REPRESENTATION AND ANALYSIS OF TOPOLOGY IN MULTI-REPRESENTATION DATABASES

M. Breunig *, A. Thomsen , B. Broscheit, E. Butwilowski, U. Sander

IGF, University of Osnabrück, Kolpingstr. 7, 49069 Osnabrück, Germany –
{martin.breunig, andreas.thomsen, edgar.butwilowski, uwe.sander, bjoern.broscheit}@uos.de

KEY WORDS: Topology, multi-scale representation, geodatabase, MRDB, data modelling, LOD, abstraction.

ABSTRACT:

Multi-scale representation and analysis of topology is playing a growing role in Photogrammetric Image Analysis. However, the standardisation of multi-scale topological data models is still at its beginning. Furthermore, the multi-representation of geo-objects poses new challenges, resulting in the development of Multi-Representation Databases. In this article the realisation of a general model based on oriented hierarchical d-Generalised Maps to represent and analyse topology in MRDB is described in detail. The model can be used as a data integration platform for 2D, 3D, and 4D topology. Examples of elementary and complex topological operations for multiple representations are presented. An application example with 2D cartographic datasets from Hannover University shows the feasibility of the new approach. Finally, an outlook on future research is given.

1. INTRODUCTION

Multi-scale representation and analysis of topology is important for GIS and will also play a growing role in Photogrammetric Image Analysis. However, to our knowledge, the database representation of topology in different levels of detail (*LOD*) has not been investigated in detail.

Multi-representation of topology poses new challenges resulting in the development of *Multi Representation Databases (MRDB),* that manage discretely and continuously changing LOD. Although generalisation operations affect the topology of a spatial model, research about the representation and management of topology in MRDB is still at its beginning. In (Thomsen and Breunig, 2007), we propose some elementary and complex topological operations for a topological database toolbox based on oriented Generalized Maps (G-Maps).

In this paper, we investigate how oriented hierarchical G-Maps can be used to handle the topology of a digital spatial model at different levels of detail in a MRDB based on the object-relational model, providing a generic, application-independent approach. The method is general enough to support 2- and 3-dimensional models, as well as 2D-manifolds in 3D space.

2. RELATED WORK

Approaches for representing topology in 3D modelling have been examined by different authors (Mäntylä, 1988). For the representation of 3D-objects in GIS by 2D-manifolds, (Gröger and Plümer, 2005) propose “2.8-D maps”, that avoid the topological complexity of true 3D-Models. Cellular complexes, and in particular cellular partitions of d-dimensional manifolds (d-CPM) have been described to represent the topology of an extensive class of spatial objects by (Mallet, 2002). The topology of d-CPM can be represented by d-dimensional Cell-Tuple Structures (Brisson, 1993), respectively d-dimensional Generalized Maps (d-G-Maps) (Lienhardt, 1994). (Lévy, 1999) has shown that 3D-G-Maps have comparable space and time behaviour as the well-known DCEL and radial edge structures, but can be used for a much wider range of applications, allowing for a more concise code. Lévy also introduces hierarchical G-Maps (HG-Maps) for the representation of nested structures. 3-G-Maps are also applied e.g. in the geoscientific 3D-Modelling software GOCAD (Mallet, 1992, 2005). (Fradin et al., 2002) use 3-G-Maps to model and visualize architectural complexes in a hierarchy of multi-partitions. Finally, an interactive graphical G-Map-based 3D-modeller MOKA has been made available by the group of graphical informatics at Poitiers University (MOKA, 2006). (Meine & Köthe (2005)) have introduced the GeoMap, a related but less general concept based on half-edges, that integrates planar topology and geometry for raster image segmentation.

