functions involving the spatial domain. In this paper combining relations on the attribute domains of maps with such proximity notions is proposed as a way to modeling spatial variability for neighborhoods.

A lot of proximity relations on the spatial domain of maps can be described by comparing measures of distance and direction implemented as functions associating pairs of locations to positive number values. Comparing such measurements through relations defined on numbers can also help defining neighborhood regions. For example:

- Neighborhood close_to_forest : $distance() < 3$ and $veg == "forest"$
- Neighborhood up_and_right : $distance() < 3$ and $direction() < 90$;

The type "Neighborhoods" is introduced here as another specialization for the type Regions. First variable describes any set of locations with forest coverage in a circular vicinity of radius 3 units around any specified focus location in the study area. The second one describes any set of locations in a sector of 90 degrees and not more than 9 units close to a specified focus. As another example consider the set of regions defined by each location and its eight, adjacent neighboring locations. In this case the adjacency relation plays the role of a proximity criterion among locations.

A natural way to involve neighboring locations in operations consists of explicitly referring them in expressions as illustrated bellow:

- $(img[-1,-1]+img[-1,0]+img[-1,1] +img[0,-1]+img[0,0]+img[0,1] +img[1,-1]+img[1,0]+img[1,1])/9$

Above expression describes the averaging of values associated to adjacent neighboring locations from the image layer represented by the variable "img". Specific neighboring locations are referred by pairs of integer numbers indicating the displacement, in terms of shifted lines (above or bellow) and columns (to left or to

right), relatively to the “focus” location associated to the pair $[0,0]$. It is then suggestive to also adopt this shifting mechanism to specifying neighborhoods. For instance, the family of regions involved in the averaging operation can be specified as follows:

- Neighborhoods ngh8 :
 $[-1,-1], [-1,0], [-1,1],$
 $[0,-1], [0,0], [0,1],$
 $[1,-1], [1,0], [1,1];$

A region satisfying such specification is essentially a function from the study area to the binary set $\{T, F\}$, that associates the focus and its neighboring locations to the value ‘T’ (true), while all remaining locations are valued ‘F’ (false). The whole set of regions implied in such specification can also be defined as a function from the study area to the set of all functions defining specific regions (see Section-6). Of course, only relative locations valued ‘T’ needs to be indicated.

For convenience, the binary value associated to a shifting pair in a neighborhoods specification may sometimes be given explicitly as a third parameter representing the location selection state, as illustrated by the equivalent specifications given below:

- Neighborhoods plus :
 $[-1,0],$
 $[0,-1], [0,0], [0,1],$
 $[1,0];$
- Neighborhoods plus :
 $[-1,0,T],$
 $[0,-1,T], [0,0,T], [0,1,T],$
 $[1,0,T];$
- Neighborhoods plus :
 $[-1,-1,F], [-1,0,T], [-1,1,F],$
 $[0,-1,T], [0,0,T], [0,1,T],$
 $[1,-1,F], [1,0,T], [1,1,F];$

Actually selecting a set of locations in the vicinity of a focus is a matter of computing the actual coordinates of neighboring locations based on those of the focus location.

Boolean operations can be naturally extended to all neighborhood specifications so that one can operate with neighborhoods variables to obtain new specifications. For example, given the expression below:

- Neighborhoods times :
 $[-1,-1], [-1,1],$
 $[0,0],$
 $[1,-1], [1,1];$

Then one can redefine variable ngh8 as:

- $\text{ngh8} = \text{plus OR times} ;$

In fact any regions specification, can be operated this way to result new variables of Neighborhoods type, as illustrated bellow:

- `ngh = plus AND (slope < 30);`

As the values ‘T’ and ‘F’ are themselves constant Boolean expressions, they can be replaced by any Boolean expression in explicit neighborhoods specifications, so that the variable “ngh” in previous example could otherwise be obtained by the following explicit specification:

- `Neighborhoods ngh :`
`[-1,0, slope < 30],`
`[0,-1, slope < 30],`
`[0, 0, slope < 30],`
`[0, 1, slope < 30],`
`[1, 0, slope < 30];`

