Geoinformatica (2009) 13:183–202
DOI 10.1007/s10707-008-0045-4

# Yet Another Map Algebra

**João Pedro Cerveira Cordeiro · Gilberto Câmara · Ubirajara Moura de Freitas · Felipe Almeida**

Received: 2 March 2006 / Revised: 3 January 2008 /
Accepted: 4 February 2008 / Published online: 2 May 2008


**Abstract** This paper describes features of a language approach for map algebra based on the use of algebraic expressions that satisfy a concise formalism. To be consistent with formal approaches such as geoalgebra and image algebra, the proposed algebraic expressions are suitable not only for the usual modeling of layers but also to describe variable neighborhoods and zones. As a compromise between language and implementation issues we present an implementation strategy based on the theory of automata. The result is an efficient way of implementing map algebra that simplifies its use on environmental and dynamic models without going too far from its well-known paradigm.

**Keywords** map algebra · cartographic modeling · spatial analysis · formal languages · automata · dynamic modeling

## 1 Introduction

The main contribution towards an algebraic foundation for modeling operations over maps came from the works of Tomlin and Berry at Yale University in the 1980's (see Tomlin and

J. P. Cerveira Cordeiro (✉) · G. Câmara
Divisão de Processamento de Imagens, Instituto Nacional de Pesquisas Espaciais (DPI–INPE),
São José dos Campos, São Paulo, Brazil
e-mail: jpedro@dpi.inpe.br

G. Câmara
e-mail: gilberto@dpi.inpe.br

U. Moura de Freitas
Departamento de Geoprocessamento, Fundação para Ciência, Technologia e Aplicações Espaciais
(FUNCATE), São José dos Campos, São Paulo, Brazil
e-mail: bira@funcate.org.br

F. Almeida
Instituto Tecnológico da Aeronáutica, Centro Técnico Aeroespacial (ITA–CTA),
São José dos Campos, São Paulo, Brazil
e-mail: felal@ita.cta.br

Berry 1979; Tomlin 1983, [1]), compiled in the book "Geographic Information Systems and Cartographic Modeling" [17]. They stated the foundations for map algebra, a formal algebraic approach to accommodate a wide range of modeling situations concerning data associated to locations of a spatial domain. Map algebra data model accommodates map layers of quantitative types such as "ratio", "ordinal", "interval", "scalar" and also a "nominal" qualitative type. Running a cartographic model within Tomlin's map algebra is a matter of interpreting a sequence of textual sentences of a language named MAP, for "Map Analysis Package". Each sentence in this language describes the assignment to a named variable representing a map layer, of the result of evaluating operations invoked through expressions conformed to well defined syntax and grammar rules. A sequence of intermediate map layers is usually generated during runtime, some of which are incorporated to the model while some others may be discarded.

Lots of map algebra implementations came up since Tomlin's work, such as the GRID module of ArcInfo, now integrated into the rich language of ArcView Spatial Analyst framework, from Environment Systems Research Institute (ESRI); IDRISI, developed at Clark University; ILWIS developed by the International Institute for Aerospace survey and marketed by PCI Geomatic; the R-Mapcalc, from GRASS-Community, and also many other well known GIS software packages. Earlier map algebra implementations were identified to the concept of "raster" GIS, although today's products also include extensive vector capabilities and strong integration with other software environments such as database management, mobile GIS and simulation.

As new techniques, computational resources and data, become available, the complexity of models also experiences a growing tendency. Coupling GIS systems 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 [4], [5], [20]). 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 runtime (see Dragosits 1996). Optimization strategies to deal with these problems, such as those suggested in [7], and the use of efficient algorithms are important issues in accommodating the problem of coupling GIS and dynamic models. PCRaster [19] is a good example of a modeling tool that integrates map algebra with a wide class of physical environment modeling, in which optimization techniques play an important role. Also a classical map algebra extension to deal with cubic 3D data in which one dimension is time was proposed in [10].

There are lots of commonalities between GIS and image processing issues. A lot of mathematics has been explored in developing algorithms, such as filtering, segmentation and classification. Even an algebraic formalism has already been proposed, the "image algebra" [13], joining image functionality into a common algebraic framework. Besides images, two new data types are incorporated to image algebra: the "template" and the "generalized template" to model the interaction between image cells and specific sets of other related cells that exert influence on them. The concept of image is then generalized by allowing image cells to range over templates instead of just whole numbers representing the "gray" level associated to it. Image algebra has made its contribution in a number of advances in image processing and computer vision, many resulting from the research agenda of the Image Algebra Project of the Center for Computer Vision and Visualization at Florida University.

In Couclelis [5], 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 such as: atmospheric, hydrological, biological and ecological systems, natural hazards, and many others typical modeling themes. The lack of a consistent framework for data modeling, process modeling and data manipulation, forces dynamic modelers to switch between GIS and other tools like simulation systems and general purpose language environments, thus leading to higher costs and loss of consistence. In order to fulfill this lack, in ([14], [15]) an algebraic structure called geoalgebra was introduced to extend map algebra into a framework analogous to image algebra, consisting of "maps", "relational maps" and "meta-relational maps", that can also model the dynamics of processes. Geoalgebra formalizes a view of the geographic space that incorporates to the classical absolute view adopted by the majority of GIS tools, a proximal view in which each georeferenced location also represents the relative space of which it is a part [4]. This view of space is intended to support the static and dynamic aspects of modeling in a common framework.

