Chapter 3

Map Updates in a Dynamic Voronoi Data Structure

Darka Mioc, François Anton, Christopher M. Gold and Bernard Moulin

Additional information is available at the end of the chapter

http://dx.doi.org/10.5772/50279

1. Introduction

Within a traditional geographic information system, it is currently difficult to ask questions about data that possess positions in both space and time. Information about objects that have spatial position, spatial relationships with nearby objects, and the time of their existence needs to be stored within a computer system, making them available for queries concerning spatial locations, dates, and attributes.

In existing GIS software spatio-temporal queries cannot be answered easily; the absence of a temporal component implies that analysis of past events and future trends is difficult or impossible [37].

In traditional geographic information systems, the temporality of spatial data has been treated separately from their spatial dimensions. The lack of mechanisms for recording incremental changes poses a serious problem for the integration of temporal data inside a GIS. In current GISs, the spatial changes affect the whole map. Therefore, the "snapshot model[1]" with global changes of map states is considered to be a basis of any spatio-temporal model [6].

Nowadays, there is a growing demand from the user community for a new type of GIS that will be able to support temporal data and spatio-temporal facilities such as: spatio-temporal queries [41] and interactive spatial updates [45].

Spatio-temporal facilities would be useful in many GIS applications such as harvesting and forest planning, cadastre, urban and regional planning as well as emergency planning.

In all these fields, there is a need for a GIS technology which can manage the history of spatial objects and their evolution, and which can efficiently answer spatio-temporal queries. In forest management, there is a need for GIS technology which can manage the history of spatial objects, map versioning and spatio-temporal queries. An example is a forest inventory map that is being frequently updated with the latest information about forest roads, cut areas, fire, etc. It would be desirable to preserve all the previous states, and to be able to track the evolution of the spatial objects.

[1] The snapshot model is a map at a single moment in time.

While in the past, most of the approaches concentrated on the extension of existent GIS models with temporal data, recent research [8] shows that dealing with time as calendar time and mapping it onto an integer domain is feasible, but does not capture the semantics of time and leaves out most of its important properties. According to Frank [8], the understanding of how time and temporal reasoning processes are conceptually structured is a prerequisite to building support for temporal reasoning into current GISs. Frank [8] further emphasized the need for formal models to determine the representation of temporal information and temporal and spatio-temporal reasoning methods.

2. Previous research

Several authors [25], [40], [5] proposed spatio-temporal data models based on extending existing GIS models, in order to include temporal information. The problem they faced is that most commercial GISs are closed systems which cannot be extended nor modified in order to include temporal information. Therefore, the solutions they propose are based on a "dual architecture" [45], which is composed of two subsystems: a commercial GIS and a relational DBMS. Van Oosterom [46] emphasized the problem of the complexity of the maintenance of the proposed spatio-temporal data models.

There are several spatio-temporal functionalities such as retroactive map updates [20] and the incorporation of temporal data without exact temporal information [7] that cannot be handled with such hybrid models. Another problem with existing GISs, is that the semantics of map topology construction [46] are lost. This problem is better known under the term of "long transactions in GIS" [38]. This is due to complex models of spatial topology which need to be processed globally after being updated. When a map is processed in batch mode, its topology is built, and it is not possible to go back to previous states, nor to reuse the past map states, because the information about spatio-temporal objects and the operations that have been executed upon them is lost. Batch processing of spatial topology is managed through "long transactions" which can be composed of nested transactions and could last for hours or days, and system or user generated aborts will not be permitted during that time [29].

Newell and Batty [38] stated that current GISs differ from standard DBMS in that only "long transactions" (and not short ones) are possible. This restricts access to intermediate map states, which is a limitation when the concern is with local updates and temporal queries. They also state that short transactions are required for problems such as emergency planning, vehicle tracking, fault logging, and other real-time system problems.

The problem of long transactions is so acute (time consumption, database inconsistency problems, etc.), that the underlying problem is not well defined. The problem lies in the fact that "dynamic topology" [14] is not used: geometric algorithms cannot be executed interactively upon spatial objects while maintaining topological relationships. Recently, several GIS researchers (Van Oosterom [45], [46], Chrisman [6], Newel and Batty [38], Kraak [23], [24]) tackled the same problem from several different perspectives.

2.1. A fundamental problem in GIS

Recently, the research of Newell and Batty [38], Gold [14], and independently Chrisman [6] shows that "batch processing" of spatial data is a fundamental problem in commercial GISs. The "batch processing" of static spatial topology used by line intersection based spatial systems does not support mechanisms for recording incremental change and poses a serious problem for the integration of temporal data inside GISs.

The "batch processing" of spatial data cannot support incremental (local) addition and deletion of spatial objects, and they cannot support the temporal evolution of spatial data.

Briefly, we will explain now what led to the batch processing of spatial data that imposed the snapshot or time-slicing approach in GIS.

The snapshot model of map time [6] was always emphasized by cartographers, resulting in current commercial GIS models. Current GISs can represent a space evolving in time only through a series of map snapshots. The snapshot approach of traditional geographic information systems, where independent coverages are generated for each time step, cannot easily maintain the incremental changes of cartographic data evolving in space and time. The limitations of this approach include high data redundancy, due to the inability of conventional GIS models to support incremental changes, causing difficulties in the maintenance of long series of cartographic snapshots [23].

The snapshot model misses the key nature of change [6], which can be seen as a composition of events. The "time slicing" idea leading to the snapshot approach collapses many events, each of which occurred separately [6], causing difficulties in determination of spatio-temporal processes. Map snapshots tend to be created independently at specific intervals, rather than incrementally, and thus there is no preservation of topological relationships between map elements in different time slices, and no effective way of determining the continuity of existence of map elements and their neighbours between snapshots. Although a snapshot method is an important capability, this historical model represents only a single point in time, and does not reveal the sequence of events, or history of the area.

For change detection, the snapshot method uses two distinct maps [6] which are measured with some error. When the map overlay is used to compute differences between these two maps [6], the errors are confounded with the actual changes, and it is difficult to distinguish error from change when using snapshots.

2.2. Overview of recent research on spatio-temporal models

Van Oosterom proposed a new approach for handling spatio-temporal information [45], [46]. He proposed a data model in which both time and topology are consistently maintained in the database updates. His topological data structure is based on the "CHAIN" method [46], similar to the winged edge data structure[2]. He also addresses the problems of long transactions in GIS, the loss of transactional semantics and possible database inconsistency problems [45], [46]. In his approach, the history of complex map objects can be obtained

[2] The winged edge data structure or a polygon data structure is a way of representing a geometric graph in the plane. The winged edge data structure was proposed by Baumgart. It stores a record for each vertex, face and edge, with the edge record being the most important. The record for a vertex v stores v's coordinate position and a pointer to one arbitrary edge that is incident upon v. Likewise the record for a face f stores the name of f and a pointer to one arbitrary edge that lies on the boundary of f. The record for an edge e stores many fields including:
Pointers to the two vertex endpoints v_1 and v_2 of e.
Pointers to the two faces f_1 and f_2 on either side of e.
Pointers to the predecessor and the successor edges of e on the boundaries of f_1 and f_2 respectively. They are the so called "wing edges".
Thus, the total number of pointers needed is | V | + | F | +8 | E |.
Most importantly, just by following the appropriate pointers, one can traverse the boundary of a face in either clockwise or counterclockwise order in time proportional to the number of edges on the boundary. Likewise one can list all the edges incident upon a vertex v in clockwise or counterclockwise order in time proportional to their number.
Source: http://www.ugrad.cs.ca/spider/cs414/winged-edge.html

only by using spatial overlap queries, with respect to the given objects over time. Indeed, the model he proposed can be queried for historic versions only for simple object changes: it does not work for splits, joins nor more complicated spatial editing, unless spatial overlay is performed. Within his data structure, retroactive map updates are not implemented, due to possible consistency problems.

Frank [7] provides a theoretical framework for spatio-temporal reasoning based on the relative order of events[3] and he proposed the integration of ordered event structures in a GIS. In his paper titled "Different Types of 'times' in GIS" [8], he made two very interesting points that will be further explored in this section. The first point is related to the exploration of the relations[4] between temporal order relations and the order relation in the lattice of partitions in geometric space [8], which he considered a challenging research question.

The second point is related to the different spatio-temporal models that he described, which will now be mentioned briefly. Some of the models he describes are not explored (and implemented) yet in current GISs. An example of such models is the movement along a path[5], that is an interaction between temporal and spatial reasoning. Another example of spatio-temporal model that cannot be implemented within current GISs is spatial data without precise temporal information, which has been described by Frank [8]. This example occurs often in history and geology related mapping, where some of the spatio-temporal sequences of the data are missing and need to be interpolated. Further, he stated a "need for more realistic GISs" that will unify different models presently used for spatio-temporal reasoning. The "realistic GIS" [8] should be able to deal with error correction and other improvements of existing data. In other words, there is a need for retroactive updates in spatio-temporal models, and they are difficult to achieve. Gold [16] proposed another spatio-temporal model based on event ordering, which responds instantly to map construction commands given by the user, changing the state of map objects and their spatial topology, and storing all the changes in topological states. In the past few years, his research efforts showed that the Voronoi diagram offers "a more intelligent, more realistic, and semantically richer" model for spatial representation. In his model, based on the Voronoi diagram for sets of points and line segments, map topology states can be reconstructed at any time because all the operations on his data structure are reversible. The Voronoi diagram for sets of points and line segments is the generalization of the ordinary Voronoi diagram (for sets of points), where the set of sites may contain not only points, but points and line segments as well (see Figure 1).

