An algebra for modeling and simulation of continuous spatial changes

André Fonseca Amâncio[1], Tiago Garcia de Senna Carneiro[1]

[1]Department of Computing – Federal University of Ouro Preto (UFOP) – Ouro Preto – MG – Brazil.

afancio@gmail.com, tiagogsc@gmail.com

Abstract. *Continuous change models are commonly based on the Systems Dynamics paradigm. However, this paradigm does not provide support for an explicit and heterogeneous representation of geographic space, nor its topological (neighborhood) structure. Therefore, using it in modeling spatial changes still remains a challenge. In this context, this paper presents an algebra that extends the Systems Dynamics paradigm to the development of spatially explicit models of continuous change. The proposed algebra provides types and operators to represent flows of energy and matter between heterogeneous regions of geographic space. To this end, algebraic sets of operations similar to those in Map Algebras are introduced, allowing the representation of local, focal and zonal flows. Finally, case studies are presented to evaluate the usefulness, expressiveness and computational efficiency of the proposed algebra.*

1. Introduction

Continuous spatial changes describe continuous flows of energy or matter between regions of geographic space. Although the System Dynamics paradigm (Forrester 1961, Meadows 2008) is widely used for modeling continuous changes, it does not provide support for an explicit and heterogeneous representation of geographic space and its topological structure (neighborhood). For this reason, it needs to be extended to construct spatially explicit models [Parker et al. 2001] of continuous spatial changes, as of interactions between society and nature.

In this work, the types and operators present in map algebras [Tomlin 1990, Karssenberg et al. 2001, Cordeiro et al. 2009, Schmitz et al. 2013] are used to extend the Systems Dynamics paradigm. Map algebras generally define three sets of operations, defined as proposed by Tomlin (1990): (a) Local operations - whose the value of a location in the output map is calculated from the values of that location in the input maps; (b) Focal operations - whose the value of a location in the output map is calculated from the neighborhood values of that location in the input map; And (c) zonal operations - whose the output values summarize values of regions (neighborhoods) defined on the input map.

Models based on Map Algebra and System Dynamics have completely different syntax, semantics, and execution flows, creating challenges for combining these paradigms. Models based on map algebras are finite sequences of algebraic expressions that cause discrete changes in space. The operations are performed synchronously and immediately, one after the other, as they appear in the model code. Models based on System Dynamics are formulated in terms of differential equations and use

infinitesimally small time-steps for numerical integration procedures [Kelly et al. 2013] that simulate continuous changes over time. Since in many models it is common to find interdependence between the several differential equations, they need to be computed simultaneously to avoid error propagation. Therefore, the equations are invoked asynchronously, that is, when they appear in the model code they are only instantiated and, they are executed only after all of them have been invoked. In addition, the combination of these paradigms needs to deal with spatio-temporal dependence generated by feedback loops [Schmitz et al. 2013]. Feedback loops generate data dependencies that need to be resolved for the coherence of simulations, that is, during simulations intermediate states of the variables need to be updated and consisted to avoid error propagation. Finally, the modeling activity requires the modeler to be a specialist in the model application domain and in computer programming to be able to code it in the form of algebraic operations or differential equations. Currently, the following questions remain: *How to promote the expressiveness of modeling tools for continuous and spatially explicit change simulation? How to combine different behavioral, spatial and temporal representations in those tools, in a transparent way for the modeler?*

In this context, this work proposes and evaluates through case studies an algebra for the development of spatially explicit models of continuous changes that take place in the geographic space. This algebra extends the Systems Dynamics paradigm with types and operators that allow the representation of energy and matter flows between heterogeneous regions of geographic space that could be connected by distinct topological relationships. To this end, we introduce sets of algebraic operations similar to those in Map Algebras, allowing the representation of local, focal and zonal flows. The case studies evaluate the usefulness, expressiveness and computational efficiency of the proposed algebra.

This article is structured as follows. Section 2 presents the related works. In section 3, the algebra is described as a generic instrument for modeling continuous spatial changes. Section 4 explains how the algebra works. In Section 5, we describe case studies of simplified models to evaluate the usefulness, expressiveness and computational efficiency of an implementation of this algebra in the TerraME tool [Carneiro et al. 2013]. Finally, the discussion of the benefits of using algebra concludes this work.