Also different Boolean expressions (or regions) can be associated to each relative location of a neighborhoods specification, for example:

- `Neighborhoods mess :`
`[-1,0, slope<30],`
`[0,-1, ndvi<0 AND slope<40],`
`[0, 0, veg=="forest"],`
`[0, 1, soil==pdz],`
`[1, 0, slope<30 OR soil=="sand"];`

Selecting values at neighboring locations is modeled through the same ‘selecting’ operator already defined for regions, as shown bellow:

- `plus * ((b4 - b3)/(b4 + b3))`

Above expression describes the selection of vegetation index values associated to locations at north-south and east-west adjacent directions relatively to each focus location in the study area.

Besides the three steps of any operation involving regions discussed in Section-3, there is an additional one regarding the “importance” to which a selected location value must be considered for recording purposes. To illustrate consider the expression bellow:

- `Sqrt (`
`((im[1,-1]+2*im[1,0]+im[1,1])`
`-(im[-1,-1]+2*im[-1,0]+im[-1,1]))^2`
`+`
`((im[-1,1]+2*im[0,1]+im[1,1])`
`-(im[-1,-1]+2*im[0,-1]+im[1,-1]))^2`
`);`

It describes a gradient or Sobel filtering operation of common use in image processing for edge enhancement, given by explicitly referring the relative neighboring locations involved. Here factors of ‘2’ indicate double weighting of values at specific locations mapped by the relative coordinate pairs [1, 0], [-1, 0], [0, 1] and [0,-1]. This suggests extending the region concept so that the modeling of such “weights” can be

incorporated to the selecting process. In this context a value '0' can be associated to each relative location exerting no influence in an operation, while any other values will indicate positive or negative "multiplicity" to which the location must be counted.

The 'selecting' concept can thus be extended to also support the 'weighting' concept, provided the selecting product table of Section-2 is adjusted as follows:

*	Value	Null
w	w copies of Value	Null

Following above table, a weight valued '0' will result in 0 copies of a specific location value, what coincides with the notion of a null value for any attribute domain; so that it can play the role of the value 'F'. The value 1 can work in the same way as the value 'T' for simple selection. Any other values essentially combine selection and weighting. Instead of introducing a new term such as "weighted neighborhood", it is preferable to go on with simply "Neighborhoods" as the only type defined.

To illustrate consider the neighborhoods specifications bellow:

- Neighborhoods up : [1,-1,1],[1,0,2],[1,1,1];
- Neighborhoods down : [-1,-1,1],[-1,0,2],[-1,1,1];
- Neighborhoods left : [-1,-1,1],[0,-1,2],[1,-1,1];
- Neighborhoods right : [-1,1,1],[0,1,2],[1,1,1];

They can model the selections and weightings involved in the Sobel filtering operation:

- `im*down-im*up`
- `im*right-im*left`

All arithmetic and mathematical functionality for numbers, as well as their properties, became now available to neighborhoods specifications. For instance, one can rewrite above expressions as follows:

- `im*(down-up)`
- `im*(right-left)`

Of course, other neighborhoods variables can be defined from existing ones through arithmetic. For example:

- `dirY = down-up ;`
- `dirX = right-left ;`

Back to the region concept one can observe that all specializing represented by the types for zones and neighborhoods are just essentially needless. Both concepts can be threatened indistinctly as simply regions in all expressions used as examples.

In this context, a model can be described by means of functions that associates locations to regions, it doesn't matter how many other locations are associated to the

same region. Zones and neighborhoods are just limit concepts, a lot of intermediate situations can be possibly explored through map algebra this way.

