Editorial Manager(tm) for Geoinformatica
Manuscript Draft

Manuscript Number: GEIN43

Title: Yet Another Map Algebra

Article Type: Manuscript

Section/Category:

Keywords: Map Algebra;
Cartographic Modeling;
Spatial Analysis;
Formal Languages;
Automata

Corresponding Author: yet another map algebra Joao Pedro Cerveira Cordeiro, MSc

Corresponding Author's Institution: INPE - National Institute for Spatial Research

First Author: Joao Pedro Cordeiro, MSc

Order of Authors: Joao Pedro Cordeiro, MSc; Joao Pedro Cerveira Cordeiro, M.D; Gilberto Camara, PHD; Felipe Almeida, PhD

Manuscript Region of Origin:

Abstract:

Yet Another Map Algebra

João Pedro Cordeiro[1], Gilberto Câmara[1], Felipe Almeida[2]

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

[2]Instituto Tecnológico da Aeronáutica (ITA) – São José dos Campos, SP, Brasil

{jpedro,gilberto}@dpi.inpe.br,felal@ita.cta.br

Abstract. *This paper describes features of a language approach for map algebra based on the use of algebraic expressions that satisfies a concise formalism. To be consistent with formal approaches such as geo-algebra and image algebra, the proposed algebraic expressions are suitable not only for the usual modeling of layers but also to describing neighborhoods and zones. A tight compromise between language and implementation issues based on the theory of automata is proposed as the necessary support to define or extend coherently operators and grammar rules. This results in an efficient way of implementing map algebra 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 for 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", is proposed to describe the operations and relations among locations and location data, usually represented as map layers. 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.

Map algebra operations have been traditionally presented as functions of one, two or more named variables usually representing map layers, lookup tables and constants. 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 Tomin'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 an overlay operation for reclassifying locations based on criteria involving their associated values at different layers. For instance it states that locations higher than 290 occurring in regions of vegetation coverage of type valued 1 to 3, should be associated to a value of 1 in a resulting layer, etc. The use of numbers to represent both quantitative and qualitative data such as heights and vegetation coverage types 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.

In Couclelis (2000), a model is conceptualized as an abstract and partial representation of some aspects of the world that can help deriving analysis, definitions and possibilities based on acquirable data. 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 typical modeling themes. As new techniques, computational resources and data, become available, the complexity of models also experiences a growing tendency. Coupling GIS with system involved in environment and dynamic modeling has been the object of intensive research in which map algebra plays a special role because of its spatial representation and descriptive characteristics, particularly considering modeling based on cellular automata applications, frequently based on raster structures (see White et al., 1994. Couclelis, 1997, 2000).

However, problems arise regarding the interpretative approach commonly adopted in the implementation of map algebra functionality, and the excess of intermediate data representation generated at model execution time (see 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 language and implementation approach integrating GIS and a wide class of physical dynamic modeling applications in which optimization techniques plays an important hole.

The growing complexity of modeling would benefit from more formal support to spatial analysis operations. Tomlin’s map algebra itself is such a formalism, in which by operations such as arithmetic, statistics (local, focal and zonal), propagation, diffusion etc, are combined with basic “overlay” operations such as rating and ranking, All sort map operations can be modeled, however, its implementations usually lacks in actually exploring the properties of operators involved in operations in both language and implementation aspects.

In Ritter (1990) a sound algebraic formalism to image domains, the Image Algebra, is proposed through which all classes of image based operations can be derived. Concepts such as mask, window, neighbor and structuring elements commonly used in image processing and mathematical morphology, are grouped together in the concepts of “template” and “generalized template”. He starts from defining image as a function from a spatial domain into a set of possible range values, its “attribute” domain. The template concept results from allowing the image attribute domain to range over values that can model the extent to which a specific sets of locations can have influence in characterizing a given specific location in some image operations. Finally, a

generalized template results from allowing the attribute domain for images to range over the set of all possible templates.