2. Related work

Most extensions of the Systems Dynamics paradigm only replicate systems-based models in discrete and regular partitions of space to deal with spatial changes, interconnecting stocks present in these models through spatial neighborhoods. All changes occur instantly and simultaneously (snapshot). Stocks in a locality are linked to stocks of same name in neighboring localities, simulating processes of spatial diffusion or mobility. Generally, the lateral flows are controlled by only one rate fixed by the modeler, with no way to represent heterogeneous lateral flows. The neighborhoods are of stationary topology, typically Moore or von Neumann. This type of approach was called Spatial System Dynamics (SSD) [Ahmad and Simonovic 2004]. Some authors consider it a simplistic extension of Systems Dynamics, it has a slow execution and is only appropriate for feedback loops between two models [Swinerd and McNaught 2012, Sahin and Mohamed 2014]. Therefore, the limitation of these approaches in dealing

with heterogeneous, non-stationary and anisotropic spaces, under different spatial and temporal scales, has motivated several innovations [Elsawah et al. 2017]. The Spatial Modeling Environment (SME) [Maxwell and Costanza 1997] platform was pioneer in this sense, by representing vertical flows between diverse representations of space and allocating different models in different regions. To represent more complex interactions the literature presents approaches based on Individual-Based Modeling [Vincenot et al. 2011], Hybrid Simulations Involving Agent-based [Swinerd and McNaught 2012] and Discrete Event Simulation [Morgan et al. 2017].

On the other hand, several papers propose generalizations and extensions of Map Algebra to represent spatial processes. Camara et al. (2005) present a generalized Map Algebra that uses spatial topological and directional predicates. Frank (2005) discusses how Map Algebra can be formalized for programming, extending it to deal with spatiotemporal data. Cordeiro et al. (2009) extend the Map Algebra by proposing the concept of Geoalgebra with generalizations for describing layers, regions, neighborhoods and zones. Schmitz et al. (2013) combine the concepts of Map Algebra and Model Algebra for the coupling of model components. Camara et al. (2014) introduce the concept of Fields for representations of continuous spatiotemporal variables, demonstrating its use in the construction of a novel Map Algebra. Silva and Carneiro (2016) developed an algebra for models based on spatially explicit agents. However, the authors of this work have not found in the literature algebras that extend the System Dynamics paradigm to operate directly on maps, or that extend Map Algebra to represent continuous flows of energy or matter.

PCRaster approach extends map algebra for the development of spatio-temporal environmental models [Burrough 1998, Wesseling et al. 1996]. However, it does not explicitly represent the flow operator from System Dynamic Theory in its algebra. It is assigned to the modeler the responsibility to implement a set of operations (map = map +/- change()) to simulate outflows from one storage or/and inflows to another. Those operations are interpreted as difference equations computed only once at each simulation time step, no numerical integration methods are applied. In contrast, we propose an extension of System Dynamic Theory to the development of geospatial models, with an explicit representation of flow operations, reducing the modeler responsibility of properly implement, simulate and compute flows of energy.

Regarding usability, Frank (2005) and Silva and Carneiro (2016) describe algebras as facilitators for model specification, since modelers did not need to become experts in different languages and modeling tools to describe models. The use of a given algebra allows the description of model components focused on the model objectives and not on its implementation [Schmitz et al. 2013]. Cardelli (1997) reinforces the idea that simplifications in models reduce the effort to understand it by future applications and prevent possible mistakes made by users. Frank (2005) says that the descriptions of algebra processing steps can be formalized and optimized. Finally, Schmitz et al. (2013) present evidence that the automation of the interaction routines between space regions, through algebra native operators guarantees the integrity and improves the readability of the models.

3. Types of Algebra Operators

The algebra proposed in this work is a generic tool for modeling continuous spatial changes, it can be implemented in several tools and languages according to the ideas presented in Silva and Carneiro (2016). Here, algebra components are specified from abstractions of their operators [Frank 1999].

The algebra operators act over spatial types that represent stocks of energy or matter (attributes) localized in the geographical space. Space topology (neighborhood and proximity relations) is also represent allowing diffusive flows. Operators are subdivided into creation operators, responsible for creating and relating types, and flow operators, responsible for defining how changes occur (behavioral rules) in relation to time and space. Finally, the execution of operator coordinates the simultaneous and interleaved execution of changes during simulation.

3.1. Spatial types

There are four spatial types present in the algebra: *Cells*, *CellularSpaces*, *Trajectories* and *Neighborhoods*. There are two basic spatial types (Figure 1 (a)): cell and neighborhood. A *Cell* represents the stocks of a space location and contains a list of attributes and a list of neighboring cells. Neighborhoods represent the space connectivity and can represent areas of influence, adjacency or proximity relations. Moore and von Neumann [Couclelis 1997] neighborhoods are often used for spatially explicit modelling.

CellularSpaces and *Trajectories* are collections (Figure 1 (b)), that is, they represent sets of entities of the same type, in this case cells. *CellularSpaces* represent regions in the geographic space. All cells in a *CellularSpace* are composed by the same set of attributes, which can assume distinct values during simulation. *Trajectories* are collections that select and order cells from a *CellularSpace*, allowing the modeler to filter the cells on which operators must focus and to establish the order in which those operators must traverse the *CellularSpace* performing changes.