3. MULTI-SCALE REPRESENTATION AND ANALYSIS OF TOPOLOGY

Aggregation, simplification, elimination, displacement and typification are well-known generalisation transformations. Aggregation and elimination directly affect the topology of a map. Simplification may affect the interior structure of an object, whereas displacement may be employed in order to maintain topological consistency under a geometrical generalisation operation - e.g. if smoothing a river bend would leave a building on the wrong side. In a first step, we concentrate on the aggregation of contiguous cells by the

* Corresponding author.

application of sequences of Euler transformations, being aware that this approach covers only a selection of generalisation operations. In a second step, we will try to model the aggregation of disjoint cells using transformations of classifications/colourings of cellular complexes. Whereas the choice of the generalisation method is taken by the geoscientist, supported by specialised software (cf. Haunert & Sester, 2005), we focus on the representation of the given transformations and of the resulting relationships between LOD in the MRDB. Relationships between cells at different levels can be defined by explicit links, or by indicating the sequence of elementary operations that transform a cellular complex at scale *A* into a cellular complex at scale *B*. It is the task of the database software, to keep track of the incurred changes, and if possible to support *transitions* with *commit* and *rollback* operations.

3.1 Representation

3.1.1 Hierarchies of maps: For the representation of multi-scale topology, Lévy (1999) proposes Hierarchical G-Maps (HG-Maps): The Aggregation of neighbouring cells results in a classification of cells on the more detailed level *A*, each class being associated with one cell on the less detailed level *B*. It can be represented by an n:1-mapping from one level *A* to level *B*. As cells are merged, and interior boundaries disappear, the number of cell-tuples is reduced. The cell-tuples on level *B* can be associated with a selection of cell-tuples on the lower level *A*, or be identified with a subset of the latter. If the geometry of the remaining cell boundaries is not changed after the aggregation step, higher level cell-tuples may delegate their geometrical embedding (co-ordinates, lengths, angles etc.) to their counterparts on the lower level (**fig. 1**), so that a higher-level edge is geometrically represented by a sequence of lower-level arcs and vertices. Otherwise, links with a new higher-level geometrical embedding must be established.

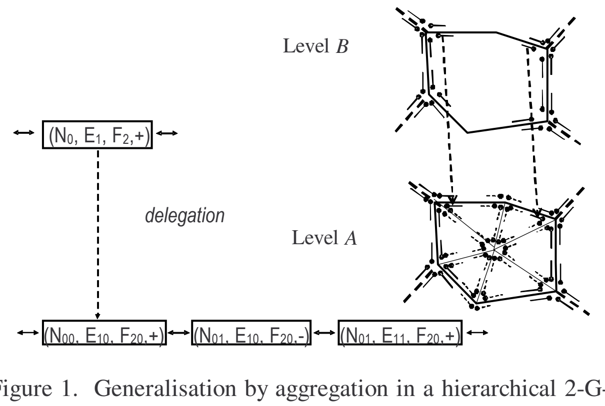

Figure 1. Generalisation by aggregation in a hierarchical 2-G-Map. Cell-tuples (darts) are symbolised by small pins.

3.1.2 Progressive Variation of LOD: Due to the necessity of keeping all levels of detail consistent with each other, any changes in an MRDB are first introduced at the greatest scale, and then propagated upwards using appropriate generalisation methods (Haunert & Sester, 2005). Carrying this "dynamic" approach a step further, we investigate the applicability of progressive meshes. The progressive triangulation method (Hoppe, 1996) uses two localised elementary operations, namely the "edge collapse" and its inverse, the "vertex split", to coarsen or to refine a triangle network incrementally in both directions, by successively applying a sequence of stored "delta" operations. This method is well suited for *progressive transmission*, as it can reduce the amount of data exchanged between a geo-database server and a local client (cf. Shumilov et al., 2002).

Generalized maps are abstract simplicial complexes, but Hoppe's method cannot be adapted: Although a d-cell-tuple is an abstract d-simplex, its d+1 components belong each to a different class defined by dimension, and therefore cannot be merged, like in an "edge collapse" operation on a triangle network. An analogous argument holds for the inverse "vertex split" operation. Instead, we investigate the possibility to use combinations of the Euler elementary split and merge operations on cells to model the transformation of topology induced by generalisation. Different from Hoppe's method, the progressive mesh transformation is controlled by the external generalisation method, and not by a given optimisation criterion. Note that the merge operations are applicable only in certain configurations and hence require supervision.

3.2 Analysis

The relational representation of d-G-Maps has been made persistent using an Object-Relational Database Management System (*ORDBMS*). Implementing a topological component for multi-representation databases (Thomsen and Breunig, 2007) we used 2D- and 3D-G-Maps with the ORDBMS PostgreSQL (PostgreSQL.org, 2006) in combination with the open source PostGIS (PostGIS.org, 2006).