In Takeyama (1996), a formal mathematical basis for extending the static modeling paradigm of map algebra is introduced, following close to Ritter's image algebra formalism for operations on images, Takeyama adopts the idea of map as a function from a spatial domain to a given attribute domain and, by allowing one such function to range over sets of functions, the "relational maps", a more general "map" structure, the "meta-relational" 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 not only the map algebra operations under a single framework, but also extend the cartographic modeling paradigm to deal with the dynamics of processes.

In Pullar (2001) an implementation founded on the formal ideas exposed previously is discussed, based on an algebraic formalism that incorporates the notion of templates. In this context, a neighborhood is defined as an arbitrary open set contained in the study area, shaped rectangular, with a centering element inside the set.

In this work an algebraic structuring is proposed, based essentially on introducing a binary operator to model the interaction of Boolean, with any other data type typically represented in map layers.. 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 essentially comparisons based on order and equality relations defined on the attribute domains representing altitudes vegetation coverage data. Putting those relations more explicitly, 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 ;
```

At the language level, describing regions by comparisons expressed as above as well as those describing the interactions between regions and maps, follow the same class of grammar rules of ordinary arithmetic and Boolean algebra. Thus, interpreting and parsing strategies can follow the same principles. 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.

The endeavor in this work is to add formalism from language and automata theories to the basis of map algebra implementation in order to reviewing its concepts in a more compatible manner regarding the requirements for its extensiveness to dynamic modeling. 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 incorporates some principles discussed in this paper. Spring GIS software is available free at www.dpi.inpe.br/spring. For most of the examples, only partial expressions or sub-expressions are relevant; only those expressions closed by the sign ';' (semicolon) may be considered complete regarding LEGAL syntax. New language constructs are introduced prototypically 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 geo-referenced, 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 can 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 of specific type. Only existing map layers and meaningful result layers should need to be actually represented in a model.

By focusing only on quantitative attribute domains, a lot of mathematical operations and functions can be induced by locally applying one-dimensional versions to values 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 multi-spectral image. If two named variables, 'b3' and 'b4', are adopted to represent the image bands, then it can be described as follows:

```
(b4 – b3)/(b4 + b3)
```

An expression as the above one can be used to define a new layer to the model that will be associated to a named variable, by means of an assignment operation, as illustrated bellow:

```
ndvi = (b4 – b3)/(b4 + b3) ;
```

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. They 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

To specifying a set of locations, as for sets of any nature, can be done either by explicitly indicating its elements or by means of their common properties. Thus, for an element to belong to a set is a matter of satisfying criteria based either on a checklist strategy or the evaluation of some properties. In any case, such criteria can be modeled as mappings from a spatial domain to a binary (or Boolean) attribute domain represented by values such as “true” and “false”, ‘0’ and ‘1’ etc. In particular, results from locally evaluating comparisons based on order and equality relations such as ‘<’, ‘>’, ‘<=’, ‘>=’, ‘==’, ‘!=’, can be identified to binary values as well, so that a wide range of sets of locations (or regions) can be expressed by using such relational criteria. Of course, Boolean algebra can be naturally extended to map domains as well, by inducing operators like ‘and’, ‘or’ and ‘not’ from their one-dimensional versions.

In this work the expressiveness of combining comparisons based on order and equality relations, and Boolean operations is the basis for characterizing sets of locations in the study area. 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 data 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 bellow:

*	value	null
true	value	null
false	null	null

The term ‘value’ above represents an arbitrary value of any valid data type, while ‘null’ is associated to the absence of data. The symbol ‘*’ is adopted here to represent the operator itself.

Extended to maps, this operator can model the selection of a set of locations, in terms of their associated values at possibly different layers, provided at least one of them be of type Boolean. Of course one need not a physical representation of a Boolean “layer”; the way operations are extended to maps is such that they work independently for each location, then a local Boolean value can be obtained whenever needed in a running model by evaluating an expression of type Boolean.

For a lot of quantitative spatial data types, such as images and possibly many other representing specific measures of ratio, interval or ordinal nature, operator ‘*’ can be induced from number multiplication, provided the integers ‘1’ and ‘0’ plays the role of ‘true’ and ‘false’. Applied to an image, this “selecting” operator 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’ 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’ cannot be represented locally by the integer ‘0’ because this is a meaningful value for a local slope. 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
```

Variable “soils” above, represents a soil types thematic (qualitative) layer.

