Multi-VMap: a Multi-Scale Model for Vector Maps

R. Viaña[1], P. Magillo[2], E. Puppo[2] and P.A. Ramos[1]

[1] Department of Mathematics, University of Alcala, Spain
[2] Department of Computer Science, University of Genova, Italy

Summary. *Multi-VMap* is a compact framework from which plane graphs representing geographic maps at different levels of detail can be extracted. Its main feature is that the scale of the extracted map can be variable through its domain, while each entity maintains consistent combinatorial relations with the rest of entities represented in the map. The model is based on a set of operators, called updates, which modify the level of detail in a small portion of a map. The set of updates is partially ordered, and can therefore be represented as a Directed Acyclic Graph, which defines our multi-scale structure. An algorithm to extract a map at the required resolution is proposed, and a lower bound for the number of different maps which can be extracted from the model is given. The model supports map data processing operations (e.g., querying), as well as progressive and selective transmission of maps over a network.

1 Introduction

In a computational environment, geographic maps admit two main types of representation: *raster* and *vector* [24, 52]. Raster maps are formed by discrete elements, or pixels, each of which is defined by its spatial position and has associated a set of attributes. A vector map, on the contrary, is defined in terms of the structured elements composing it, called entities. A plane vector map is formed by a set of points, lines and regions, characterized by their topological relations (representing the linking of entities), geometry, and semantics. Both types of map description, as well as the hybrid one, have been broadly employed over the last decades. None of them has proved to be more appropriate than the other one in absolute terms, the suitability of which one to adopt depending on the particular application. In this paper, we will follow a vector approach to develop an entity-based multiresolution model.

The advantages of having a multiresolution model of any description of an object are well known: for some applications a representation of the object in its full complexity is not required. The bigger the resolution the bigger the space necessary to store the model and the time complexity to process it.

Therefore, it is quite inefficient to work at full resolution if this is not necessary. A multi-scale model is a compact structure relating representations of the same geographic map at different scales, where by *scale* we mean level of detail of the entities forming in the map. Multi-scale modelling has a fundamental importance in allowing for map storage, processing and visualization with an acceptable trade-off between cost and performance, in case of very large data sets. In fact, maps extracted according to application requirements will be definitely smaller than the original map at full resolution. The multi-scale representation for maps developed in this work also allows the transmission of maps over a network, both progressively (i.e. sending a coarse map first, followed by details to incrementally improve the level of detail), and selectively (i.e. concentrating the transmission of details in some parts of the map).

It is well known that the structure of 2-D vector maps can be suitably represented by means of *plane graphs* (properly speaking, plane pseudographs, allowing multiple lines and self-loops). Given a plane graph $\mathcal{G}$, the *geometry* of $\mathcal{G}$ is defined by the coordinates of its vertices and the geometry of its edges, whereas the *combinatorics* (also called *topology*) of $\mathcal{G}$ is given by the incidence relations, which reflect how the entities in $\mathcal{G}$ are linked. Geometry and combinatorics can best be treated by means of different mathematical tools. The combinatorial structure of plane graphs is mathematically modelled by *Abstract Cell Complexes*. This algebraic structure can capture the whole topological nature of maps, independently of their geometry [10, 26, 29]. In this work only the combinatorial aspects of maps are considered, the model developed being fully combinatorial.

Two maps of the same area at different resolutions must be *topologically consistent*, i.e., objects that appear in both maps must maintain compatible spatial relationships. For instance, objects that meet before simplification cannot possibly be disjoint in the simplified map [31]. This might seem contradictory with traditional cartographic generalization principles. However, we believe that the digital management of spatial data must not necessarily be ruled by ad-hoc cartographic rules, based on practical experience responding mainly to a visualizing purpose.

We consider an input map at a high level of detail and we generalize it through a procedure, which consists of a sequence of continuous functions, called *updates*, over the corresponding complex, as indicated in [33, 3], until we obtain a drastically generalized map. The sequence of updates is recorded and a partial order among them is defined, which is based on a concept of *dependency*. Two updates are mutually dependent if the combinatorial structure of the map at the highest available level of detail cannot be retrieved if the two updates are permuted in the sequence. The resulting framework can be represented as a Directed Acyclic Graph (DAG) having updates as its nodes and direct dependency links as its arcs. Each update in the DAG is a local modification of the map, and thus corresponds to a local change of scale.

We propose an algorithm for extracting variable-scale maps from such a framework, which is based on a top-down traversal of the DAG starting at its

root, that corresponds to the most generalized map. For each node traversed, the corresponding local generalization update is possibly undone (i.e., the map is locally refined), depending on input parameters that specify the desired level of detail.

Obtaining a map which contains as less entities as possible and which allows spatial analysis is the main functionality of our model. An example is shown in Figure 1. In (a), we show a map at a small scale, as depicted in the screen of an electronic device. Instead of managing the map at the highest available level of detail that contains the zone of interest, shown in (b), our model allows the user to manage a map in which detail is high just where necessary, hence saving computational resources. The user could also do vectorial flight simulation.

The rest of this paper is organized as follows. In Section 2 we report on related work. In Section 3 we give preliminaries on representing vector maps with plane graphs, capturing the combinatorial structure of a map by an abstract cell complex, and we describe the atomic generalization updates. In Section 4 we introduce their corresponding refinement updates that are at the basis of our work. In Section 5 we consider collections of refinement updates applied on a base map. In Section 6, we define feasible subsequences. In Section 7, we present the concept of dependency among updates, and the corresponding partial ordering on the set of updates. In Section 8 we study feasible subsequences of updates, and we construct the operator to combine updates in a feasible subsequence. In Section 9 we define our multiresolution model, which naturally derives from the concept of feasible sequence, we introduce an algorithm to extract a map at variable scale, and we evaluate the expressive power of the model. Finally, in Section 10 we make some concluding remarks.

2 Related Work

Cell complexes have proven to be of great value in geographic information systems to model vector objects [23, 50]. The suitability of using cell complexes is based on the theoretical results of *algebraic topology* [1, 39], which provide means to check the consistency of representations defined on the basis of such concepts. Simplicial complexes are particular cell complexes suitable to encode triangulated maps [15]. They could as well be used to represent plane graphs after a pre-processing triangulation phase [18, 51]. However, this solution is inefficient, and it is not conceptually clean, since it focuses on the triangles (that were introduced artificially just to decompose the domain) rather than on the semantically relevant entities of the map. The same problem arises for the rest of *euclidean cell complexes*, as those complexes with convex cells [17], or CW-complexes [8, 32, 35, 53].

First existing multi-scale spatial databases, known as as *multiple representations*, consisted of collections of maps at different scales linked by hierar-

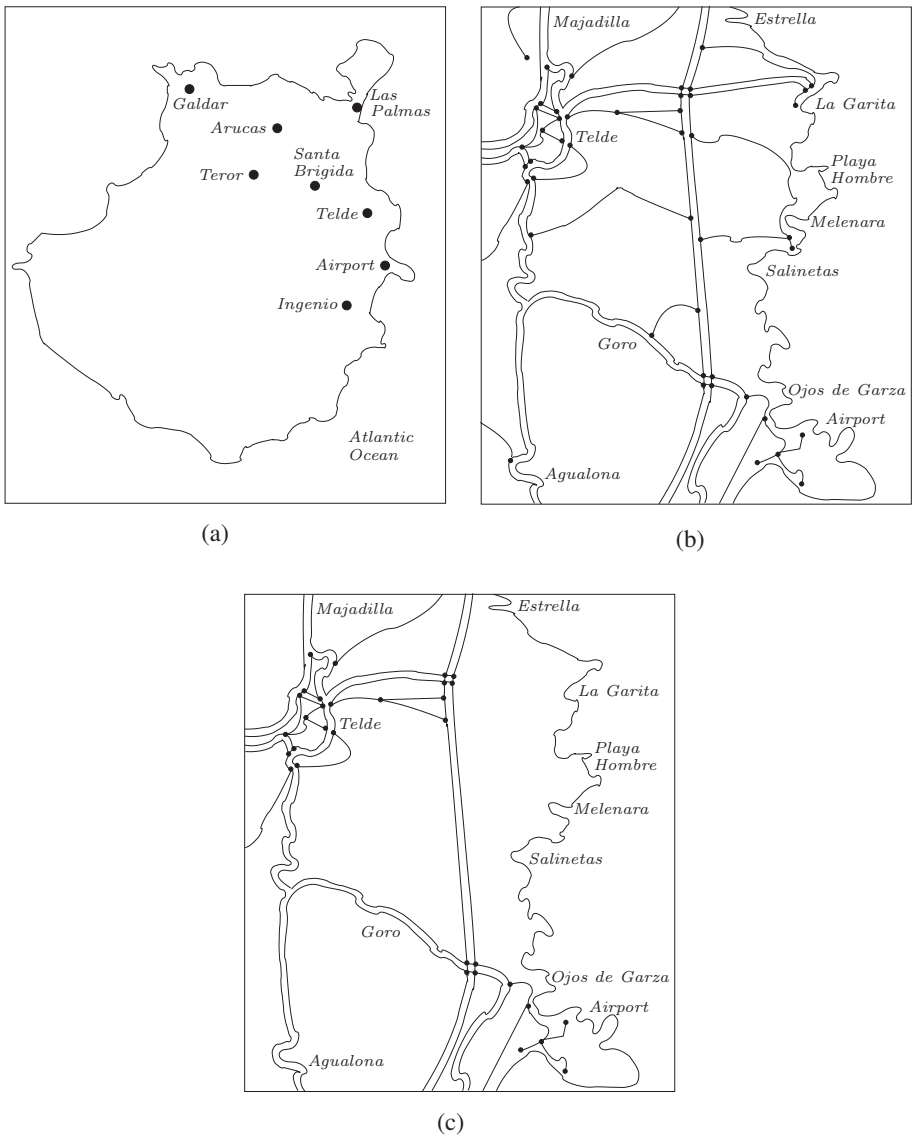


Fig. 1. (a) Map at a small scale. (b) Following a hierarchical approach, the map of the same area at larger scale is divided in portions, and each of them is stored separately as another map. (c) In our approach, the level of detail can be increased just where necessary, while all the entities forming the rest of the map maintain consistent topological relations.

chical structures [16, 44, 43]. This is the principle underlying several internet

applications (www.multimap.com, www.mapblast.com, www.mapquest.com, etc.). In this view there is no relation between the different representations of the same object at distinct scales. Moreover, it is not possible to obtain a map in which the level of detail is variable through space (see Figure 1).

For the sake of consistency, other models have been developed which include information about how different entities are related in maps at different levels of detail [19]. On this basis, the first hierarchical model formally defined on a mathematical basis is [4], consisting of a tree of maps at different resolutions, where each map is a refined description of a region of its parent vertex in the tree. This model is improved in [33], where combinatorics is separated from geometry and semantics. In [3], a set of generalization operators has been proposed, which support the encoding of relations between different representations of the same entity at different scales, by consistent mapping of entities from an input map onto their corresponding entities in a generalized map. *Euler operators* are general topological operators which allow the manipulation of any plane graph, by supporting both insertion and removal of any set of entities in the graph [30]. However, Euler operators do not allow to keep track of the simplified entities and their mutual relationships in a consistent way, and this is fundamental in order to build a multi-scale model. The work of Stell and Worboys [40, 41, 42], based on the formalisation of models for plane graphs at different levels of detail, presents quite similarities to our work here. They use different simplification operators, with much more cartographic meaning.

Spatial indexes, such as *quadtrees*, are techniques for processing collections of entities based on their spatial position, and are suitable to locate entities in a map [37, 38]. *Reactive data structures* [46] are hierarchical structures which allow the retrieval of descriptions of an object at different levels of detail by traversing a tree. The R-tree is a spatial indexing which consists of a hierarchical representation of minimum bounding rectangles of objects. Some examples of reactive structures are the *reactive tree* (based on the R-tree), the *BLG-tree* for line generalization, and the *GAP-tree*, to support generalization of area partitioning [49].

Recent approaches to multi-scale are often aimed at progressive transmission of maps. In [6, 7], a method is proposed for the progressive transmission of lines, which is based on the popular Douglas-Peucker simplification algorithm [14]. It uses tree structures, as in the BLG tree of [48]. The model avoids undesired intersection between lines, but the topological consistency of the simplified model, as we define here, is not ensured. In [5] a hierarchical model maintaining vertical links between representations of the same entity at two consecutive levels is proposed, which is based on the same set of operators, proposed in [3], that we consider here. This model supports the progressive transmission of geographic maps in vector format, but it does not support multi-scale operations such as focusing on a given area or extracting a representation with variable scale through the domain of the map.

Instead of considering several maps at different fixed scale levels, *on-demand* mapping refers to the creation of a map whose scale and purpose are appropriate to user needs. This approach has been the subject of research [25] focussing on the generalization issues. In [9, 36], other procedures to perform map generalisation while preserving topological consistency are proposed. One main feature is that instead of considering consistency regarding to complete objects, only lines are considered. The concept of topological consistency only involves geometric constraints. In [47], a vertex labelling procedure is proposed that guarantees topological consistency across retrieved levels of detail.