Cell: (name, attributes, neighbors)
- *name : String*
- *attributes: [Attribute]*
- *neighbors: SpatialNeighborhood*

SpatialNeighborhood : (type, d, self, cells)
- *type: String*
- *d: (width: Number, lenght: Number)*
- *self: Boolean*
- *cells: [Cell]*

CellularSpace: (cells, dimension)
- *cells: [Cell]*
- *dimension: (width: Number, lenght: Number)*

Trajectory: (cs, selectFunction, sortFunction, cells)
- *cs: CellularSpace*
- *selectFunction: Boolean Function (Cell)*
- *sortFunction: Boolean Function(Cell, Cell)*
- *cells:[Cells]*

Figure 1. (a) Definition of the neighborhood cell; (b) Definition of collections.

3.2. Creation Operators

Creation operators are intended to ensure that all basic types belong to, at least, one collection. In this way, after creating basic types, the modeler needs to relate them to a

collection in order to use than as operands in other operators. Figure 2 presents the definition of creation operators.

The *CellularSpace* creation operator uses a *Cell* instance that provides the archetype for cloning the other cells it aggregates, the size of the *CellularSpace* determines the number of *Cells* that will be created. A *SpatialNeighborhood* is created from a *CellularSpace*, the type and dimensions of the neighborhood (leght, width), and from the definition of whether or not the cells are self-contained in their own neighborhood structures.

createCell: Cell Function (name, attributes) • *name : String* • *attributes: [Attribute]* *createCellularSpace: CellularSpace Function(cell, dimension)* • *cell: Cell* • *dimension: (width: Number, lenght: Number)*	*createTrajectory: Trajectory Function(cs, select, sort)* • *cs: Cellular Space* • *select: Boolean Function(Cell)* • *sort: Boolean Function(Cell, Cell)* *createSpatialNeighborhood: SpatialNeighborhood Function(cs, type, d, self)* • *cs: CellularSpace* • *type: String* • *d: (width: Number, lenght: Number)* • *Self: Boolean*

Figure 2. Definition of creation operators

3.3. Flow operator

In this algebra, the *Flow* operators (FLOW) use only collections (*CellularSpace* and *Trajectory*) as operands. It (Figure 3) represents continuous transference of energy between regions of space. The differential equation supplied as the first operator parameter determines the amount of energy transferred between regions.

FLOW (f(), a, b, step, Collection1, "Attribute", "Neight1", Collection2, "Attribute", "Neight2")

- *f(): Differential equation that describes, as a function of one or two parameters, the rate of change (point derivative) of energy f (t, y) at time t, where t is the simulation current instant time, and y is the past value of the rate of change f ().*
- *a: Number - Beginning of the integration interval.*
- *b: Number - End of integration interval.*
- *step: Number - An infinitesimal time interval used in numerical integration.*
- *Collection1: Cellular Space or Trajectory - A collection of cells that will be used to calculate and subtract flow output.*
- *Attribute1: String - Name of the attribute of the cells contained in the collections over which the flow will operate.*
- *Neight1: Neighborhood - Neighborhood name defined on the source collection of the energy flow. Optional.*
- *Collection2: Cellular Space or Trajectory - Target collection of energy flow.*
- *Attribute2: String - Name of the attribute of the cells contained in the collections of the energy flow.*
- *Neight2: Neighborhood - Neighborhood name defined over the recipient collection of the energy flow. Optional.*

Figure 3. Flow Operator Definition

Flow operations are classified as local, focal and zonal [Câmara et al. 2014] and their semantics depend on the parameters reported at the moment they are invoked, as described in Table 1 and illustrated in Figure 4.

