Designing a library to support model-oriented generalization

Giuliana Dettori
I.M.A C.N.R, Via De Marini 6
16149 Genova, Italy
e-mail: dettori@ima.ge.cnr.it

Enrico Puppo
I.M.A C.N.R, Via De Marini 6
16149 Genova, Italy
e-mail: puppo@ima.ge.cnr.it

1. ABSTRACT

Model-oriented generalization modifies the internal representation of a map to make it simpler. We present a layered library supporting this task, based on a separation of the topological, metric and semantic aspects of maps. We outline the theoretical framework of our approach, and describe the main structure of its software layers.

1.1 Keywords

Model-oriented generalization, cartographic generalization, plane map, layered library, topology.

2. INTRODUCTION

The process of modifying maps in order to suitably represent them at smaller scales is known as generalization. We can distinguish two different kinds of generalization, which involve different operations [9, 6, 11]. Model-oriented generalization aimes at reducing the *information density* in a map and affects its internal representation in the context of a GIS. Here a map is first of all a digital model that includes numerical data, relations between them and semantic information. Great importance is given to the consistency of internal data through increasing levels of abstraction. On the other hand, cartographic generalization occurs at graphic representation level and is traditionally performed by cartographers to improve legibility. Both kinds of generalization are non-algorithmic tasks since they involve both exact rules and as yet unquantifiable considerations. Nevertheless, it is important to provide tools that support generalization operations, and help maintaining consistency of results through such processes. Most literature in this context offers a non-homogeneous choice of tools to solve specific subtasks, while a systematic treatment of the whole problem is still missing.

In our previous work, we have proposed a formal framework to model and manipulate maps through different levels of abstraction [10], and we have analysed and formalised in this context the main classes of operations in generalization [4]. The distinction between topological aspects, related to the structure of a map, and metric aspects, related to the size and appearance of its features, is a key point of our approach. Our framework permits one to modify maps by means of a minimal kernel of elementary operations, and to preserve topological consistency between representations at different levels of abstraction in the context of a unified multi-resolution model. The theory underlying our framework, as well as issues on multi-resolution, are developed further in a companion paper [2]. Here, we address the design of a library to support generalization, which is based on the mentioned model and kernel of operations.

Our library has a layered architecture with increasing levels of specialization, conceived as a part of a layered model of GIS. Its kernel is formed by several groups of primitive operations that permit construction and modification of a graph representing a map, as well as inquiries on the attributes and relations of its elements. This basic kernel is made available to three independent transformation modules, devoted to perform topological, metric and semantic transformations and consistency checks, as well as to a higher-level layer, performing model-oriented generalization. The transformation modules are available to the model-oriented layer, which combines their operation. Cartographic generalization can be added on top of this layer, and, in turn, is available to a drawing layer. On top of all is the layer of applications.

The paper is organised as follows. Section 3 shortly reviews the theoretical tools we use to model plane maps; Section 4 discusses model-oriented and cartographic generalization; Section 5 describes the architecture of the proposed library; Section 6 describes the content of the kernel, in particular of its generalization primitives; Section 7 outlines the combination of operations in the model-oriented layer; Section 8 gives some concluding remarks.

3. MODELING PLANE MAPS

A plane map is defined as a disjoint covering of a portion of the plane with a collection of spatial entities, namely points, lines, and regions. This definition is analogous to those adopted by other authors and in some commercial systems [7, 8]. In this section, we review the formal models for maps that we introduced in our earlier work [3, 10].

Lines in a map are always simple (i.e., a line cannot self-intersect). A line can be either open, or closed. We approximate generic lines with polylines, hence a line l is described by a sequence of vertices. Extreme vertices in the sequence, called endpoints of l, form its boundary. If l is closed, its two endpoints are coincident, thus its boundary is made up of a single point.

A plane map is a triple $M = (P, L, R)$, where R, is the set of connected components, called regions, obtained by removing the plane graph (P, L) from the real plane. Therefore, each region is an open set. A region can have holes, cuts, and punctures, caused by internal loops of lines, dangling lines, and isolated points, respectively. There is a unique unbounded region, called the infinite region, which surrounds the domain of the map.