Multiresolution models have been broadly studied for triangulated meshes [11, 20, 28]. Such approaches could be applied to geographic maps as well, provided that maps could be triangulated in a pre-processing step. On a more abstract perspective, such multiresolution models are all based on two simple concepts: a set of updates that are applied locally to modify the structure of an object; and relations among those updates that control how they can be combined to extract representations at variable resolution. In our approach, we follow a similar strategy, in a rather different context, though.

3 Background

In this Section, we review basic concepts on plane graphs, the modelling of the combinatorics of a plane graph by abstract cell complexes, and the topological operators which will be used in our approach as functions to decrease the level of detail in a map.

3.1 Vector Maps Represented as Plane Graphs

A *plane pseudograph* is a set of *vertices* and *edges* in the plane satisfying:

1. each edge joins either two different vertices, or a vertex to itself,
2. the *faces* of the graph are the connected components of the plane obtained by removing the vertices and edges from it,
3. no two elements of the graph intersect.

We will refer to plane pseudographs as *plane graphs*. Edges joining a vertex to itself are called *loops*, and different edges joining the same pair of vertices are called *multiple edges*. The boundary of an edge is formed by the vertices it joins. The boundary of a face is formed by the vertices and edges delimiting it, and can be further subdivided into *proper boundary* and *features*. The proper boundary of a face is formed by those vertices and edges bounding both it and at least another different face. The elements in the proper boundary of a face form cycles. There is one unbounded face called the *infinite face*. The proper boundary of a face which is not the infinite one will be formed by exactly one *outer boundary* (surrounding the whole face), and possibly some *inner boundaries* (surrounding holes in the face). The infinite face has no outer

boundary, while it may have one or more inner boundaries. The *features* of a face are those connected sets of vertices and edges which bound just such face. The *feature-vertices* or *feature-points* of a face are the isolated vertices puncturing it, and its *feature-edges* or *feature-lines* are those edges dangling inside.

A plane graph has a geometric and a a combinatorial structure, implicitly related. The geometry of the map is given by the coordinates of its vertices and the geometry of its edges. The combinatorial structure of the graph reflects how the elements of the graph are connected. The connectivity information of a plane graph is usually expressed by *incidence* and *adjacency relations*, also known as *topological relations*, between the elements of the graph. An edge and each of the vertices it joins are said to be mutually *incident*, as well as a face and each of the vertices and edges bounding it. Two vertices or two faces which are incident at a common edge, as well as two edges incident at a common vertex are said to be *adjacent*. For further details on this representation refer to [10, 33].

A vector map M can be effectively represented by a plane graph as the collection of its vertices, edges and faces, also known as points, lines and regions respectively, together with their connectivity structure. We will use the term *entity* to denote a point, a line, or a region of a map. In practice, a map will not cover the whole plane, but a portion of it, called the *domain* of the map. This is the portion of plane covered by all entities of the map, except for the infinite region (see Fig. 2).

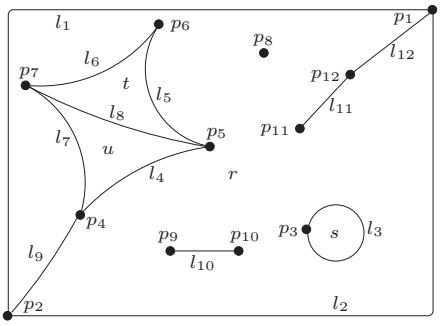


Fig. 2. Map whose domain is covered by regions r, s, t, u, and the entities bounding them. The boundary of region r is decomposed into its *outer boundary*, formed by cycle p_1, l_1, p_2, l_2; two *inner boundaries*: one formed by loop l_3 and its endpoint p_3, and the other one formed by cycle p_4, l_4, p_5, l_5, p_6, l_6, p_7, l_7; and four *features*: l_9 (*feature-line*); p_9, l_{10} (*feature-line*), p_{10}; p_{11}, l_{11} (*feature-line*), p_{12}, l_{12} (*feature-line*); and p_8 (*feature-point*).

3.2 Modelling the Combinatorial Structure of a Vector Map

An *Abstract Cell Complex* (ACC), defined in [26], is a purely topological structure which captures the combinatorial structure of a plane graph.

Definition 1. *An ACC Γ is a triple $(C, \prec, dim)$, where:*

- *C is a finite set, called the* set of cells*;*
- *$\prec$ is a strict partial ordering on the elements of C (i.e., $\prec$ is an irreflexive, antisymmetric, and transitive binary relation) called the* bounding relation*;*
- *$dim : C \to \mathbb{N}$, called the* dimension function*, is such that*

$$\gamma \prec \gamma' \Rightarrow dim(\gamma) < dim(\gamma'), \qquad \forall\ \gamma, \gamma' \in C.$$

Given a plane graph, let C be the set formed by the union of its vertices, edges and faces, and dim : $C \to \mathbb{N}$ be the function which takes value 0 on the vertices, 1 on the edges, and 2 on the faces of the graph. Defining partial order $\prec$ as:

$$a \prec b \iff b \text{ belongs to the boundary of } a,$$

it is clear that $(C, \prec, dim)$ is an ACC. Topological relations between entities in a map can be translated in terms of cells in the corresponding complex. We provide next some definitions referring to ACCs:

Definition 2. *Given ACC $\Gamma = (C, \prec, dim)$:*

(a) The boundary *of a cell γ of Γ is defined as:*

$$\partial\gamma = \{\xi \in C \mid \xi \prec \gamma\}.$$

(b) A cell γ for which $dim(\gamma) = k$ is called a k-cell*. An ACC is called* d-dimensional *complex or a* d-complex *if $max_{\gamma \in C}(dim(\gamma)) = d$.*

(c) A subcomplex *$\Gamma_a = (C_a, \prec_a, dim_a)$ of $\Gamma = (C, \prec, dim)$ is a complex whose set C_a is a subset of C, and both relation $\prec_a$ and function dim_a are restrictions of $\prec$ and dim to C_a respectively. We call the* difference of Γ and Γ_a*, and denote it $\Gamma \setminus \Gamma_a$, as the subcomplex of Γ whose set of cells is $C \setminus C_a$.*

(d) Given ACC $\Gamma' = (C', \prec', dim')$, and strict partial order relation I on $C \cup C'$, we call the I-union *of Γ and Γ', and denote it $\Gamma \cup_I \Gamma'$, to complex $\Gamma'' = (C'', \prec'', dim'')$, where:*

- *$C'' = C \cup C'$,*
- *dim'' is defined as dim on C and dim' on C',*
- *function $\prec''$ is defined as:*

$$\xi \prec'' \gamma \iff \begin{cases} \xi \prec \gamma, & \forall\ (\xi, \gamma) \in C \times C, \\ \xi \prec' \gamma, & \forall\ (\xi, \gamma) \in C' \times C', \\ \xi\ I\ \gamma, & \forall\ (\xi, \gamma) \in (C \times C') \cup (C' \times C)\ s.t.\ dim''(\xi) < dim''(\gamma), \end{cases}$$

We say that I is a bounding relation between Γ *and* Γ'.

(e) Let k be the dimension of Γ, with $0 \leq k \leq 2$. A planar embedding of Γ is a mapping $f : C \to \mathbb{R}^2$ such that no two images of cells in C intersect, the image of a 0-cell is a point, the image of a 1-cell is a line, and the image of a 2-cell is a region in the plane, and satisfying that:

$$\gamma \prec \gamma' \Leftrightarrow f(\gamma) \text{ belongs to the boundary of } f(\gamma').$$

In the remainder of the paper, a map M will be represented as a pair (Γ, f). With an abuse of notation, we will indistinctly speak about the entities of a map, namely points, lines and regions, and the $0-$, $1-$ and $2-$ cells of its associated complex.

3.3 Topological Abstraction Operators

In the description of topological operators to decrease the level of detail on a map, we refer to the theory of map generalization proposed in [33] and developed in [3]. Abstraction operators are regarded as functions between ACCs Γ and Γ' corresponding to two maps M and M', where M' is a generalized version of M. A function is said to be *consistent* if it is *surjective, monotonic*, and *preserves connected sets of cells by inverse image*:

- Due to the surjectivity assumption, an entity cannot appear in Γ' without being the image of some entity in Γ.
- Monotonicity is defined in terms of the bounding relation of ACCs Γ and Γ'. It guarantees that two entities that are not incident in the co-domain cannot be the images of entities incident in the domain.
- The preservation of connected sets by inverse image does not permit to map completely disconnected sets of entities onto connected sets.

The reader is referred to [12, 33] for a discussion about the suitability of atomic abstraction updates as functions to decrease the level of detail on a map while maintaining a trade-off between cartographic principles and combinatorial consistency of the model.

The pair $C = (O, A)$, where O is the set of all ACCs and A is the set of all possible consistent combinatorial transformations between objects in O, allows to give a formalization of the problem in terms of *category theory* [2]. There exists a set of seven functions, called *atomic operators*, that are necessary and sufficient to generate *all* strictly non-injective consistent combinatorial transformations; thus, consistency is implicit in multiple representation models built on the basis of such operators. They will be taken in our approach as the set of abstraction updates. The interested reader can find the complete theoretical development in [3].

An atomic operator is a consistent function mapping two or three entities onto one entity, or, in other words, it deletes two or three entities from a

map containing them, and replaces them with a new entity. All remaining entities are mapped one-to-one. The seven atomic operators are described in the following:

(a) line-to-point

$$ltp : \{(p, p', l), \prec\} \rightarrow \{p_0\},$$

with $p \prec l$ and $p' \prec l$.

(b) region-to-point

$$rtp : \{(p, l, r), \prec\} \rightarrow \{p_0\},$$

where $p \prec l$, $p \prec r$, $l \prec r$, and there is no any entity different from p or l which bounds r.

(c) region-to-line

$$rtl : \{(l, l', r), \prec\} \rightarrow \{l_0\},$$

where $l \prec r$, $l' \prec r$, and there is no entity different from l, l', or some of the endpoints of these two lines which bounds r.

(d) line-merge

$$lm : \{(p, l, l'), \prec\} \rightarrow \{l_0\},$$

where $p \prec l$, $p \prec l'$, and there is no line l'' different from l or l' s.t. $p \prec l''$.

(e) region-merge

$$rm : \{(l, r, r'), \prec\} \rightarrow \{r_0\},$$

with $l \prec r$ and $l \prec r'$.

(f) isolated-point-removal

$$ipr : \{(p, r), \prec\} \rightarrow \{r_0\},$$

where $p \prec r$ and there is no line l s.t. $p \prec l$.

(g) feature-line-removal

$$flr : \{(l, r), \prec\} \rightarrow \{r_0\},$$

where $l \prec r$ and there is no region r' such that $r' \neq r$ and $l \prec r'$.

The seven atomic operators are depicted in Fig. 4 together.

4 Update Operations on a Map

Given map $M = (\Gamma, f)$, we address the problem of creating another map $M' = (\Gamma', f')$, at a lower level of detail. The application of an *update* on M consists in replacing some cells of Γ by some other cells, and modifying the combinatorics so that another map is obtained. We are interested in how updates change the combinatorics of a map. Of course every combinatorial variation involves a geometric change, but in this paper we are concerned with creating a multiresolution model combinatorially consistent. Once a framework satisfying this constraint is obtained, geometry should be fixed for the model being also geometrically consistent.

Definition 3. *Let $M = (\Gamma, f)$ be a map, and u be a triple (Γ_a, Γ_b, I), where:*

1. *Γ_a is a subcomplex of Γ,*
2. *Γ_b is an Abstract Cell Complex (ACC) of dimension less than or equal to 2,*
3. *I is a bounding relation between cell complexes Γ_b and Γ_A, where Γ_A is the set of cells of $\Gamma \setminus \Gamma_a$ which are incident at some cell of Γ_a,*

If a planar embedding f' of $\Gamma_b \cup_I \Gamma_A$ exists, such that the replacement in M of $f(\Gamma_a \cup \Gamma_A)$ by $f'(\Gamma_b \cup_I \Gamma_A)$, is another map, we say that u is an update *on M.*

In Figure 3 (a), update u_1 is applied on map $M = (\Gamma, f)$ to the left of the figure, where $u_1 = (\Gamma_a, \Gamma_b, I)$ is defined as follows:

1. Γ_a is the subcomplex of Γ formed by line l and region r.
2. Γ_b is an ACC formed by region r'.
3. I specifies that each cell of $\Gamma \setminus \Gamma_a$ that is incident at some cell of Γ_a will bound r'.

Clearly, as the effect of this update is the removal of one feature-line, maintaining the planar embedding of the rest of points and lines in the map produces another map.

In Figure 3 (b), on the contrary, we show an example of application of a triple $u_2 = (\Gamma_a, \Gamma_b, I)$ which is not an update. It is defined by:

1. Γ_a is the subcomplex of Γ formed by point p and region r.
2. Γ_b is an ACC formed by region r'.
3. I specifies that each cell of $\Gamma \setminus \Gamma_a$ that is incident at some cell of Γ_a will bound r'.

As a result of the replacement of Γ_a by Γ_b according to I, there is a line which is not a loop and which has exactly one endpoint. Hence, there is not any embedding from which a map can be obtained.

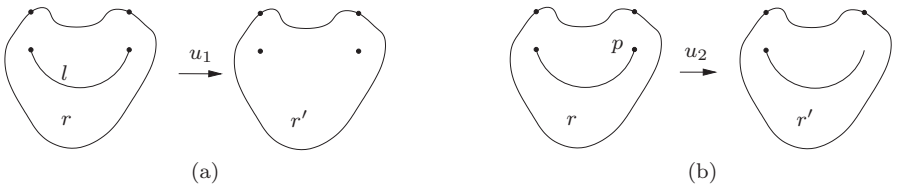


Fig. 3. Modification u_1 in (a) is an update, whereas u_2 in (b) is not.