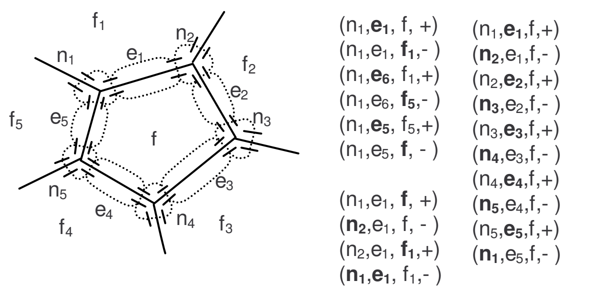

Figure 2. 2-G-Map with darts and involutions, and cell-tuple representation of the orbits around node n_1, edge e_1 and face *f*.

3.2.1 Oriented Generalized Maps: An *oriented Generalized Map of dimension d (d-G-Map)* (Lienhardt, 1994) represents a cellular complex that is used as a discrete model of the topology of an orientable manifold of dimension d. It consists of a set of *darts*, d+1 transformations of the set of darts, $\alpha_i,\ i = 0 \ldots, d$, that are *involutions* verifying $\alpha_i(\alpha_i(x)) = x$ (**fig. 2**). The involutions must further verify the condition that $\alpha_i(\alpha_{i+2+k}())$ is an involution for $k >= 0$. Subsets of darts that can be reached from a starting dart x_0 by any combination of involutions $\alpha_i \ldots \alpha_i$ are called *orbits*. We note them $orbit^d(i,\ldots,j,x_0)$ or $orbit^d_{i\ldots j}(x_0)$, where *d* is the dimension of the G-Map, the indexes *i, …, j* are a subset of $\{0,\ldots,d\}$, and x_0 is the starting dart. Certain orbits, namely those of the form $orbit^d(\ldots{\sim}k\ldots)$, that use all involutions except α_k, determine the *k*-dimensional cells of the cellular complex, i.e. nodes, edges, faces, and solids for dimension $d=3$ (**fig. 2**). In a d-G-Map, $orbit_{0\ldots d}(x_0)$ returns the connected component containing x_0. Orbits $orbit_i()$, and by Lienhardt's condition, orbits of the form $orbit_{i,i+2+k}()$ have a fixed length. Other orbits can be implemented by single or nested programming loops, a small number of orbits however, are more complicated – they can be implemented recursively, returning the subset of cell-tuples as a collection of connected sequences possibly interrupted by discontinuities. For some topological operations, especially the solid split operation, we need *continuous loops* that generally are defined by the user, and not produced by an orbit. Different from linear iterators, orbits and loops are examples of *circulators* (Fabri et al., 1998) that can begin at any object in the circular sequence, and advance until the starting point is again encountered.

3.2.2 Realisation by means of an ORDBMS: Whereas G-Maps can be implemented focusing on the involution transitions represented e.g. as references between anonymous darts, we prefer the relational realisation to focus on the darts, which are represented by signed *d-cell-tuples* $(c_0, \ldots, c_d, +/-, \ldots)$ (**fig. 2**), cf. (Brisson, 1993), collected in the tables of an ORDBMS. The c_i are identifiers of cells of dimension i, i.e. nodes, edges, faces, solids. The identifiers of the *neighbour cells* c_inv_j, $0 \quad j \quad d$, are also attached to the cell-tuples. The involutions α_j are implemented as "switch" operations that transform the *cell-tuple key* $(c_0, \ldots, c_j, \ldots, c_d)$ into $(c_0, \ldots, c\_inv_j, \ldots, c_d)$, exchanging c_j and c_inv_j and then retrieve the corresponding cell-tuple record from the database.