The *combinatorial boundary* of a region r is the collection of points and lines of M that form the *topological boundary* of r. The regularization r^* of r is obtained by filling its cuts and punctures: more formally, r^* is the interior of the closure of r [12]. The elements of (P, L) inside r^* (cuts and punctures) are called the *features* of r, while those belonging to the topological boundary of r^* form the *proper boundary* of r. The proper boundary can be subdivided into loops: an external loop surrounding r, and zero or more internal loops surrounding its holes. Each loop is a closed chain formed by some lines and their endpoints. The combinatorial boundary of r is formed by the union of its features and of its proper boundary, and it completely characterises the geometry of r.

We model the topological structure of a map M through a purely combinatorial structure, called an Abstract Cell Complex (ACC). An ACC is a triple $\Gamma = (C, \triangleright, ord)$, where C is a set of atomic cells, $\triangleright$ is a strict partial ordering on C, called *bounding relation*, and ord: $C \longrightarrow \{0, 1, 2\}$, called the *order function*, is such that for every pair of cells γ_1 and γ_2 in C, $\gamma_1 \triangleright \gamma_2 \Rightarrow ord(\gamma_1) < ord(\gamma_2)$.

The ACC describing a map M has one cell for each element (point, line, region) of M; for each pair of cells γ_1 and γ_2, we have $\gamma_1 \triangleright \gamma_2$ if and only if the entity of M corresponding to γ_1 belongs to the combinatorial boundary of the entity of M corresponding to γ_2; the function ord gives the dimension of the entity corresponding to each cell: 0 for points, 1 for lines, and 2 for regions. In the sequel, we overload the name of a cell to denote also its corresponding entity in the map. The *star* of an entity γ_1 is the set $*\gamma_1$ of all entities containing γ_1 in their topological boundaries. Based on the definitions above, all topological relations among entities in a map can be translated into relations between cells of the corresponding abstract cell complex, expressed in terms of $\triangleright$, ord, and $*$ (see [10, 1] for details).

A complete description of a map is given by adding geometric and semantic attributes to cells of an ACC. The geometry of each point is given by its Cartesian coordinates. The geometry of a polyline is given by its endpoints, which can be retrieved from the ACC through the relation $\triangleright$, and by the sequence of its internal vertices. The geometry of a region is given by its boundary, which can be retrieved from the ACC through the relation $\triangleright$. Moreover, each entity in the map can have a set of semantic attributes attached to it.

4. MODEL-ORIENTED AND CARTOGRAPHIC GENERALIZATION

The classical literature on generalization lists a set of basic operations that affect the structure and appearance of maps in various ways [9], namely: *selection*, which discards some objects that are no longer relevant in the generalized map; *collapse*, which represent some object with a lower-dimensional entity (e.g., a city is represented with a point); *aggregation*, which represents a group of objects with a single entity; *simplification*, which reduces the level of detail in the geometric representation of lines; *displacement,* which moves entities apart from each other in order to improve legibility; *exaggeration*, which modifies the shape of some entity to a caricature in order to highlight some shape features; and *symbolization*, which replaces an entity representing an object with a proper graphical symbol. In [4], we have discussed how such operations affect the topological and metric structure of a map, pointing out that only some of them modify the information in a map, while others are just pertinent to its graphical representation.

Model-oriented generalization refers to the modification of the internal representation, in order to feed computer-aided tools for map analysis and geographic reasoning with a lighter model to work on. While performing this task, it is important to apply inherently consistent operations, which allow any tool working on a generalized map to produce results that are not in contradiction with those obtained on a more detailed map. Model-oriented generalization may affect the topological structure of the map by reducing the number of objects represented (detail); it may affect metrics by reducing the accuracy of representation of each entity (shape); it may affect semantics by reducing the attributes attached to each entity and the discrimination among different types of objects represented (classification). Topological abstraction seems somehow more basic than metric and semantic ones, and it can induce changes of geometrical and/or semantic nature, while the latter two can occur independently of each other and usually do not influence the topological structure. Model-oriented generalization may perform *selection*, *collapse*, *aggregation*, and *simplification* operations. The first three operations modify the structure of a map, while the last one just modifies its shape.