Given update $u = (\Gamma_a, \Gamma_b, I)$, let $|\Gamma_a|$ be the number of cells in Γ_a and $|\Gamma_b|$ be the number of cells in Γ_b.

- if $|\Gamma_a| > |\Gamma_b|$, u is an *abstraction update*, and we will represent u as:

$$\underline{u} : \Gamma^+ \xrightarrow{I} \Gamma^-,$$

  with $\Gamma^+ = \Gamma_a$ and $\Gamma^- = \Gamma_b$. Superscripts $+$ and $-$ are used to denote more or less amount of cells respectively of the corresponding complex.
- if $|\Gamma_a| < |\Gamma_b|$, u is a *refinement update*, and we will write:

$$\overline{u} : \Gamma^- \xrightarrow{I} \Gamma^+,$$

  with $\Gamma^- = \Gamma_a$ and $\Gamma^+ = \Gamma_b$.

The atomic operators described in the previous Section are functions deleting two or three entities (complex Γ^+) from a map containing them, and replacing them with a new entity (complex Γ^-). All remaining entities are mapped one-to-one. As they are monotonic functions, each entity that was incident at one of the entities which are removed becomes incident at the entity replacing it. Thus, relation I of Definition 3 is implicitly given. It is immediate to see that a planar embedding of Γ^- and all the cells which become incident at it can be given, such that the result of replacing Γ^+ by Γ^- on the corresponding map continues being a map. Thus, atomic operators correspond to abstraction updates, called *atomic abstraction updates*. As the definition of I is the same for every type of atomic abstraction update, it can be omitted, and we can denote an atomic abstraction update as:

$$\underline{u} : \Gamma^+ \rightarrow \Gamma^-$$

or equivalently:

$$\underline{u} : \{(\gamma_a^{(i)}, \gamma_b^{(j)}, \gamma_c^{(k)}), \prec\} \rightarrow \{\gamma_d^{(l)}\},$$

with possibly $\{\gamma_c^{(k)}\} = \emptyset$. Superindexes (i), (j), (k), (l), with $0 \leq i, j, k, l \leq 2$ indicate the dimension of the corresponding cell, i.e., if $i = 0$, $\gamma_a^{(i)}$ is a point, if $i = 1$, $\gamma_a^{(i)}$ is a line, and if $i = 2$, $\gamma_a^{(i)}$ is a region. With this notation, function *dim* of Definition 1 is implicitly given.

4.1 Atomic Refinement Updates

For each atomic abstraction update $\underline{u} : \{(\gamma_a^{(i)}, \gamma_b^{(j)}, \gamma_c^{(k)}), \prec\} \rightarrow \{\gamma_d^{(l)}\}$, a corresponding *atomic refinement update*, which will replace entity $\{\gamma_d^{(l)}\}$ by $\{(\gamma_a^{(i)}, \gamma_b^{(j)}, \gamma_c^{(k)}), \prec\}$ will be defined. In order to univocally define an atomic refinement update, incidence relation I on those entities incident at $\gamma_d^{(l)}$ must be specified. This latter part is crucial to ensure the correctness of refinement, but it makes refinement more complicated than abstraction. Note that, depending on how combinatorial relations are recovered, different maps could

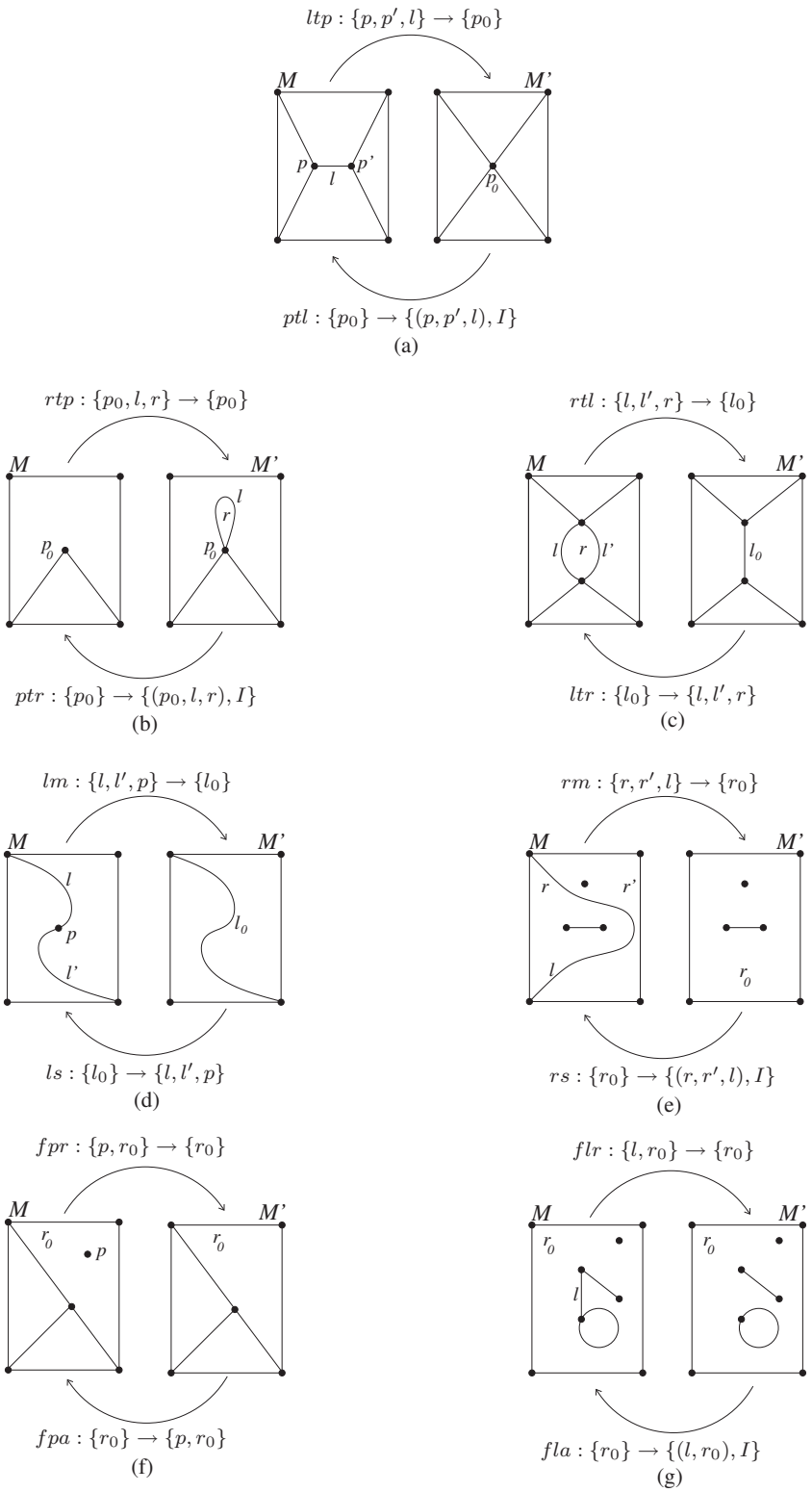


Fig. 4. Pairs of mutually inverse atomic abstraction/refinement updates.

be obtained. In Fig. 5 we show alternative atomic refinement updates, applied on each map M' which was obtained applying the atomic abstraction updates shown in Fig. 4 (a), (b), (e), and (g). We observe that, although the entities replaced are exactly the same, the corresponding original map M of Fig. 4 is not retrieved.

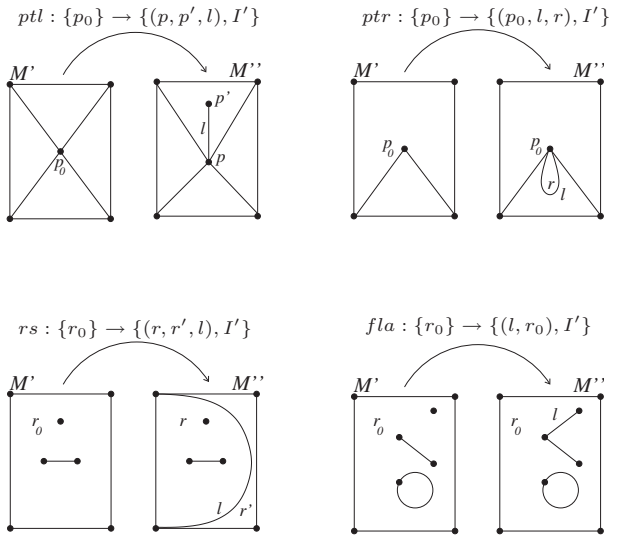


Fig. 5. The atomic refinement updates shown here are not inverse updates of the corresponding atomic abstraction updates in Fig. 4 (a), (b), (e), and (g).

The atomic refinement updates are described in the following, and pairs of inverse atomic updates of each type are represented in Fig. 4.

(a) *point-to-line* $ptl : \{p_0\} \rightarrow \{(p, p', l), I\}$,
where l is a line whose endpoints, p and p', are different. To uniquely specify the combinatorial relations of all the lines and regions incident at p_0, it can be easily proved that it is sufficient that I specifies which of the lines incident at p_0 become incident at p and/or p' and, in case l is a feature-line, which of the regions incident at p_0 will contain such line (see Fig. 4 (a)).

(b) *point-to-region* $ptr : \{p_0\} \rightarrow \{(p, l, r), I\}$,
where r is a simply-connected region without internal features, whose boundary is formed just by a loop l and its endpoint p, and I must specify which of the regions incident at p_0 must contain line l (see Fig. 4 (b)).

Since points p and p_0 are coincide their only combinatorial difference being that p is incident at l and r, while p_0 is not, points p_0 and p will be considered to be the same point, and we will write $ptr : \{p_0\} \rightarrow \{(p_0, l, r), I\}$.

(c) *line-to-region* $ltr : \{l_0\} \rightarrow \{l, l', r\}$,
where r is a simply-connected region without internal features, whose boundary is formed by two distinct lines, l and l' and their common endpoints (see Fig. 4 (c)). As the flattening of $\{l_0\}$ to $\{l, l', r\}$ is univocal, the explicit specification of I is not required.

(d) *line-split* $ls : \{l_0\} \rightarrow \{l, l', p\}$,
where l, l', with $l \neq l'$, are two (non-loop) lines incident at a common point p, which does not bound any other line (see Fig. 4 (d)). In this case, I is univocally defined from the corresponding atomic abstraction update.

(e) *region-split* $rs : \{r_0\} \rightarrow \{(r, r', l), I\}$,
where r, r', with $r \neq r'$, are two regions having a common bounding line l, and I must specify the two points p_1 and p_2 that will bound line l, and how to partition the inner boundaries and features of r_0 which will continue being inside one of the two newly created regions into two subsets: the inner boundaries and features that will bound r, and the inner boundaries and features that will bound r' (see Fig. 4 (e)).

(f) *feature-point-addition* $fpa : \{r_0\} \rightarrow \{p, r\}$,
where p is an isolated point inside region r (see Fig. 4 (f)).
Combinatorially, regions r_0 and r only differ in that r is incident at point p and r_0 is not. Therefore, we will consider r to be the same region as r_0, and we will write $fpa : \{r_0\} \rightarrow \{p, r_0\}$.
Relation I is univocally defined, so it has not been specified.

(g) *feature-line-addition* $fla : \{r_0\} \rightarrow \{(l, r), I\}$,
where l is a feature-line in region r, and I must specify the endpoints of line l (see Fig. 4 (g)).
The only combinatorial difference between r_0 and r is that r is incident at l and r is not. Thus, we will consider r_0 to be the same region as r and we will write $fla : \{r_0\} \rightarrow \{(l, r_0), I\}$.

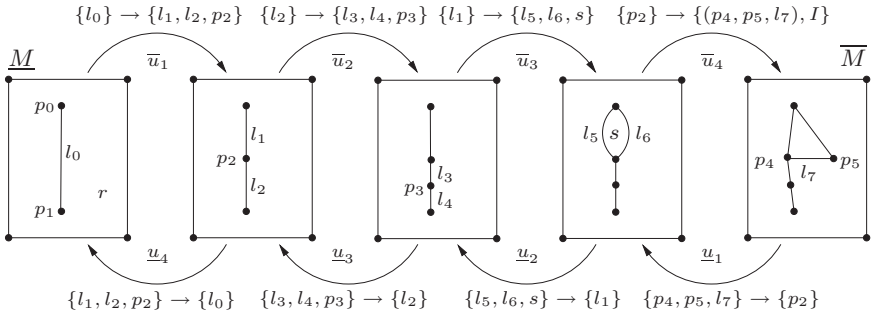
Given a map on which a sequence of abstraction updates has been applied, atomic refinement updates allow the sequence to be reversed.