Orbits and loops. By definition, an orbit $orbit_{i..j..}(ct_0)$ consists of the subset of cell-tuples that can be reached from ct_0 using any combination of $_{i,\ldots,}$ $_{j,\ldots}$ The components of dimension k where k is not contained in the set of indices i,…,j remain fixed, e.g. if $ct_0=(n,e,f,s)$, then $orbit_{012}(ct_0)$ leaves solid s fixed, and returns all cell-tuples of the form (*,*,*,s).
The implementation of the darts of a G-Map as cell-tuples in a relational DBMS is straightforward, the involutions can be implemented using queries or joins, supported by foreign keys and indexes, and iterators can be realised as database cursors, but a normal relational DBMS does not provide the equivalent of circulators, i.e. closed loops of undetermined, albeit finite, length. The representation of orbits therefore needs additional code controlling repeated database queries. As such implementations are not very efficient, we try to replace orbits by subset queries, wherever the circular arrangement is dispensable.
The trivial orbits of the form $orbit_i()$ can be treated like the corresponding involutions, and by Lienhardt's condition, orbits of the form $orbit_{i,i+2+k}()$ have a constant length of four and can be modelled by a limited number of queries or join operations. Whereas RDBMS do not support cyclic cursors that would correspond to circulators, result sets of queries can be ordered, e.g. the query:

```
SELECT * FROM celltuples
WHERE <condition>
ORDER BY face, edge, sign;
```

returns the retrieved cell-tuples ordered according to faces, in ordered pairs corresponding to the edges of the face boundary, although not in a cyclic arrangement. In some application cases, this may be sufficient. Whenever the orbit arrangement must be reproduced exactly, however, a true orbit can be implemented by stepwise executing the involution operations:

Start with
  node n_0, *edge* e_0, *face* f_0, *sign* sg_0, n_inv_0, e_inv_0, f_inv_0;
i=0;
repeat {
  ++i;
  update j; / j: selector of the next involution* α_j **/*
  case j {
    0: SELECT node as n_i, edge as e_i, face as f_i, n_inv_i ...
      FROM celltuples
      WHERE n_i = n_inv_{i-1} AND e_i= e_{i-1} AND f_i= f_{i-1}
    1: SELECT node as n_i, edge as e_i, face as f_i, e_inv_i...
      FROM celltuples
      WHERE n_i = n_{i-1} AND e_i= e_inv_{i-1} AND f_i= f_{i-1}
    2: SELECT node as n_i, edge as e_i, face as f_i, f_inv_i ...
      FROM celltuples
      WHERE n_i = n_{i-1} AND e_i= e_{i-1} AND f_i= f_inv_{i-1} *}*
} until $n_i= n_0$ *and* $e_i=e_0$ *and* $f_i= f_0$;

We use a *selector* variable to determine the next transition step. This procedure can be modified to implement any closed loops in the G-Map, by attaching to the cell-tuples a selector variable the current value of which controls the choice of the next α_i transition.

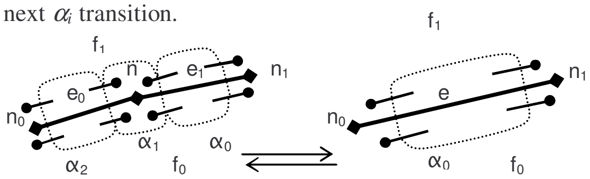

Figure 3. Merging two edges e_0, e_1 that separate faces f_0, f_1, by deletion of a node n.

3.2.3 Realisation of simple generalisation operations: At the present stage, we concentrate on basic split and merge operations, which serve to build more complex aggregation operations in 2D and 3D.

Merging two edges. The merging of two edges, i.e. 1-cells, by removal of an intermediate node is straightforward: consider a sequence $n_1\ e_1\ n\ e_2\ n_2$ consisting of nodes n_i and edges e_j. We wish to replace $e_1\ n\ e_2$ by a new edge e, hence we delete all cell-tuples $(n,\ldots)$ having n as node component, and in all cell-tuples containing e_1 or e_2 as edge component, we replace e_1 and e_2 by e. Then, we update all cell-tuples related to $(n_1,e_1,\ldots)$ or $(n_2,e_2,\ldots)$ by $_1$ involutions. If node n and edges e_1, e_2 are not used elsewhere, we delete them as well (**fig. 3**). A necessary condition for the edge merge operation to be applicable is that there are only two edges incident with node n. This can be checked counting the length of an $orbit_{12}((n,e_1,f_1))$, or by counting the number of darts returned by a corresponding SQL query. In the following, we tacitly assume that whenever a sequence of edges without branches that separates two faces is to be submitted to a merge operation, it is first transformed into a single edge by a succession of edge merges.