3. The dynamic spatial Voronoi data structure

Cartographic objects on maps are composed of points, curves and surfaces. The ordinary point Voronoi diagram does not allow us to accurately model linear or areal objects (curves and surfaces) needed in many GIS applications.

---

[3] Orderings of events in which it is not known for every event if it is before or after another one are called partial orderings [8]. Completely ordered sets of events are known as totally ordered [8] with a single viewpoint of all events.

[4] *"A set with a partial order relation is called a **partially ordered set** or **poset**. Subsets of a poset may be totally ordered; this is, for example, the case for each sequence of events that apply to a single parcel (and all its predecessors). The elements in a poset can be linearly ordered, but there is more than one possible solution (topological sorting). A special case of "a poset" is a **lattice**. The intersection of the successors of two points have a single earlier point. The same concept can be applied to a family of partitions of the plane, where the partitions of different levels of subdivisions form a lattice (ordered by a "refinement" relation)."* from [8]

[5] Movement along a path is one of the facilities needed in GIS, for applications related to emergency planing, robotics, and marine GIS.

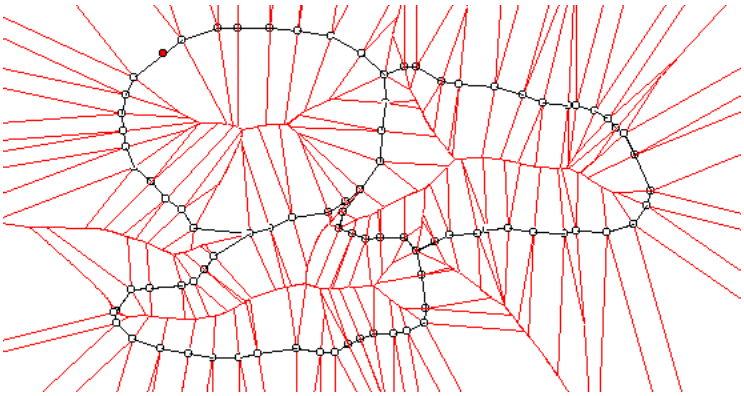

Figure 1. A line Voronoi diagram, from [33]

Therefore, in order to represent the different kinds of cartographic objects, we need to use the Voronoi diagram for sets of points and line segments instead of just the ordinary point Voronoi diagram.

Gold [11], [18] developed a dynamic spatio-temporal data structure based on the Voronoi diagram for a set of points and line segments (also known as a line Voronoi diagram, see Figure 1). His data structure is dynamic: objects may be added to and removed from the data structure. This data structure is also kinematic: objects are created by splitting the nearest point into two and then moving the newly created object to its desired location, and deleted by moving it to the nearest point location and merging it with the nearest point. In Section 3.1, we present the properties of this data structure: the dynamic spatio-temporal Voronoi data structure [33]. In Section 3.2, we develop a formalism for specifying the operations upon this data structure and their changes in topology [33], [36],[35].

The Voronoi diagram for a set of map objects (points and line segments) is the tessellation of space where each map object is assigned an influence zone (or Voronoi region), that is the set of points closer to that object than to any other object (see [39] and Figure 1).

The algorithm used to construct the Voronoi vertices has been described in [4]. The boundaries between the regions of this tessellation form a net (the Voronoi diagram), whose dual graph (the Delaunay quasi-triangulation or Delaunay graph) stores the spatial adjacency (topology) relationships among objects. Within such a dynamic Voronoi spatial data structure, as developed by Gold [12], map objects (points and/or line segments) are stored as nodes of the dual spatial adjacency (topology) graph: the Delaunay triangulation. The underlying data structure used is the Quad-Edge data structure [19].

The Quad-Edge data structure was used for computing the line Voronoi diagram [17], which is the basis of the dynamic Voronoi data structure for points and line segments. The Quad-Edge data structure was introduced by Guibas and Stolfi [19] as a primitive topological structure for the representation of any subdivision on a two-dimensional manifold. The Quad-Edge data structure is the implementation of an edge algebra [19], which is the mathematical structure that defines the topology of any pair of dual subdivisions on a two-dimensional manifold. In the context of the application of the Quad-Edge data structure to the computation of Voronoi diagrams, both a primal planar graph (the Voronoi diagram) and its dual graph (the Delaunay triangulation) are stored in the Quad-Edge data structure - see [19].

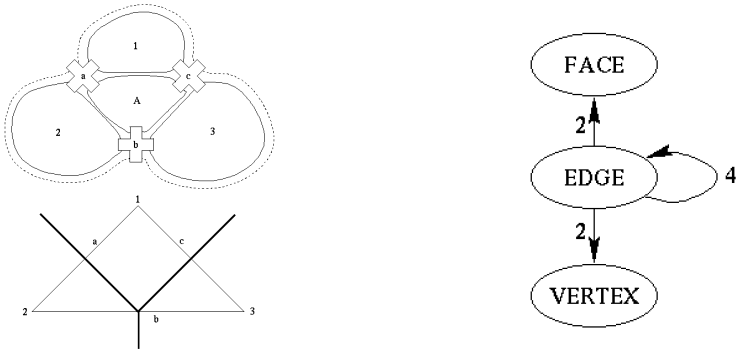

Figure 2. Left: a simple Voronoi diagram and its corresponding Quad-Edge; right: the PAN graph of the Quad-Edge data structure on the left

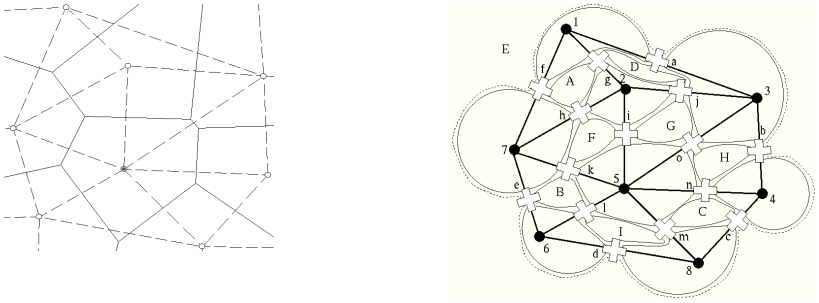

Figure 3. Left: a Voronoi diagram; right: the corresponding Quad-Edge data structure

The PAN graph [10] of the Quad-Edge data structure gives a representation more suitable within a GIS context (see Figure 2). The Quad-Edge data structure (see Figure 3) represents a graph and its geometric dual. In the context of the application of the Quad-Edge data structure to the computation of Voronoi diagrams, both a primal planar graph (the Voronoi diagram) and its dual graph (the Delaunay quasi-triangulation) are stored in the Quad-Edge data structure.

The definition of spatial adjacency relationships within the Voronoi model is the adjacency of the Voronoi regions of two objects. In such a case, this spatial adjacency relationship is stored in the dual representation of the Voronoi diagram: the Delaunay quasi-triangulation.

The main characteristic of the topology[6] within the Voronoi spatial data model which distinguishes it from other models such as the vector model, is that it does not need any other computation than the incremental construction of the Voronoi data structure. Topology in the Voronoi spatial data model is given by the fact that two objects have adjacent Voronoi regions, which is stored in the dual representation of the data structure.

The dynamic Voronoi spatial data model is based on an event-condition-action paradigm [16], which seems to provide many advantages over traditional GIS data models. The main

[6] Topology is the branch of mathematics, that deals with the properties of points of some space that are invariant under some continuous transformations. Topological relationships are spatial adjacency relationships.

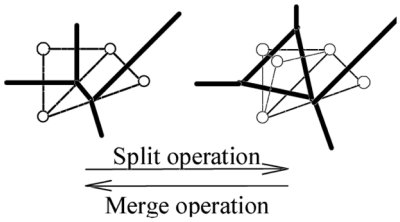

Figure 4. The topological changes due to the Split and Merge operations

advantage of the dynamic Voronoi data structure is its dynamic, incremental and explicit topology, which allows one to automatically keep track of each event and change of map state [16].

The changes in this data structure are therefore the changes in the spatial adjacency relationships, that is to say the changes in the Delaunay triangulation [39]. Within this data structure, the user's commands are changing the map incrementally and locally, and the map objects and their spatial adjacency relationships are all visualized at any point in time [2].

Furthermore, this approach allows real-time dynamic maintenance of the spatial data structure, as well as dynamic sequential processing of events [16].

3.1. The atomic actions on the dynamic Voronoi data structure

These map state changes are produced by map commands [12], that are composed of atomic actions. Each atomic action in the map command executes the geometric algorithm for addition, deletion or change of map objects and corresponding Voronoi cells.

The *atomic actions* are:

- the *Split* action inserts a new point into the structure by splitting the nearest point from the pointed location into two points (see Figure 4);
- the *Merge* action deletes the selected point by merging it with its nearest neighbour (see Figure 4);
- the *Switch* action is performed when a point moves and a topological event occurs (i.e. the moving point enters or exits a circle circumscribed to a Delaunay triangle, see Figure 7), switching[7] the common boundary of two adjacent triangles (see Figure 5). On Figure 7 we can see the topological event caused by the "Switch" atomic operation.
- the *Link* action adds a line segment[8] between the points obtained after a Split action (see Figure 6). A Link action must occur after a Split action, and adds a line segment between the point selected for splitting and the newly created point.

[7] The *Switch* action will be used in the construction of the *Move*action. The *Move* (topological event) action moves the selected point from its current position to a new position or until the next topological event.

[8] A line segment is composed of two half-line segments, whose Voronoi regions are on each side of the line segment, having the line segment as a common boundary (see Figure 6).