5 Sequences of Atomic Refinement Updates

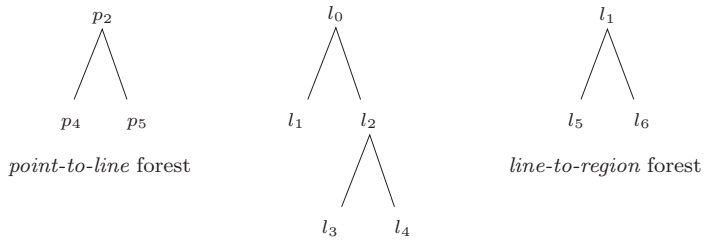
Given a map $\overline{M}$, any generalization of $\overline{M}$ can be obtained from it by applying a suitable sequence of abstraction updates [3]. Let us define a *sequence of atomic abstraction updates* for $\overline{M}$ as a pair $\underline{S} = (\overline{M}, U = (\underline{u}_1, \underline{u}_2, \ldots, \underline{u}_k))$, where $U = (\underline{u}_1, \underline{u}_2, \ldots, \underline{u}_k)$ is a collection of atomic abstraction updates, satisfying that $\underline{u}_1$ is an update on $\overline{M}$, and every $\underline{u}_i$, $2 \leq i \leq k$, is an update on the map obtained by applying to $\overline{M}$ all updates preceding $\underline{u}_i$ in the collection. We will denote $\underline{M}$ the generalized map obtained by applying all updates of $\underline{S}$ to $\overline{M}$ in

the given sequence. Note that, since atomic abstraction updates are defined as functions, their composition is also a function, called the *generalization function*, which provides a correspondence between each entity in $\overline{M}$ and its representative in $\underline{M}$. In general, each entity in $\underline{M}$ will be the representative of one or more entities in $\overline{M}$, forming its inverse image through the generalization function.

We will also denote as $\overline{S} = (\underline{M}, U = (\overline{u}_1, \overline{u}_2, \ldots, \overline{u}_k))$ the inverse *sequence of atomic refinement updates*, where, for $1 \leq i \leq k$, $\overline{u}_{k-i+1}$ and $\underline{u}_i$ are mutually inverse updates, and $\underline{M}$ is the map obtained from $\overline{M}$ by applying all modifications of $\underline{S}$ to it. In our notation, the upper bar is meant to suggest higher detail and, conversely, the lower bar is meant to suggest lower detail. Fig. 6 (a) shows a sequence of atomic abstraction updates (when going from $\overline{M}$ to $\underline{M}$), and its inverse sequence of atomic refinement updates (when going from $\underline{M}$ to $\overline{M}$).



(a)



(b)

Fig. 6. (a) $\underline{S} = (\overline{M}, U = (\underline{u}_1, \underline{u}_2, \underline{u}_3, \underline{u}_4))$ is a sequence of atomic abstraction updates and $\overline{S} = (\underline{M}, U = (\overline{u}_1, \overline{u}_2, \overline{u}_3, \overline{u}_4))$ is its inverse sequence of atomic refinement updates. (b) Forests of updates corresponding to the sequence represented in (a).

Map $\underline{M}$ is the map at the smallest scale (lowest detail) that we will consider, and we will call it the *base map*. Conversely, map $\overline{M}$ is the map at the largest scale (largest detail) that we will consider, and we will call it the *reference map*.

Sequence $\overline{S} = (\underline{M}, U = (\overline{u}_1, \overline{u}_2, \ldots, \overline{u}_k))$ is *non-redundant* if each entity c_i, that has been removed from an update $\overline{u}_i$ in the sequence, is never introduced again from another update $\overline{u}_j$ following $\overline{u}_i$ in the sequence. Redundant sequences are not useful, so we will always assume to deal with non-redundant sequences.

The following forests formed by entities appearing in sequence $\overline{S}$ will be created:

- In a *point-to-line* forest: roots are points created from updates of type *line-split*, or type *feature-point-addition*; and the children of a node are the two points created from it in an update of type *point-to-line*.
- In a *line-split* [*line-to-region*] forest: roots are lines created from updates of types *point-to-line*, *point-to-region*, *region-split*, or *feature-line-addition*; and the children of a line are the two lines obtained from it through an update of type *line-split* [*line-to-region*].
- In a *region-split* forest: roots are those regions created from updates of type either *point-to-region* or *line-to-region*; and the children of a region are the two regions obtained from it through an update of type *region-split*.

Given an entity e, we define its *most abstract representative* E to be the root of the tree e belongs to. If e does not belong to any forest of updates, the most abstract representative of e is itself.

As all the entities in a tree of updates derive from the same entity (the root of the tree), under certain conditions a group of entities in the tree will be allowed to be represented by the same entity in maps obtained at intermediate scale between $\underline{M}$ and $\overline{M}$ which did not originally appear in $\overline{S}$.

Map $\overline{M}$ can be obtained from $\underline{M}$ by applying all updates in $\overline{S}$. However, it is also possible to apply the updates in $\overline{S}$ selectively, thus obtaining maps that are at an intermediate scale between $\underline{M}$ and $\overline{M}$. One obvious possibility is to truncate the sequence at some point, but it is also possible to apply other subsequences, which skip some of the updates in the original sequence. However, not all (combinatorially many) subsequences produce meaningful results. This subject will be the matter of the following section.

6 Feasible subsequences

Given sequence $\overline{S} = (\underline{M}, U = (\overline{u}_1, \overline{u}_2, \ldots, \overline{u}_k))$, we must analyse the problematic of applying a subcollection $U' = (\overline{u}_{i_1}, \overline{u}_{i_2}, \ldots, \overline{u}_{i_h})$ of U on $\underline{M}$. In this and the next three sections, we will tackle the following subjects:

- Define the necessary conditions to apply a given update, hence a partial order among updates. This is done by analysing each of the seven refinement updates and, for each of them, by defining what entities are strictly necessary to belong to a map in order to apply the update on it.
- Given a collection of updates consistent with the partial order, define an operator that specifies how to apply on $\underline{M}$ such collection of updates. The definition of this operator is necessary because each update in the collection is not applied in exactly the same situation as in the original sequence. Therefore, it is not always obvious what the effect of an update will be on a given map.
- Throughout this process, the combinatorial identity of the original sequence should not be lost, i.e, if a collection containing some of the updates in U is applied on $\underline{M}$, the subsequent application of the rest of updates in U should give as final result $\overline{M}$.

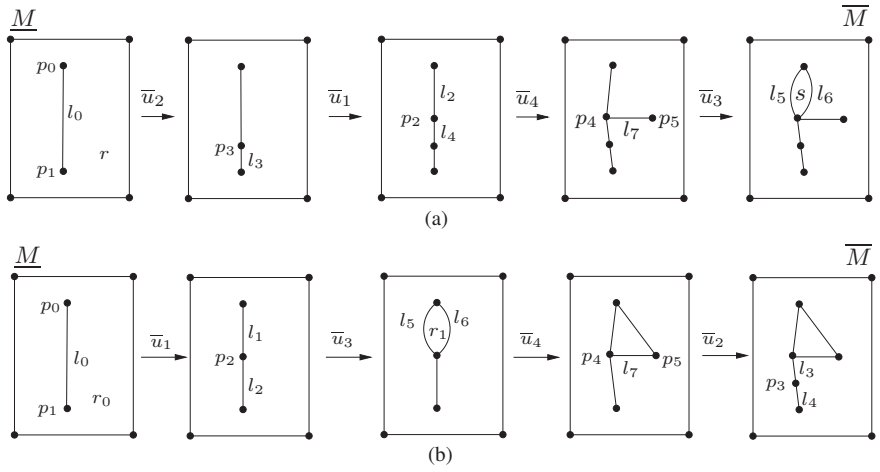


Fig. 7. Two different subsequences obtained from sequence in Fig. 6 (a). Sequence in (a) is not feasible, whereas sequence in (b) is.

On the basis of the previous explanation, we give the formal definition of subsequence and of subsequences we are interested in.

Definition 4. *Let* $\overline{S} = (\underline{M}, U = (\overline{u}_1, \overline{u}_2, \ldots, \overline{u}_k))$ *be a sequence of refinement updates.*

(i) A subsequence $\overline{S}'$ *of* $\overline{S}$ *is a triple* $(\underline{M}, U' = (\overline{u}_{i_1}, \overline{u}_{i_2}, \ldots, \overline{u}_{i_h}), \oplus)$*, where* $U' = (\overline{u}_{i_1}, \overline{u}_{i_2}, \ldots, \overline{u}_{i_h})$ *is a collection of updates belonging to* U*, and* $\oplus$ *is a composition operator specifying how the updates in* U' *must be sequentially*

applied on $\underline{M}$*, so that maps, denoted* $\underline{M} \oplus \overline{u}_{i_1} \oplus \overline{u}_{i_2} \oplus \ldots \oplus \overline{u}_{i_j}$*, for all* $1 \leq j \leq h$*, are successively obtained.*

(ii) Let $m = max\ \{i_1, i_2, \ldots, i_h\}$*, and let* M *be the map obtained by applying* $(\overline{u}_1, \overline{u}_2, \ldots, \overline{u}_m)$ *in* $\overline{S}$*. We say that subsequence* $\overline{S}' = (\underline{M}, U' = (\overline{u}_{i_1}, \overline{u}_{i_2}, \ldots, \overline{u}_{i_h}), \oplus)$ *is* feasible *if:*

$$M = \underline{M} \oplus \overline{u}_{i_1} \oplus \overline{u}_{i_2} \oplus \ldots \oplus \overline{u}_{i_h} \oplus \sum_{\substack{j=1 \\ j \neq i_1, \ldots, i_h}}^{m} \overline{u}_j$$

where the summations are performed by $\oplus$ *operator.*

In other words, we say that subsequence $\overline{S}'$ is feasible if the map obtained after applying all the updates in U on $\underline{M}$ is the same map as the map obtained after the application on $\underline{M}$ of the updates in U' followed by the updates in $U \setminus U'$. The subsequence shown in Fig. 7 (a) is not feasible, whereas the one shown in (b) is feasible.

In the next two sections, we analyse how to construct feasible subsequences of a given sequence $\overline{S} = (\underline{M}, U = (\overline{u}_1, \overline{u}_2, \ldots, \overline{u}_k))$. In section 7, we define a dependency relation between updates, that will be modelled as a partial order. In section 8, we consider any collection $U' = (\overline{u}_{i_1}, \overline{u}_{i_2}, \ldots, \overline{u}_{i_h})$, formed by updates of U, which is consistent with such partial order, and we define an operator, $\oplus_{\overline{S}}$, such that $\overline{S}' = (\underline{M}, U' = (\overline{u}_{i_1}, \overline{u}_{i_2}, \ldots, \overline{u}_{i_h}), \oplus_{\overline{S}})$ is a feasible subsequence of $\overline{S}$.

7 Partial order

Consider sequence $\overline{S} = (\underline{M}, U = (\overline{u}_1, \overline{u}_2, \ldots, \overline{u}_k))$. For each update $\overline{u}$ in U, it will be said to be *directly dependent* on all and only those updates that introduce some entities. We will see that the transitive closure of the direct dependency relation is a *strict partial order*.

The definition of direct dependency of update:

$$\overline{u} : \{\gamma_d^{(l)}\} \xrightarrow{I} \{(\gamma_a^{(i)}, \gamma_b^{(j)}, \gamma_c^{(k)}), \prec\}$$

is based on the analysis of what entities must have been created before applying $\overline{u}$, and relies on the following key factors:

- To apply update $\overline{u}$, instead of demanding that $\gamma_d^{(l)}$ belongs to the map, it will be only required that its most abstract representative has been previously created.
- If $\overline{u}$ is an update of types either *point-to-line* and *point-to-region*, exactly one of the entities in $\{\gamma_a^{(i)}, \gamma_b^{(j)}, \gamma_c^{(k)}\}$ is a line. Such line was incident in $\overline{S}$ at either one or two already existing regions. We will require that the most

abstract representative of such regions have been created before $\overline{u}$ is applied. According to the arrangement of entities in form of forests described in Section 5, the most abstract representatives of the regions must have been created in an update of type either *point-to-region* or *line-to-region*. This situation corresponds to the example shown in Fig. 8, where in (b) we show a subsequence of sequence in (a) which is not feasible, because when update $\overline{u}_2$ [$\overline{u}_3$], of type *point-to-line* [*point-to-region*] is applied, line l_3 [loop l_4] becomes incident at region r_2. In subsequence shown in (b), we observe that when feature-line l_3 [loop l_4] was created, region r_2 did not exist. Thus, although r_2 is created in a posterior update, l_3 [l_4] will be outside it, and the combinatorics of map $\overline{M}$ cannot be retrieved, as there is no atomic refinement update which can make a line that is outside a region *jump* inside it.

- Let $\overline{u}$ be an update of type either *feature-line-addition* or *region-split*. In this case, also exactly one of the entities in $\{\gamma_a^{(i)}, \gamma_b^{(j)}, \gamma_c^{(k)}\}$ is a line, which was incident in $\overline{S}$ at two existing points (possibly coincident). We will require that the most abstract representative of such points have been created before $\overline{u}$ is applied. Such representatives must necessarily have been created in an update of type either *line-split* or *feature-point-addition*.