Merging two faces. Let us consider the following situation:
Two faces f_1 and f_2 are separated by one edge e between nodes n_1 and n_2. By removing e, f_1 and f_2 are merged into one face f (**fig. 4**). Again, we first remove all cell-tuples containing edge e. Then in all cell-tuples containing f_1 or f_2, we replace these by f. Next, we replace f_1, f_2 by f in all cell-tuples relating f_1, f_2 by

$_2$ involutions, and "repair" the involutions at nodes $n1$ and n_2 replacing sequences of the form
$(n_i,e_x,f)\ \alpha_1\ (n_i,e,f)\ \alpha_2\ (n_i,e,f)\ \alpha_1\ (n_i,e_y,f)$
by $(n_i,e_x,f)\ \alpha_1\ (n_i,e_y,f)$ (**fig. 4**).

The face merge operation can be applied if none of the faces separated by e belongs to the outside ("universe") of the G-Map. Otherwise, it has to be verified that the operation doesn't produce a "bridge" configuration – a single edge incident on both sides to the outside, linking two connected parts of the G-Map. Though bridge configurations could be modelled in 2D using the orientation of the cell-tuples, we exclude them because they do not fit well with our definition of an involution as exchange of two distinct k-cells.

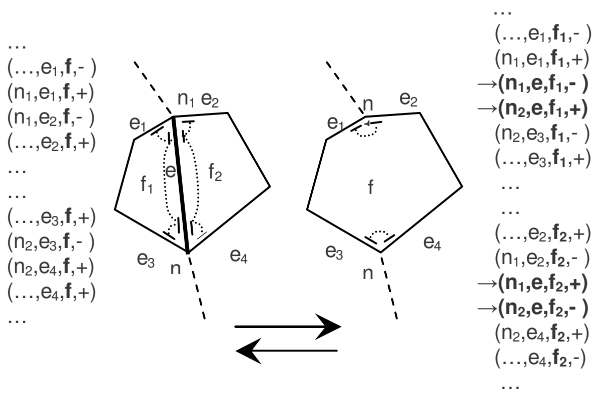

Figure 4. Merging two faces f_1, f_2 by removing edge e.

Splitting a solid. The inverse operations are decomposed analogously, exchanging the roles of insert and delete operations. Let us discuss the splitting of a solid s by the insertion of a separating face f, into two solids s_1, s_2 (**fig. 5**).
Besides set operations, splitting a 3d-cell requires the use of an $orbit_{012}()$. We start with the definition of a closed connected sequence of nodes and edges that define the contact – the seam – between the circumference of the face and the meshing of the inner surface of the solid. This can be done using a sequence of cell-tuples connected by $_0$- $_1$-, and $_2$-involutions forming a closed loop. This seam location has to be defined by the user or by a client program, and the number of its nodes and edges must coincide with that of the boundary of f. The operation then consists of the following steps:
First, insert face f, and solids s_1 and s_2. Next, for each pair of cell-tuples situated on either side of the seam location, replace
$(n_i,e_j,f_k,s,+)\ _2\ (n_i,e_j,f,s,-)$
by a sequence
$(n_i,e_j,f_k,s_1,+)\ _2\ (n_i,e_j,f,s_1,-)\ _3\ (n_i,e_j,f,s_2,+)\ _2\ (n_i,ej,f_l,s_2,-)$.
Finally, starting from a cell-tuple $ct_0(n_i,e_j,f,s_1,+)$, use an $orbit_{012}(ct_0)$ to replace s by s_1 on every cell-tuple encountered, and all cell-tuples related by $_3$-involutions. By the use of an $orbit_{012}()$, we assure that all cell-tuples $ct(.,.,.,s)$ selected for update are situated on the boundary of solid s and on one side of face f, independent of the value of the solid component. Next we repeat the same procedure starting with $(n_i,e_j,f,s_2,-)$, replacing s by s_2 on the other side of face f.
Obviously, such sequences can be implemented using the insert, delete and update operations of a relational database within a transaction. For the solid merge to be applicable, we have to check that there is no other contact between s_1 and s_2, and that none of the solids s_1 and s_2 is part of the outside of the G-Map. Otherwise, we have to check that no 3D-bridge configurations result, i.e. a single face incident on both sides with the same solid, or with the outside. The latter configurations can be avoided by first ensuring that none of the other neighbouring cells are part of the outside.