If our purpose is to obtain a graphical representation of a map to be interpreted by people, generalization is aimed at organizing data both to convey the proper amount and kind of information, and to make the graphical representation as clearly legible as possible. Once we have decided *what* information we want to include (through model-oriented generalization) we must decide *how* to display such information. This latter task consists in modifying the map model further through cartographic generalization, and to generate a proper drawing from it.

Cartographic generalization may perform *displacement* and *exaggeration*, as well as further line *simplification*. It does not change the amount of information encoded in the map, but rather makes it better readable by people. In this case, we can relax some consistency constraints, as long as the result is improved from the point of view of human interpretation. Indeed, some operations can produce inherently inconsistent results.

5. GENERAL ARCHITECTURE OF OUR LIBRARY

Our library is devoted only to model-oriented generalization. Since model-oriented and cartographic generalization have different purposes, use different operations, and the former must be used before the latter also when producing only drawings, we decided that they should be organized in two separate software layers. The general architecture of our library and its integration with Cartographic Generalization, Drawing and Application layers are depicted in Figure 1.

The library is essentially composed of a basic *kernel*, on the lowest layer, an intermediate *transformation layer*, made up of three independent modules devoted to topological, metric and semantic transformations, and an upper layer for *model-oriented generalization*, which lays on top of both the kernel and the transformation layers. As concerns its integration within a complete GIS system, the model-oriented generalization can interact with cartographic generalization, with drawing modules, and with application programs, which, in turn, can be seen as being one above the other. The *kernel* must be available to all higher layers, since it contains several groups of functions necessary to set and retrieve information related to all aspects of a map, i.e. relations among entities, numerical values describing their shapes, and their semantics.

The *transformation layer* is composed of three independent modules:

The *topological module* includes macro operations built as sequences of the atomic topological primitives in the kernel, such as the contraction of a chain to a single line or to a single point; the contraction or abstraction of a complex feature, made up of several points and lines; etc. While the set of atomic primitives in the topological kernel has been proved to be complete [1], a group of macro operations of this kind obviously can not be exhaustive, though we have included in it a rich set of combinations of primitives; however, the fact that the kernel is also available to both the Model-oriented generalization and Application layers can compensate for possible missing operations. This module manipulates only the relations between entities, but does not compute shape or semantic changes involved by such manipulations.

Shape transformations are computed by the *metric module*. This module includes both high-level operations, like line simplification, and low-level ones, like shape modifications induced by a region contraction or required by drawing procedures. For this reason, the metric module, unlike the topological and semantic ones, must be made available also to the cartographic layer. While topological operators perform structure manipulation, metric functions are essentially based on numerical and geometrical algorithms, which do not influence a map's structure but only its shape.

The *semantic module* contains information such as thresholds for the existence of an entity in different dimensions, or a description of its state and use, which can help automate the generalization process, determining, for instance, if a line should be contracted or abstracted, or with which adjacent region a given region is most suitably merged. The mechanism of inheritance and merge of semantic attributes over generalization is not a trivial problem, and requires a proper set of semantic generalization functions; the more accurate these functions, the more effective the contribution of semantics to automate the generalization process. However, a thorough analysis of this issue is beyond the scope of this paper.

These three modules have been designed to work independently of each other, since each kind of operation has its proper underlying theoretical framework. However, the three kinds of operation are not conceptually independent, since topological primitives can induce changes both in the shape and in the semantics of the affected entities. The results of the three modules are combined at the upper level by the *model-oriented generalization layer*, whose functions activate suitable metric and semantic functions to perform the changes in shape and meaning induced when topological modifications are performed. A reason for making this blend at this level, instead of at the previous one, is that topological macro operations are built as (possibly) long sequences of primitive operations; hence, performing the correspondent shape transformation each time would be unnecessarily time consuming, besides being meaningless for the sake of generalization. The structure and working of this layer is illustrated in Section 7.