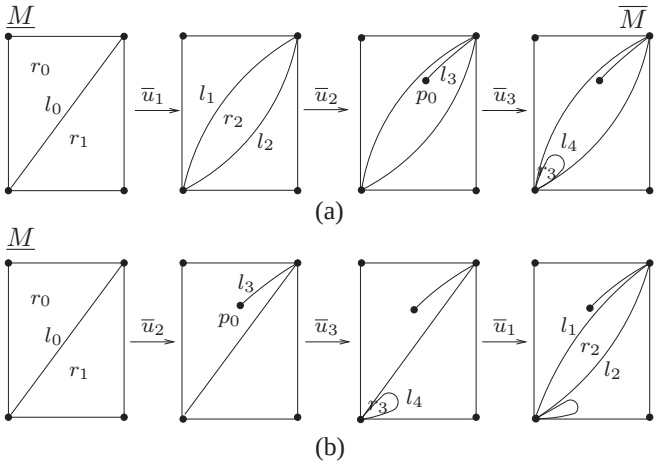


Fig. 8. (a) Sequence $\overline{S}$ of atomic refinement updates. (b) Subsequence of $\overline{S}$ which is not feasible.

- In updates of types *feature-line-addition* and *region-split*, besides requiring that the most abstract representatives of the endpoints of l exist, it must also be required that they are disconnected or connected respectively, oth-

erwise the corresponding update cannot be applied. Assume $\overline{u}$ is an update of type *region-split*, and let p_1 and p_2 be the endpoints of the splitting line when this update was applied in the original sequence. Points p_1 and p_2 had to be connected before $\overline{u}$ was applied. The only type of atomic refinement update which can connect two points that are disconnected is *feature-line-addition*. Thus, provided the most abstract representatives of p_1 and p_2 were not connected when they were created, some feature-lines must have been created in updates previous to $\overline{u}$ in $\overline{S}$ to connect them. We will require to apply $\overline{u}$ that such feature-lines have been created (see Fig. 9).

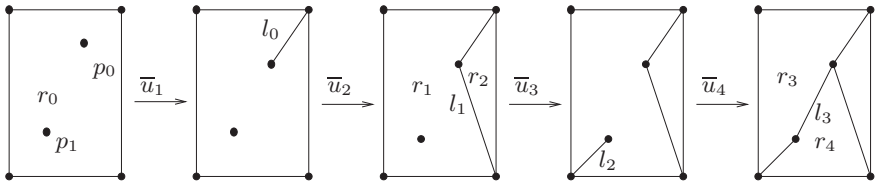


Fig. 9. If any of the feature-lines l_0, l_2 has not been created before $\overline{u}_4$ is applied, the line introduced by $\overline{u}_4$ will not split any region in two. Thus, update $\overline{u}_4$ is directly dependent on the updates in which l_0 and l_2 where created, $\overline{u}_1$ and $\overline{u}_2$ respectively.

According to this criteria, we give the formal definition of the direct dependency relation.

Definition 5. *Given sequence* $\overline{S} = (\underline{M}, U = (\overline{u}_1, \overline{u}_2, \ldots, \overline{u}_k))$*, we say that update* $\overline{u}_j : \Gamma_j^- = \{\gamma_j\} \rightarrow \Gamma_j^+$ *is* directly dependent *on update* $\overline{u}_i : \Gamma_i^- \rightarrow \Gamma_i^+$ *if some of the following conditions hold:*

1. *The most abstract representative of* $\gamma_j \in \Gamma_i^+$.
2. $\overline{u}_j$ *is an update of type either* point-to-line *or* point-to-region*, and* $\overline{u}_i$ *is an update of type either* point-to-region *or* line-to-region*, such that the line created in* $\overline{u}_j$ *is incident in* $\overline{S}$ *at the region created in* $\overline{u}_i$.
3. $\overline{u}_j$ *is an update of type either* feature-line-addition *or* region-split*, and* $\overline{u}_i$ *is an update of type either* line-split *or* feature-point-addition*, such that the line created in* $\overline{u}_j$ *is incident in* $\overline{S}$ *at some point created in* $\overline{u}_i$.
4. $\overline{u}_j$ *is an update of type* region-split*, and* $\overline{u}_i$ *is an update of type* feature-line-addition*, such that the feature-line created in* $\overline{u}_i$ *connects the endpoints of the line created in* $\overline{u}_j$.

Note that, as there is no atomic refinement update which can make two points that are connected to become disconnected, and *feature-line-addition* is the only atomic refinement update which can make two points that are

disconnected to become connected, the set of feature lines which in $\overline{S}$ connect the representatives of the endpoints of l is univocally defined.

A collection $U' = (\overline{u}_{i_1}, \overline{u}_{i_2}, \ldots, \overline{u}_{i_h})$, formed by updates of U, is said to be *closed with respect to the relation of direct dependency*, if for each update in U', all the updates it directly depends on are previous to it in U'.

It is clear that, if sequence $\overline{S} = (\underline{M}, U = (\overline{u}_1, \overline{u}_2, \ldots, \overline{u}_k))$ is non-redundant, and update $\overline{u}_j$ in $\overline{S}$ is directly dependent on $\overline{u}_i$, update $\overline{u}_i$ is previous to $\overline{u}_j$ in U. From this fact, the next proposition follows.

Proposition 1. *Let $\overline{S} = (\underline{M}, U = (\overline{u}_1, \overline{u}_2, \ldots, \overline{u}_k))$ be a non-redundant sequence of atomic refinement updates. The transitive closure of the direct dependency relation is a* strict partial order*, $\prec$.*

Given sequence $\overline{S} = (\underline{M}, U = (\overline{u}_1, \overline{u}_2, \ldots, \overline{u}_k))$, from now on we will only consider collections $U' = (\overline{u}_{i_1}, \overline{u}_{i_2}, \ldots, \overline{u}_{i_h})$, obtained from U, consistent with $\prec$, i.e., closed with respect to the relation of direct dependency.

In the next section, we define an operator, $\oplus_{\overline{S}}$, which specifies how to apply U' on $\underline{M}$, so that subsequence $\overline{S}' = (\underline{M}, U' = (\overline{u}_{i_1}, \overline{u}_{i_2}, \ldots, \overline{u}_{i_h}), \oplus_{\overline{S}})$ is feasible.

8 Constructing a feasible subsequence

In this Section, we develop algorithms to build a feasible subsequence from any collection of updates which is closed with respect to the relation of direct dependency. For the sake of clarity, the proofs of this section are given in a separate Appendix.

Let $\overline{S} = (\underline{M}, U = (\overline{u}_1, \overline{u}_2, \ldots, \overline{u}_k))$ be a sequence of atomic refinement updates, and $U' = (\overline{u}_{i_1}, \overline{u}_{i_2}, \ldots, \overline{u}_{i_h})$ be any collection of updates closed with respect to the relation of direct dependency. Each update in U':

$$\overline{u} : \{\gamma_d\} \xrightarrow{I} \{(\gamma_a, \gamma_b, \gamma_c), \prec\},$$

where for the sake of clarity we have omitted dimension superscripts, must be applied on the map M obtained after applying on $\underline{M}$ all updates preceding $\overline{u}$ in U'. As map M is not necessarily the same map on which $\overline{u}$ was applied in $\overline{S}$, M might not contain γ_d and/or the rest of entities that were incident at γ_d when $\overline{u}$ was applied in $\overline{S}$. Thus, to apply $\overline{u}$ on M operator $\oplus_{\overline{S}}$ has to specify:

(a) which entity γ'_d of M will play the role of γ_d, i.e., must be removed from M when $\overline{u}$ is applied. Entity γ'_d will be said to *represent* γ_d in M,

(b) the entities which will replace γ'_d in M,

(c) how the entities replacing γ'_d will be combinatorially related with the rest of entities in M.

We will refer to (a) and (b) as the *representation process*.

Given an entity γ appearing in $\overline{S}$, consider all the entities it is incident at in the maps associated to sequence $\overline{S}$. Although all such entities were known, in a feasible subsequence entity γ might appear in a map not containing any such entities. For example, in sequence $\overline{S}$ of Figure 6 (a) we observe that the points at which line l_2 is incident are p_1 and p_2. Nevertheless, in feasible subsequence of $\overline{S}$ shown in Figure 7 (b), when update $\overline{u}_4$ is applied, point p_2 is removed from the map and replaced by two other points, and we should know at which of them line l_2 must become incident. To define rules specifying how to modify combinatorics when an update is applied, we define the *univocal incidence* concept, which will be also used in the representation process.

Definition 6. *Given entities a and b, let*

- $\mathcal{D}(a)$ *be the set of entities incident at a or some descendant of a.*
- $\mathcal{A}(b)$ *be the set of entities incident at b or some ancestor of b.*

We say that a is univocally incident *at b if the following three conditions hold:*

1. $a \in \mathcal{A}(b)$,
2. $CHILDREN(b) \subset \mathcal{D}(a)$,
3. *there is no ancestor of b satisfying 1. and 2.*

We observe that if b is a leaf, $CHILDREN(b) = \emptyset \subset \mathcal{D}(a)$.

Considering again the example of Figure 7 (a), we notice that line l_2 is univocally incident at points p_1 and p_4, and when update $\overline{u}_4$ is applied in feasible subsequence of Figure 7 (b), line l_2 has as endpoints are p_1 and p_4. We will see later on that the univocal incidence concept is essential to update combinatorics in a feasible subsequence.

We can consider the straightforward extension of the definition of univocal incidence when cell γ represents an inner boundary or feature.

From each tree τ in the forests of updates, another tree, called *representation tree* and denoted $\mathcal{R}_\tau$, is created. It encodes the incidence relations of all the entities created in the updates associated to τ, and allows keeping track of which nodes of τ are represented by the same entity in M. Whenever a node of τ is split up into its two children, an entity incident at both children is also created. Such entities do not belong to τ. However, they will play a fundamental role in the representation process, as they connect the nodes of τ. We provide next the definition of the representation tree $\mathcal{R}_\tau$ corresponding to τ.

Definition 7. *Let $\overline{S} = (\underline{M}, U = (\overline{u}_1, \ldots, \overline{u}_n), \oplus_{\overline{S}})$ be a sequence of atomic refinement updates closed with respect to the relation of direct dependency. For each tree τ in the forests of updates of $\overline{S}$, its* representation tree $\mathcal{R}_\tau$ *is created as follows:*

Nodes of $\mathcal{R}_\tau$.- The set of nodes will be the set of entities created in the updates associated to τ but which are not nodes of τ (denoted ν), plus the set of nodes of τ at which each such entities is univocally incident (denoted c).

Arcs of $\mathcal{R}_\tau$.- There is an arc between two nodes ν and c if either ν is univocally incident at c, or c is an internal node of τ and ν is the node that represents the entity created when c is split up into its two children.

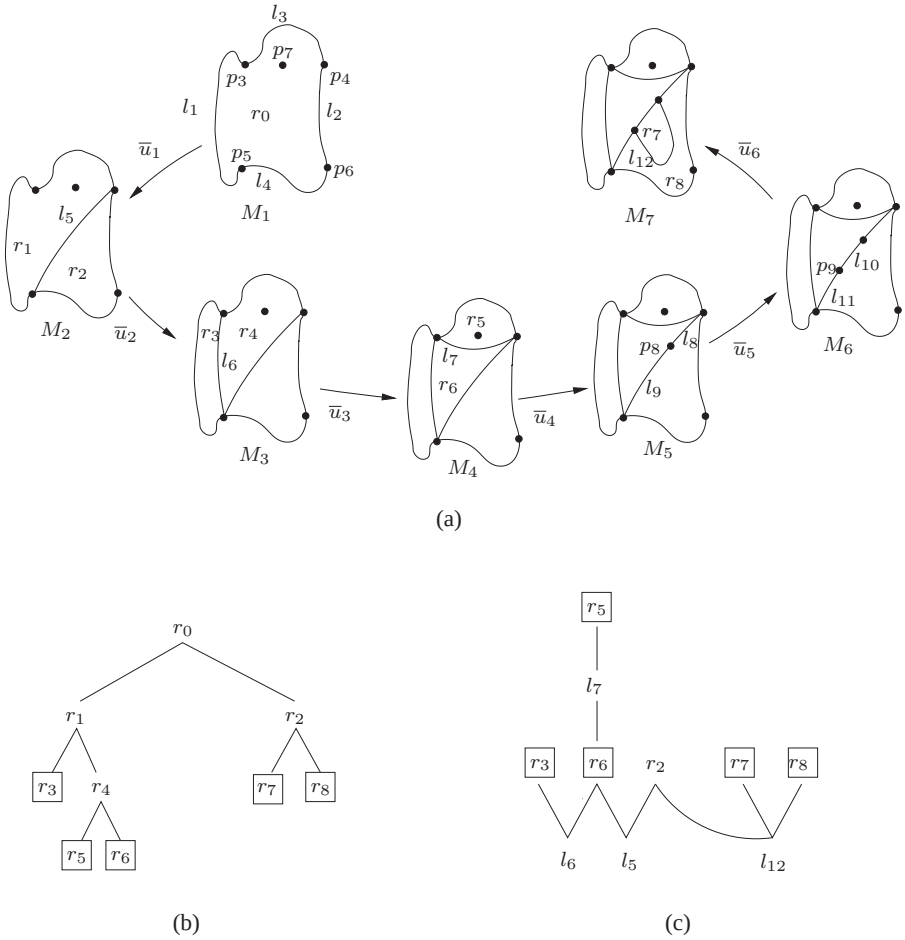


Fig. 10. (a) Sequence $\overline{S}$ of atomic refinement updates. (b) Region-split tree τ corresponding to the sequence in (a). (c) Representation tree $\mathcal{R}_\tau$.