5. Summary Functions

After selecting and weighting locations through map algebraic local operations, it follows the last step of a region operation: summarizing a value to characterize an entire region, or a specific focus location in the study area. Typical summarizing functions are simple statistics like 'average', 'sum', 'maximum', 'majority' etc, applied to the sets of location values previously selected and weighted. For example:

- `Sum (img * ngh8)`
- `Majority (veg * (slope < 30 AND soil == "lacto", 30 <= slope < 50 AND veg == "forest"))`

In the first above expression every location will be characterized through the summation of the product of each 8-neighboring location values from an image layer represented by variable "img" by corresponding weighting values from the neighborhood specification. Evaluating the second will extract the predominant vegetation class from a map layer represented by the variable "veg", at zones described by a list of Boolean expressions based on data from slope, soils and vegetation layers

As another example, consider the case of the Sobel or gradient filtering operation used as example in previous section. It will involve the following summarizing expressions:

- `Sum (im * dirX)`
- `Sum (im * dirY)`

The complete Sobel filtering operation can then be expressed as follows:

- `sqrt((Sum(img*dirX))^2 + (Sum(img*dirY))^2)`

Arguments to statistical functions are anything that can be thought of as samples in a sample space. In the case of spatial data, sample space definition involves regions typically given by neighborhoods and zones. By the language counterpart, the values at locations in any region can be described by any valid algebraic expression, while the sample spaces themselves are explicitly or implicitly defined through Boolean expressions combining comparisons based on adequate relations. The resulting approach to map algebra proposed in this paper thus avoids the need for specializing statistical concepts regarding the way sample spaces are recorded. By separating selection and weighting from summarizing, it become natural to think about modeling with a minimal set of concepts

6. Fitting Geo-algebra

Regions are particular cases of a more general structure for spatial data representation, namely that of meta-relational map introduced by Takeyama et al., 1997. A single

region can be generalized to an entire map, a “relational map”, in which 0-valued cells correspond to non-influencing locations while any other value will indicate a weighting factor.

Maps are defined as elements of the set V^L of functions from a set L of locations in the cartographic plan into a set V of attribute values observable at the locations. Formally a map m is defined as a set of ordered pairs for which the first entry is the location position and the second is a value associated to it, as shown below:

- $m = \{(l, m(l)) \in L \times V\}$

A meta-relational map is a type of generalized map such that for each location an entire binary map - a relational-map - is associated, instead of a single value. More formally a relational-map R_l will be associated to each location $l = (i, j)$ so that:

- $R_l = \{(l, R_{ij}(l)) \in L \times \{0, 1\}\}$

New values at locations in a study area are computed by first “multiplying” maps and meta-relational maps resulting in new meta-relational maps that can be further operated in order to model adequate influence sets for each location. After all, adequate summarizing functions - the influence functions - can be applied to each influence set in order to generate new location values and then new maps.

Associating regions to locations in the study area to defining sets of regions corresponds to associating relational maps to locations to defining meta-relational maps. Hence the situational information for locations represented in maps, for a wide range of operations, can also be modeled by such sets of regions. In the operational sense the role played by the Regions type in realizing the functional aspect of a meta-relational map is also evident. Both concepts characterize functions from the spatial domain represented by the study area to its power set (the set of all its subsets). Maps and relational maps are also modeled in a similar way, in which, arithmetic and Boolean expressions play the role of general functions, whose arguments’ attribute domains may involve, any qualitative, quantitative or binary set.

To illustrate the expressiveness of geo-algebra, Takeyama proved in his thesis the equivalence between the cellular automata and a sub-algebra of geo-algebra (see Takeyama et al, 1997). The game “Life”, a popular example of cellular automata was used as an example.

The game “Life” was invented by the mathematician John Conway at Princeton University in 1970, based on a set of rules carefully chosen after trying many possibilities some of which caused the cells in the associated cellular space to die (0-valued) too fast and others which caused too many cells to be born (1-valued). Life balances these tendencies, making it hard to tell whether a pattern will die out completely, form a stable population, or grow forever (Gardner, 1970). The rules are simple:

- A live cell (1-valued) with two or three live neighbors will survive (keep its value).
- An empty (0-valued) cell with three live neighbors will come alive.
- Otherwise the cell will not survive.