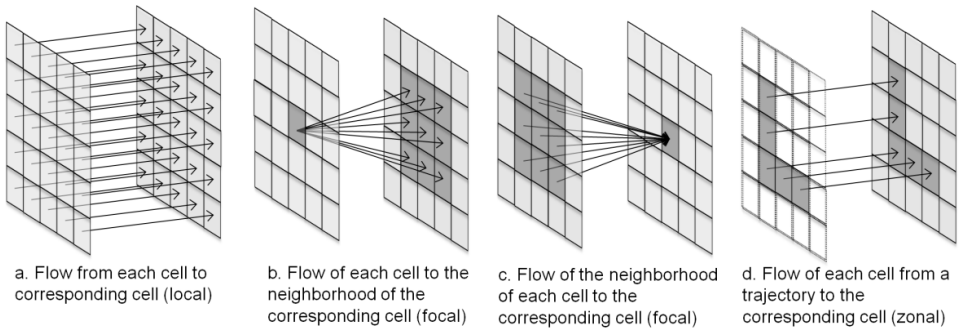

Figure 4. Flow operator examples between two colections

Table 1 - Behavior rule, syntax and semantic definition of the flow operator

RULE	SYNTAX	SEMANTICS
Flow local execution Rule: From Cell To Cell	*flow(Collection, Collection)* ● *Collection* ● *Collection*	Each cell in a cellular space transfers part of its attribute stock at a rate defined by f (t, y) to the spatially corresponding cell attribute of another cellular space, Figure 4 (a). Example: precipitation of cloud water to ground.
Flow focal execution Rule: From Cell To Neigth Of Cell	*flow(Collection, Collection, Neigth)* ● *Collection* ● *Collection* ● *Neight*	Each cell in a cellular space transfers part of its attribute stock at a rate defined by f (t, y) to the attributes of cells in the neighborhood of the spatially corresponding cell of another cellular space, Figure 4 (b). Example: Heat dispersion in fire propagation modeling.
Flow focal execution Rule: From Neight Of Cell To Cell	*flow(Collection, Neight, Collection)* ● *Collection* ● *Neight* ● *Collection*	Each cell from neighborhood of a cell in a cellular space transfers part of its attribute stock at a rate defined by f (t, y) to the cell attribute spatially corresponding to the central cell of the neighborhood of another cellular space, Figure 4 (c). Example: Condensation of water in clouds.
Flow execução zonal Rule: From Selected Cell To Cell	*flow(Trajectory, Collection)* ● *Trajectory* ● *Collection*	Each cell in a trajectory transfers part of its attribute stock at a rate defined by f (t, y) to the spatially corresponding cell attribute of another cellular space, Figure 4 (d). Example: Evaporation of water from a river to clouds.

Flows from the collection A to the collection B are calculated only for cells in intersection A ∩ B. Figure 5 illustrates the possible topological relations between collections and the *Flow* operator semantics, named by Egenhofer and Herring (1993) as: a. equal, b.contains, c.inside and d.overlap.

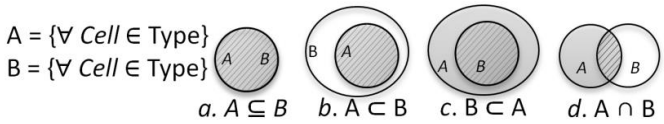

Figure 5. Venn diagram of the topological relation between two collections

Due to space restrictions, the semantics of some *Flow* operators are not detailed in the Table 1, such as flows between distinct trajectories (zonal - example: flow of water from the mainland to the ocean at a beach), or flows from a trajectory to its neighborhood (zonal and focal composition - example: heat flowing from fire border), among others. In the *Flow* operator, it is possible to construct several combinations of collection and neighborhood parameters, both in the source or destination of flows.

3.4. Execute operator

The *Execute* operator described in Figure 6 starts the simulation execution. The simulation will run until the simulation clock reaches the time received as parameter

(*finalSimulationTime*). All flows have the definition of its integration interval, defined by the lower time and upper time limits. The lowest time limit for all flows is used as the initial simulation time.

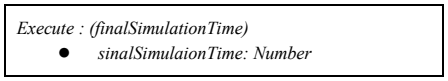

Figure 6. Definition of the execution operator

4. Simulation execution and its implementation in TerraME

The proposed algebra simulator was implemented based on the temporal types of the TerraME platform: Timer and Event (Figure 7). Timer is a discrete event scheduler that operates according to Discrete Event Driven Simulation (DEVs) [Wainer 2009]. It maintains a queue of chronologically ordered events and the current time record of the simulation. Events are instants in the simulated time in which the modeler the TerraME platform performs input and output operations, or computations defined by the modeler. Events are defined by the instant, periodicity, final instant, and action parameters. The instant parameter determines the moment in the simulation in which the event must occur, triggering an action defined by the modeler. The periodicity determines the instant that the event will occur again. The final instant (finalInstant) determines when the event will cease to occur. The action is a function that implements the behavioral rules of the model or commands for TerraME to load, view, and store data. The return value of an action is used as a stop condition, if true the event returns to the Timer queue at the position determined by its periodicity (event.instant = event.instant + event.periodicity), otherwise the event is permanently canceled.

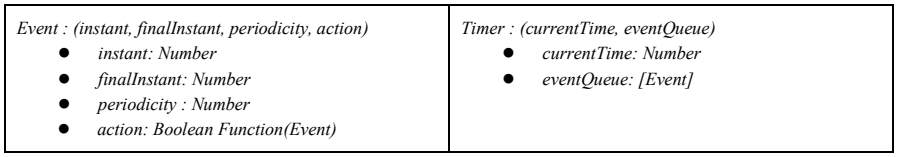

Figure 7. Definition of temporal types

During the simulation, the events are removed from the queue, the simulation current time is updated (currentTime = event.instant), and then the event action is performed. Eventually, the event will be rescheduled if its action returns true.