In Fig. 10 (a), a sequence of atomic refinement updates is shown. The representation tree corresponding to the tree of *region-split* updates of (b) is

shown in (c). The next Lemma states that the set of nodes and arcs of $\mathcal{R}_\tau$ has a tree structure.

Lemma 1. *$\mathcal{R}_\tau$ is a tree.*

We give next the definition of operator $\oplus_{\overline{S}}$ as a constructive algorithm that builds a feasible subsequence from any collection of updates closed with respect to the relation of direct dependency.

Algorithm Building_Feasible_Subsequence

Input.- Sequence $\overline{S} = (\underline{M}, U = (\overline{u}_1, \overline{u}_2, \ldots, \overline{u}_k))$, and collection of updates $U' = (\overline{u}_{i_1}, \overline{u}_{i_2}, \ldots, \overline{u}_{i_h})$, closed with respect to the relation of direct dependency.
Output.- $M = \underline{M} \oplus_{\overline{S}} \overline{u}_{i_1} \oplus_{\overline{S}} \overline{u}_{i_2} \oplus_{\overline{S}} \ldots \oplus_{\overline{S}} \overline{u}_{i_h}$.

1. **for** $\overline{u}_{i_j} : \{\gamma_d\} \xrightarrow{I} \{(\gamma_a, \gamma_b, \gamma_c), \prec\}$, with $1 \leq j \leq h$, **do**
2. **if** γ_d does not belong to any tree in the forest of updates
3. **then** $\gamma'_d = \gamma_d$
4. **else** $\gamma'_d =$ root of tree τ to which γ_d belongs.
5. $M = \underline{M}$
6. **for** $\overline{u}_{i_j} : \{\gamma_d\} \xrightarrow{I} \{(\gamma_a, \gamma_b, \gamma_c), \prec\}$, with $1 \leq j \leq h$, **do**
7. remove γ'_d from M and add entities of the same types as $\gamma_a, \gamma_b, \gamma_c$
8. **Representation_Process**$(\overline{u}_{i_j}, M)$
9. **Update_Combinatorics**$(\overline{u}_{i_j}, M)$

Procedure **Representation_Process** is given next.

Representation_Process

Input.- Map M and update $\overline{u} : \{\gamma_d\} \xrightarrow{I} \{(\gamma_a, \gamma_b, \gamma_c), \prec\}$, where entity γ_c is not of the same type as γ_d.
Output.- Entities of $\overline{S}$ represented by the entities added to the map, which are of the same types as $\gamma_a, \gamma_b, \gamma_c$.

1. **if** γ_d does not belong to any tree in the forest of updates
2. **then** α represents only α, $\forall\ \alpha \in \{\gamma_a, \gamma_b, \gamma_c\}$
3. **else** γ_c and its incident arcs are deleted from $\mathcal{R}_\tau$
4. **if** an incident arc connects γ_c with an internal node of τ
5. **then** also such node, and all its incident arcs, are removed from $\mathcal{R}_\tau$.
6. All leaves of τ contained in $\mathcal{R}_\tau^{(i)}$, with $1 \leq i \leq 2$, are represented by the entity added to the map of the same type as γ_d.

7. Each entity in $\mathcal{R}_\tau^{(i)}$, with $1 \leq i \leq 2$, which is not a node of τ the entity of τ in whose splitting such entity was created, is represented by the same entity as $\mathcal{R}_\tau^{(i)}$.

In procedure **Update_Combinatorics**, the univocal incidence concept plays a fundamental role. We must make a remark on how many points can be univocally incident at a line. In case l is a loop all over $\overline{S}$, the two points it is univocally incident at will be coincident. But if at some stage of $\overline{S}$ line l stops being a loop (i.e., the endpoint of the loop is the input of an update of type *point-to-line*), we will assume we have a different line, and we will find the two points it is univocally incident at. We will consider l is univocally incident at exactly those two points.

Update_Combinatorics

Input.- Map M and update $\overline{u} : \{\gamma_d\} \rightarrow \{\gamma_a, \gamma_b, \gamma_c\}$.
Output.- Entities of M at which entities $\gamma_a, \gamma_b, \gamma_c$ are incident.

1. **if** the endpoints of line l are required
2. they will be the points l is univocally incident at.
3. **if** the region in M containing an inner component is required
4. it will be the region f is univocally incident at.

The correctness of **Algorithm Build_Subsequence** is proved in the next proposition.

Proposition 2. *Let $\overline{S} = (\underline{M}, U = (\overline{u}_1, \overline{u}_2, \ldots, \overline{u}_k))$ be a sequence of atomic refinement updates, and $U' = (\overline{u}_{i_1}, \overline{u}_{i_2}, \ldots, \overline{u}_{i_h})$ be a collection closed with respect to the relation of direct dependency. Then, the output $M = \underline{M} \oplus_{\overline{S}} \overline{u}_{i_1} \oplus_{\overline{S}} \overline{u}_{i_2} \oplus_{\overline{S}} \ldots \oplus_{\overline{S}} \overline{u}_{i_h}$ of Algorithm BuildSubsequence is a map.*

Once the definition of $\oplus_{\overline{S}}$ is complete, we are in conditions to prove the result we have been searching throughout this section.

Theorem 1. *Let $\overline{S} = (\underline{M}, U = (\overline{u}_1, \overline{u}_2, \ldots, \overline{u}_k))$ be a sequence of atomic refinement updates, and let $U' = (\overline{u}_{i_1}, \overline{u}_{i_2}, \ldots, \overline{u}_{i_h})$ be a collection of updates closed with respect to the relation of direct dependency. Then, $\overline{S}' = (\underline{M}, U' = (\overline{u}_{i_1}, \overline{u}_{i_2}, \ldots, \overline{u}_{i_h}), \oplus_{\overline{S}})$ is a feasible subsequence of $\overline{S}$.*

9 The Multiresolution Model

In this Section, we define our multi-scale model for maps, called *Multi-VMap*, which is inspired by the Multi-Triangulation developed for triangle meshes [11, 34].

We define a Multi-VMap as the pair $(\overline{S}, \prec)$, where:

- $\overline{S} = (\underline{M}, (\overline{u}_1, \overline{u}_2, \ldots, \overline{u}_k))$ is a sequence of atomic refinement updates, which is the inverse of a sequence $\underline{S} = (\overline{M}, (\underline{u}_1, \overline{u}_2, \ldots, \underline{u}_k))$ of abstraction updates that have been performed during a generalization process.
- $\prec$ is the partial order on $\{\overline{u}_1, \overline{u}_2, \ldots, \overline{u}_k\}$ defined by direct dependency in Section 7.

Any partially ordered set can be represented as a Directed Acyclic Graph (DAG). In the case of the Multi-VMap, it will have an initial update $\overline{u}_0 = (\emptyset, \underline{M})$ as root, updates as its nodes and direct dependency links as its arcs (see Fig. 11).

A *subMulti-VMap* is the restriction of a Multi-VMap to a set of updates closed with respect to the relation of direct dependency. We have proved in Proposition 2 that the application of all the updates in a subMulti-VMap to the base map $\underline{M}$ produces a map. The map obtained after the application of all the updates in a subMulti-VMap is called an *extracted map*.

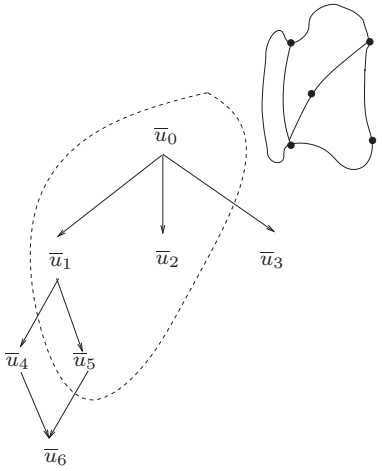


Fig. 11. DAG representing the Multi-VMap corresponding to the sequence of Fig. 10, and extracted map obtained from the subMulti-VMap surrounded by a dotted line.

9.1 Extracting a map at resolution variable in space

We are now interested in obtaining a map whose scale is variable in space, according to arbitrary user requirements. We assume that user requirements are given by means of an external Boolean function $b()$, defined over the updates of a Multi-VMap, which decides whether an update is *necessary* or not in order to achieve (locally) the level of detail needed by the user/application.

For instance, if all updates acting in a certain area of interest are necessary, then refinement will produce the maximum level of detail inside that area, while leaving the rest of the map at a lower detail.

The *selective refinement* will produce the smallest map, extracted from the Multi-VMap, in which all updates necessary according to $b()$ have been performed. Such map is generated from the minimal set of updates which contains all updates necessary according to $b()$, and is closed with respect to the relation of direct dependency. To find such set, the DAG representing the Multi-VMap will be traversed, starting from its root. A subset N of nodes closed with respect to the relation of direct dependency must be maintained, and nodes will be successively added to N until every entity composing the corresponding extracted map, M_N, attains the required resolution.

The data structure encoding the Multi-VMap, which is the subject of future work, must support the following primitives, referring to a node V involving update $\overline{u}$ in the DAG representing the Multi-VMap:

children(V) : returns those nodes children of node V in the DAG
refine_test (V, N): given node V not in N, returns true if all its parents belong to N;
dependencies_retrieval (V, N): given node V not in N, returns those nodes that are parents of node V and are not in N;
map_refinement (M_N, V): refines map M_N by applying $\overline{u}$, i.e., produces map $M_N \oplus_{\overline{S}} \overline{u}$;

Let us assume that primitives **children, refine_test** and **dependencies_retrieval** can be performed in time linear in its output size. Let us evaluate the worst case complexity of primitive **map_refinement**.

Let $\overline{S} = (\underline{M}, U = (\overline{u}_1, \overline{u}_2, \ldots, \overline{u}_k))$ be a sequence of atomic refinement updates, and let $\overline{S}' = (\underline{M}, U' = (\overline{u}_{i_1}, \overline{u}_{i_2}, \ldots, \overline{u}_{i_h}), \oplus_{\overline{S}})$ be a subsequence of $\overline{S}$ such that U' is closed with respect to the relation of direct dependency. Let us now study the time complexity of applying update $\overline{u}_{i_j} : \{c\} \rightarrow \{a_1, a_2, a_3\}$ of U' on map $M = \underline{M} \oplus_{\overline{S}} \overline{u}_{i_1} \oplus_{\overline{S}} \ldots \oplus_{\overline{S}} \overline{u}_{i_{(j-1)}}$. The entity representing c in M is found in constant time, provided c has a reference to the entity representing it. To find which entities of $\overline{S}$ represents each entity created in $\overline{u}_{i_j}$, if $\overline{u}_{i_j}$ is of type *point-to-region, feature-point-addition*, or *feature-line-addition*, it takes constant time. Otherwise, a representation tree must be cut into two, and in the worst case this requires $O(k)$ time. Finally, the update of combinatorics takes $O(m)$ time, where m is the number of combinatorial relations which must be updated. Hence, the next result follows.

Proposition 3. *Given sequence* $\overline{S} = (\underline{M}, U = (\overline{u}_1, \overline{u}_2, \ldots, \overline{u}_k))$*, and feasible subsequence* $\overline{S}' = (\underline{M}, U' = (\overline{u}_{i_1}, \overline{u}_{i_2}, \ldots, \overline{u}_{i_h}), \oplus_{\overline{S}})$ *of* $\overline{S}$*, the application of update* $\overline{u}_{i_j}$*,* $1 \leq j \leq h$*, on map* $\underline{M} \oplus_{\overline{S}} \overline{u}_{i_1} \oplus_{\overline{S}} \ldots \oplus_{\overline{S}} \overline{u}_{i_{(j-1)}}$ *requires* $O(k + m)$ *time, where* m *is the number of combinatorial relations which must be updated due to the application of* $\overline{u}_{i_j}$*.*

Let us take into account that, assuming a node V must be added to N, provided any node W which is a parent of V is not in N, W must be added to N before the inclusion of V. This is done in the following procedure.

Add_Node (V, N)

1. **if** (**refine_test** (V, N)=false) **do**
2. **for all** node $W \in$ **dependencies_retrieval** (V, N) **do**
3. ADD_NODE (W, N)
4. $S = S \cup \{V\}$
5. **map_refinement**(M_N, V)

Algorithm **Selective_Refinement** performs a DAG traversal to find the required subgraph of the DAG. We adopt the following notation:

- $b(N)$, with N a set of nodes, is used for convenience with the meaning that $b(N)$ is true iff $b(V)$ is true for all node V belonging to N.
- **children** (N) denotes all nodes in N which are children of some node in S.

Algorithm Selective_Refinement

Input. DAG of k nodes representing a Multi-VMap, and boolean function $b()$
Output. Extracted map $M_N = \underline{M} \oplus \overline{u}_{i_1} \oplus \ldots \oplus \overline{u}_{i_j}$, of the lowest possible size, such that $b(M_N)$ is not true for any update which is not in $\{\overline{u}_{i_1}, \ldots, \overline{u}_{i_j}\}$ is true

1. node V = root of the DAG, $N = \{V\}$, $M_N = \underline{M}$
2. $C =$ **children**(S)
3. **while** ($b(C)$ is false) **do**
4. **for all** nodes $W \in C$ such that $b(W) = false$ **do**
5. ADD_NODE (W, N)
6. $C =$ **children** (S)

In the worst case, condition of line 3 in **Algorithm Selective_Refinement** must be checked k times. We are assuming that primitives **children, refine_test** and **dependencies_retrieval** can be performed in time linear in its output size. As **map_refinement** can be performed in $O(k+m)$ time, assuming r refinement updates must be performed to have the required level of detail, **Algorithm Selective_Refinement** can be performed in $O(r(k+m))$ time.