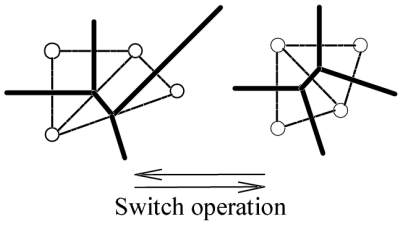

Figure 5. The topological changes due to the Switch operation

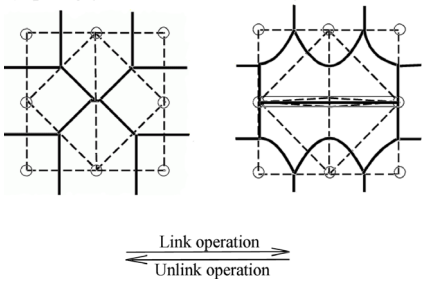

Figure 6. The topological changes due to the Link and Unlink operations

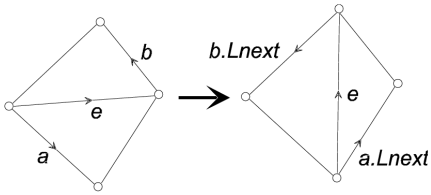

Figure 7. The topological event caused by a "swap" atomic operation

- the *Unlink* action removes the selected line segment. An Unlink action must occur before a Merge action, and removes the line segment between the selected point and its nearest object.

Figure 6 shows the succession of these atomic actions. These actions compose the set of atomic actions of the dynamic spatial Voronoi data structure [32].

3.2. Topological changes in the Voronoi data structure induced by the atomic actions

The map changes produced by the atomic actions on this data structure are the changes in the spatial adjacency relationships among spatial objects. The spatial adjacency relationships are defined as follows: two objects are Voronoi neighbours if, and only if, their Voronoi regions have one portion of the Voronoi diagram (a Voronoi edge) in common (see Figure

Atomic Operation	Symbol	Newly created Voronoi regions	Inactivated Voronoi regions
Split	S	1	0
Merge	M	0	1
Switch	N	0	0
Link	L	2	0
Unlink	U	0	2

Table 1. The atomic actions and their associated changes in topology expressed in the numbers of newly created and inactivated Voronoi regions

3). Therefore, the only map state changes of the dynamic Voronoi data structure produced by events are the changes in the Delaunay triangulation/Voronoi diagram preserved in the Quad-Edge structure. These events are ruled by the Delaunay triangulation empty circumcircle criterion (see Figure 7).

When a point comes inside a circumcircle or exits from a circumcircle - a "topological event" occurs - the boundary between the two triangles inscribed in the circumcircle "switches" [44].

Within the dynamic Voronoi spatio-temporal data model, all the operations are local and "kinematic": the addition of a new point is performed by splitting the nearest point into two and moving the newly created point to its destination; and the deletion of an existing point is performed by moving it to its nearest point and merging them. It is easy to see that the two actions described previously are mutually reversible: the reverse of a split being a merge and the reverse of a merge being a split (see Gold [12]). A Split action takes the Voronoi cell of a "parent" point and splits it into two, generating a "child" point that may then be moved to the desired destination. A Merge action reverses this process, combining two adjacent cells into one.

Each *atomic action* produces different changes in spatial topology. The possible changes are:

- the triangle switches (topological events) changing the corresponding Voronoi edges (see Figure 7),
- the creation of a new map object (point or line) and the corresponding appearance of its Voronoi region,
- the inactivation of a map object and the corresponding disappearance of its Voronoi region. Objects and spatial adjacency relationships are not removed, but inactivated, in order to be able to record all the history information.

The atomic actions of the dynamic spatio-temporal Voronoi data structure, their reverse actions, and their corresponding changes in topology are described in Tables 1 and 2. We also introduce in Tables 1 and 2 the symbols for each atomic action (N, S, M, L and U) that will be used latter for specifications of complex map operations. The topological changes for each atomic action (see Table 2) in the map are represented by the numbers of newly created and inactivated spatial adjacency links (i.e. Voronoi edges or Quad-Edges). Each atomic action is uniquely characterized by the numbers of new Quad-Edge (or Voronoi) edges and inactivated Quad-Edge (or Voronoi) edges (see Table 2). This means that from changes in topology we can determine which atomic action was applied, and vice versa. In other words, the actions on the data structure have a deterministic behaviour. More precisely, we can say that the set

Atomic Operation	Symbol	New Edges	Inactivated Edges
Split	S	6	3
Merge	M	3	6
Switch	N	1	1
Link	L	11	5
Unlink	U	5	11

Table 2. The atomic actions and their associated changes in topology

of atomic actions is naturally isomorphic to the set of the number of new edges as well as to the set of the number of inactivated edges.

3.3. The map construction commands

The *atomic actions* are the basis upon which *map commands* have been built. All the map construction commands [12] of this dynamic Voronoi data structure are complex operations composed of atomic actions (illustrated in Figures 5, 4 and 6). The composition of atomic actions into map commands is provided by syntactic rules. The meaning of the word "syntax" is based on the theory of formal languages and grammars [22]. In the theory of formal languages, the semantics of the basic operations that can be applied on the set of objects is described by a grammar. A grammar provides a set of rules, known as production rules, specifying how the sequence of atomic actions will be applied to the elemental map object (currently a point or a line segment).

Rewriting is a useful technique for defining complex objects by successively replacing parts of elemental map objects using a set of rewrite rules or production rules [42]. Given a set of productions we can generate an infinite number of map objects [1]. In the Voronoi spatial data system there is more than one rule we can apply, and the user is given the freedom of selecting the production rule appropriate to the map update needed. Rewriting context is extended from topological context to include geometrical (spatial) context[9] (position). Therefore rewriting is done sequentially at the specific locations selected by the user or given by coordinates. Thus, the update of the Voronoi data structure given by map commands can be interpreted as the execution of production rules which constitute a map grammar [42]. Graph-grammars may be used as a natural and well-established syntax-definition formalism [43] for languages of spatial relationships graphs. The map grammar shows the hierarchical presentation of the production rules and spatial objects. For example, the move command is a part of any other production rule described as a map command.

The *map construction commands* are illustrated in Figure 8. On the left side of Figure 8, we can see the map objects on which the map command will be applied, and on the right side, the map objects that have been rewritten. In the graphical illustration of map commands, the topological part of the model is left out for better understanding of the general principle, and

[9] In context sensitive systems the selection of the production rule is based on the context of the predecessor. A context sensitive system is needed to model information exchange between neighboring elements [47].

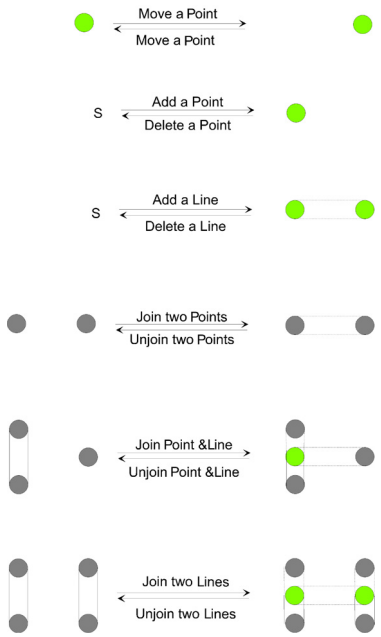

Figure 8. The map commands (S is the starting symbol in the production rule

also the line-line collisions and their effects are not shown. The detailed description together with the graphical illustrations will be presented in Section 4.

The *map commands* (see Figure 8) are composed of atomic operations, and the exact decomposition of map commands into sequences of atomic actions is given in Table 3. The atomic operations are denoted by the symbols (*N, S, M, L* and *U*) from Table 2. For example, the map command *"Move a Point"* corresponds to the sequence of movements of the point from its initial position to its destination through all the intersections of its trajectory with circumcircles, and the corresponding triangle switches (*"N"*) in the Voronoi data structure.

The map command *"Move a Point"* is possible in this Voronoi data structure because the Voronoi data structure is kinematic: one point may move at a time, and this point is called the *"moving point"* [14]. In fact, all the operations on this kinematic Voronoi data structure use this concept of the moving point. For example, when a point is to be created at some location, the nearest point from that location is split into two (*S* term in the decomposition of "*Add a Point*" operation SN^t), and then the newly created point is moved to its final destination (N^t term in the decomposition of "*Add a Point*" operation SN^t). In fact, the triangle switch operation incorporates the movement of the moving point to the intersection of the trajectory of the moving point with the circumcircle that induced the triangle switch. In Table 3, the exponents denote how many times the operation is executed repeatedly, e.g. N^t denotes N executed t times, where t denotes the number of topological events. Whenever more than one connected sequence of topological events is executed in a map command, such as in the *"Add a Line"* command $(SN^{t_1}\ SLN^{t_2}\ (SLN^{t_{2i+1}}\ MSLN^{t_{2i+2}}))$, the total number of topological events is broken down into the number of topological events in the first connected sequence (N^{t_1}), the number

Map construction command	Decomposition (the terms in parentheses appear at each line-line collision, i = collision index, $i \in \{1,...,c\}$, c = number of collisions; t, t_x denote numbers of topological events
Move a Point	N^t
Add a Point	SN^t
Delete a Point	$N^t M$
Add a Line	$SN^{t_1} SLN^{t_2} \left(SLN^{t_{2i+1}} MSLN^{t_{2i+2}}\right)$
Delete a Line	$\left(N^{t_{2i+2}} UMSN^{t_{2i+1}} UM\right) N^{t_2} UMN^{t_1} M$
Join 2 Points	$SLN^{t_1} \left(SLN^{t_{2i}} MSLN^{t_{2i+1}}\right) M$
Unjoin 2 Points	$\left(N^{t_{2i+1}} UMSN^{t_{2i}} UM\right) N^{t_1} UM$
Join Pt & Line	$SLN^{t_1} \left(SLN^{t_{2i+1}} MSLN^{t_{2i+2}}\right) SLN^{t_2} M$
Unjoin Pt & Line	$SN^{t_2} UM \left(N^{t_{2i+2}} UMSN^{t_{2i+1}} UM\right) N^{t_1} UM$
Join 2 Lines	$SLN^{t_1} SLN^{t_2} \left(SLN^{t_{2i+2}} MSLN^{t_{2i+3}}\right) SLN^{t_3} M$
Unjoin 2 Lines	$SN^{t_3} UM \left(N^{t_{2i+3}} UMSN^{t_{2i+2}} UM\right) N^{t_2} UMN^{t_1} UM$

Table 3. The map commands and their decomposition into atomic actions

of topological events in the second connected sequence (N^{t_2}), and so on. The parameter i denotes the number of times the line segment being added has already intersected existing line segments. This type of intersection with an existing line segment is called a collision, and i is called the collision index. The terms in parentheses are repeated for each intersection with an existing line (i.e. each collision).