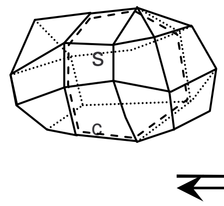

Figure 5. splitting a 3D solid s by the insertion of a 2D face f. The location of the seam is defined by the loop c.

A non-Euler operation. Geo-data from external sources cannot be expected to carry an explicit representation for their topology ready for representation as a G-Map. Rather, one of the first steps of the import of geo-data consists in extracting topological relationships that are implicit within the data. As an example, consider a land use map encoded as a *shapefile*: each parcel is defined by one or several polygons, that are not linked to each other, so that topologically each parcel is an island disconnected from the rest. In this particular case the vertex co-ordinates, however, of neighbouring polygons match exactly, so that it is possible to reconstruct the neighbourhood relationships between parcel boundaries by matching vertex co-ordinates. In the general case, we have to modify the geometrical matching criterion such as to accommodate small numerical fluctuations, e.g. resulting from digitisation.
We introduce the newly gained information into the G-Map by *sewing* corresponding cell-tuples, i.e. by establishing the α_i involution links. This operation starts with the merging of a pair of nodes from two neighbouring polygons. It is not an Euler operation, as the number of nodes is reduced by one, whereas edges and faces remain unchanged. The resulting configuration of two polygons having one point in common is theoretically admissible, but it poses practical problems, therefore we require it to be immediately followed by the merging of a second pair of nodes, and of the two edges joining the nodes to be merged. This second sewing operation, and any others following without interruption on the same boundaries, do not affect the Euler-Poincaré characteristic.

Integrity constraints. Whereas basic split operations do not affect the consistency of the G-Map, merging of cells may lead to singular and inconsistent configurations. As an example, consider a map of land use, comprising a number of parcels of identical land use A that surround one or more parcels of land use B (**fig. 6**). A complex merge operation that aggregates all cells of type A eventually results in a ring, which is multiply connected and hence is not consistent with the definition of a cell in a cellular complex. Another consequence is that the cells of type B could never be reached by an orbit starting from the outer boundary of a type A cell. We must therefore detect these configurations and stop the merging process such as to conserve two cells of type A separated by two bridging edges (**fig. 6**). The occurrence of a bridge configuration during a face merge operation is not detected by a change in the Euler characteristic of the G-Map subset defined by the class A.

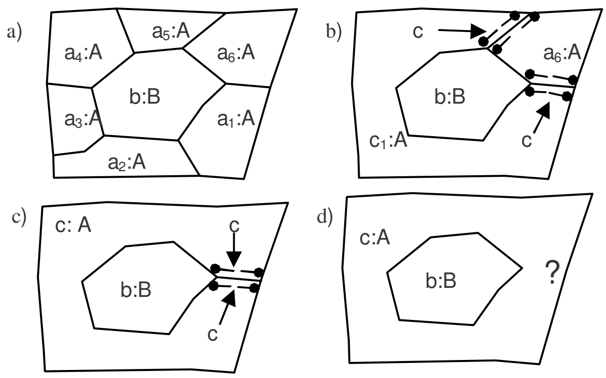

Figure 6. (a) A face b of class B is completely surrounded by faces a_i of a different class A. (b) Stepwise merging all cells of class A results in a bridge configuration (c) and finally in a ring-shaped cell (d).

It can be detected before the merge operation by verifying that the boundary between the cells to be merged is simply connected, or after the operation by searching for α_2-transitions that link two cell-tuples having the same face:

```
SELECT count(*)
   FROM celltuples
   WHERE <condition>  AND face_id= face_inv;
```

Any result different from 0 indicates an error. If the bridge configuration is detected after a merge operation, it can be corrected either using DBMS transition rollback, or by performing the inverse edge split operation.
The transition to a ring configuration (**fig. 6d**) can be detected by a change of the Euler characteristic N-E+F, where N, E, F are the numbers of nodes, edges and faces respectively. In fact, deleting the last bridging edge doesn't change N or F, but reduces E by one. Though irregular configurations can be avoided during a merge after classification, it is an inconvenience that in some cases contiguous cells of the same class nevertheless must be kept separate. To handle the connected components of a partition, (Fradin et al. 2002) use boolean flags to distinguish those α_i transitions that join cells of the same class, from α_i transitions that link different classes. Moreover, they implement multiple partitions using an array of flag bits associated with the α_i transitions, supporting several different classifications on the cells of the same basic G-Map.
Since their G-Map implementation is based on α_i transitions of darts, rather than on explicitly modelled cell-tuples, we cannot use this approach without modification.