Some ideas from image algebra and geoalgebra have been used in [11], [12] to specify map algebra functionality to model spatial simulations and neighborhood analysis when dealing with heterogeneous physical processes. The resulting framework named MapScript is based on grids and template as data types. The way the author defines the grid type also include graph networking of cells in a rectangular array of cells so that besides algebraic, also interesting topological properties can be easily described. However the need to actually represent templates as a data structure in the model may impose some limitations regarding the variability of the shape and weights in a spatial-temporal modeling context. In this paper we concentrate on the spatial context to demonstrate a simple way to add variability at the basis of the template, or relational map concepts, in a way that avoids the need for physically represent it, thus resulting in a map algebra in which essentially local operations are the basis to describe and implement the so called zonal and neighborhoods operations. Explicit time considerations are left for future works.

Though focused in the formal aspects of these algebraic structuring tendencies, our research's main contribution concerns the formal aspects of an expression language to describe map algebra. The usual way to write arithmetic, relational and Boolean expressions in mathematics corresponds to specific expressions involving symbols, names and operations that satisfy conventional (and consensual) grammar rules to govern their understanding and evaluation. In a formal language theory, these algebraic expression's classes correspond to sentences of a "context free" language (CFL) [9], so that interpreting and parsing strategies to model their understanding and evaluation may follow automata theory principles. A revision of map algebra principles, based on this formal compromise between language and implementation thus suggests that it can be more naturally extended to deal with complex spatial and dynamic modeling issues while also avoiding traditional overheads associated with interpretative solutions and the excessive intermediate data generation steps ([7]; Dragosits 1996).

Results from this research are being implemented as part of the Spring GIS project of the Image Processing Division at the National Institute for Spatial Research in Brazil, as an enhancement for the language LEGAL available as a module in the Spring GIS environment (http://www.dpi.inpe.br/spring). LEGAL implements map algebra functionality in a framework that accommodates the map concept into categories such as Thematic, Surface, Image and Objects. An overview of Spring's data model and the language LEGAL can be found in [2] and [3]. For most examples in this text only partial expressions or subexpressions are relevant; only those expressions closed by a semicolon may be considered complete sentences in the proposed language syntax.

This paper is structured as follows: the correspondence between language expressions and operations is discussed in Section 2 at the specification and implementation levels, just to roughly introduce the reader to some very basic and specific aspects of the theories of formal languages and automata. In Section 3 we discuss the extension of basic algebraic structure to geo-spatial domains; the concept of region and region coverage is introduced in Section 4 and developed in Section 5 and Section 6 as the basic concept behind non-local map algebra operations. In Section 7 the concept of region is enhanced so that differences among locations regarding their influence in operations can be modeled. Section 8 presents a discussion on the expressiveness of the proposed language in describing cellular automata applications such as the Life game. Also some basic language requirements are discussed for describing environment models in a proximal and absolute spatial perspective, such as those involved in landscape ecology issues. As concluding remarks, some performance issues that may benefit from this approach are pointed out regarding its use within distributed and parallel architectures.

## 2 Language and automata

A language consists of a set of expressions that convey some meaningful information. It can be so extensive as the natural languages such as Portuguese and English, or as restrictive as the language accepted by a coffee machine. The language to express arithmetic and Boolean operations is interesting because of its simplicity and consensus. Grammar rules governing the building of algebraic expressions can be specified by means of a recursive approach in which each language element definition depends on a series of other elements' definitions. For instance the following specification rules may define what, in general, an arithmetic expression can be.

```
<expression> ::   <variable> |                              (1)
                  <constant> |                              (2)
                ( <expression> ) |                          (3)
                  <expression><op><expression>|             (4)
```

The concept of expression can thus be equated to the concepts of variable, constant and function. A variable will refer to an element of the application domain; a constant will explicitly refer to a specific number, and operators($\langle op \rangle$) are used to combine expressions into new ones. Grammatical rules are applied in a stepwise way toward building (or understanding) of acceptable sentences of the language they are supposed to specify. To roughly illustrate consider two variables named "a" and "b", by rule-1 (rules are separated by the symbol '|') these are expressions per se. Next, by using rule-4 with the operators '−' and '+', one can combine them to build new expressions:

```
a - b
a + b
```

Now using rule-3 twice, followed by rule-4 with the operator '/', lead to the final expression:

```
(a - b)/(a + b)
```

The formalism behind compiler implementation for computer languages such as 'C', 'Pascal' etc. was based on automata and formal languages' theories. Formal languages can be categorized into classes associated to corresponding classes of conceptual machines that can

model their sentences understanding. For instance the class of context-free languages (CFL), in which the language of algebraic expressions is included, can be modeled by means of a formal machine approach commonly referred as "pushdown automata" (see [9], Chapter 4). In this approach, a stack structure is used to communicate arguments and operators.

To illustrate the pushdown approach, consider again the example expression given before. After firing any grammar rule in the parsing (syntactical analysis) process, an adequate instruction call is recorded, so that in the end a sequence of such calls is generated. The resulting sequence of instructions constitutes the code that implements the control of the operation evaluation flow. Essentially this code can be described as follows:

```
push(a) push(b) sub push(a) push(b) add divide
```

Each primitive instruction above consists of a function call whose execution will cause some values to be popped up from a stack structure and some action to be done that result in a new value to be pushed back into the stack for further instruction usage. An expression evaluation is concluded whenever the stack gets empty. The columns of the table showed in Fig. 1 illustrate the stack content along the sequence of states achieved at the code runtime.

The stack is initially empty, then the 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 "*a*" and "*b*" are pushed in the stack again so that the instruction "add" pops them up as arguments from the stack, performs addition then pushes the result back into the stack. Finally both arguments feed the "divide" instruction that pops them out of the stack and a final result is obtained.