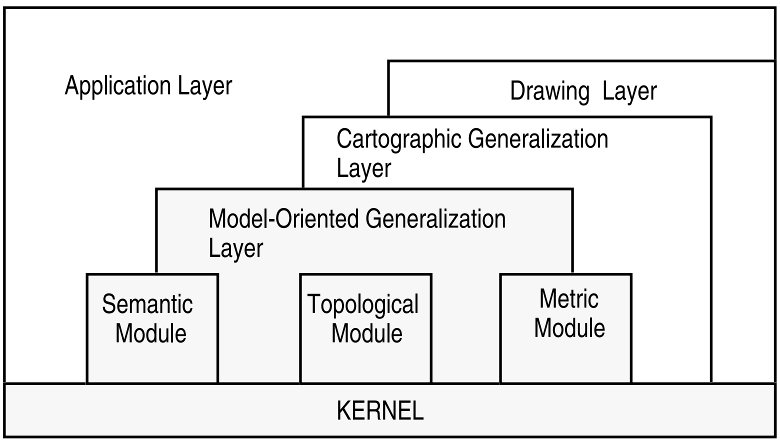

Fig.1 - Structure of our library (shaded) and its integration with cartographic operations, drawing and applications

Cartographic generalization, drawing and application functions are not part of our library; nevertheless, we have included them in the diagram of Fig.1, since their interaction with our library does not appear to be simply "one on the top of the other", but more articulated, which makes them worth a few words discussion.

Cartographic generalization works on a map model and produces another map model, modified according to aesthetic needs, in such a way to preserve some, but not always all, the topological relations in the map (the fewer relations are violated, the more adherent to reality is the map). In Fig.1, we put this layer on top of model-oriented generalization, since a displacement or an exaggeration typically turns out to be necessary for representation at a given scale after a structural generalization has been performed. Moreover, this layer must be granted access also to the metric module, which contains numerical algorithms to change the shape of lines, and to the kernel, which performs topological modifications and consistency checks on the manipulated entities and their neighbourhoods.

A *drawing layer*, devoted only to the external representation of maps, should sit on top of both generalization layers, since either layer can output a map model ready for drawing. This module gets a map model and represents it graphically on some external support, using for each entity the appropriate symbols; in order to efficiently scan the map model, it needs to access the data structure's navigation and inquiry functions, and this explains why it is granted access also to the kernel of our library.

Finally, an *application layer*, which sits on top of the previous ones, needs to access both the model-oriented generalization and the drawing layers, since map generalization is of interest of a wide range of applications, which can be aiming both at performing operations on different elaborations of the original model and/or at producing graphical representations of them. Since applications might need to operate directly on models, it is necessary that the semantic module and the kernel be accessible also to applications.

6. THE LIBRARY'S KERNEL

The *kernel,* on the very bottom level, includes several groups of functions, devoted to construction, modification and inquiry of the data-base representing a map. All functions are designed so that the consistency of the graph structure is always preserved.

For *graph creation*, the kernel includes:

- operators to build the topological structure; these operators support the creation of a map by adding elements to a simpler one. These operators can be viewed as the inverse of the transformation operators described below.
- operators to set numerical information for each entity, such as *set-line-geometry*, which describes the shape of a line through a list of points; or *set-point-position,* which specifies the coordinates of a point; consistency checks guarantee that numerical values are assigned to entities only if they are compatible with the position of the entity in the graph;
- operators to associate semantic information to each entity.

Inquiry functions allow the retrieval of information from the internal representation of the map.

Transformation operators perform basic topological generalization. They allow only modifications that consistently reduce the amount of information in the map. In [10, 2] we show that structural consistency corresponds to the fulfilment of some constraints when modifying the ACC representing the structure of a map, and constitutes the base of our generalization framework. This group is formed by seven atomic primitives which have been proved to be necessary and sufficient to express all consistent generalization sequences [1]. Each of them defines not only a local modification, but also the relation between each entity in the initial map and its correspondent entity in the generalized map.

Consistent atomic operators can be of three kinds:

Contractions refer to rubber-sheet modifications of a map; they affect a single entity by reducing its dimensionality:

- *line-to point-contraction (in line l, out point p)* maps an open line l and its two end points p_0 and p_1 to a single point p in the simplified map (Fig. 2a); all lines and regions incident at p_0 and p_1 must be stretched consistently to become incident at p;
- *region-to-point-contraction (in region r, out point p)* applies to a simply connected region r without features, whose boundary consists of a single closed line l and of its endpoint q (Fig. 2b); it maps r, l and q to a single point p; all lines and regions incident at q must be stretched consistently to become incident at p;
- *region-to-line-contraction (in region r, out line l)* applies to a simply connected region r without features, whose boundary consists of a pair of open lines l_0 and l_1 that share common endpoints p_0 and p_1 (Fig. 2c); it maps r, l_0, and l_1 to a single line l having its endpoints at p_0 and p_1; the regions incident at l_0 and l_1 from out-side r are stretched consistently to become incident at l.