It is possible, however, to adapt this feature to our cell-tuple-based representation by associating the flag array with the cell-tuple variables node_inv, edge_inv, etc. that define the α_i transitions, or simply use queries like the following one:

```
UPDATE celltuples ct1
SET face_class_flag= TRUE
WHERE  EXISTS(
  SELECT * FROM celltuples ct2
  WHERE ct2.face_id= ct1.face_inv
  AND ct2.face.class =  ct1.face.class );
```

Then, iterators can be derived from the α_i orbits that yield all cell-tuples associated with a given class A, its associated flag values indicating a position at a class boundary or in the interior. Using a nested query or a join, the corresponding relational query can directly return all cell-tuples belonging to a given class A, together with the associated flag for further processing (e.g. for skipping interior cell-tuples).

3.2.4 Realisation of complex generalisation operations: A Multi-Resolution Database (*MRDB*) of land use (Haunert & Sester, 2005) consists of a stack of maps at different scale and LOD, and a hierarchy of partitions of the map of highest LOD. In this example, the maps are encoded as *shapefiles*, and the aggregation hierarchy is represented by n : 1-relations between successive LODs that are stored in a table.
To establish the topological properties of the MRDB, at each LOD first the isolated polygons are sewed to form a partition of part of the map plane. Then, the classification induced by the lower LOD B and the aggregation table is used to "paint" the faces of the map at higher LOD A. The resulting partition of A can be used to introduce flags distinguishing inter-class α_i transitions from intra-class transitions.
In a next step, in order to reduce the amount of data and to establish a more detailed relationship between successive LODs, neighbouring faces of A that belong to the same class are merged wherever this is possible without violating the integrity of the G-Map. The result of this operation is an aggregated G-Map A' of A that, with a number of exceptions, corresponds to the G-Map B. At this stage, the user may intervene and modify the aggregation table in order to reduce the number of problematic configurations.
Let us now extend the hierarchical relationship between faces of A and B, and to establish relationships between nodes, edges and faces of A' and B respectively. As generalisation may have involved displacements, a simple comparison of co-ordinates is not a sufficient matching criterion. Instead we use the G-Map to find corresponding nodes in A' and B by comparing the configuration of their neighbourhoods. E.g. if a connected set of class C in the G-Map of A' corresponds to a face f in B, we can search for nodes on the boundaries of C and f that have similar neighbours. As A' has been developed from A by aggregation of cells, the nodes, and cell-tuples of A' correspond to a subset of those of A. Thus a finer correspondence between the topologies of A and B is established, than the initial aggregation hierarchy.

4. AN APPLICATION EXAMPLE

The Hannover Institute of Cartography (IKG) is investigating methods that generalise land use maps by an automatic aggregation of parcels using thematic and/or geometric criteria (Haunert & Sester, 2005). The resulting hierarchies of maps at different LOD are stored in a MRDB (Anders & Bobrich, 2004). The n:1 relationships between polygonal faces between different scales are represented in tabular form (**fig. 7**).

From a set of separate maps at different scales imported into PostGIS/PostgreSQL, we derive corresponding G-Maps. Topological consistency is checked and the Euler characteristic and some basic statistics are established. The n:1 relationship between maps at different LOD induces a classification of the cells of greater scale. Using the elementary merge operations described above, groups of cells of the same class are aggregated either until a 1:1 correspondence is established, or until inconsistent configurations are detected. Thereafter, unnecessary nodes on the boundaries of the aggregated cells are eliminated while edges are merged. If no premature stop has been encountered, the 1:1 relationship between faces and aggregated cells is used to determine the relationships between edges, nodes, and in consequence cell-tuples. The resulting hierarchical G-Map represents the interrelations between the topologies at different LOD.