Tomlin's original specification for map algebra suggests a functional implementation approach in which function composition is used to model operations and communicate partial results until the final result is returned, in this context one should rather rewrite our example expression as follows:

```
divide(subtract(a, b),add(a, b))
```

If "*a*" and "*b*" stands for maps one can represent them by two-dimensional raster structures so that all intermediate data generated will probably be also represented this way. Then, for more complex expressions more sophisticated allocation strategies at the model runtime would be demanded thus imposing limitations to the size of expressions. For instance, two intermediate representations are needed to accommodate the results of the addition and subtraction operations in the expression above before division can take place. On the other hand, in the automata approach lots of allocation problems can be avoided because only "one-dimensional" operator versions will be active throughout the whole evaluation process. One may even forget the usual notions of map, image and grid as the basic representations of data over a study area, and focus on the study area as a set of locations each characterized by an adequate expression, regardless of the local, zonal or neighborhood nature of the operation it describes.

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | | | | | b | | | |
| | | b | | a | a | a+b | | |
| | a | a | a-b | a-b | a-b | a-b | (a-b)/(a+b) | |

**Fig. 1** Stack states of a pushdown automaton for simple arithmetic

## 3 Extending algebra to maps

Exploring algebraic formalism over maps allows the modeling of phenomena as operational descriptions that translate the model semantics in a consistent and consensual way. Concepts manipulated in a model must be translated into algebraic expressions that may involve operators, variables, constants and other symbols. For instance, the concept of "vegetation index" defined as the normalized local difference of radiometric values from image data representing red and near-infrared spectral frequency measures from a specific sensor device, can be described as follows:

```
(nir - red)/(nir + red)
```

Also expressions describing comparison operations based on relations such as order and equality can be used to induce local operations. For example, a set of locations with 'forest' coverage, a set of locations with less than 30% slope and a set of locations with vegetation indexes higher than 0.5 can be described by the following three expressions:

```
use == 'forest'
slope < 30
(nir - red)/(nir + red) > 0.5
```

Results from evaluating comparisons can be visualized as maps having binary sets such as {'true', 'false'} or {'0', '1'} as attribute domains. Hence Boolean algebra can be naturally extended to map domains as well, by allowing operations combining comparisons through operators such as 'and', 'or' and 'not', as illustrated below:

```
use == 'forest' AND (ndvi > 0.5 OR slope >= 30)
```

Finally, arithmetic and Boolean expressions can be assigned to variables through assignment statements indicated by the equal (=) sign, as illustrated below:

```
ndvi = (nir - red)/(nir + red);
bestplace = (use == 'forest') AND (ndvi > 0.5 or slope >= 30);
```

Differently from classical map algebra languages, an assignment statement to a named variable will behave as a synonym for its defining expression, to be evaluated whenever needed in the modeling process. An assigned variable will be made into a new map only if a physical representation is already associated to it. In the case of Boolean typed evaluations we don't ever need to physically represent the results, this class of operations is intended just to select the sets of locations involved in further operations, either in a local, zonal or focal context. The term "Boolean" will be used hereafter in this text to refer to both comparison and Boolean, expressions or operations.

The discussion up-to this point concerned essentially the class of local operations of Tomlin's map algebra taxonomy, other classes such as zonal and neighborhood operations can model the influence of sets of locations over single locations. Evaluating operations within these non-local contexts would involve three basic steps:

```
1. sets of influencing locations are selected;
2. sets of values at selected locations are recorded from maps;
3. a value is summarized for each recorded set.
```

Figures 2, 3 and 4 illustrate these steps for a simplified study area represented by a 4 × 4 array of cells in Fig. 2a. Initially there are no maps or any other sort of spatial data

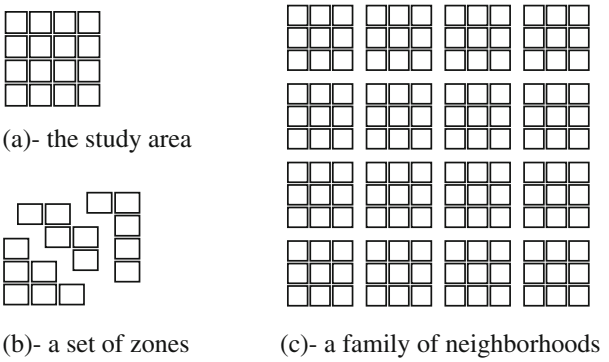



**Fig. 2** Zonal and a neighborhoods' coverage of a given study area

representation involved, only spatial locations to which basic cartographic premises were assumed. Figure 3 illustrates the selection of values associated to locations by means of maps, or expressions involving maps. In Fig. 4, new information is produced for each location by summarizing the values previously selected by zones or neighborhoods using simple majority criterion.

The next three sections will discuss these steps with more details. Actually the second step concerning the recording of values to be summarized will be underestimated at the first glance, in Section 7 this concept will be enhanced to accommodate situations in which the influence exerted by specific locations of a common region may vary.

## 4 Selecting regions

In the cartographic modeling paradigm, a map often represents the partitioning of the study area into a set of disjoint regions whose union covers the whole study area. Each element in such coverage is named a "zone", and it is intended to aggregate locations with common properties. Concepts such as, "states in a country", "parcels", "thematic classes", and "cell spaces" among others can be modeled as zones. Other maps such as images and numerical grids usually represent continuous distributions of quantitative or qualitative values over the study area. In this case map analysis is usually based on a coverage type consisting of sets of regions that may overlap such as "neighborhoods", "masks" and "structuring elements", to model the spatial variability of the represented data.