Regions to be contracted can be isolated or have some lines incident in the point(s) on their boundary; the lines to be contracted can either be features or belong to a boundary. The new entity created should inherit the semantics of the entities which have been contracted.

Merge operations refer to the immersion of adjacent entities obtained by abstracting their common boundaries:

- *line-merge (in point p, out line l)* applies to a point p, which represents the shared boundary of a pair of open lines l_0 and l_1 (Fig. 2d); there must be no lines other than l_0 and l_1 incident at p ; it maps l_0, l_1, and p to a single line l, having its endpoints at the endpoints of l_0 and l_1 other than p; regions incident at l_0 and l_1 become incident at l ;
- *region-merge (in line l, out region r)* applies to a line l which belongs to the shared boundary of a pair of regions r_0 and r_1 (Figure 2e); it maps r_0 r_1, and l to a single region r, whose boundary coincides with the union of boundaries of r_0 and r_1 minus line l.

The semantics of the new entity created by merging operations should be a merge of the two original entities

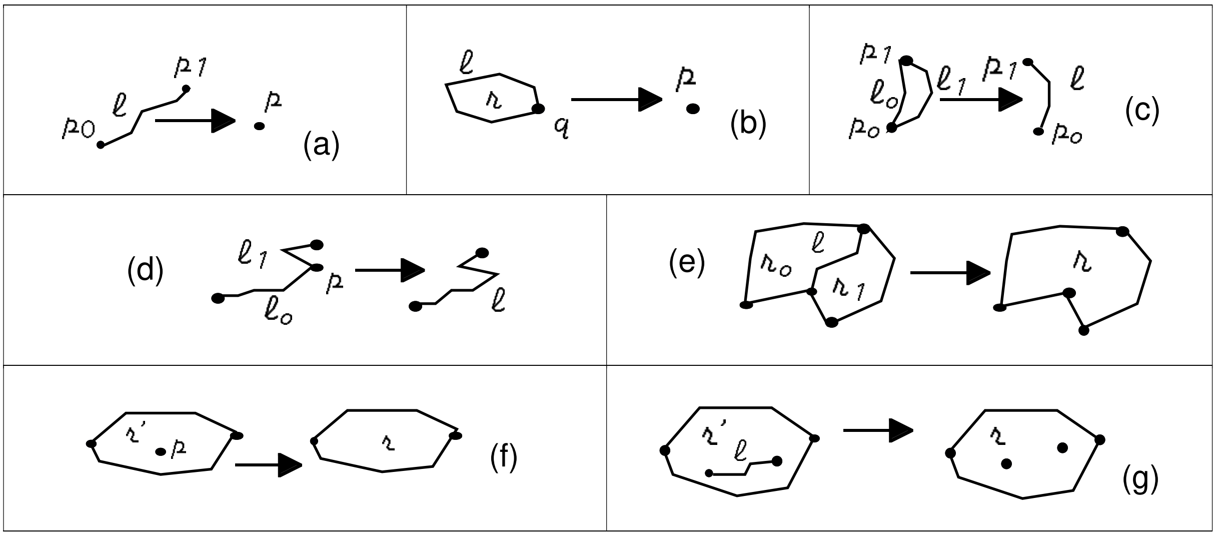

Fig.2 - Topological primitives: line to point contraction (a); region to point contraction (b); region to line contraction (c); line merge (d); region merge (e); point abstraction (f); line abstraction (g).

(e.g.; wheat field + corn field = crop field) or include both meanings when it is not possible to merge (e.g. lake + solid ground does not make marsh). Rules for semantic merge should be properly set at application level.