Proposition 4. *Given a Multi-VMap $\mathcal{M}$ and a Boolean condition $b()$, the map of the smallest size M_N such that $b(M)$ is true can be extracted in $O(r(k+m))$ time, where k is the number of nodes in $\mathcal{M}$, m is the maximum number of combinatorial changes required to performed any update, and r is the number of updates which must be applied to have the required resolution.*

The following proposition provides a lower bound for the expressive power of the Multi-VMap, i.e., the number of different maps which can be extracted from it.

Proposition 5. *Given map $\overline{M}$ with n entities, from which a Multi-VMap with height h has been built, at least $2^{\lceil \frac{n}{2h} \rceil}$ different maps can be extracted from the Multi-VMap*

Proof

$(\overline{S}, \prec)$ is a finite partially ordered set. A chain in $\overline{S}$ is a set of pairwise comparable elements (i.e., a totally ordered subset). And an antichain in $\overline{S}$ is a set of pairwise incomparable elements. By Dilworth's Lemma ([13]), calling k to the number of elements in $\overline{S}$ and h to the cardinality of the longest chain, a partition of the k elements into h antichains must exist. Hence, an antichain must exist with at least $\lceil \frac{k}{h} \rceil$ elements. In our case, as the DAG has height h, and each atomic abstraction update reduces the cardinality of a map for at most two elements, the number of nodes must be greater than $\frac{n}{2}$. Thus, an antichain must exist with at least $\lceil \frac{n}{2h} \rceil$ elements. □

Although the process of obtaining a sequence of simplified versions of a reference map $\overline{M}$ greatly relies on the cartographic practice of generalization, we will show in the next proposition that, disregarding the semantics in the generalization process, independent atomic abstraction updates can be performed allowing to build a trivial map (formed by one only region) from $\overline{M}$ in two steps, i.e., a DAG of height 2 can be built.

Proposition 6. *Given map $\overline{M}$, the set of atomic abstraction updates allows to build a Multi-VMap whose base map has just one simple region with height of the DAG less than or equal to* 2

Proof

We consider a partition of the entities in M into feature entities (feature-points, feature-lines, and points joining two adjacent feature-lines) and non-feature entities (simply called points, lines and regions).

We know that updates of type feature-point-removal can be applied independently on every isolated point, and updates of type feature-line-removal can also be applied independently to remove every feature line. Hence, in the first level of the DAG, there is no feature-point or feature-line, and those points which were incident at several feature-lines and which have become isolated, can be removed independently in the next step, thus in the next level of the DAG there will not be any feature-entity.

Let us now see how many non-feature entities (points, lines, and regions) can be removed by independent updates. All the regions can be reduced to one only region by the independent application of updates of type region-merge. Update region-merge is independent of update line-to-point provided the line to be contracted is not a line of merging two regions. Thus, the application

of independent updates of these two types allows the transformation of all non-feature entities into one only region bounded by exactly one point and one line.

Thus, after two steps, each of which is formed by independent sets of updates, map $\overline{M}$ has become an empty loop. □

10 Concluding Remarks

The main contribution of this paper is the introduction of a multiresolution model for vector maps in which all extracted maps are combinatorially consistent. The number of extractable maps is so high, that we can assume to have a virtual continuum of scales between the reference map at high detail and the base map at low detail, where detail can be variable through the domain of the map.

The model is fully combinatorial. Once topological consistency is guaranteed, our next efforts will be devoted to assign geometry to entities, in a way such that there is no any undesired intersection all over the model. In a final step, semantics shall be considered.

The operators which are used in the multiresolution model to perform the generalization of the map were not developed on a cartographic basis. They were created so that spatial analysis on the resulting model could be performed. On the basis of such operators, which ensure consistency of the model, macro-operators must be found to translate cartographic generalization operations into composition of atomic operators. The main applications of the model are support to map generalisation and automated cartography, efficient browsing over large GIS, structured solutions in wayfinding and planning, etc.

Data structures which encode the extracted map and the whole multi-scale model, and that support the primitives indicated in 9.1, are beyond the scope of this paper and are the subject of current work. A clear formalization and a sound theory are essential for an efficient implementation of models.

References

1. Alexandroff, P.: *Elementary Concepts of Topology.* Dover Publications, New York (1961).
2. Barr, M., Wells, C.: *Category Theory for Computer Science.* Prentice-Hall (1995).
3. M. Bertolotto. Geometric modeling of spatial entities at multiple levels of resolution. Ph.D.Thesis, Department of Computer Science, University of Genova, DISI-TH-1998-01, 1998.

4. Bertolotto, M., De Floriani, L., Puppo, E.: *Multiresolution topological maps.* Advanced Geographic Data Modelling - Spatial Data Modelling and Query Languages for 2D and 3D Applications, M. Molenaar, S. De Hoop (eds.), Publications on Geodesy - New Series, N. 40, Netherland Geodetic Commission, pp. 179-190 (1994).
5. M. Bertolotto and M.J. Egenhofer. "Progressive transmission of vector map data over the World Wide Web", GeoInformatica, Vol. 5(4), 345-373, 2001.
6. B. Buttenfield. "Progressive transmission of vector data on the Internet: A cartographic solution", in Proc. 19th Int. Cartographic Conf. Ottawa, Canada, 581-590, 1999.
7. Buttenfield, B.: *Transmitting Vector Geospatial Data across the Internet.* Second International Conference, GIScience, Lecture Notes in Computer Science, Springer, Berlin, pp.51-64 (2002).
8. Corbett, J.P.: *Topological principles in cartography.* Technical Report 48, U.S. Bureau of Census, USA (1979).
9. De Berg, M. van Kreveld, M, Schirra, S.: *Topologically Correct Subdivision Simplification Using the Bandwidth Criterion.* Cartography and Geographic Information Systems, 25 (4), pp. 243-257 (1998).
10. L. De Floriani, P. Marzano and E. Puppo. "Spatial queries and data models". Spatial Information Theory - A theoretical basis for GIS, A.U. Frank, I. Campari (Eds.), LNCS Vol.716, Springer-Verlag, 113-138, 1993.
11. L. De Floriani, P. Magillo. "Multiresolution mesh representation: Models and data structures", Tutorials on Multiresolution in Geometric Modelling, M.Floater, A.Iske, E.Qwak (Eds.), Springer-Verlag, 363-418, 2002.
12. G. Dettori and E. Puppo. "How generalization interacts with the topological and metric structure of maps", in Proc. 7th Int. Symp. on Spatial Data Handling, Delft, 9A.27-9A.38, 1996.
13. R.P. Dilworth. "A decomposition theorem for partially ordered sets", Ann. Math., Vol. 51, 161-166, 1950.
14. D.H. Douglas and T.K. Peucker. "Algorithms for the reduction of the number of points required to represent a digitized line or its caricature", The Canadian Cartographer, 10, 2, 112-122, 1973.
15. M. Egenhofer, A. Frank, J. Jackson. "A topological data model for spatial databases". Proceedings of the Symposium on the Design and Implementation of Large Spatial Databases, Santa Barbara, CA, Lecture Notes in Computer Science, Vol. 409, 271-286, 1989.
16. Egenhofer, M.J., Clementini, E., di Felice, P.: *Evaluating inconsistencies among multiple representations.* Proceedings 6th International Symposium on Spatial Data Handling, Edinburgh, Scotland, 901-920 (1994).
17. Finke, U., Hinrichs, K.H.: *The quad view data structure - A representation for planar subdivisions.* Advances in Spatial Databases, M.J. Egenhofer, J.R. Herring (eds.), Lecture Notes in Computer Science, Vol.951, pp.29-46 (1995).
18. A.U. Frank, W. Kuhn. "Cell graph: a provable correct method to the storage of geometry". Proceedings 2nd Symposium on Spatial Data Handling (SDH'86), Seattle, WA, pp.411-436 (1986).
19. Frank, A., Timpf, S.: *Multiple representations for cartographic objects in a multiscale tree - An intelligent graphical zoom.* Computers and Graphics, 18, 6, pp. 823-829 (1994).

20. C. Gotsman, S. Gumhold and L. Kobbelt. "Simplification and Compression of 3D Meshes", in Proc. European Summer School on Principles of Multiresolution in Geometric Modelling (PRIMUS), Munich, 2001.
21. Güting, R.H., Schneider, M.: *Realms: a foundation for spatial data types in database systems*, Proceedings 3rd International Symposium on Large Spatial Databases, Singapore, pp. 14-35 (1993).
22. Güting, R.H., Ridder, T., Schneider, M.: *Implementation of the ROSE algebra: efficient algorithms for realm-based spatial data types*, Proceedings 4th International Symposium on Large Spatial Databases, Portland, ME, Lecture Notes in Computer Science, 951, Springer-Verlag, pp. 216-239 (1995).
23. J. Herring. "The Mathematical Modeling of Spatial and Non-Spatial Information in Geographic Information Systems", in Cognitive and Linguistic Aspects of Geographic Space, D.Mark and A.Frank (Eds.), Kluwer Academic Publishers, pp.313-350, 1991.
24. C.B. Jones. Geographical Information Systems and Computer Cartography, Ed. Longman, 1997.
25. Jones, C.B., Abdelmoty, A.I., Lonergan, M.E., van der Poorten, P., Zhou, S.: *Multi-Scale Spatial Database Design for Online Generalisation.* Proceedings of the 9th International Symposium on Spatial Data Handling, pp. 7b-34 7b-44, Beijing (2000).
26. V.A. Kovalevsky. "Finite topology as applied to image analysis", Computer Vision, Graphics, and Image Processing, 46, 141-161, 1989.
27. Lawson, C.L.: *Software for C^1 surface interpolation.* Mathematical Software III, J.R. Rice (ed.), Academic Press Inc., pp. 161-164 (1977).
28. D. Luebke, M. Reddy, J. Cohen, A. Varshney, B. Watson and R. Huebner, "Level of Detail for 3D Graphics", Morgan-Kaufmann, Inc., 2003.
29. A.T. Lundell and S. Weingram. The Topology of CW Complexes, Van Nostrand Reinhol Comp., 1969.
30. M. Mantyla. An Introduction to Solid Modeling, Computer Science Press, 1987.
31. Müller, J.C., Weibel, R., Lagrange, J.P., Salge, F.: *Generalization: state of the art and issues.* GIS and Generalization: Methodology and practice, J.C.Muller, J.P.Lagrange and R.Wibel eds., Taylor and Francis, pp. 3-17 (1995).
32. Pigot, S.: *Generalized singular 3-cell complexes.* Proceedings 6th International Symposium on Spatial Data Handling, Edinburgh, Scotland, pp.89-111 (1994).
33. E. Puppo and G. Dettori. "Towards a formal model for multiresolution spatial maps". Advances in Spatial Databases, M.J. Egenhofer, J.R. Herring (Eds.), LNCS Vol.951, Springer-Verlag, 152-169, 1995.
34. E. Puppo. "Variable resolution triangulations", Computational Geometry Theory and Applications, 11, 219-238, 1998.
35. Saalfield, A.: *Comflation: automated map compilation.* Technical Report CAR-TR-670, Center for Automation Research (1993).
36. Saalfield, A.: *Topologically Consistent Line Simplification with the Douglas-Peucker Algorithm.* Cartography and Geographic Information Science, 26 (1), pp. 7-18 (1999).
37. Samet, H.: *The design and analysis of spatial data structures.* Addison-Wesley, Reading, MA (1990).
38. Samet, H.: *Applications of spatial data structures.* Addison-Wesley, Reading, MA (1990).
39. Spanier, E.: *Algebraic Topology.* McGraw-Hill, New York (1966).

40. Stell, J.G.: *Granulation for Graphs*. Proceedings of the International Conference COSIT'99. Springer-Verlag Lecture Notes in Computer Science, V.1661, pp. 417-432 (1999).
41. Stell, J.G.: *The Representation of Discrete Multi-Resolution Spatial Knowledge*. Proceedings of the Seventh International Conference on Principles of Knowledge Representation and Reasoning (KR2000). Morgan Kaufmann, pp.38-49 (2000).
42. Stell, J.G., Worboys, M.: *Generalizing Graphs using Amalgamation and Selection*. Proceedings of the 6th International Symposium on Advances in Spatial Databases (SSD'99). Springer-Verlag Lecture Notes in Computer Science, V.1651, pp.19-32 (1999).
43. Timpf., S.: *Hierarchical Structures in Map Series*. Ph. D., Technical University Viena (1998).
44. S. Timpf and A. Franck. "A multi-scale DAG for cartographic objects", Proc. Auto Carto 12, Charlotte, North Caroline, USA, 157-163, 1995.
45. S. Timpf. Hierarchical Structures in Map Series, Ph.D. thesis, Technical University Vienna, 1998.
46. van Oosterom, P.: *Reactive Data Structures for Geographic Information Systems*. Oxford University Press (1993).
47. van der Poorten, P., Zhou, S., Jones, C.B.: *Topologically-Consistent Map Generalisation Procedures and Multi-scale Spatial Databases*. GISscience, M.J.Egenhofer and D.M.Marks (eds.), pp. 209-227, Springer-Verlag Berlin Heidelberg (2002).
48. van Oosterom, P., Schaukelaars, V.: *The development of an interactive multi-scale GIS*. International Journal of GIS, 9, 5, pp. 489-507 (1995).
49. van Putten, J., van Oosterom, P.: *New Results with Generalised Area Partitionings*. 8th International Symposium on Spatial Data Handling, Vancouver, International Geographical Union (1998).
50. White, M.: *Technical requirements and standards for a multipurpose geographic data system*. The American Cartographer, Vol.11, N.1, pp.15-26 (1984).
51. M. Worboys. A generic model for planar geographic objects, International Journal of Geographical Information Systems, Vol.6, N.5, pp.353-372, 1992.
52. M. Worboys. GIS: A Computer Perspective, Taylor and Francis, 1995.
53. Worboys, M.F., Bokaofs, P.: *A canonical model for a class of areal spatial objects*. Advances in Spatial Database (SSD93), D.Abel, B.C.Ooi (Eds.), Lecture Notes in Computer Science, Springer-Verlag, pp.36-52 (1993).