We will now briefly explain the decomposition of each map command. We have already seen the description of "*Move a Point*" and "*Add a Point*" map commands. Map command "*Delete a Point*" is exactly the reverse of "*Add a Point*" map command: the point to be deleted is moved to the location of the nearest point (N^t), and then they are merged with this nearest point (M).

The remaining map commands involve the addition or removal of one or more new line segments. For all these map commands, the decomposition includes a fixed sequence of atomic actions that is executed only once (the sequence outside the parenthesis), and a sequence that is executed at each collision (replicating sequence).

In the case of "*Add a Line*" and all the join map commands, the replicating sequence has always the same pattern in terms of atomic operations ($\left(SLN^{t_{2i+1}} MSLN^{t_{2i+2}}\right)$, although the actual indices may vary). This corresponds to the splitting of the existing line ($SLN^{t_{2i+1}}$), the merging of the newly created point (by the S atomic action in this last sequence) with the extremity of

the line segment being added (M), and the continuation of the new line segment after collision ($SLN^{t_{2i+2}}$).

In the case of "*Delete a Line*" and all the unjoin map commands, the replicating sequence has always the same pattern in terms of atomic operations ($\left(N^{t_{2i+2}}UMSN^{t_{2i+1}}UM\right)$, although the actual indices may vary). This is exactly the reverse of the previous replicating sequence.

Now, we will explain the fixed sequence for all these map commands. In order to "*Add a Line*", the nearest point to the starting extremity location has to be split into two (S), then it has to be moved to the starting extremity location (N^{t_1}). Then, the ending extremity has to be created by splitting the starting extremity into two (S). At this point the two extremities must be linked (L) in order to form a line segment. Finally, the ending extremity has to be moved (N^{t_2}) to its expected location. The sequence for "*Delete a Line*" is exactly the reverse of the preceding sequence.

In order to join two points with the "*Join two points*" map command, the first point must be split into two (S) in order to create the ending extremity of the line segment that starts at the first point. Then, these two points must be linked (L) in order to form a line segment. Then, the ending extremity must be moved (N^{t_1}) to the location of the second point (including eventually the replicating sequence in case of collisions). Finally, the ending extremity must be merged with the second point (M).

The sequence for the "*Unjoin two points*" map command is exactly the reverse of the sequence for "*Join two points*" map command. The sequences of the remaining map commands follow immediately from the sequence of the "*Join two points*" map command. Indeed, the other join map commands fixed sequence involve several sequences corresponding to the same atomic actions as the SLN^{t_1} sequence already encountered in the fixed sequence of "*Join two points*" map command. The unjoin map commands are the exact reverse of their join counterpart.

3.4. Reversibility of the map commands in the dynamic spatio-temporal Voronoi data structure

For each map command, the reverse map command is composed of reverse atomic actions in exactly the reverse order [34]. Due to the local scope of its spatio-temporal topology, all the atomic actions of the dynamic Voronoi spatio-temporal model are reversible. Indeed, each atomic action has its reverse atomic action shown in the Table 4. The consequence of the property of reversibility of the atomic actions inside the Voronoi dynamic data structure is that a sequence of atomic actions applied in a map construction command can be reconstructed from the predecessor and successor map states. This proves in another way that the atomic actions are reversible: the input can be deduced from the output; or, in other words, computation happens without any loss of information [9].

The resulting complex operations (map commands) are reversible (see Figure 8 and Table 5), as long as their decomposition into atomic actions is exactly known (including the numbers of topological events and the number of line-line collisions).

The reversibility of the addition and deletion of intersecting line segments has been studied in [3]. This strictly showed that in order to perform backwards visualization through reverse execution, we need to access the sequences of atomic actions stored in a log file. This can become cumbersome due to the potentially large number of line-line collisions and of their

Atomic action	Reverse atomic action
Split	Merge
Switch	Switch is self-reversible
Link	Unlink

Table 4. The reversibility of the atomic actions

Map construction command	Reverse map construction command
Move a Point	Self-reversible
Add a Point	Delete a Point
Add a Line	Delete a Line
Join 2 Points	Unjoin 2 Points
Join Pt & Line	Unjoin Pt & Line
Join 2 Lines	Unjoin 2 Lines

Table 5. The reversibility of the map commands

associated replicating sequence (see terms in parenthesis in Table 3 and [32]). Moreover, if we want to perform spatio-temporal queries or spatio-temporal analysis, we need to access the previous map states. However, the undoing and redoing of atomic actions consume large amounts of time and storage and are not spatially localized. This is the consequence of the total ordering of the map construction events, that does not allow spatially local redoing or undoing map commands. Therefore, in order to avoid these problems, we keep a spatio-temporal structure which captures the execution traces of map commands in the dynamic Voronoi data structure [32]. This spatio-temporal structure, along with the reversibility property of the atomic actions, guarantees exact reverse execution of map construction.

3.5. Logging the transactions

The incremental updates (given by the user) are recorded in the log file. Transactions are logged at two levels: the log file that stores map command transactions, and the history of the map (the hierarchical Voronoi data structure) that is logging the changes in topology as newly created, deleted and modified Voronoi regions. The operations recorded in a log file have a direct translation into the spatio-temporal topology as the hierarchical Voronoi data structure. Temporal (parent-child, or modification) links between two update levels correspond to the applied (executed) map commands (see Figure 9).

In the next subsection we will briefly introduce a theoretical approach needed for the execution of map commands or transaction processing.

3.6. Trace systems

Execution traces [28] give us a formalism to represent map history at a more abstract level. The theory of traces established by Mazurkiewicz [28] deals with the transaction processing in concurrent systems. Trace systems are special in that they enable both very detailed modelling and also offer opportunities for abstraction. In order to specify a trace system one has to specify an underlying spatial language and a dependency relation [28].

The trace system that represents the history of the map is maintained as a hierarchy of map objects and their ancestral dependency relationships [27]. Hence for the corresponding topological changes in the Quad-Edge data structure, a dependency relationship occurs when one atomic action uses the results of another: e.g. a Link action always follows a Split action, and it uses both the point that has been split and the point from which it occurred [32].

4. Map updates and map history

The user incrementally constructs or updates a map by giving map commands. A typical example could be updating a forest map to show the previous year's clear-cuts. Each map command is a sequence of atomic actions which will be executed on the Voronoi diagram, producing changes in spatial topology. All the state changes produced by map commands (induced by past map events) are permanently stored in the spatio-temporal data structure as levels of map updates [32].

The left side of Figure 9 shows the effects of a map update *"Add a Point P_6"* on a small portion of a map consisting of several line segments and points, illustrating the growth of the Voronoi diagram and of the Delaunay triangulation. The right side of Figure 9 shows the corresponding history structure for the map update shown in the left side, as described in detail below. The parent-child links (represented in Figure 9 by thick solid straight lines) allow us to know from which point a point has been split, or with which point a point has been merged. The connectivity links (represented in Figure 9 by solid curved lines) allow us to know if a given point has been linked to another point, and if it is the case, with which point, and through which half-line segments. The parent-child links associated with the creation of the two endpoints allow us to know the direction in which the digitizing of the line has been done between the two endpoints. The modification links (represented in Figure 9 by light solid straight lines) allow us to know if the Voronoi cell of a given object has been changed during a given map update (map command). The spatial adjacency links (corresponding to edges in the Delaunay quasi-triangulation) are drawn in dashed lines on both sides of Figure 9.

The map construction commands have a direct translation into a spatio-temporal topology, as the hierarchy of map objects and their corresponding Voronoi cells that have been added, moved, or inactivated over time.