Abstraction operations refer to the elimination of features by merging them with their containing regions:

- *point-abstraction (in point p, out region r)* applies to a point *p* that is isolated inside a region *r'* (Fig. 2f); it maps both *p* and *r* into a region *r,* whose combinatorial boundary coincides with the combinatorial boundary of *r'* minus *p*;
- *line-abstraction (in line l, out region r)* applies to a feature line *l* inside a region *r* (Fig. 2g); it maps both *l* and *r'* into a region *r,* whose combinatorial boundary coincides with the combinatorial boundary of *r'* minus *l.*

With these operators, the semantics of the surrounding region can be updated to mark (up to a certain threshold) the existence of entities which have been abstracted.

Aim of the above set of topological primitives is just to manipulate the data structures, not to determine new values for their information fields. Metric and semantic changes which are conceptually implied by most of these operators are not implemented in the kernel, since these modifications conceptually belong to a higher level of generalization.

7. MODEL-ORIENTED GENERALIZATION

Let us now illustrate on a simple example how the model-oriented generalization layer splits a generalization task among the three transformation modules and combines their results. Let the configuration in Fig.3a be given; we want to manipulate the map in order to arrive to a configuration whit only three large regions (Figs.3b and 3c). This can be done in different ways, according to the meaning given to this operation.

Case 1: We want the central region to be represented by a point. In terms of generalization operations this corresponds to a *collapse*. In topological terms, this corresponds to a *region-to-point contraction*; since region D is not in simple form (its boundary has more than one line), we need to apply one of the macro-operators in the topological module, not simply a topological operator: *complex-region-to-point-contraction (in region R, out point P)*. This function first eliminates possible holes and features inside region D, then applies a line-to-point-contraction to all lines but one on the region's boundary; at this point the region is in simple form, and the topological primitive *region-to-point-contraction* can be applied, replacing D in the data structure with a single point. Now the metric transformations must be tackled, which include: 1) to compute suitable coordinates for the new point P; unless different indications are given in the semantics or are required by the user, the centroid of the original region is chosen; 2) to compute an extra segment for the lines incident in D, in order to connect them with P; unless different prescriptions, the segment joining P with P_1, P_2, P_3 is added to the geometry of the line. As concerns semantics, the new point will inherit the semantics of region D. Joining all parts together, a specification of this function looks as in Fig. 4 below. It is evident that metric operations do not need to be paired with every single topological change performed. The map's name is not

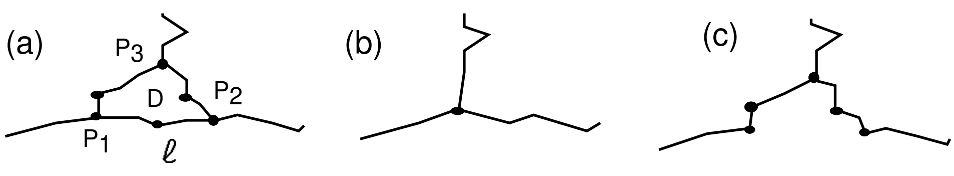

Fig.3 - Region D (a) can be eliminated by region collapse (b) or by region merge (c)

specified as parameter since this function is a method of a data type *map* in an object-oriented approach.

```
function collapse-region (in region R, out point P);
{collapse-semantics (R, sr);
compute-centroid (R, (x,y));
inquire-incident-lines (R, List);
for each L in List, update geometry of L;
complex-region-to-point-contraction (R, P);
set-point-coordinates (P, (x,y));
set-semantics (P, sr) }.
```

Fig. 4 - A specification of the generalization function *collapse-region*; *collapse-semantics* belongs the Semantic module; *compute-centroid, update-geometry* to the Metric one; *complex-region-to-point-contraction* to the Topological one; *set-semantic*, *set-point-coordinates, inquire-incident-lines* to the Kernel.