Figure 7. Application example by courtesy of J. Haunert, IKG Hannover University: a section of ca. 2 % of a digital map on land-use at three different scales.

5. CONCLUSION AND OUTLOOK

In this article the realisation of a general model based on oriented hierarchical d-Generalized Maps to represent and analyse topology in MRDB has been described in detail. The model can be used as a data integration platform for 2D, 3D, and 4D topology. Typical examples for elementary and complex topological operations for multiple representations have been presented and illustrated. An application example with 2D cartographic datasets from Hannover University showed the feasibility of the new approach. It can also be used to combine 2D maps and 3D models, the last-mentioned being the specialisation of the 2D map. The advantage of this approach is to have a single representation for describing 2D and 3D topology. In our future work we intend to focus on this aspect, e.g. in the context of 3D urban planning.

REFERENCES

Anders, K.-H., Bobrich, J., 2004. MRDB Approach for Automatic Incremental Update. In: *ICA Workshop on Generalisation and Multiple Representation,* Leicester.

Brisson, E., 1993. Representing Geometric Structures in d Dimensions: Topology and Order. In: *Discrete & Computational Geometry (9),* pp. 387-426.

Gröger, G., Plümer, L., 2005. How to Get 3-D for the Price of 2-D-Topology and Consistency of 3-D Urban GIS. *Geoinformatica, 9 (2)*, pp. 139-158.

Fabri, A., Giezeman, G.-J., Kettner, L., Schirra, S., Schönherr, S., 1998. On the design of CGAl, the Computational Geometry Algorithms Library. *Research Report MPI-I-98-1-007*, Max-Planck-Institut für Informatik, Saarbrücken.

Fradin, D., Meneveaux, D., Lienhardt P., 2002. Partition de l'espace et hiérarchie de cartes généralisées. In : *AFIG 2002,* Lyon, décembre 2002, 12p.

Haunert, J.-H., Sester, M., 2005. Propagating updates between linked datasets of different scales. In: Proceedings *XXII Int. Cartographic Conference,* A Coruna, Spain July 11-16.

Hoppe, H., 1996. Progressive meshes. In: *ACM SIGGRAPH* 1996, pp. 99-108

Lévy, B., 1999: *Topologie Algorithmique - Combinatoire et Plongement.* PhD Thesis, INPL Nancy, 202p.

Lienhardt, P., 1994. Topological models for boundary representation: a comparison with n-dimensional generalized maps. In: *Computer Aided Design 23(1),* pp. 59-82.

Mallet, J. L., 2002. *Geomodelling.* Oxford University Press, 599 p.

Mallet, J.L., 1992. GOCAD: A computer aided design programme for geological applications. In: Turner, A.K. (Ed.): *Three-Dimensional Modelling with Geoscientific Information Systems,* NATO ASI 354, Kluwer Academic Publishers, Dordrecht, pp. 123-142.

Mäntylä M., 1988. *An Introduction to Solid Modelling.* Computer Science Press, 401 p.

Meine, H., Köthe, U., 2005. The GeoMap: A Unified Representation for Topology and Geometry. in: Brun, L., Vento, M. (Eds.): *Graph-Based Representations in Pattern Recognition*, Proc. GbR 2005, LNCS 3434, pp. 132-141, Springer, Berlin.

MOKA, 2006. Modeleur de Cartes. http://www.sic.sp2mi.univ-poitiers.fr/moka/ (accessed 21.03.2007).

PostGIS.org, 2006. http://postgis.refractions.net/documentation (accessed 21.03.2007).

PostgreSQL.org (2006): http://www.postgresql.org/docs (accessed 21.03.2007).

Shumilov, S., Thomsen, A., Cremers, A.B., Koos B., 2002. Management and visualisation of large, complex and time-dependent 3D objects in distributed GIS, In: Proc. *ACM-GIS 2002*, pp. 113-118.

Thomsen, A., Breunig, M., 2007. Some remarks to topological abstraction in multi representation databases. *Proc. Int. Workshop on Information Fusion and Geographical Information Systems IF&GIS´07,* St. Petersburg, 12p. (in print).

ACKNOWLEDGEMENTS

This work is funded by the German Research Foundation (DFG) in the project "MAT" within the DFG joint project "Abstraction of Geoinformation", grant no. BR 2128/6-1.