At the beginning of the simulation, all collections created by the modeler are synchronized through the TerraME's synchronize() function. That is, temporary copies of all cells in each collection are created, recording their immediate state. During the simulation, all readings are performed on the temporary copies of the attributes and the writes are performed directly on the attributes. This strategy ensures that all computations start from the same shared and consistent value, ensuring consistency of the simulations.

The flow operator is implemented according to Algorithm 1 that evolves in two stages: (1) **Flow Execution** - Behavioral rules (BehavioralRules) are executed, that is, TerraME iterates over all cells of the involved collections, applying the differential equations (flows) defined by the modeler that receive the temporary values as parameters, the results of the equations are written directly on the attributes of cells; (2) **Synchronization** - Temporary copies of the cells of the collections affected by the flow

are updated instantly, causing the changes to be persisted and to be noticed by the next computations. All events present in the algorithm remain re-queued until the end time of the simulation is reached (timer.currentTime == finalSimulationTime).

Algorithm 1 FLOW

```
1: function FLOW(f(), a, b, step, colle1, attr1, neig1, colle2, attr2, neig2)
2:     –FLOW EXECUTION                                   return currentTime < b
3:     timer.add( Event(a, b, step, BehavioralRule()))
4:     –SYNCHRONIZATION
5:     timer.add( Event(a, b, step, synchronize(colle1, colle2)))
6: END FLOW
```

5. Case study

Three case studies are used to evaluate the usefulness, expressiveness, and computational efficiency of the TerraME implementation. Case study 1 uses a simplistic "Hello World" model to simulate fire propagation in a forest. Case study 2 simulates the water cycle exemplifying operations frequently used in the representation of continuous spatial changes. The case study 3 evaluates the response time of the simulator implemented in this work. More detailed descriptions, as well as other examples and codes for reproduction of case studies can be found in [ExtraCases 2017].

5.1. Case Study 1

Fire Spread model is composed of a cellular space (lines 3-4) and a von Neumann neighborhood (line 5) through which fire will propagate. Each cell has the attribute heat that represents the thermal energy stored in it, initially equal to 0 (green). A cell is considered to be burning (brown) if its stock is greater than zero. A random fire starting point is created (line 6), whose value is 1. The flow operator (line 7) simulates heat going from burning cells to their neighbors according to the exponential differential equation defined in line 2. Figure 8 shows a series of images with the simulation result.

Algorithm 2 FireSpred Model

```
1: dispersionRate = 0.99
2: function dispersion (t, stock) return dispersionRate * stock
3: createCell(groundslice, heat = 0)
4: createCellularSpace(ground, groundslice, 50)
5: createSpatialNeighborhood(groundNeight, ground, vonneumann, nil, true)
6: ground.RandomCell(heat = 1)
7: FLOW(DISPERSION, 1, 100, 1, ground, heat, nil, ground, heat, groundNeight)
8: Execute(100)
```

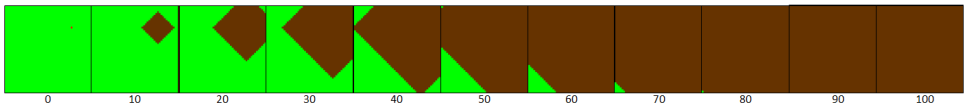

Figure 8. Heat spread over the cellular space. Green (inert), Brown (Burning).

5.2. Case Study 2

The simplified water cycle model (Figure 9) is composed of four flow operations: (1) Cloud water precipitation to the soil - local flow; (2) Evaporation of soil water to the clouds (with water vapor dispersion) - focal flow; (3) Surface runoff of soil water through neighborhood - focal flow; and (4) Condensation of water in the clouds - focal flow.

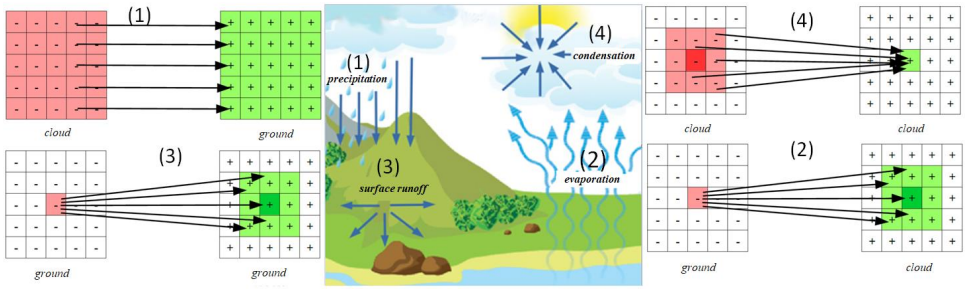

Figure 9. Illustration of water cycle operations

In this case study, both cloud and soil are represented by 5x5-sized cellular spaces, which work as water stocks. Algorithm 3 presents the complete model code. All flows defined in lines 16 to 19 have different start and end times. However, they use equal step intervals, 1. These flows have behavior governed by exponential differential equations defined in lines 5 to 8, whose rates are defined in lines 1 to 4.