In Takeyama's geo-algebra this situation can be modeled as the single map equation:

- $n = X_{=2}(I(m*R)) * m + X_{=3}(I(m*R)) * m$

The product $m*R$ involves a binary map and a relational map, it essentially selects all the neighboring location values around each location of the study area, thus resulting in a meta-relational map. The influence function $I()$, defined on meta-relational maps will summarize a value from the previously selected values for each location in the study area, thus resulting in a new map. The functions $X_{=2}()$ and $X_{=3}()$ are known as the characteristic functions for the one element subsets $\{2\}$ and $\{3\}$. Its evaluation in the first case will return the value 'True' whenever a value of '2' is found, and in the second case, when a value '3' is found. For any other value the value 'False' is returned. The application of the characteristic functions results in two new binary maps, the product of which with the original binary map m will result in two intermediate binary maps. Finally a logical (Boolean) 'or' operation, represented by the symbol '+' is applied so that the map n representing the new states after applying "Life" rules to m keeps completely determined.

With the help of some iterating and control statements, the expressions describing the rules of "Life", in the approach of this paper, would look like this:

- ```
Numeric state,new_state ;
Neighborhoods ngh8:
 [-1,-1],[-1,0],[-1,1],
 [0,-1],[0,0],[0,1],
 [1,-1],[1,0],[1,1];
t = 0;
While (!end)
{
new_state=
1: ((state==1) AND (2<=Sum(state*ngh8)<=3))
OR
((state==0) AND (Sum(state*ngh8)==3));
state = new_state;
t = t+1;
}
```

Many different versions of "Life" can be obtained by means of the product defined in Section-2. For example, one could restrict the domain by conditions involving other map layers as in the following sub-expression:

- $Sum(state * ngh8 * (veg == "forest"))$

Here only locations contained in a region represented by the thematic class "forest" of a vegetation map layer associated to the variable "veg" are to be considered.

A language based on algebraic expressions must couple with grammatical rules that must be interpreted, and parsed in a uniform manner so that sentences in the language can be understood. This process results in an executable code that implements an execution strategy for operations.

## 7. Implementation Model

The way local operators are extended from one to more dimensional domains in map algebra was originally based on a functional approach. In this context, adding two map layers of quantitative type corresponds to calling of a function possibly named 'add' with two map layers as arguments. A new layer thus results in which each location value is defined by adding corresponding values in both layers. Expressions involving more than one operator can be implemented by function composition. To illustrate consider an expression such as:

- $(a - b) / (c + d);$

Its evaluation will involve two intermediate results representation, one for addition and another for subtraction, before division can be done. It could rather be expressed like:

- `divide(subtract(a, b), add(c, d))`

If the arguments passed above are actually two-dimensional (or more), so will be all intermediate data generated before completion..

The formalism behind compiler implementation for computer languages such as 'C', 'Pascal' etc was founded on automata and formal languages theories (see Hopcroft et al., 1969). A formal compromise among algebraic structures, language expressions, grammar rules, and implementation were then stated for language expressions interpretation, parsing and code generation. Languages were then categorized into classes associated to other classes of conceptual machines that can model the understanding of sentences of a language. For instance the class of context-free languages (CFL), can model algebraic expressions such as arithmetic and Boolean, typically extended to map algebra language. The formal machine approach to implement the interpreting and parsing of CFL's is commonly referred to by "pushdown automata" (see Hopcroft, et al., 1969, ch.4). In this approach, a stack structure is used to communicate arguments and operators. At the location level, operators and functions do not need to deal with multi-dimensional structures, thus avoiding limitations regarding the size and complexity of expressions that describes operations.

To illustrate the pushdown approach, consider the example expression given before. After interpreting and parsing it will result in a code such as:

- `push(a) push(b) sub push(c) push(d) add divide`

Each "instruction" above can be implemented as a function whose execution will typically cause some values to be popped up from the stack, some action to be done, and a result to be pushed back into the stack for next instruction usage.

The columns of the tables showed bellow illustrates the sequence of states achieved by the stack at code execution time.

|  |   |   |     |     |     |     |             |  |
|--|---|---|-----|-----|-----|-----|-------------|--|
|  |   |   |     |     | d   |     |             |  |
|  |   | b |     | c   | c   | C+d |             |  |
|  | a | a | a-b | a-b | a-b | a-b | (a-b)/(c+d) |  |

The stack is initially empty, then contents of variables 'a' and 'b' are pushed into the stack and the instruction 'sub' is called which pops up its arguments from the stack, performs the subtraction and pushes the result back into the stack. Next the variable 'c' and 'd' are pushed and the instruction 'add' is called which pops up its arguments from the stack, performs addition then pushes the result into the stack. Finally both arguments to feed the 'divide' instruction are in the stack and a final result is obtained.