Thus, the data structure encodes the history of map construction at two different levels: as spatial topology (or neighbouring relations), and as temporal topology (temporal adjacency relationships; see [32] also shown on the Figure 9 as the vertical links). Temporal topology is maintained through temporal or history links, which are the parent-child relationship links for the objects that have been split or merged, and modification links for the objects whose Voronoi regions have been changed. In Figure 9 the execution of the map command *"Add a Point"* is shown. We can see the addition of the Point P_6 to the Voronoi diagram. The

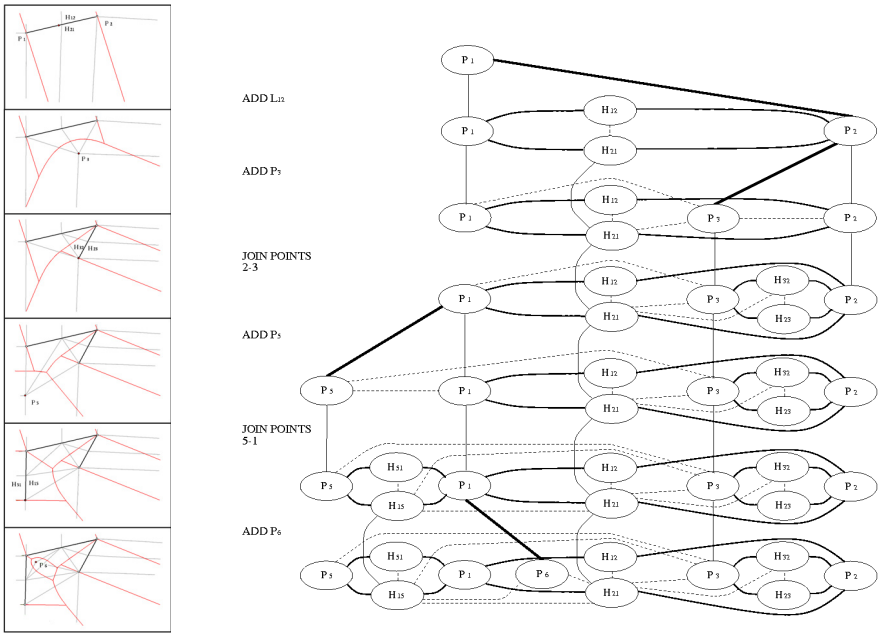

Figure 9. A sequence of map construction commands and the history of the map

Predecessor and the Successor map state are shown (as two update levels) together with the changes in the spatio-temporal topology. Each application of map commands is time-stamped and represented as an update level in our spatio-temporal structure. The hierarchical Quad-Edge data structure stores all the states of the Quad-Edge data structure (spatial topology states) and the connections between the update levels (the temporal topology in the Quad-Edge data structure). These connections between update levels correspond to the temporal topology, while the relationships in an update level correspond to spatial topology. These connections between update levels are the vertical component of the hierarchical Quad-Edge data structure.

5. Hierarchical Voronoi data structure

Ancestral dependency relationships are defined as timed division hierarchies of elemental map objects and their corresponding Voronoi regions, which are ordered by the dependency of one atomic action upon another.

In event-driven systems, such dependency relationships are isomorphic to event structures [48]. Event structures may be seen as a generalization of such structures [48]. We can say that the hierarchical Voronoi data structure is equivalent to an event structure. In event-driven systems the ordering of event occurrences is partial: there is no means to decide which of several independent events occurs first [28]. The only way to establish objective ordering of event occurrences is to find their mutual causal dependencies and to agree that a cause must always occur earlier than its effect [28]. Therefore, the dependency (or independency) of event occurrences should be a basis for the behaviour description of the event driven system [28].

Dependency graphs are a way of visualizing complex relationships [26]. In the following example, we can see update levels corresponding to the map updates shown in Figure 9, and their dependency links. Real-time sequencing of user-invoked events produces a temporal ordering of spatial objects inside the dynamic Voronoi data structure. Inside each map update level, the numerical order of the IDs of the objects corresponds to the temporal order of the commands that generated them. The temporal ordering in the hierarchical Voronoi data structure is maintained through history links (see Figure 9: parent-child links corresponding to the origin/destination of a Split/Merge or Link/Unlink action, and modification links representing a change in the Voronoi region of an object).

This temporal order has direct implications for the reversibility of the Voronoi data structure. Inside each map update level, it is possible to perform reverse execution without accessing a log file following the correspondence between the temporal ordering (maintained through the temporal topology links in Figure 9) of the atomic actions and the numerical ordering of the objects IDs (resulted from the decomposition sequences of Table 3).

Thus, the spatio-temporal model combines events and corresponding state changes in topology. Event structures together with "semantics of change" are essential for spatio-temporal reasoning and answering spatio-temporal queries.

This formal model for spatio-temporal change representation is used to develop the hierarchical Voronoi data structure (hierarchy of map objects ordered by their ancestral dependency relationships) suited for imprecise temporal data representation and spatio-temporal reasoning in the ordered event structures[10].

6. Spatio-temporal change representation

The lack of theory and formalisms for spatio-temporal change representation is a serious problem in research in spatio-temporal GIS [21]. One of the problems is related to the lack of the incremental map updates in current GISs [6]. In the Voronoi spatial data structures the clear specification of map updates (presented in the previous chapter) leads to a method for spatio-temporal change representation.

The formalism for representation of spatio-temporal changes in a dynamic Voronoi data structure and the method for map updates is based on the topological (or structural) properties of the line Voronoi diagram (see [16] and Figure 1). The map updates produce changes in spatio-temporal topology that are different for each map command and can be expressed using the theory of numbers as the structural topology changes.

The corresponding changes in topology are described in Tables 6 and 7. In Table 6, the corresponding "state changes" for each map command are represented as the number of newly created and inactivated Voronoi cells. We can observe that except for the *"Move a Point"* map command, the formulas for the numbers of inactivated Voronoi regions and newly created Voronoi regions generate couples of numbers, which pertain to different couples of residual classes modulo the prime number 5. The corresponding couple of numbers of

[10] There are significant implications of the temporal ordering of map construction events in the Voronoi data structure. The model has an implicit time ordering of events, visible through changes in topology. The dynamic Voronoi spatial data structure can support temporal data without precise temporal information. The changes in the spatio-temporal data structure capture the temporal and spatial semantics.

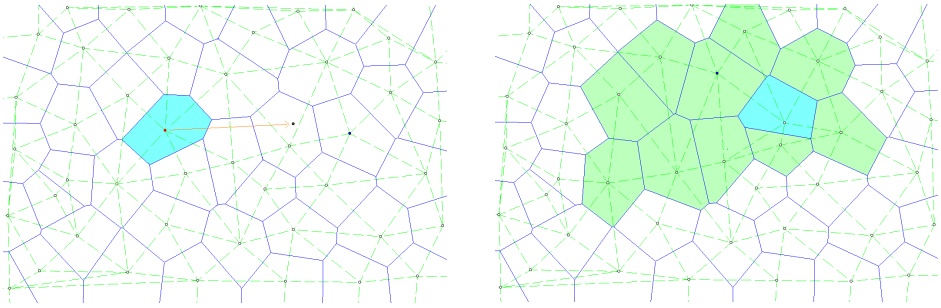

Figure 10. Predecessor and successor map states for a "Move a point" map command

inactivated Voronoi regions and newly created ones for the *"Move a point"* map command is $(0, 0)$, which does not pertain to any other corresponding couple of numbers for other map commands. Therefore, we have an isomorphism between the set of map commands and the set of couples of numbers of inactivated Voronoi regions and of newly created ones. In the following figures the predecessor and successor map topology states for each map command are presented, together with the spatio-temporal changes that occurred, the appearance or disappearance of the Voronoi regions, as well as the modification of the neighbouring Voronoi cells. The newly inserted regions are displayed in dark gray, and the modified Voronoi regions in light gray. In Figure 10, the result of the map command *"Move a Point"* is shown. Figure 11 shows the effect of a map command *"Add a Point"*. Figure 12 shows the addition of a line segment to the Voronoi diagram, by an *"Add a Line"* map command. The last three sets of map commands show all possible combinations of the joining of different objects, points and line segments. Firstly, we can see the result of a map command *"Join two Points"* in Figure 13. Then, Figure 14 shows the effects of a map command *"Join Point and Line"*. Finally, Figure 15 shows the result of a map command *"Join two Lines"*.

Moreover, this isomorphism gives rise to a discrimination of map commands, that allows one to determine the number of line-line collisions that occurred in a given map update. The discrimination just described allows us to determine which map commands were applied just by knowing the predecessor and successor map topology states expressed in the number of newly created Voronoi regions and of inactivated ones (see Table 6). In formal terms, the connections between update levels are formally described by the surjective homomorphism from the Cartesian product (D) of the set of the numbers of new Voronoi regions by the set of the numbers of inactivated Voronoi regions, to the set of map commands.