Algorithm 3 Water Cicle Model

1: $precipitationRate = 0.2$
2: $evaporationRate = 0.1$
3: $surfacerunoffRate = 0.3$
4: $condensationRate = 0.5$
5: function precipitation $(t, stock)$ return $precipitationRate * stock$
6: function evaporation $(t, stock)$ return $evaporationRatee * stock$
7: function surfacerunoff $(t, stock)$ return $surfacerunoffRate * stock$
8: function condensation $(t, stock)$ return $condensationnRate * stock$
9: createCell$(groundslice, water = randon())$
10: createCellularSpace$(ground, groundslice, 100)$
11: createSpatialNeighborhood$(groundNeight, ground, moore, nil, true)$
12: createCell$(cloudslice, water = randon())$
13: createCellularSpace$(cloud, cloudslice, 100)$
14: createSpatialNeighborhood$(cloudNeight, cloud, vonneumann, nil, true)$
15: createSpatialNeighborhood$(cloudNeight5x5, cloud, nil, 5, true)$
16: FLOW$(precipitation, 2, 7, 1, cloud, water, nil, ground, water, nil)$
17: FLOW$(evaporation, 5, 16, 1, ground, water, nil, cloud, water, cloudNeight)$
18: FLOW$(surfacerunoff, 15, 19, 1, ground, water, nil, ground, water, groundNeight)$
19: FLOW$(condensation, 5, 16, 1, cloud, water, cloudNeight5x5, cloud, water, nil)$
20: Execute(20)

Flow operator 1 at line 16 transfers water from the cloud to soil, simulating rainfall. Flow operator 2 at line 17 simulates evaporation, transporting water from soil to cloud, so that each cell receives a proportion of water proportional to the weights they

have in the neighborhood. Flow operator 3 at line 18 simulates the water surface runoff in the soil, transporting water from a cell to its neighbors. Finally, flow operator 4 at line 19 simulates the condensation of water in the cloud, transferring water from neighboring cells to the central cell.

Figure 10 and Figure 11 graphically display the volumes of water stored in the cloud and soil during simulation. Arrows indicate the start points of flows between cellular spaces. In Figure 11, precipitation during instants 2 to 7 causes ground darkening and cloud whitening. In Figure 10, from moment 5, evaporation reduces the slope of curves by combining its effects with the precipitation. After instant 7, continuous evaporation causes cloud darkening and ground whitening in Figure 11. The surface runoff (instants 15 to 19) and condensation (instants 5 to 20) make homogenous the water stocks in the cells, observed in Figure 11.

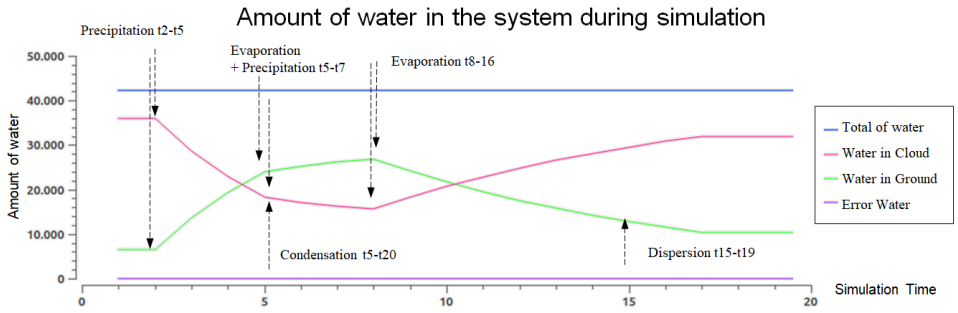

Figure 10. Graph representing the total water quantity of the model

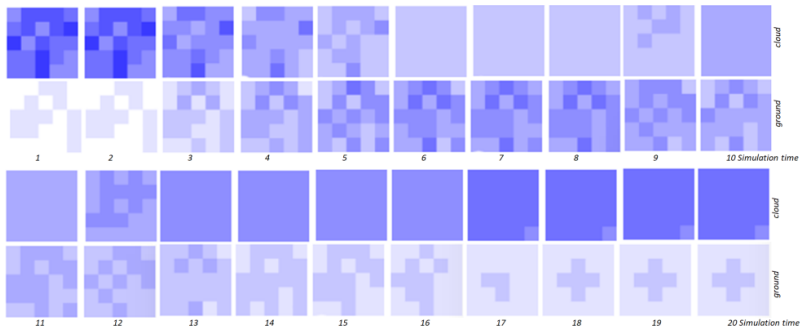

Figure 11. Representation of the quantity of water contained in each cell of ground and cloud cellular spaces according to simulation time