The “selecting” operator has properties similar to those of ordinary multiplication so one can adopt the same symbol ‘*’ to represent both operators over maps, without any syntactic or semantic ambiguity, as illustrated by the equivalent expressions bellow.

```
(veg == “crop” AND slope < 30) * heights * distances
heights * (veg == “crop” AND slope < 30) * distances
heights * distances * (veg == “crop” AND slope < 30)
```

Each expression above involves two occurrences of the symbol ‘*’, one for selection and the other for usual multiplication.

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 be 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. Of course one can have more than a single location characterized by the same influence set as is the case for zonal operations. In implementing such non-local operations three basic steps are essentially of concern:

```
1. sets of influencing 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 1 and 2 above. The recording of values at selected locations is usually driven by the characteristics of the "summary functions" to be applied on then, typically consisting of simple statistics such as average, maximum, majority etc.

Expressions involving the selecting operator follows the rules of a context-free grammar similar to those for arithmetic and Boolean expressions, so that their interpretation and parsing may follow the same pushdown automata strategy (Hopcroft et al., 1969) adopted for other algebraic expressions in LEGAL map algebra approach. In its current version, LEGAL generates a pseudo-code to be executed after completion of interpretation and parsing of a complete set of expressions usually referred to by "program". An actual compiler approach for map algebra language can then be foreseen.

3. Regions and zones

The term "region" will be adopted here to mean set of locations. The type "Region" can also 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 this non-overlapping 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 thematic class named “main”, represented in a layer associated to the variable rivers.

Indeed, the selecting operator can be also applied on arguments of Boolean (or Region, or Zone) type, in a way that it is equivalent to a Boolean “and” operator, with the effect of refining a zonal partitioning, as illustrated by the two equivalent expressions bellow:

```
districts * vegetation
districts AND vegetation
```

New values at locations in a study area can be unambiguously characterized from available data by evaluating expressions of any type and their interactions with expressions describing sets of zones. For example:

```
(b4 – b3)/(b4 + b3) * buffers
(b4 – b3)/(b4 + b3) * districts * vegetation
```

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 a set of regions that do not constitute a set of zones in cartographic modeling. In a functional sense, operations always involves the association of a region to each location, be it the same region for a group of locations or even different regions to distinct locations.

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 involving the spatial domain.

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 help defining neighborhood regions. For example:

```
distance()<3 and veg == “forest”
distance()<3 and direction()<90
```

First above expressions describes the set of all regions with forest coverage in a circular vicinity of radius 3 units around some specified set of foci locations, typically consisting of all locations represented in a study area. The second expression describes any set of locations in a sector of 90 degrees, not more than 3 units close to a specified focus.

This paper proposes combining relations on the attribute domains of maps with such proximity notions as a natural way to modeling some “spatial variability” for the classic neighborhood concept. The type “Neighborhoods” is introduced here as another specialization for the type Regions so that one could associate the expressions in last example to variables such as:

```
Neighborhood close_to_forest = distance()<3 and veg ==
“forest”
Neighborhood up_and_right = distance()<3 and
direction()<90;
```

Other simple relations in the spatial domain, such as “adjacency” can also play the role of a proximity criterion among locations, thus defining regions characterized by a focus and its (one to eight) adjacent neighboring locations. De Moore and Von Neumman' vicinities are classical examples.

Operations based on formal approaches to exploit the cartographic plane can explore the partial ordering properties of regions. For instance, concepts from lattice theory of modern algebra can then be become available. A quite complete such applications is mathematical morphology, which offers a formal approach to modeling a wide class of image analysis operations taking the partially ordered framework as support.

A natural way to involve neighboring locations in operations, consists of referring them explicitly in expressions by their relative positioning regarding each location in the study area, taken as a reference or focus location, as illustrated by the expression 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
```