In Table 7, the corresponding "state changes" for each map command are represented as the difference between the predecessor state and the successor state, expressed as the difference between the numbers of inactivated and newly created Voronoi edges. These state changes take into account the number c of intersections (between the newly created line segment and any existing objects) that occurred in the execution of the map command. When a moving point of a newly added line segment enters the last circumcircle of the line segment - this situation is named "the insertion context" after [31] and [30] the mutual splitting of the line segments occurs. The mutual splitting of line segments operation is visible as a replicating sequence in Table 3. Terms in parentheses are repeated for each line intersection (see Figure 16) detected in drawing the new line specified in the map command. When the first intersection

Map construction command	Inactivated Voronoi regions	Newly created Voronoi regions
	c = number of line intersections	
Move a Point	0	0
Add a Point	0	1
Delete a Point	1	0
Add a Line	0	$4+5c$
Delete a Line	$4+5c$	0
Join 2 Points	0	$2+5c$
Unjoin 2 Points	$2+5c$	0
Join Point & Line	0	$5+5c$
Unjoin Point & Line	$5+5c$	0
Join 2 Lines	0	$8+5c$
Unjoin 2 Lines	$8+5c$	0

Table 6. The changes induced by map commands in Voronoi regions

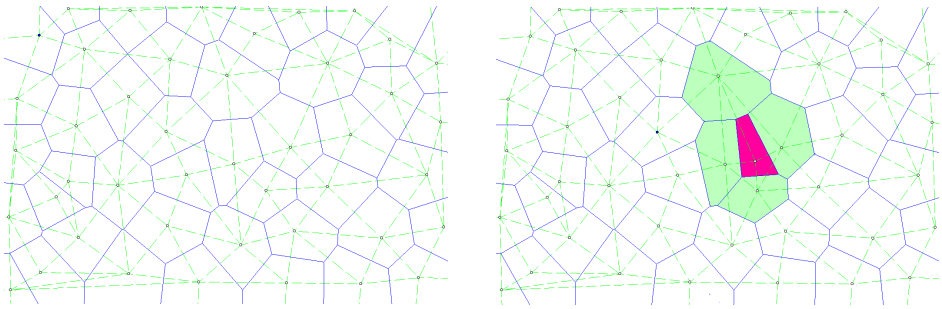

Figure 11. Predecessor and successor map states for an "Add a point" map command

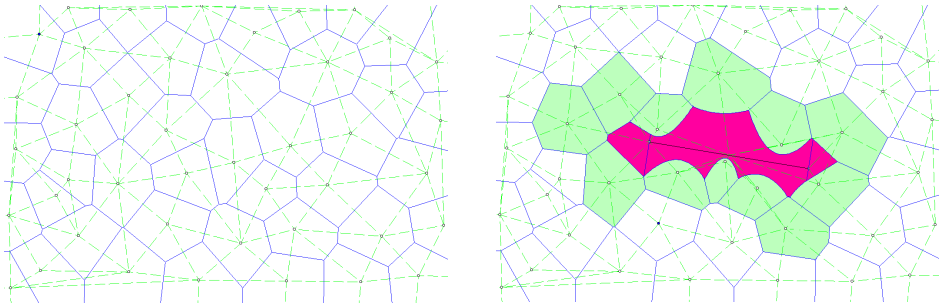

Figure 12. Predecessor and successor map states for an "Add a line" map command

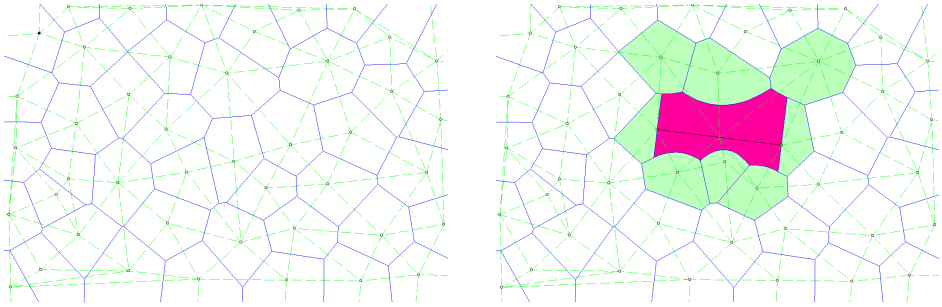

Figure 13. Predecessor and successor map states for a "Join two points" map command

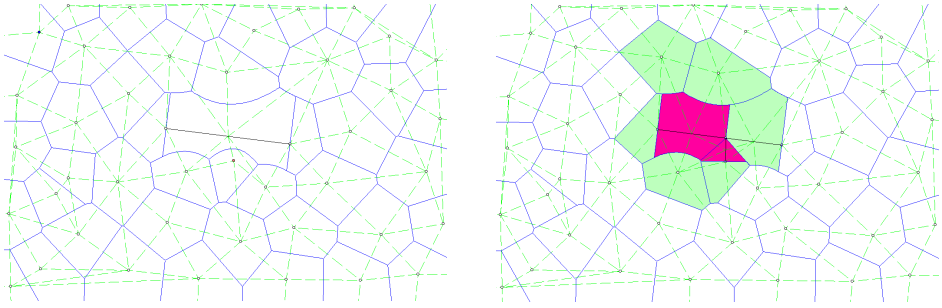

Figure 14. Predecessor and successor map states for a "Join point and line" map command

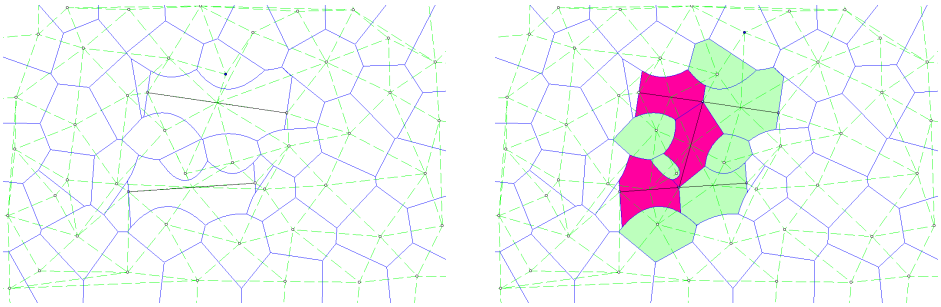

Figure 15. Predecessor and successor map states for "Join two lines" map command

(also called a collision) happens, the collision index is 1. Each time an intersection happens, the collision index is incremented. The intersections are computed incrementally, following the trajectory of the "moving point". In the "Add a Line" command, when the first intersection happens the sequence ($\mathrm{SLN}^{t^{2i+1}}$ $\mathrm{MSLN}^{t^{2i+2}}$)=(SLN^{t^3} MSLN^{t^4}) is added. Then, at the next intersection, the sequence (SLN^{t^5} MSLN^{t^6}) is added.

The language theoretic aspects of replicating sequences can be found in [31]. Longer and longer sequences [30] are produced by new map updates, only rearranging and replicating already existing information. Here, we can see that replication results in a growth of the sequences of the atomic actions. The sequence length is growing linearly. The replication mode is deterministic, therefore we have a growth function associated with a replicating system.

The map commands can be recognized by the changes between the predecessor and successor map topology states, expressed by the difference between the numbers of newly created Voronoi edges and of inactivated Voronoi edges (see Table 7), and vice versa. Indeed, the numbers generated by the formulas of differences (in the third column) pertain to sets (in the fourth column) which are mutually exclusive. Moreover, this discrimination allows us to determine the number of line-line collisions, and then the number of topological events that occurred in a map update (see example treated below). Mathematically speaking, there is an isomorphism between the set of map commands and the set of sets (in the fourth column) of possible corresponding changes between the predecessor and successor map topology states.

In the following example illustrated on the Figure 16, we will see the changes in topology induced by the map update. On the left Figure 16 the one line segment is shown and on the right Figure 16 the execution of the new map command is displayed. We can clearly see that the nine new Voronoi regions appeared in this map update level. From that result, we can determine (from Table 6) the number of line-line collisions that occurred in the map update: $9 \equiv 4 \bmod (5)$. This implies that the update was *"Add a line"* and therefore we get the final result: $4 + 5c = 9 \Rightarrow c = 1$. We can see that the difference in the numbers of newly created and of inactivated Voronoi edges is 27. From this result ($27 \equiv 12\,(15)$, $12 + 15c = 27 \Rightarrow c = 1$), we arrive (see Table 7) at the same conclusion: the map update corresponds to the addition of a new line segment intersecting one existing line segment.

Knowing the numbers of new Voronoi edges and of inactivated Voronoi edges, we can determine the exact number of topological events: $t + 23 + 37$ equals the number of new

Map construction command	New Voronoi Edges	Inactivated Voronoi Edges	Difference New - Inactiv.	Discrimination (set of the numbers corresponding to the difference New - Inactivated Voronoi Edges)
	t = total number of topological events			
Move a Point	t	t	0	$\{0\}$
Add a Point	$t+6$	$t+3$	3	$\{3\}$
Delete a Point	$t+3$	$t+6$	-3	$\{-3\}$
Add a Line	$t+23+37c$	$t+11+22c$	$12+15c$	$\left\{ \begin{array}{c} z \in \mathbb{Z}, z \equiv 12 \bmod 15 \\ \wedge z \geq 12 \end{array} \right\}$
Delete a Line	$t+11+22c$	$t+23+37c$	$-12-15c$	$\left\{ \begin{array}{c} z \in \mathbb{Z}, z \equiv 3 \bmod 15 \\ \wedge z < -12 \end{array} \right\}$
Join 2 Points	$t+20+37c$	$t+14+22c$	$6+15c$	$\left\{ \begin{array}{c} z \in \mathbb{Z}, z \equiv 6 \bmod 15 \\ \wedge z \geq 6 \end{array} \right\}$
Unjoin 2 Points	$t+14+22c$	$t+20+37c$	$-6-15c$	$\left\{ \begin{array}{c} z \in \mathbb{Z}, z \equiv 9 \bmod 15 \\ \wedge z < -6 \end{array} \right\}$
Join Pt & Line	$t+37+37c$	$t+22+22c$	$15+15c$	$\left\{ \begin{array}{c} z \in \mathbb{Z}, z \equiv 0 \bmod 15 \\ \wedge z \geq 15 \end{array} \right\}$
Unjoin Pt & Line	$t+22+22c$	$t+37+37c$	$-15-15c$	$\left\{ \begin{array}{c} z \in \mathbb{Z}, z \equiv 0 \bmod 15 \\ \wedge z < -15 \end{array} \right\}$
Join 2 Lines	$t+54+37c$	$t+30+22c$	$24+15c$	$\left\{ \begin{array}{c} z \in \mathbb{Z}, z \equiv 9 \bmod 15 \\ \wedge z \geq 24 \end{array} \right\}$
Unjoin 2 Lines	$t+30+22c$	$t+54+37c$	$-24-15c$	$\left\{ \begin{array}{c} z \in \mathbb{Z}, z \equiv 6 \bmod 15 \\ \wedge z < -24 \end{array} \right\}$

Table 7. The discrimination of map commands by means of their changes in topology, from [32]

Voronoi or Quad-Edge edges, (see Table 7). Therefore we know that the decomposition of the map command in atomic actions has the following form: $SN^{t_1}SLN^{t_2}\left(SLN^{t_3}MSLN^{t_4}\right)$ where $t_1+t_2+t_3+t_4$ is the known total number of topological events.