5.3. Case study 3

Three abstract models were used to evaluate the computational efficiency of the implementation of algebra developed in this work, all have flows based on exponential differential equations: (1) Local - Containing a local flow; (2) Focal - Containing a focal flow; and (3) Case Study 2 - Containing the 4 flows described in Algorithm 3. Simulations were performed on the Ubuntu 12.04 operating system on an Intel® Xeon (R) CPU E5620 2.40GHz x8 32GB memory. The graph in Figure 12 shows the CPU time consumed by the simulation during model execution for cell spaces containing up to two million cells.

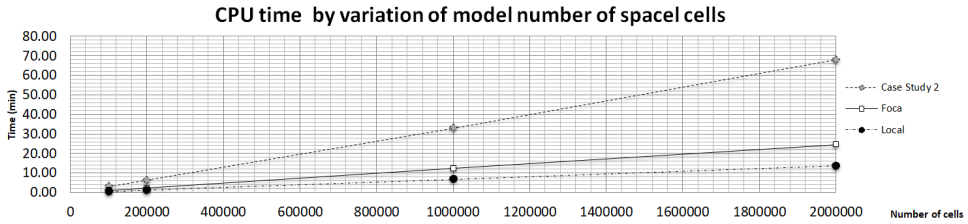

Figure 12. (a) CPU chart for simulations of large cell spaces

6. Final considerations

This paper proposes an algebra for the development of spatially explicit models of continuous changes that evolve in geographic space. This algebra extends the Systems Dynamics paradigm by introducing a set of algebraic operations similar to those in Map Algebra, allowing the representation of local, focal, and zonal flows. Experiments demonstrated how the algebra can be easily used to model and simulate scenarios containing several simultaneous and interleaved energy flows between heterogeneous regions of space. The algebra has good expressiveness and is able to concisely represent models where there are dependences between variables of several differential equations, that is, feedback loops. Briefly, the algebra contributions can be listed as:

1. Allowing the definition of rules of behavior in a declarative way;
2. Providing operators that act on a high level of abstraction, which use collections of cells as operands;
3. Allowing the representation of local, focal and zonal spatial flows;
4. Shifting the modeler's focus from model implementation to its conception and design, since operators encapsulate implementation difficulties;
5. Allowing to model and to simulate changes involving spatiotemporal discretizações of different scales (extent and resolution).

These contributions facilitate the modeling and simulation of continuous spatial changes by reducing the programming fundamentals required during model development, reducing errors arising from implementation of feedbacks, synchronization of simultaneous flows and mechanisms to avoid the propagation of errors due to numerical integrations methods. The results also show that it is possible to use personal computers to simulate flows between millions of cells in a reasonably time. The simulation times grow linearly with the number of cells. Future work includes evaluating the use of this algebra for modeling and simulation of other models found in literature and improving current implementation to simulate large-scale models over high-performance hardware architectures.

References

Aguiar, A. P. D. de, Câmara, G., Monteiro, A. M. V., Souza, R. C. M. de. (2003) Modelling Spatial Relations by Generalized Proximity Matrices, In: V Simpósio Brasileiro de Geoinformática – GeoInfo 2003, Campos do Jordão, SP, Brasil.

Ahmad, S., Simonovic, S.P., (2004). Spatial system dynamics: new approach for simulation of water resources systems. J. Comput. Civil Eng. 18, 331e340.

Burrough, P.A. (1998). Dynamic Modelling and Geocomputation. In: Geocomputation:

A Primer. Edited by P. A. Longley, S.M. Brooks, R. McDonnell, B. Macmillan. John Wiley & Sons Ltd.

Busch, J. (2013). Continuous Simulation with Ordinary Differential Equations. Seminar paper, University of Hamburg. Department of Informatics. Scientic Computing.

Câmara, G., Palomo, D., de Souza, R. C. M., & de Oliveira, O. R. F. (2005). Towards a generalized map algebra: Principles and data types. In GeoInfo (pp. 66-81).

Cardelli, L. (1997). Type Systems. Handbook of Computer Science and Engineering. A. B. Tucker, CRC Press: 2208-2236.