It describes the averaging of values associated to neighboring locations, coming from an image data layer represented by the variable "img". 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 a "focus" location associated to the pair `[0,0]`. Actually selecting a set of locations in the vicinity of a focus is a matter of computing the coordinates of neighboring locations based on the focus coordinates. It is suggestive to adopt this shifting mechanism to specifying neighborhoods. The family of regions involved in the above expression can thus be specified as follows:

```
Neighborhoods ngh8 =
            [-1,-1],[-1, 0],[-1, 1],
            [ 0,-1],[ 0, 0],[ 0, 1],
            [ 1,-1],[ 1, 0],[ 1, 1];
```

One such specification is essentially a function from the set Z^2, the set of all ordered pairs of integer number to a binary set such as {true, false} or {0, 1}. By adding or subtracting units of resolution from the coordinates of an arbitrary focal location in a raster spatial domain one can actually define a family of functions from a study area into a binary set as before. For each location, its neighboring locations are then specified and associated to the value 'true' or 'false'. Usually only 'true' valued locations need to be explicitly specified.

The "shape" of the neighborhood regions can also be modeled this way, as for example the families of neighborhoods "plus" and "times" specified bellow:

```
Neighborhoods plus =
                  [-1, 0],
          [ 0,-1],[ 0, 0],[ 0, 1],
                  [ 1, 0];
Neighborhoods times =
          [-1,-1],        [-1, 1],
                  [ 0, 0],
          [ 1,-1],        [ 1, 1];
```

For convenience in variable “ngh8”, “plus” and “times” above only cells valued ‘true’ are indicated, all other locations are omitted. However it is sometimes useful to have the values associated to relative coordinate pairs in a neighborhoods specification given explicitly, as a third parameter representing the selection state of a location, as illustrated by the new versions for neighborhoods “plus” and “times” specifications given bellow:

```
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];

Neighborhoods times =
          [-1,-1,T],[-1,0,F],[-1,1,T],
          [ 0,-1,F],[ 0,0,T],[ 0,1,F],
          [ 1,-1,T],[ 0,0,F],[ 1,1,T];
```

Values such as ‘T’ and ‘F’ above are essentially constant Boolean expressions, so that it is suggestive now to allow any Boolean expression, or any region specification, in an explicit neighborhoods specifications. The immediate consequence is the possibility of modeling the variability for neighborhoods in terms of spatial data available for the locations involved in its specification. For each focus locations, conditions involving values at neighboring locations can be evaluated in order to decide their presence or absence in the set of influencing locations regarding the focus. For instance, one can have an specification such as:

```
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”];
```

As for regions in general, Boolean algebra can be naturally extended to neighborhood specifications. For instance, one can redefine variable ngh8 as:

```
ngh8 = plus OR times ;
```

In fact any Boolean operation involving regions and at least one neighborhoods specification can be assigned to a variable of Neighborhoods type, as illustrated by the two equivalent expressions bellow:

```
Neighborhoods ngh = plus AND (slope < 30);
Neighborhoods ngh =
     [-1,0, slope < 30],
     [0,-1, slope < 30],
     [0, 0, slope < 30],
     [0, 1, slope < 30],
     [1, 0, slope < 30];
```

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 table of Section-2 is adjusted a bit, as follows:

*	value	null
x∈R	X 'copies' of value	null

A weight valued '0' will mean 0 'copies' of some specific location value, thus meeting the notion of a 'null' for any attribute domain, so that it can play the role of the value 'false' in the previous table version. By the same argument, the value '1' can work in the same way as the value 'true' for simple selection. By extending the binary set {0, 1} to a wider set of quantitative values such as the set of integer numbers, or even the set of all real numbers, the concepts of selection and weighting just can be joined together. At this point, instead of introducing a new concepts such as "weighted neighborhood", it is preferable to go on with simply "Neighborhoods" as the only type defined.