Appendix

Proofs of Section 8

Lemma 1. *$\mathcal{R}_\tau$ is a tree.*

Proof

If tree τ has $2n+1$ nodes, as it is a full binary tree it will have $n+1$ leaves. Besides, there will be n entities created in the updates associated to τ which are not nodes of τ. From Definition 6, it follows that every leaf of τ is a node of $\mathcal{R}_\tau$. Assuming $\mathcal{R}_\tau$ contains i internal nodes of τ, $\mathcal{R}_\tau$ will have $2n+1+i$ nodes and $2n+i$ arcs.

We will now prove by induction on the number n of updates in τ that $\mathcal{R}_\tau$ is connected. If $n=1$, clearly $\mathcal{R}_\tau$ is connected. Assuming that $\mathcal{R}_\tau$ corresponding to $n-1$ updates is connected, we must see that $\mathcal{R}_\tau$ for n updates is also connected. Let c be the node of $\mathcal{R}_\tau$ which is split by the nth update. The two children of c are both connected to the node representing the entity created in the nth update that is not any of them. Let ν be an entity that was connected by an arc to c before it was split. Either ν continues being univocally incident at c, and then there will be an arc between c and the node connected to its children, or ν is univocally incident at exactly one of the children of c. □

Proposition 2. *Let $\overline{S} = (\underline{M}, U = (\overline{u}_1, \overline{u}_2, \ldots, \overline{u}_k))$ be a sequence of atomic refinement updates, and $U' = (\overline{u}_{i_1}, \overline{u}_{i_2}, \ldots, \overline{u}_{i_h})$ be a collection closed with respect to the relation of direct dependency. Then, the output $M = \underline{M} \oplus \overline{u}_{i_1} \oplus \overline{u}_{i_2} \oplus \ldots \oplus \overline{u}_{i_h}$ of Algorithm BuildSubsequence is a map.*

Proof

1. For all $\overline{u}_{i_j} : \{\gamma_d\} \to \{(\gamma_a, \gamma_b, \gamma_c), \prec\}$, with $1 \leq j \leq h$, we will prove that there is one and only one entity c' in M representing γ_d, where M is the map obtained by applying on $\underline{M}$ all updates preceding $\overline{u}$ in U'.
   a) If γ_d does not belong to any forest of updates, its most abstract representative is itself. As U' is closed with respect to the relation of direct dependency, γ_d must belong to M.
   b) In case a tree τ of updates exists containing γ_d, let us write $\overline{u} : \{c\} \to \{(c_1, c_2, \nu), I\}$, according to the notation established for $\mathcal{R}_\tau$.
   We will first see that the removal of ν, and the rest of entities described in lines 3., 4., 5. of Procedure RepresentationProcess splits up the representation tree containing ν into two trees. Due to Definition 6, it can be easily shown that ν can be univocally incident at at most two nodes of τ. Thus, by construction of the representation tree, there can only be either two or three arcs incident at node ν. For ν having

degree three, one of the nodes i that is connected by an arc with ν must be an internal node of τ. For i belonging to $\mathcal{R}_\tau$, at least another entity univocally incident at i must exist. Let ν_1 be any such entities. We have that:

- If τ is a region-split tree, node i must be a region, and ν_1 must be a line, such that each descendant of ν_1 by an update of type *line-split* became incident in $\overline{S}$ at a different child of i.
- In case τ is a line-split or a line-to-region tree, a node in $\mathcal{R}_\tau$ which is an internal node of τ cannot exist.
- Finally, if τ is a tree of points, ν_1 must be a line, and i has to be a point, such that in $\overline{S}$, an update of type *line-to-region* was applied to ν_1, and each child of ν_1 became incident at a different child of i.

In all cases, as U' is closed with respect to the relation of direct dependency, ν_1 must have been created before the update replacing i by its children is performed. Thus, when $\overline{u}$ is to be applied, neither i, nor the arc connecting it to ν, can belong to $\mathcal{R}_\tau$. Hence, there can only be two arcs incident at ν. And when ν is removed from $\mathcal{R}_\tau$, it will become split up into two trees.

Let us now see by induction on the number of updates associated to τ that have been already applied, that for all update $\overline{u} : \{c\} \rightarrow \{(c_1, c_2, \nu), I\}$ associated to τ which still has not been performed, there is one and only one entity c' in M representing c.

i. The root C of τ represents all nodes of τ.
ii. Assume $n-1$ updates associated to τ have been performed, and there is exactly one node in the map representing c. All nodes except the one split up in the update will be clearly represented by one of the resulting trees. When an entity univocally incident at an internal node is inserted, although the internal node is removed from $\mathcal{R}_\tau$, the node ν corresponding to the entity splitting it will still belong to $\mathcal{R}_\tau$.

2. the modification of combinatorics required to apply each update of U' can be univocally performed on the corresponding map.

From the definition of atomic refinement updates given in Section 4, it follows that there are four types of updates, namely *point-to-line*, *point-to-region*, *region-split*, and *feature-line-addition*, which are not uniquely defined from their inverse atomic abstraction update, i.e., for which the specification of incidence information I is necessary. The combinatorial information that must be specified for these four types of updates belongs to the following classes:

a) If a line l is added to M from an update of type either *region-split,* or *feature-line-addition* then its endpoints (which must be connected if the update is of type *region-split* and disconnected in case it is of type *feature-line-addition*) must be specified in I;

b) The regions surrounding the new entities added to M from an update of type either *point-to-line*, or *point-to-region* must be specified in I;
c) If an existing line l in M has (at least) one endpoint p_0 that is affected from an update of type *point-to-line*, then it must be specified in I to which new point (p or p') line l will be incident;
d) If a feature or inner boundary f belongs to a region r_0 that is affected from an update of type *region-split*, and when the update is applied it remains being a feature or inner boundary respectively, then it must be specified in I to which new region (either r or r') f will belong.

The four classes of information only affect to two groups of entities. First, given line l which must be either added to M (i.e., that is created in an update of type either *region-split* or *feature-line-addition*), or that already exists in M, but whose endpoints must change (due to the application of an update of type *point-to-line*), the endpoints of l must be specified in I. Second, given a feature or inner boundary which must be either added to M (in case of having a feature or inner boundary created in an update of type either *point-to-line* or *point-to-region*), or that already exists in M, but which must be assigned to one of the newly created regions (when an update of type *region-split* is applied on M), the region which must contain the feature or inner boundary must be specified in I.

We will first see that for each line [inner boundary or feature] appearing in $\overline{S}$ whose endpoints [containing region] must be specified when an update $\overline{u}$ belonging to U' is applied, the points [containing region] specified by the rules to modify combinatorics, belong to the map M in which the update has to be applied.

a) Let l be a line appearing in $\overline{S}$ univocally incident at points p_1 and p_2. We must see that exactly one point representing p_1 (the same for p_2) exists in map M. We know that $l \in \mathcal{A}(p_1)$. Thus, either p_1 is created with l in $\overline{u}$, or as U' is closed with respect to the relation of direct dependency, the most abstract representative of p_1 must have been already created. We also know that $CHILDREN(p_1) \in \mathcal{D}(l)$. Two possibilities arise: 1) if $CHILDREN(p_1) = \emptyset$, then some representative of p_1 must exist in M, and 2) if $CHILDREN(p_1) \neq \emptyset$, then l must exist before p_1 disappears, thus also some representative of p_1 must exist in M.

   If $\overline{u}$ is an update of type *region-split*, the two endpoints of the line to be inserted must be connected. And if $\overline{u}$ is an update of type *feature-line-addition*, the two endpoints of the line to be inserted must be disconnected. But as U' is closed with respect to the relation of direct dependency, both conditions are satisfied.

   If line l does not belong to $\overline{S}$, it will represent a set of lines in either a *line-split* tree or a *line-to-region* tree. In both cases, assuming an update of type *point-to-line* affecting the endpoints of l is applied, as we know the lines of $\overline{S}$ represented by l, we will be able to update combinatorics univocally.

b) Analogue. □

Teorema 1. *Let* $\overline{S} = (\underline{M}, U = (\overline{u}_1, \overline{u}_2, \ldots, \overline{u}_k))$ *be a sequence of atomic refinement updates, and let* $U' = (\overline{u}_{i_1}, \overline{u}_{i_2}, \ldots, \overline{u}_{i_h})$ *be a collection of updates closed with respect to the relation of direct dependency. Then,* $\overline{S}' = (\underline{M}, U' = (\overline{u}_{i_1}, \overline{u}_{i_2}, \ldots, \overline{u}_{i_h}), \oplus_{\overline{S}})$ *is a feasible subsequence of* $\overline{S}$.

Proof

1. $\overline{S}' = (\underline{M}, U' = (\overline{u}_{i_1}, \overline{u}_{i_2}, \ldots, \overline{u}_{i_h}), \oplus_{\overline{S}})$ is a subsequence of $\overline{S}$.
   This part has been already proven in Proposition 2.
2. $\overline{S}'$ is feasible.
   We must prove that $M = M'$, where:
   - M is the map of $\overline{S}$ obtained after applying collection $(\overline{u}_1, \overline{u}_2, \ldots, \overline{u}_m)$ on $\underline{M}$, with $m = max\ \{i_1, \ldots, i_h\}$.
   - M' is given by:

$$M' = \underline{M} \oplus_{\overline{S}} \overline{u}_{i_1} \oplus_{\overline{S}} \overline{u}_{i_2} \ldots \oplus_{\overline{S}} \overline{u}_{i_{(h-1)}} \oplus_{\overline{S}} \overline{u}_{i_h} \oplus_{\overline{S}} \sum_{\substack{j=1 \\ j \neq i_1, \ldots, i_h}}^{m} \overline{u}_j.$$

   a) We will first see that M and M' contain the same entities. Obviously, both maps have the same number of entities. Thus, it is sufficient to prove that each entity of M also appears in M'. Let b be an entity appearing in M, created in update $\overline{u}_b$. Assume b is the most abstract representative of itself. Then, according to rule R4 of the representation process, $b \in M'$. If, on the contrary, b has been created in some tree of the forests of updates, as all the ancestors of b have been already split up, and none descendant of it has been split up, b belongs to a representation tree containing just it and all its descendants, as we show next. Thus, $b \in M'$.
   Let us first show that if none node in the subtree $\mathcal{S}_b$ of b has been split up, all the nodes in the $\mathcal{S}_b$ belong to the same representation tree. We will prove it by induction on the number of nodes in τ_b which have been split up. If $n = 0$, all nodes in τ_b belong to the original representation tree $\mathcal{R}_\tau$. Assume now $n - 1$ nodes out of $\mathcal{S}_b$ have been split up, and all the nodes in $\mathcal{S}_b$ correspond to the same representation tree. If one more node not belonging to $\mathcal{S}_b$ is split up, according to the representation process all the nodes in $\mathcal{S}_b$ will correspond to the same representation tree.
   Now we have to see that no any other node of τ corresponds to the same representation tree. For each node of τ that is split up, a node ν is created, which by Definition 6 will be univocally incident at the two created nodes, or descendants of each of them. Thus, for some node

connecting $\mathcal{S}_b$ to some other node to exist, some ancestor of b should not have been split up.

b) Finally, we must see that the configuration of entities incident at b is the same in both M and M'. The representation process ensures that if b and such entity belong to the same representation tree, they will be incident in M'. Thus, we must only pay attention to the rules to modify combinatorics.
   i. Let b be a point. If it is isolated, it will be univocally incident at either the region or some descendant of the region it belongs to in M. If it is not isolated, the configuration of lines incident at it univocally defines the configuration of regions incident at it. Let l be any line incident at b in M. Line l belongs to M', and according to the rules to specify combinatorics, the endpoints of l are the points in M which represent the points l is univocally incident at in $\overline{S}$, one of which must be either b or some descendant of b.
   ii. Let b be a line. The two regions it is incident at must be the same in both M and M'. Let r be a region incident at b in M. If b belongs to some inner boundary or feature of r, according to rule C2 to modify combinatorics, b will be incident at r. In case b does not belong to any inner boundary or feature, when it was created it was incident at some region representing r. As lines are considered to be oriented, when the rest of updates of type *region-split* affecting such region are applied, b will also be incident at r. □