Case 2: We want the central region to be merged into the adjacent region which is semantically closer to it. From the point of view of generalization, this corresponds to an operation of *selection* if region D is considered meaningless in the context of the map under construction, or to an operation of *aggregation*, if we want that a trace of the existence of this regions be preserved in the semantics. From a topological point of view, in both cases a region-merge operation is performed. Since the boundary *l* separating the two regions to be merged might not be a single line, macro operations (belonging to the Topological module) must be used in place of topological primitives (belonging to the Kernel). If the extreme points of *l* are already known, we can apply the function *chain-to-line*, which merges all lines of a chain into a single line; then the two regions can be merged by means of the region-merge primitive. If the extreme points of the common boundary are not known, we should use the function *merge-adjacent-regions*, which first finds the common boundary between two regions, then collapses this chain into a single line, and finally merges the two regions by applying the *region-merge* primitive. The graphical result of the two operations is the same. What differentiates the two approaches is the result of the semantic update. If a selection is made, the semantic of region D is simply disregarded; if a region merge is applied, the semantic of the new region created is a merge of the semantics of the two regions involved in the operation. In neither case the geometry of the nearby lines has to be changed. As concerns points P_1 and P_2, the decision whether to erase them or not depends on whether they have relevant semantic attributes or are just endpoints of *l.*

This example points out that, in order to have a rich, ductile, accurate and fast library, several functions must be provided to perform each generalization operation in different contexts.

8. CONCLUDING REMARKS

Our library aims at providing a consistent software layer to develop applications that either need to manipulate maps at different levels of detail, or directly perform generalization in the context of GIS and cartography. The library can be used both to support interactive systems, either manual or semi-automatic, equipped with graphical interfaces, and to support fully automatic systems, possibly assisted by intelligent agents.

All library parts regarding structural manipulation of maps are at an advanced stage of development. We are also working on techniques to solve main metric problems, while the semantic part is at an earlier stage of design.

A data structure similar to that proposed in [3] has been adopted. The kernel is implemented in C++ by following an object-oriented approach, and it is built on top of CGAL [5], a computational geometry library that allows its users to exploit a full spectrum of geometric data types and algorithms.

9. REFERENCES

[1] Bertolotto, M. : Geometric Modeling of Spatial Entities at Multiple Levels of Resolution, Ph.D. Thesis, Dept. of Comp. & Inform. Science, Univ. Genoa, 1998

[2] Betolotto, M., Dettori, G., Puppo, E. : A combinatorial framework for managing levels of detail in geographic databases (in preparation)

[3] De Floriani, L., Marzano, P. Puppo, E. : Spatial queries and data models. In Spatial Information Theory - A theoretical basis for GIS, A.U. Frank, I. Campari (Eds.), Lect. Notes in Computer Science **716**, Springer-Verlag, 1993, pp.113-138

[4] Dettori, G., Puppo, E. : How Generalization Interacts with the Topological and Metric Structure of Maps. Advances in GIS Research II, M.J.Kraak, M.Molenaar eds, Taylor & Francis, 1997, pp. 550-570

[5] Fabri, A., Giezman, G.-J., Kettner, L., Schirra, S., Schoenherr, S. : On the design of CGAL, the computational geometry algorithms library, Res.Rep. MPI-I-98-007, Max-Plank Insti. for Computer Science, 1998; http://www.cs.uu.nl/CGAL/

[6] Frank, A.U., Timpf, S. : Multiple representations for carto-graphic objects in a Multiscale tree - an intelligent graphical zoom. Comput & Graphics, vol.18, n.6, 1994, pp.823-829

[7] Herring, J. : TIGRIS: topologically integrated GIS. Proc. Autocarto 8, ASPRS/ACSM, Baltimore, MD, pp.282-291, March 1987

[8] Molenaar, M.: Single valued vector maps - a concept in GIS. Geo-Informations-systeme, **2** (1), 1989

[9] Müller, J.C., Weibel, R., Lagrange, J.P., Salgé, F. : Generalization: state of the art and issues. In GIS and Generalization: Methodology and Practice, J.CMüller, J.P.Lagrane and R.Weibel eds., Taylor and Francis 1995, pp.3-17

[10] Puppo, E., Dettori, G. : Towards a formal model for multiresolution spatial maps. Advances in Spatial Databases (Proc. SSD95), M.J. Egenhofer, J.R.Herring eds. LNCS 951, 1995, pp.152-169

[11] Rigaux, P. , Scholl, M.: Multiple Representation Modeling and Querying. Geographic Information Systems, J. Nievergelt, T. Roos, H.-J.Scheck, P. Widmayer, Eds., LNCS 884, Springer 1995, pp.59-69

[12] Tilone, R.B.: Set membership classification: a unified approach to geometric intersection problems, IEEE Trans. on Computers, C-29, 10, pp.874-883