All arithmetic and mathematical functionality for numbers, along with their properties, became now available to neighborhoods specifications. To illustrate consider the neighborhoods specifications bellow:

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

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

```
im*down–im*up
im*right–im*left
```

By exploring associative law for numbers one can rewrite the 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 ;
```

So that one can replace sub-expressions by equivalent variables, as follows:

```
im*dirY
im*dirX
```

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 this map algebraic approach.

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 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 by summing the product of each 8-neighboring location values from the image layer represented by variable "img" by their specified weighting values from the neighborhood specification. The second expression extracts predominant vegetation from a map layer represented by the variable "veg", at zones described by a list of Boolean expressions involving data from slope, soils and vegetation data layers

In the case of the Sobel or gradient filtering operation used as example in previous section the following summarizing expressions can be involved:

```
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. The values at locations in any such regions are described by valid algebraic expressions describing operations over available data, while the sample spaces themselves are expressed as regions, defined through Boolean expressions combining comparisons based on adequate relations. The approach to map algebra adopted in this work thus eliminates 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

Concepts such as "meta-relational" map, a map in which each location is associated to an entire map (a "relational" map) representing the resulting interaction of a map and the family of influence sets involved in characterizing each location in a spatial domain. A region is a particular case of a 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.

A map is defined as a element of the set V^L of functions from a set L of locations in the cartographic plan into a set V of attribute values, usually L is a subset of R^2, then its a relation in R^2. More specifically a map m is hence a set of ordered pairs to which the first entry represents a location position, that is a coordinate pair, while the second entry is its associated attribute value, as shown bellow:

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

A relational map to be associated to a single location in the study area can be defined in the same way an ordinary map would, except for the requirement that the set V must be of quantitative nature. That happens because such values will represent

selecting and weighting information, while an ordinary map can also represent data of qualitative nature.

A meta-relational map is thus a sort of generalized map such that for each location an entire relational-map is associated, instead of a single value. More formally a relational-map R will be associated to each location l so that:

$$R = \{(l, R_l) \in L \times V^L\}$$

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 a set of regions corresponds to associating relational maps to locations to defining a meta-relational map, so that the situational information for locations can be equally modeled by such sets of regions. Both concepts characterize functions from the spatial domain represented by the study area to its power set (the set of all its subsets). Arithmetic and Boolean expressions plays the role of general functions, whose arguments' attribute domains may involve, any qualitative, quantitative or binary set.

In Takeyama and Couclelis (1997) the authors proved that geo-algebra not only extends, but also generalizes the dynamic modeling formalism known as cellular automata. Geo-algebra expressiveness is illustrated in Takeyama's thesis by the equivalence between the cellular automata, and a sub-algebra of geo-algebra. 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:

$$n = \chi_{=2}(I(m*R))*m + \chi_{=3}(I(m*R))*m$$

The product m*R involves a binary 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 return one (summary) value from each set of influence locations´ values selected to characterize each location in the study area. This results in a new map. The functions $\chi_{=2}()$ and $\chi_{=3}()$ 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 '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 the 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 can result in an executable code that implements an execution strategy for operations that corresponds to the concept of a compiler, thus avoid the efficiency problems deriving from the usual interpretation approach usually adopted to map algebra languages.

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, 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 the algebraic expressions such as arithmetic and Boolean 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 to operators implemented as primitive functions. At the location level, operators do not need to deal with multi-dimensional structures, thus avoiding limitations regarding the size and complexity of their expression couterpart.

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 to serve as arguments, 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 from interpreting a "map" algebraic expression essentially implements the automaton governed by a context free grammar, so that it can be thought of as the operation counterpart of an expression. Of course, sub-expressions describing regions will be translated into sub-automata by themselves, that shall be inserted in the main flux whenever needed, in order to characterize the pertinence of a locations to one or more regions in the study area at the model execution time.

8. Concluding Remarks

In this work map algebra has been essentially 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 was 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.

Pullar, D., 2001, Map Algebra and Neighborhood Functions, *GeoInformatica,* 5, 145-163.

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*, Adisson-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., Karssenberg, 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: 40-48.

White R., Engelen G. 1994 Cellular dynamics and GIS: modeling spatial complexity", *Geographical Systems* 1: 237-253

3/2/2006 3:09 PM