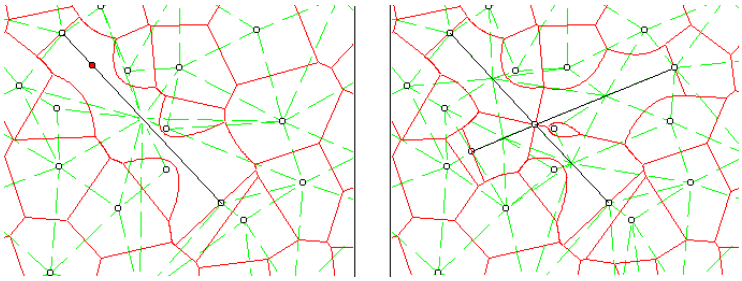

Figure 16. Mutual splitting of line segments

6.1. Retroactive map updates

For some cartographic applications users need to be able to access and update not only the current map, but also the past map states. In the case of retroactive map updates, all the map commands that happened after the date of the beginning of the retroactive map update are undone using reverse execution, returning to the corresponding starting state of the map.

Indeed, in the case of retroactive map updates, all the changes that occurred after the beginning of the retroactive map update are undone, and then the retroactive map update is performed, and the spatio-temporal data structure is updated. This is of particular interest in the cadastre due to the fact that land marking and division operations are rarely inserted in a cadastre information system at the same time as they are officially registered. In most cases the update time ordering does not match the official time ordering [7]. Moreover, the semantics of such operations (especially land marking) cannot be retrieved by comparing the cadastre maps before and after they occurred. Often, the documents and/or the landmarks disappear, making any operation much harder and more subjective. In this particular case of land marking, in order to retrieve the semantics we need to be able to reconstruct the sequence of operations that were performed together with their spatial adjacency relationships. In other words, we need to retrieve their spatio-temporal topology. The parent-child links and the modification links described above (see Figure 9) allow us to retrieve the temporal topology links between these operations, while the spatial adjacency links allow us to retrieve the spatial topology links between these operations.

Further benefits of the **formal model for spatio-temporal change representation** presented in the previous chapter include retroactive map updates. Even though there are some other methods, such as transaction logging, to keep track of the operations that have been applied, these methods have several drawbacks in the case of retroactive map updates. For example, for map updates in the past, an additional "audit file" needs to be maintained and updated with the exact sequence of atomic actions, while in our approach the "semantics of change" is simply maintained as the difference in the number of Voronoi edges between two update levels.

The theoretical work on the formalization of map update operations and the hierarchical Voronoi based spatio-temporal data structure has led to the three different methods for retroactive map updates presented here:

- Recomputation for a portion of the hierarchical data structure with retroactive updates. This method seems to be complex to implement, because it involves maintaining a large

number of history links during the update operation. It could demand very complex algorithms for structure maintenance during the updates as well, due to the necessity for resolving potential spatio-temporal inconsistencies or conflicts[11]

- The second method for retroactive map updates is based on reversibility. The reversibility of map commands gives us a unique tool for another method for retroactive map updates. By exploiting reversibility the structure can be first undone until the moment in the past where the retroactive update has to take a place, and then redone up to the present. Here the efficiency of such retroactive update depends on how far in the past the user has to make the changes. Therefore for updates that need to access deep in the past this approach is not efficient enough. This gives rise to the third model for map updates that is based on the recomputation from the "log file".
- Within this kinematic, geometrical Voronoi structure [15], the map history is difficult to update locally. Indeed, for many updates, it could be a complex and time consuming task, because addition or removal of each point has to be recomputed up to the surface of the map. Therefore, for complex updates, the sequence (log file) could be updated, and the whole map with history recomputed! Therefore, the better solution is to rebuild the whole structure. Complete rebuild is also needed in the case of multiple updates and retroactive map updates from multiple sources.

The last method for retroactive map updates is the one that is most frequently applied in data structures. It is based on rebuilding the data structure from the list of entries.

7. Conclusions

In this research, a new spatio-temporal model based on a dynamic Voronoi data structure for points and line segments is presented. The approach is based on local changes in topology induced by spatio-temporal map updates. These map updates are performed through map construction commands that are composed of atomic actions on the dynamic Voronoi data structure. Even though previous research [13] on the dynamic Voronoi data structure for points and line segments describes the set of atomic actions and map commands used for the construction of the map based on the Voronoi diagram, those atomic actions and map commands were not formalized in a proper way. They were not deterministic and as a consequence not reversible.

This research succeeded in the formalization of the operations needed for constructing a Voronoi diagram for points and line segments, and of the corresponding topological changes. These operations are formalized at the lowest level, as the basic algorithms for addition, deletion and moving of spatial objects in the Quad-Edge data structure; defined as the atomic actions. Furthermore the map commands that are composed of these atomic actions are defined as well.

[11] **Spatio-temporal conflicts and inconsistencies caused by updates**
One of the problems arising from spatio-temporal updates in the Voronoi data model is that if we are deleting, creating or moving an object in the past, we are creating spatio-temporal inconsistencies in the present. This changed object is now inconsistent with the part of the map history which was previously rolled back, and now needs to be rolled forward to modify the present state accordingly.
The fact that the chronology of the events could be different after the change may produce spatio-temporal conflicts. Deletion of an object in the past do not remove its activities - there are a lot of links and interactions with other objects which have been recorded in the map history, and they should be resolved and recomputed up to the present state of map history and the log file.
The same conflict arises with the creation of a new spatial object - changes will affect its neighbours.

This research shows that the result of the formalization of the operations on the dynamic Voronoi data structure is a spatial language or a map grammar that is deterministic and reversible.

It was shown that the behaviour of the basic map operations is deterministic, and well defined in terms of topology changes.

Furthermore, a formal model for spatio-temporal change representation has been defined, where each map update is uniquely characterized by the number of newly created Voronoi regions. The recognition of map commands from the corresponding changes in spatial topology allows us to extend our data structure towards the hierarchical Voronoi data structure (hierarchy of map objects ordered by their ancestral dependency relationships). The hierarchical Voronoi data structure is well suited for spatio-temporal data representation even in the case of imprecise temporal data, and spatio-temporal reasoning in ordered event structures. The model has an implicit time ordering of events, visible through changes in topology. The changes in the spatio-temporal data structure capture the temporal and spatial semantics.

This research has shown that the hierarchical Voronoi data structure is equivalent to an event structure. The temporal ordering of past map events as well as states in a hierarchical Voronoi data structure provide a suitable model for the integration of Allen's temporal algebra, that is needed for reasoning about spatial objects and their temporal relationships [7]. Furthermore, a spatio-temporal structure that combines events and past map states is essential to answer queries about map changes over space and time.

This research has presented several applications of the hierarchical Voronoi data structure that are difficult to implement within the traditional GIS. The formal model of spatio-temporal change representation is currently applied to retroactive spatio-temporal map updates and visualization of map evolution.

The benefits of this approach reside in the possibility of reverting to previous states in order to visualize the evolution of the map or to perform spatio-temporal queries and analysis, as well as in performing reverse execution of the map commands previously applied to the Voronoi spatial data structure, to achieve retroactive map updates. The visualization of map changes enables fast comprehension of the events and processes that occurred in space and time. Visualization of map changes offers a powerful tool for spatio-temporal reasoning, which is needed in many GIS applications.

The map grammar described here allows us to build the deterministic spatio-temporal representations in which all the rules used in their construction are preserved. This is one of the main contribution of this paper, where for the first time in GIS research, a map grammar has been proposed as a method for handling map updates and building spatio-temporal representations.

Author details

Darka Mioc and François Anton
Technical University of Denmark, Denmark

Christopher M. Gold
University of Glamrogan, United Kingdom

Bernard Moulin
Université Laval, Canada

8. References

[1] Angel, E., 1997, *Interactive computer graphics: a top-down approach with OpenGL*, Addison-Wesley, Reading, MA.

[2] Anton, F., 1995, Le système de numérisation Voronoi intelligent, interactif et dynamique et ses applications à la foresterie, *Thèse de maîtrise, Université Laval, Faculté de Foresterie et Géomatique*, Québec, Canada.

[3] Anton, F., 1996, Reversible splitting of line segments within a Voronoi diagram for a set of points and straight line segments, *Internal document, Industrial Chair of Geomatics, Université Laval, Faculté de Foresterie et Géomatique*, Québec, Canada.

[4] Anton, F. and Gold, C. M., 1997, An iterative algorithm for the determination of Voronoi vertices in polygonal and non-polygonal domains, *Proceedings of the 9th Canadian Conference on Computational Geometry (CCCG'97)*, Kingston, Canada, pp. 257-262.

[5] Bedard, Y., Caron, C., Maamar, Z., Moulin, B. and Vallière, D., 1996, Adapting data models for the design of spatio-temporal databases, *Comput., Environ. and Urban Systems*, Vol. 20, No. 1, pp. 19-41.