Carneiro, t. G. S., Maretto, e. V., Câmara, g. (2008) Irregular Cellular Spaces: Supporting Realistic Spatial Dynamic Modeling over Geographical Databases. In: Simpósio Brasileiro de GeoInformática, Rio de Janeiro, RJ. Simpósio Brasileiro de GeoInformática. 1: 1, 2008. v.1. p.1 - 1

Carneiro, T. G. S., Câmara, G. (2009) An Introduction to TerraME. INPE Report, 2009, version 1.2. Available in: <http://www.terrame.org/>

Carneiro, T. G. S., DE Andrade, P. R., Câmara, G., Monteiro, A. M. V., Pereira, R. R. (2013) An extensible toolbox for modeling nature–society interactions, Environmental Modelling & Software, Volume 46, Agosto 2013, Pg. 104-117.

Cordeiro Cerveira, J. P., Câmara, G., Moura de Freitas, U., & Almeida, F. (2009). Yet another map algebra. *Geoinformatica*, *13*(2), 183-202.

Couclelis. H. (1997). From cellular automata to urban models: new principles for model development and implementation, Environment and Planning: Planning & Design, Vol. 24:165–174, 1997.

Egenhofer, M. J., & Herring, J. (1990). Categorizing binary topological relations between regions, lines, and points in geographic databases. The, 9(94-1), 76.

Elsawah, S., Pierce, S. A., Hamilton, S. H., Van Delden, H., Haase, D., Elmahdi, A., and Jakeman, A. J. (2017). An overview of the system dynamics process for integrated modelling of socio-ecological systems: Lessons on good modelling practice from five case studies. Environmental Modelling and Software, 93, 127-145.

ExtraCases (2017) Extra case study of an algebra for modeling and simulation of continuous spatial changes. Available in: < http://bit.ly/2gCmXrg>.

Forrester, J.W. (1961) Industrial dynamics. MIT Press Cambridge, MA.

Frank, A. U. (1999). One step up the abstraction ladder: Combining algebras-from functional pieces to a whole. In International Conference on Spatial Information Theory, pages 95–107. Springer.

Frank, A. (2005). Map algebra extended with functors for temporal data. *Perspectives in conceptual modeling*, 194-207.

Kelly, R. A., Jakeman, A. J., Barreteau, O., Borsuk, M. E., ElSawah, S., Hamilton, S. H., and van Delden, H. (2013). Selecting among five common modelling approaches for integrated environmental assessment and management. Environmental modelling & software, 47, 159-181.

Law, A. M. and Kelton, W. D. (2000). Simulation modeling and analysis (Vol. 3). New York: McGraw-Hill.

Maxwell, T., and Costanza, R. (1997). An open geographic modeling environment. Simulation, 68(3), 175-185.

Meadows, D. H. (2008). Thinking in systems: A primer. Chelsea green publishing.

Morgan, J. S., Howick, S., and Belton, V. (2017). A toolkit of designs for mixing Discrete Event Simulation and System Dynamics. European Journal of Operational Research, 257(3), 907-918.

Parker, D. C., T. Berger, et al. (2001). Agent-Based Models of Land-Use and Land-Cover Change. Report and Review of an International Workshop. L. R. No.6. Irvine, California, USA.

Sahin, O., and Mohamed, S. (2014). Coastal vulnerability to sea-level rise: a spatial–temporal assessment framework. Natural hazards, 70(1), 395-414.

Silva, W. S. F.; Carneiro, T. G. S., (2016) An algebra for modelling the simultaneity in agents behavior in spatially explicit social-environmental models. GeoInfo - Brazilian Symposium on Geoinformatics.

Schmitz, O., Karssenberg, D., De Jong, K., De Kok, J. L., & De Jong, S. M. (2013). Map algebra and model algebra for integrated model building. Environmental modelling & software, 48, 113-128.

Schmitz, O., de Kok, J. L., & Karssenberg, D. (2016). A software framework for process flow execution of stochastic multi-scale integrated models. Ecological Informatics, 32, 124-133.

Swinerd, C., and McNaught, K. R. (2012). Design classes for hybrid simulations involving agent-based and system dynamics models. Simulation Modelling Practice and Theory, 25, 118-133.

Tomlin, C. D. (1990) Geographic Information Systems and Cartographic Modeling.

Vincenot, C. E., Giannino, F., Rietkerk, M., Moriya, K., and Mazzoleni, S. (2011). Theoretical considerations on the combined use of system dynamics and individual-based modeling in ecology. Ecological Modelling, 222(1), 210-218.

von Neumann, J., (1966). Theory of Self-Reproducing Automata. Edited and completed by A.W. Burks., Illinois.

Wainer, G. A. (2009). Discrete-event modeling and simulation: a practitioner's approach. CRC press.

Wesseling, C.G., Karssenberg, D., van Deursen, W.P.A. and Burrough, P.A., (1996), Integrating dynamic environmental models in GIS: the development of a Dynamic Modelling language. Transactions in GIS, 1, pp. 40-48, Link.