The code generated implements the automaton associated to a context free language expression, so that it can be thought of as the operation counterpart of such expression, associated to a location in the study area at the model execution time.

## 8. Concluding Remarks

In this work map algebra essentially is generalized to deal not only with the description of layers, but also with the description of regions and thus to the description of neighborhoods and zones. Also the interaction among these concepts, in a natural and quite consensual manner is devised in a way that is consistent with principles of geo-algebra general algebraic formalism.

Starting from a pushdown automaton implementation the possibility of a compiler approach to map algebra implementation is foreseen. A "program" in the resulting language would first be fully interpreted; resulting in an executable code. Between interpreting and executing phases, optimization issues can be considered so that performance can meet dynamic model requirements. At any time of such a model running process exactly one single stack operation is in charge so that a lot of concern about memory management for temporary data is naturally removed.

As the concept of neighborhood adopted here is based on language expressions implemented as pushdown automaton, it also suggests exploring modeling approaches such as cellular automata by its descriptive language counterpart. Another point to be explored in future works comes from the simplicity and low memory demand for pushdown automata implementation that can ease the task of extending map algebra for distributed environments as well as parallel architectures.

## References

- Berry, J.K., 1987, 'Fundamental operations in computer-assisted map analysis', *International Journal of Geographic Information Systems*, 2, 119-136.
- Camara, G., Freitas, U.M., Cordeiro, J.P., 1994, 'Towards an Algebra of Geographical Fields', SIBGRAPI, Campinas, SP.
- Camara, G., Souza, R.C., Freitas, U.M., Garido, J.C. 1994, 'SPRING: Integrating Remote Sensing and GIS with Object-Oriented Data Modeling'. *Computers and Graphics*, 15, 6..
- Couclelis, H., 1997, 'From cellular automata to urban models: new principles for model development and implementation', *Environment and Planning: Planning & Design*, 24, 165-174.

- Couclelis, H., 2000, Modeling frameworks, paradigms, and approaches, In Clarke KC, Parks BE, and Crane MP (eds). *Geographical Information Systems and Environmental Modeling*. New York: Longman & Co. Ch2.
- Chan, K.K.L. & White, D., 1987, Map Algebra: An Object Oriented Implementation. Proceedings, International Geographic Information Systems (IGIS) Symposium: The Research Agenda. Arlington, Virginia, November.
- Gardner, M., 1970. 'Mathematical Games: The fantastic combinations of John Conway's new solitaire game 'life'' *Scientific American*, 223, 120-123.
- Hopcroft, J. E., Ullman, J.D., 1969. *Formal Languages and Their Relation to Automata*, Addison-Wesley, Reading, Mass.
- Ritter, G. X., 1990, Wilson, J., Davidson, J., 1990. 'Image Algebra An Overview', *Computer Vision, Graphics and Image Processing*, 49, 297-331
- Serra, J, *Image Analysis and Mathematical Morphology*, Academic Press, New York, 1983
- Takeyama, M., 1996, *Geo-Algebra: A mathematical approach to integrating spatial modeling and GIS*, PhD dissertation, Department of Geography, University of California at Santa Barbara.
- Takeyama, M.; Couclelis, H., 1997. 'Map dynamics: integrating cellular automata and SIG through Geo-Algebra', *International Journal of Geographical Information Science*, 11, 73-91.
- Tomlin, D., 1990. *Geographic Information Systems and Cartographic Modeling*. Prentice Hall, Englewood Cliffs, NJ.
- Wesseling, C.G., Karssenbergh, D.J., Burrough, P.A. and Van Deursen, W.P.A. 1996. 'Integrated dynamic environmental models in GIS: The development of a Dynamic Modelling language. *Transactions in GIS*, 1-1, 40-48.
- White R., Engelen G., 1994, "Cellular dynamics and GIS: modeling spatial complexity", *Geographical Systems 1*, 237-253