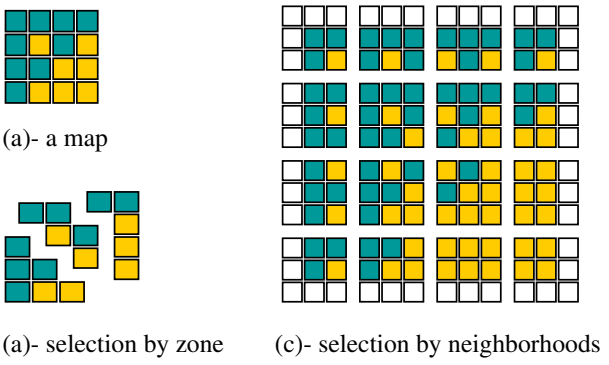
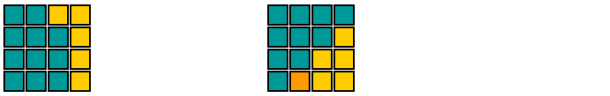



**Fig. 3** Selection of values at zones and neighborhoods from map data



(a)-summarizing by zone (b)- summarizing by neighborhoods

**Fig. 4** Summarizing values for each location based on simple statistics criteria, such as majority, to both zones and neighborhoods previously selected values

Early works such as Berry [1] and Chan and White [6] were based on the "map layer" as a primary data structure to represent maps, along with a "topographic layer" conceived to model continuous surfaces. Operations were then classified around these basic two-dimensional arrays of data as: "local operations", "neighborhood operations" and "region operations". The term "region" was replaced latter on by the term "zone" to reinforce the non-overlapping condition usually imposed on regions represented by map layers as a basic cartographic premise. The concept of a region was then focused on problems concerning the extraction of quantitative properties such as area and topological properties such as Euler numbers and clumps. We choose to recycle the term "region" just as a synonym to "set of locations". If a set of regions constitutes a partitioning for a given study area, then each of its component may be called a "zone", while "neighborhood" is just another instance of the region concept in which sets of locations are defined relatively to specific reference locations.

As pointed out in the previous section, evaluating a local Boolean operation concerns the selection of a set of locations, so that we can identify regions to the pertinence conditions imposed to its locations. We actually treat the terms—"region" and "Boolean"—as synonym in this text.

Named variables can be assigned to such "region expressions", so that regions and lists of regions can be easily incorporated to the model as illustrated by variables 'bestplace" and "goodplaces" shown below:

```
bestplace = use == 'forest' AND ndvi > 0.5 AND slope <= 30;
goodplaces = bestplace,
             use == 'crop' AND district == 'd1',
             use == 'urban' AND ndvi > 0.5 ;
```

Assignment statements such as above constitutes complete sentences of the language so that they must end with the symbol ';' (semicolon). Some syntax shortcuts may be also available in the language to avoid long lists of repetitive terms when writing region descriptions, as for example:

```
district == "d1", "d2", "d3"
district.All
use.*
```

Besides attribute domain relations such as order and equality, also proximity relations defined on the spatial domain can be used as criteria to specify regions. For instance, measures of distance and direction relative to a given focus location can be used to characterize its neighborhood region provided that adequately defined functions for distance and direction evaluation are available in the language as illustrated by the following two variable definitions:

```
near = distance()<3 ;
upright = distance()<3 AND direction()<90;
```

Variable "near" describes a family of circular vicinities of radius 3 units around each location in the study area, while for variable "upright" each region is further restricted to a

sector of 90° from a focus location. The null parameter in the function calls above indicates the focus location as the only reference location to be considered, more parameters may indicate other situations possibly involving fixed locations. Other proximity relations such as adjacency and connectedness are commonly used to describe neighborhood regions in raster domains, as for example the classical von Neumann and Moore neighborhood templates illustrated in Fig. 5.

Despite any proximity relation, neighbor locations can be explicitly involved in expressions by indicating their displacement in terms of shifted lines (above or below) and columns (to left or to right) from a specified focus location. To illustrate consider 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
```

This expression describes an image filtering operation used to characterize each location in the study area by averaging data associated to neighboring locations identified by pairs of integer coordinates. The focus location is associated to pair [0, 0]. It is thus suggestive to adopt this shifting mechanism in the specification of neighborhood regions. For instance, the whole family of regions involved in the filtering operation above can be specified by the following set of relative coordinate pairs:

```
[-1,-1],[-1,0],[-1,1],
[ 0,-1],[ 0,0],[ 0,1],
[ 1,-1],[ 1,0],[ 1,1] ;
```

Each neighborhood of the family so defined corresponds to a function from the study area, that associates ‘false’ to every location out of those belonging to its specified vicinity, and ‘true’ otherwise. In order to put this functional aspect more explicit, each pair should be replaced by a triple in which the third coordinate indicates the selection state for each location, as in the following improved version:

```
[-1,-1,true],[-1,0,true],[-1,1,true],
[ 0,-1,true],[ 0,0,true],[ 0,1,true],
[ 1,-1,true],[ 1,0,true],[ 1,1,true]
```

By switching some values to ‘false’ other configurations can be obtained such as the following new version for von Neumann’ neighborhood:

```
[-1,-1,false],[-1,0,true],[-1,1,false],
[ 0,-1,true ],[ 0,0,true],[ 0,1,true ],
[ 1,-1,false],[ 1,0,true],[ 1,1,false]
```

Only ‘true’-valued relative locations must be present in specifications, all remaining locations are implicitly valued ‘false’. As the constant values ‘true’ and ‘false’ are just primitive Boolean conditions, then it is also suggestive to allow any Boolean expression to

**Fig. 5** Standard von Neumann and Moore neighborhoods

replace them in specifications, so that arbitrary conditions can be imposed at specific relative location as in the expression below:

```
[-1,-1, use=='forest'],[-1, 0, slope<30],[-1, 1, use=='forest'],
[ 0,-1, slope<30],[ 0, 0, use=='forest'],[ 0, 1, slope<30],
[ 0,-1, use=='forest'],[ 0, 0, slope<30],[ 0, 1, use=='forest']
```

Variables can also be assigned to expressions describing neighborhoods, as illustrated below:

```
N =   [-1,-1],[-1,0],[-1,1],
      [ 0,-1],[ 0,0],[ 0,1],
      [ 1,-1],[ 1,0],[ 1,1] ;
```

Boolean algebra is also extensible to this class of neighborhood specifications so that one can easily build new specifications from existing ones, as illustrated below:

```
N AND bestplace
```

The above expression could be also specified as a list of pertinence conditions to be satisfied at each relative location, as follows:

```
[0,-1, bestplace],[0, 0, bestplace],[0, 1, bestplace],
[0,-1, bestplace],[0, 0, bestplace],[0, 1, bestplace],
[0,-1, bestplace],[0, 0, bestplace],[0, 1, bestplace]
```

The locations' selecting strategy discussed in this section centered in the region concept thus offers a flexible way to express zones and neighborhoods when modeling spatial situations either in a relative or absolute sense. Also the ability to specify different conditions at neighboring locations adds spatial variability to neighborhoods' shape modeling in that each location's vicinity may shape a particular union of 'true'-valued cells.

## 5 Interacting regions and maps

The interactions involving regions and maps can be modeled by means of a local binary operation in which at least one argument is of Boolean type, while the other, and so the result, may assume any valid data type of quantitative or qualitative nature, as defined by the following table:

| * | value | null |
|---|---|---|
| true | value | null |
| false | null | null |

For spatial data types such as images this operator can be extended from number multiplication, assuming the integers '1' and '0' play the role of Booleans 'true' and 'false'. The result of locally applying this operator to image data can be visualized as a new image for which some selected locations keep their original values, while the others become '0'-valued. To illustrate consider the expression:

```
(ndvi > 0.5) * img
```

Its evaluation would select image values at locations with vegetation index greater than '0.5', while remaining locations become '0'-valued. The defined "interacting"

operator has properties very similar to those for ordinary number multiplication, so that we can adopt the same symbol '*' to represent both operators with no syntactic or semantic ambiguity. The notion of a "null" value for image data can be represented by the integer value '0', but this is not always the case for other data types. For instance, the null value in a slopes' map can't be represented by the integer '0' because this is a meaningful slope value, besides, it would be also desirable to have the operator working for qualitative data as illustrated by the following expressions:

```
(ndvi > 0.5) * slope
(ndvi > 0.5) * soils
```

For neighborhood regions specifications the interaction with maps is expressed in the same way used for regions defined by Boolean conditions, however in this case the interaction of each element of a family of neighborhoods with a map is implied. To illustrate consider the following interacting expressions:

```
(distance() < 3) * use
N * img
```

Evaluating the first expression above would result in recording of values from a map represented by variable "use", at locations belonging to the family of circular vicinities of radius 3 units from each location in the study area. In a similar way, the second expression above would record values from data represented by variable "img", at locations specified based on the specification given by the variable "N". Assuming a 'mxn' array representation for the maps involved in expressions, their evaluations would imply in '*m* times *n*' groups of recorded values. In practice, at least for non-parallel computer architectures, these groups will never be represented simultaneously.

An "interacting" expression" can also be assigned to a variable to represent it in other expressions, as illustrated below:

```
good_slope = (ndvi > 0.5) * slope ;
use_around = distance() < 3 * use ;
```

As for any Boolean operation, there are no physical spatial data structures intended to represent the results from evaluating interactions, only pieces of code, resulting from parsing the corresponding textual expression are associated to the variable to be used at runtime (see Section 2).

## 6 Summarizing values by regions

The last step of a non-local operation consists of summarizing values to characterize each location of the study area in terms of the values recorded from locations inside their influence regions. For instance, consider the interacting expression below:

```
(use == 'forest' AND slope >10) * heights;
```

Its evaluation can model the interaction of local data represented in maps associated to variables "use", "slope" and "heights". This interaction can then be

used as argument to summarizing functions such as "average", as illustrated by the typical zonal operation described below.

```
Average ((use == 'forest' AND slope >10) * heights);
```

As neighborhood regions specifications can be combined with any other regions specifications through Boolean operations, the expression above can move into the following neighborhoods operation:

```
Average (N AND landscape == 'forest' AND slope >10) * heights)
```

Typical summarizing functions are simple statistics such as "average", "summation", "maximum" and "majority". These functions just take the samples consisting of values resulting from the interaction of regions and data as arguments so that no prefixing such as "zonal" or "focal" is needed in the language repertoire. Of course one may explore the ability of the language to assign variables to expressions as discussed in previous sections, so that one can write customized expressions such as:

```
Maximum (good_slope)
Average (good_slope * N)
Majority(use_around)
```

As usual, these "summarizing" expressions can also be assigned to variables that may, or may not, be physically represented in the model, as illustrated below.

```
main_use = Majority(use_around);
```

At the implementation level when non-local operations are involved the parsing of summarizing expression will trigger the execution of a particular piece of code intended to derive the interaction between variables and regions. The following version of our example expression of Section 2 may illustrate the situation:

```
(a - Average(b * N))/(a + Average(b * N))
```

In Section 2 the code instruction "push(b)" was intended to feed the second arguments for subtraction and addition, in this new version a summarizing function must do the job. But prior, a local interaction operation must be executed restricted to the set of relative locations specified by variable "*N*". The resulting code new version would then look like:

```
push(a) push(b) push(N) sel average sub push(a) push(b) push(N)
sel average add divide
```

The "sel" instruction above indicates the interaction operation starting point. Lets assume the following neighborhood specification:

```
N = [-1,-1, use=='forest'],[-1, 0, slope<30],[-1,1, use=='forest'],
    [ 0,-1, slope<30],[ 0, 0, use=='forest'],[ 0,1, slope<30],
    [ 0,-1, use=='forest'],[ 0, 0, slope<30],[ 0,1, use=='forest'];
```

Then, depending on the relative location to be characterized the following pieces of code that implement the "equality" and "less than" local comparison operations taking variables "use" and "slope", and constants 'forest' and '30' as arguments, must be executed:

```
push(use) push('forest') eq
push(slope) push(30) lt
```

If distance or direction conditions are to be considered, then evaluations would involve special scanning techniques besides the straightforward line oriented ordering typically used to implement local operations. Other approaches such as spiral and Morton sequences (Sammet 19xx) illustrated in Fig. 6 may be useful.

The approach to map algebra focused in the location and the region concept just states that each location of the study area is always associated to a set that may include one or more locations exerting influence on it. This leads to a modeling paradigm with a minimal set of concepts and a concise set of properties.

## 7 Enhancing the region concept

In our previous discussion on regions, each location in the study area was affected with equal weights by its influencing locations. However there are situations for which different importance degrees are assigned to locations, in particular regarding neighborhood regions. To illustrate consider the gradient (or Sobel) filtering operation used in image processing for edge detection, expressed as:

```
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 );
```

Factors of 2 above indicate double weighing for the values selected by pairs [1, 0], [−1, 0], [0, 1] and [0, −1]. This “weighing” concept is intended to model the extent to which each recorded value must be accounted for in computations. For instance, if the value ‘0’ is interpreted as a “weighting” factor to which specific local values must be considered then it can naturally replace the value ‘false’ in our interacting operator definition of Section 4. Moreover the equivalence among expressions such as “true * number”, “number * true”, or simply “number”, resulting from applying the interacting operator defined in Section 5 to real numbers, also suggests extending our neighborhood’s specifications by allowing any real number to appear as a local weight in the specifications. By mapping the value ‘false’ into the real number ‘0’ and the value ‘true’ into any other real number, we not only maintain the Boolean structuring previously discussed, but also add some arithmetic to this mixed Boolean-quantitative domain. Within this new “weighted” region perspective, the family of neighborhoods involved on the Sobel filtering operation could be stated in terms of the following specifications:

```
   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] ;
```

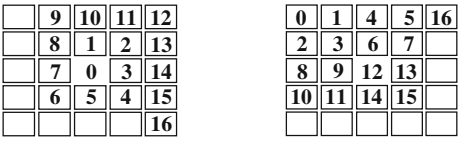



**Fig. 6** Spiral and Morton scanning order for neighborhoods selection

It follows that all selection and weighing involved in the Sobel filtering operation can be described by the following expressions:

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

Using algebraic properties such as the distributiveness regarding subtractions, one can rewrite the expressions above as follows:

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

Next a summarizing step follows as shown below:

```
Sum (im * (down – up))
Sum (im * (right – left))
```

To conclude, the complete Sobel filtering expression can then be stated and assigned to a named variable as follows:

```
Sobel = sqrt((Sum(img*(down-up)))^2+(Sum(img*(right-left)))^2);
```

The example discussed in this section also illustrated meaningful applications of arithmetic additive operations extended to this new enhanced region concept. However multiplicative operations must be considered more carefully to avoid confusion with the interacting operation itself. Actually there are more concerns regarding operations in such a "Boolean-quantitative" domain still left for further investigations.

## 8 Regions and cellular automata

In [4] it is clearly suggested that the rules of cellular automata might be considered as a map algebra, and that some aspects from the theories of formal languages and automata should be explored into the cellular automata context. This is based on a spatial modeling premise in which the geo-referenced location is the central concept so that it is the link between absolute and relative (proximal) spaces. The approach adopted in this research considers CA in a formal language context in which map algebra is actually used to express local conditions to be applied on neighbor locations, in the same way they are used to characterize locations in an absolute spatial context. In both cases the same automata approach is used at the implementation level so that it suggests an easy transition between static and dynamic modeling.

The Life game, invented by mathematician John Conway at Princeton University in 1970 is possibly the simplest instance of a cellular automata model based on rules carefully chosen, some of which may cause cells in a cellular space to "die", while others cause them to "live". Life balances lots of tendencies, making it hard to tell whether a pattern in the cellular space will die out completely, form a stable population, or grow forever [8]. 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.**

The evaluation of the above rules involves interacting a map "*m*" representing the initial configuration of the CA's cell space, with a family of regions "*R*". This results in a family of sets "$m \times R$" intended to record the values associated to each location inside any region specified by "*R*". Figure 7 illustrates the situation assuming the following specification for "*R*":

```
R = [-1,-1],[-1, 0],[-1, 1],
    [ 0,-1],        [ 0, 1],
    [ 1,-1],[ 1, 0],[ 1, 1];
```

Then we apply a summarizing or, using geoalgebra jargon (see [16]), an "influence" function "*I*", to "$m \times R$" that simply counts the elements in the previously recorded samples. This can be visualized as a new map "$I(m \times R)$" associating each location to the number of elements in its associated sample. Finally a new version for map "*m*" is generated in which each location is switched from '1' to '0' or vice-versa depending on conditions involving the number of live neighbors around it. Three maps involved are illustrated by Fig. 8.

Describing Life in our proposed language can be done, with the help of some iterating and control statements, as follows:

```
{
m = Retrieve(name="InitialState");
R = [-1,-1],[-1, 0],[-1, 1],
    [ 0,-1],        [ 0, 1],
    [ 1,-1],[ 1, 0],[ 1, 1];
t = 0;
end = 12;
While (t<end)
{
m =((m==1) AND (2<= I(m * R)<=3)) OR
       ((m==0) AND (I(m * R)==3)) ? 1 otherwise 0;
t = t+1;
}
}
```

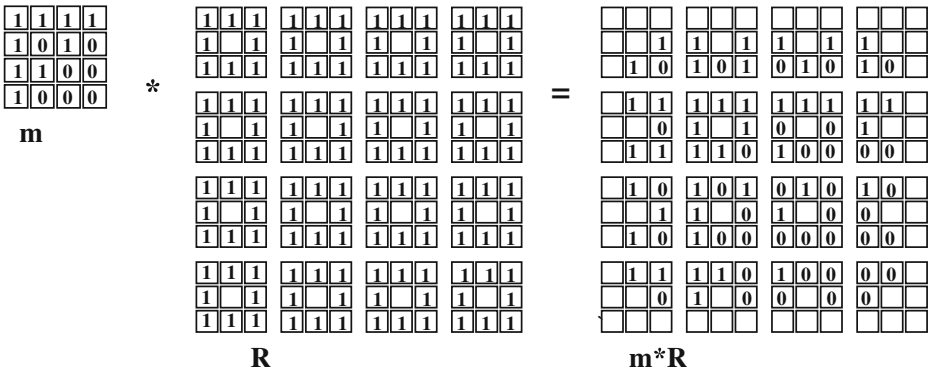



**Fig. 7** Interaction between a map and a family of regions as the base for modeling state change rules for the game Life

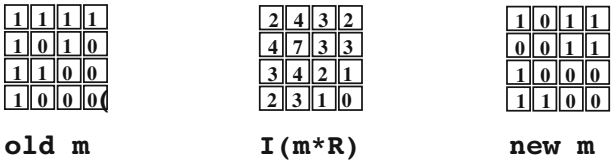



**Fig. 8** Sequence of operations to evaluating state changes based on rules of the game Life

The expression assigned to variable "m" in the above program describes the conditional assignment of the values '0' or '1', depending on a Boolean condition given by the first part of the expression, before the '?' sign. The other parts consist of the two constant expressions separated by the "otherwise" term.

Cellular automata are among the best ways to design and model the full complexity of natural events and processes [18]. In fact we are living now a change of ecological paradigm, from a stable and closed system, without human interference perspective to a unstable, complex and open system where man plays an important role. This new ecological paradigm includes the landscape ecology perspective, in which species answer in different ways to the common landscape properties and across time and scale variations.

The mobility of a species also responds to complex interacting factors such as population's density, resource availability, edge effects, life phase, corridors' existence, matrix dispersal capability, around habitat amount (at several scales), phonological phases of vegetation, predator presence and many others. As the number of variables grows, so does the need for comprehensive means to express the interaction among these variables at the language level. Mobility may change as a function of individual positioning regarding forest patches so that different dispersal abilities for interior and edge positioning must be considered. This can be evaluated in terms of measures and statistics summarized for the individual's dispersal area. For instance, a simple set of rules to characterize an individual position adequacy could be stated as follows:

```
position =
      'interior' :
            Majority (LC * (distance() <= dispersal))
            ==('MF'||'YF')AND
            Minority(LC * (distance() <= dispersal))
            == ('MF'||'YF') ,
      'edge' :
           Majority (LC * (distance()<= dispersal)) ,
      'exterior' :
            otherwise ;
```

At any time two viewpoints must be of concern, one based on cellular regions at different resolutions and the other based on (proximal) regions focused at each individual location. The proximal space (Couclelis et al. 1997) for each individual (represented as a cell at the lowest resolution), may consist of the circular region with radius equal to its dispersal factor, as illustrated in Fig. 9.

Both geoalgebra and the image algebra frameworks lack in providing a way to actually describe their templates and influence sets in a systematic manner, thus strongly relying on classical assumptions on the shape and extent of neighborhoods and zones. Our map algebra language focused on the regions' concept adds more structuring to these contexts by allowing some spatial variability for neighborhood regions given in terms of Boolean expressions that can be evaluated whenever each single neighbor influence needs to be quantified (or qualified).

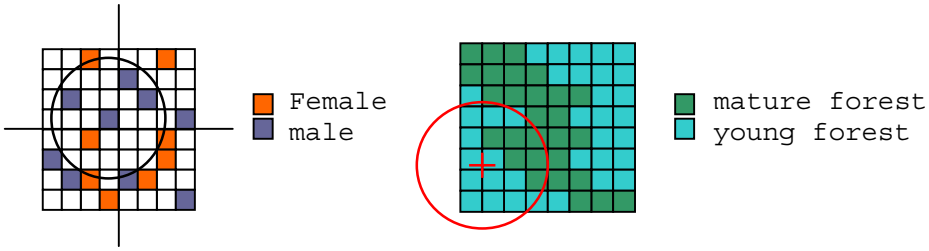



**Fig. 9** Proximal spaces to extract population and landscape information

## 9 Concluding remarks

In this work map algebra has been 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. Furthermore the interaction among these concepts were developed in a natural and consensual way that is consistent with principles of geoalgebra [14] and image algebra [13] theories. We also showed an implementation strategy for basic concepts in these formal approaches based essentially on local operations of classical map algebra to produce layers and other local (Boolean) operations intended to produce regions.

There are multiple intermediate situations between the concepts of zones and neighborhoods that can be modeled following the approach discussed in this work. For instance, one could define neighborhoods in terms of sums of distances to specific locations, or use visibility criteria to specify objects in the landscape, and so on. Behind any regions' specification there should be always a mix of relations to be considered based on the spatial and the attribute domains represented, so that other relational issues, such as "equivalence relations", as well as topology issues, particularly regarding "compact" topological spaces must be addressed in the future.

Expressions that describe interacting operations follow the rules of a context-free grammar similar to those for arithmetic and Boolean expressions, so that their interpretation and parsing may be integrated in the same pushdown automata strategy [9] already adopted for the other classes of algebraic expressions in this map algebra approach. By adopting a compromise between languages and automata theories; syntax, semantic, understanding and implementation issues can be formally tied together so that all mathematical flexibility for writing expressions regardless of their complexity may be explored in modeling, thus avoiding some drawbacks of the classical map algebra paradigm. Between the parsing and running phases of a program, optimization issues must be addressed in future works so that performance can meet dynamic model requirements. Evaluating language requirements for environment modeling disciplines such as landscape ecology, and how the ideas in this paper may fit then is another research front to be explored.

As the concept of region adopted here is based on language expressions implemented as pushdown automata, 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 of pushdown automata implementations, that may possibly ease the task of extending map algebra to parallel architectures and distributed environments.

## References


1. J.K. Berry. “Fundamental operations in computer-assisted map analysis,” International Journal of Geographic Information Systems, Vol. 2:119–136, 1987.
2. G. Camara, U.M. Freitas, and J.P. Cordeiro. Towards an Algebra of Geographical Fields. Campinas, SP: SIBGRAPI, 1994.
3. G. Camara, R.C. Souza, U.M. Freitas, and J.C. Garido. “SPRING: integrating remote sensing and gis with object-oriented data modeling,” Computers and Graphics, 15–16, 1994.
4. H. Couclelis. “From cellular automata to urban models: new principles for model development and implementation,” Environment and Planning: Planning & Design, Vol. 24:165–174, 1997.
5. H. Couclelis. “Chapter 2: Modeling frameworks, paradigms, and approaches,” in K.C. Clarke, B.E. Parks and M.P. Crane (Eds.), Geographical Information Systems and Environmental Modeling. New York: Longman & Co., 2000.
6. K.K.L. Chan and D. White. Map Algebra: An Object Oriented Implementation. Proceedings, International Geographic Information Systems (IGIS) Symposium: The Research Agenda. Arlington, Virginia, November 1987.
7. C. Dorenbeck and M.F. Egenhofer. Algebraic Optimization of Combined Overlay Operations. Auto-Carto 10: Technical Papers of the 1991 ACSM-ASPRS Annual Convention. Baltimore: ACSM-ASPRS, 6, 296–312, 1991.
8. M. Gardner. “Mathematical games: The fantastic combinations of John Conway’s new solitaire game ‘life’,” Scientific American, Vol. 223:120–123, 1970.
9. J.E. Hopcroft and J.D. Ullman. Formal Languages and Their Relation to Automata. Reading, MA: Adisson-Wesley, 1969.
10. J. Mennis, R. Viger, C.D. Tomlin.“Cubic map algebra functions for spatio-temporal analysis,” Cartography and Geographic Information Systems, Vol. 30–1:17–30, 2005.
11. D. Pullar. “MapScript: A map algebra programming language incorporating neighborhood analisys,” GeoInformatica, Vol. 5–2:145–163, 2001.
12. D. Pullar. “A modeling framework incorporating a map algebra programming language,” in A.E. Rizzoli and A.J. Jakeman (Eds.), Proceedings, Biennial Meeting of the International Environmental Modeling and Software Society. Arlington, Virginia, 2002, November.
13. G.X. Ritter, J. Wilson, J. Davidson. “Image algebra an overview,” Computer Vision, Graphics and Image Processing, Vol. 49:297–331, 1990.
14. M. Takeyama. Geoalgebra: A mathematical approach to integrating spatial modeling and GIS. PhD dissertation, Department of Geography, University of California at Santa Barbara, 1996.
15. M. Takeyama. “Building spatial models within GIS through geoalgebra,” Transactions in GIS, Vol. 2:245–256, 1997.
16. M. Takeyama and H. Couclelis. “Map dynamics: integrating cellular automata and GIS through Geoalgebra,” International Journal of Geographical Information Science, Vol. 11:73–91, 1997.
17. D. Tomlin. Geographic Information Systems and Cartographic Modeling. Englewood Cliffs, NJ: Prentice Hall, 1990.
18. H.H. Wagner and M.J. Fortin. “Spatial analysis of landscape: Concepts and statistics,” Ecology, Vol. 86–8:1975–1987, 2005.
19. C.G. Wesseling, D.J. Karssenberg, P.A. Burrough, and W.P.A. Van Deursen. “Integrated dynamic environmental models in GIS: The development of a dynamic modelling language,” Transactions in GIS, Vol. 1:40–48, 1996.
20. R. White and G. Engelen. “Cellular dynamics and GIS: modeling spatial complexity,” Geographical Systems, Vol. 1:237–253, 1994.

**João Pedro Cerveira Cordeiro** is a senior Technologist in Computer Science in Brazil's National Institute for Space Research (INPE) since 1990, working on GIS software and application development since then. He is running a PhD at the Institute for Air-Space Technologies (ITA) in São José dos Campos, Brazil, since 2004. He holds a MSc in Computer Science from INPE (1989) and a Bachelor in Mathematics from the Catholic University of Rio de Janeiro (PUC-RJ). His interests are on dynamic models and the role mathematical structures such as algebra and topology can play in attending their descriptive language requirements.

**Gilberto Câmara** is Director of Brazil's National Institute for Space Research (INPE) for the period 2006 to 2010. His research interests include: geographical information science and engineering, spatial databases, spatial analysis and environmental modelling. He has published more than 140 full peer-reviewed papers on journals and conferences, and he is also involved in the development of SPRING, a free object-oriented GIS, and of TerraLib, an open source GIS library.

**Ubirajara Moura de Freitas** is a Geoprocessing Research and Application Manager at FUNCATE, a foundation for research, development and application of spatial technologies particularly focused on urban, ecological, social and economic applications. He holds his MSc in Computer Science from INPE (1981), being one of the pioneers of GIS science and development in Brazil. His research interest now includes spatial database issues with focus on the development of Corporate GIS.

**Felipe Almeida** is researcher at the Institute for Air-Space Technologies (ITA) in São José dos Campos, Brazil. His research interests include: mathematical modeling for dynamic systems. He holds a PhD in Computer Science from University of Kent at Canterbury, where he worked in program parallelization using Transputer based supercomputers.