[6] Chrisman, R. N., 1998, Beyond the Snapshot: Changing the Approach to Change, Error and Process, *Spatial and temporal reasoning in geographic information systems*, edited by Max J. Egenhofer, Reginald G. Golledge, Oxford University Press, New York, Chapter 6, pp. 85-93.

[7] Frank, A.U., 1994, Qualitative temporal reasoning in GIS-ordered time scales, *Proceedings of the Sixth International Symposium on Spatial Data Handling, Edinburgh, Scotland, In Advances in GIS Research, Proceedings*, Vol. 1, pp. 410-430.

[8] Frank, A.U., 1998, Different Types of "Times" in GIS, *Spatial and temporal reasoning in geographic information systems*, edited by Max J. Egenhofer, Reginald G. Golledge, Oxford University Press, New York, Chapter 3, pp. 40-62.

[9] Frank, M., Knight, T., Margolus, N., 1998, Reversibility in optimally scalable computer architectures, *The First International Conference on Unconventional Models of Computation*, January 1998, pp. 165-182.

[10] Gold, C. M., 1988, PAN graphs - An aid to GIS analysis, *International Journal of Geographical Information Systems*, Vol. 2, No. 1, pp. 29-41.
Proceedings of the Fourth International Symposium on Spatial Data Handling, Zurich, Switzerland, pp. 175-189.

[11] Gold, C. M., 1992, Dynamic spatial data structures - the Voronoi approach, *Proceedings of the Canadian Conference on GIS*, Ottawa, Canada, pp. 245-251.

[12] Gold, C. M., 1992, An object-based dynamic spatial data model, and its applications in the development of a user-friendly digitizing system, *Proceedings of the Fifth International Symposium on Spatial Data Handling*, Charleston, pp. 495-504.

[13] Gold, C. M., 1994, The Interactive map, In: *Advanced Geographic Data Modelling - Spatial Data Modelling and Query Languages for 2D and 3D Applications*, Eds. M. Molenaar and S. de Hoop, Netherlands Geodetic Commission Publications on Geodesy (New Series), No. 40, pp. 121-128.

[14] Gold., C. M., 1994, Three approaches to automated topology, and how computational geometry helps, *Proceedings of the Sixth International Seminar on Spatial Data Handling*, Edinburgh, Scotland, pp. 145-158.

[15] Gold, C. M., Remmele, P. R., Roos, T., 1995, Voronoi Diagrams of Line Segments Made Easy, *Proceedings of the Seventh Canadian Conference in Computational Geometry, (CCCG'95)*, Québec, Canada, pp. 223-228.
[16] Gold, C. M., 1996, An Event-Driven Approach to Spatio-Temporal Mapping, *Spatio-Temporal special issue, Geomatica*, Vol. 50, pp. 415-424.
[17] Gold, C. M., 1997, The Global GIS, *Proceedings of the International Workshop on Dynamic and Multi-Dimensional GIS*, Hong-Kong, China, 12 pp.
[18] Gold, C.M., Mioc, D. and Anton, F., 2008, Dynamic GIS, *ISPRS congress Book, Advances in Photogrammetry, Remote Sensing and spatial Information Sciences*, Eds: Z. Li, J. Chen, E. Baltsavias, Taylor & Francis Group, London, UK, pp. 289-303.
[19] Guibas, L. and Stolfi, J., 1985, Primitives for the Manipulation of General Subdivisions and the Computation of Voronoi Diagrams, *ACM Transactions on Graphics*, Vol. 4, No. 2, pp. 74-123.
[20] Hazelton, J. W. N., 1998, Some operational Requirements for a Multi-Temporal 4-D GIS, *Spatial and temporal reasoning in geographic information systems*, edited by Max J. Egenhofer, Reginald G. Golledge, Oxford University Press, New York, Chapter 4, pp. 63-73.
[21] Hirtle, S., 1998, Epilogue In *Spatial and Temporal Reasoning in Geographic Information Systems*, edited by Max J. Egenhofer, Reginald G. Golledge, Oxford University Press, New York.
[22] Hopcroft, J. and Ullman, J., 1979, *Introduction to Automata Theory*, Addison-Wesley, Reading, Mass.
[23] Kraak, M-J., MacEachren, A. M., 1994, Visualization of the Temporal component of Spatial data, *Proceedings of the Sixth International Symposium on Spatial Data Handling*, Edinborough, Scotland, pp. 391-409.
[24] Kraak, M-J., Edsall, R., MacEachren, A. M., 1998, Cartographic Animation and Legends for Temporal Maps: Exploration and/or Interaction, *Proceedings of the Seventh Annual Conference of Polish Spatial Information Association*, Warsaw, Poland, May 1998, pp. 287-296.
[25] Langran, G., 1992, Time in Geographic Information Systems, *Technical Issues in Geographic Information Systems*, Taylor & Francis, London, UK.
[26] Linz, P., 1996, *An introduction to formal languages and automata*, D. C. Heath and Company, Lexington, Massachusetts.
[27] Lück, H. B. and Lück, J., 1976, Cell number and cell size in filamentous organisms in relation to ancestrally and positionally dependent generation times, In *Automata, Languages, Development* , North-Holland, Amsterdam, Netherlands, pp. 109-124.
[28] Mazurkiewicz, A., 1989, Basic Notions of Trace Theory, *Proceedings of REX Workshop and Summer School "Linear Time, Branching Time, and Partial Order in Logics and Models for Concurrency"*, Noordwijkerhout, May/June1987, In *Lecture Notes in Computer Science* Vol. 354, Springer-Verlag, Berlin, pp. 285-363.
[29] McFarland, G. and Rudmik, A., 1993, Object-Oriented Database Management systems, *A Critical Review/ Technology Assessment*, Source: http://www.utica.koman.com/tech/oodbms.ToC.html
[30] Mihalache, V., Paun, G., Rozenberg, G. and Salomaa, A., 1996, Generating strings by replication: a simple case, *TUCS Technical Report No. 17,* Turku Centre for Computer Science, Turku, Finland.
[31] Mihalache, V. and Salomaa, A., 1998, Language-Theoretic Aspects of String Replication, *International Journal for Computer Mathematics* 66, pp. 163–177.

[32] Mioc, D., Anton, F., Gold, C. M. and Moulin, B., 1998, Spatio-temporal change representation and map updates in a dynamic Voronoi data structure, *Proceedings of the Eight International Symposium on Spatial Data Handling*, Vancouver, Canada, pp. 441-452.
[33] Mioc, D., Anton, F., Gold, C.M., and Moulin B., 2006, Map updates in a dynamic Voronoi data structure, *Proceedings of ISVD'06*, Banf, Alberta, pp. 264-269.
[34] Mioc, D., Anton, F., Gold, C.M., and Moulin B., 2007, Reversibility of the Quad-Edge operations in the Voronoi data structure, *Proceedings of ISVD'07*, Glamorgan, UK, pp. 135-144.
[35] Mioc, D., Anton, F., Gold, C.M., and Moulin B., 2009, On Kinetic Line Voronoi Operations and Finite Fields, *Proceedings of ISVD'09*, Copenhagen, Denmark, pp. 65-70.
[36] Mioc, D., Anton, F., Gold, C.M., and Moulin B., 2010, Kinetic Line Voronoi Operations and Their Reversibility, *Transactions on Computational Science*, pp. 139-165.
[37] NCGIA, 1988, Proposal to the geography regional science program at the National Science Foundation, *Technical Paper 88-1, National Center for Geographic Information and Analysis*, University of California, Santa Barbara, CA.
[38] Newell, R. and Batty, M., 1994, GIS databases are different, *AM/FM'94*, pp. 279-288.
[39] Okabe, A., Boots, B., Sugihara, K., 1992, *Spatial Tessellations - Concepts and Applications of Voronoi Diagrams*, Wiley & Sons, Chichester.
[40] Pequet, D. and Wentz, E., 1994, An Approach for Time-Based Spatial Analysis of Spatio-Temporal Data, *Advances in GIS Research, Proceedings 1*, pp. 489-504.
[41] Proulx, M-J., 1995, Développement d'un nouveau langage d'interrogation de bases de données spatio-temporelles, *Thèse de Maîtrise, Département des Sciences Géomatiques, Université Laval*, Québec, Canada.
[42] Prusinkiewicz, P., Lindenmayer, A., 1990, *The Algorithmic Beauty of Plants*, Springer-Verlag, New York.
[43] Rekers, J., 1995, A Parsing Algorithm for Context-Sensitive Graph Grammars, *Technical report 95-05 of Leiden University*, Leiden, The Netherlands. Source: ftp.wi.leidenuniv.nl, file /pub/CS/TechnicalReports/1995/tr-05.ps.gz.
[44] Roos, T., 1991, Dynamic Voronoi diagrams, *Ph.D. Thesis, University of Würzburg*, Germany.
[45] Van Oosterom, P. J. M., 1993, *Reactive Data Structures for Geographic Information Systems*, Oxford University, Bookcraft Ltd.
[46] Van Oosterom, P. J. M., 1997, Maintaining Consistent Topology including Historical Data in a Large Spatial Database, In *1997 ACSM/ASPRS Annual Convention & Exposition, Seattle, Washington, Auto-Carto 13,* Vol. *5*, pp. 327-336.
[47] Vaario, J., 1993, An Emergent Modeling Method for Artificial Neural Networks, *Doctoral dissertation, University of Tokyo*, Japan.
[48] Winskel, G., 1989, An introduction to event structures, *Proceedings of REX Workshop "Linear Time, Branching Time, and Partial Order in Logics and Models for Concurrency", Lecture Notes in Computer Science*, Vol. 354, Springer, Berlin, pp. 365-397.