Rule-based topology system for spatial databases to validate complex geographic datasets

J. Martinez-Llario[a], E. Coll[a], M. Núñez-Andrés[b], C. Femenia-Ribera[a]

[a] *Department of Cartographic Engineering, Geodesy and Photogrammetry, Universitat Politècnica de València, Camino de Vera s/n, Valencia 46022, Spain*

[b] *Departamento de ingeniería del terreno, cartográfica y geofísica, Universitat Politécnica de Catalunya, Barcelona, Spain*

Abstract

A rule-based topology software system providing a highly flexible and fast procedure to enforce integrity in spatial relationships among datasets is presented. This improved topology rule system is built over the spatial extension Jaspa. Both projects are open source, freely available software developed by the corresponding author of this paper.

Currently, there is no spatial DBMS that implements a rule-based topology engine (considering that the topology rules are designed and performed in the spatial backend). If the topology rules are applied in the frontend (as in many GIS desktop programs), ArcGIS is the most advanced solution. The system presented in this paper has several major advantages over the ArcGIS approach: it can be extended with new topology rules, it has a much wider set of rules, and it can mix feature attributes with topology rules as filters. In addition, the topology rule system can work with various DBMSs, including PostgreSQL, H2 or Oracle, and the logic is performed in the spatial backend.

The proposed topology system allows users to check the complex spatial relationships among features (from one or several spatial layers) that require some complex cartographic datasets, such as the data specifications proposed by INSPIRE in Europe and the Land Administration Domain Model (LADM) for Cadastral data.

1. Introduction

In recent years, certain regulations, including the INSPIRE data specification (INSPIRE, 2010) and its subsequent regulations, or the Land Administration Domain Model (LADM) – ISO 19152 (ISO, 2012) are forcing GIS datasets to be highly structured to consider the spatial relationships among features (geographic objects).

Before structuring the datasets using these models, it is necessary to perform a quality control check to assure the spatial relationships. One of the ways to accomplish this task is to use topology rules to model all of the spatial relationships. We will then be able to test and ultimately edit and fix the geometries to build these complex datasets in a non- traumatic way.

Topology is a branch of mathematics that studies the relative positions among geometric features, particularly focusing on the spatial relation- ships that are maintained when the embedding space is altered with topological transformations, such as translation, rotation and scaling. The GIS community has long understood that topology can help manage spatial data; hence, how to efficiently apply its principles has been a largely discussed issue in the GIS world and is not yet definitely solved.

In this section, we overview its main topics. The second section describes the different topology software approaches. The created topology rule system is presented in the third section. Section four studies a real cartographic model case.

1.1. ***GIS basic topological structures***

Vector-based GIS models spatial features from the real world by means of geometry primitives such as points, lines and polygons. To accomplish this, several abstractions or simplifications must be performed, and consequently a series of conventional rules must be satisfied to coherently characterize the geographic reality. Standardization has been already achieved with OGC Simple feature access (SFA)—Part 1: Common architecture (OGC, 2011), providing a widely adapted way to describe geometry (Yan et al., 2011). It repeatedly relies on topological concepts to define geometry representation, but it not addresses how to store topologically structured data. It was published as the ISO 19125-1 Standard (ISO, 2004). Each geometry is independently described by a set of coordinates (Worboys and Bofakos, 1993) and sharing geometries between features (geographic objects) is not precluded.

Conversely, the arc-node model is the paradigmatic topological structure for GIS topology. In this model, the primitives are nodes, arcs and faces, all of which are interrelated. As a result, in the arc-node model, relationships such as connectivity, direction, and adjacency are explicitly stored (Murray, 2009). This is a very old model, but the still recent ISO 13249-3 or SQL/MM (ISO, 2016) represents its evolution. The SQL/MM standard defines a way of modelling and accessing a topological dataset and allows multiple levels of topological connectivity.

Both models (ISO 19125-1 or ISO 13249-3) present advantages and disadvantages, so the usage of each model depends on the task at hand. The main disadvantage of ISO 19125-1 is the repeated storage of features, but its structure is less complex and can be seamlessly used for display tasks. The main advantages of explicit storage of the topology structure model (ISO 13249-3) are the avoidance of redundant data storage and the ability to maintain data consistency by performing validation rules (De Hoop and van Oosterom, 1992).

The standard ISO 19125-1 is also known as the OGC Simple Feature Access (OGC-SFA) (OGC, 2011) Model. In addition, the standard OGC-SFA has a more recent version than ISO 19125-1.

1.2. ***Topology rules***

A rule-based topology model consists of a set of spatial rules that constrain the spatial relationships

among the features. The GIS user is able to choose which relationships are relevant for the data. After validating the rules, the GIS user handles the cases where the rules are violated and fixes the errors with the aid of topological editing tools. When a geometry that participates in some rule is added, deleted or updated, the spatial extent affected is flagged as a "dirty area" to note that this area should checked again (ESRI, 2003).

With a rule-based topology engine, it is possible to model the spatial relationships among datasets easily. A few real cases of some topology rules area as follows:

- Buildings (polygons) from a dataset must be inside cadastral parcels (polygons)
- Lakes (polygons) and land parcels (polygons) from two different layers must not overlap
- Vegetation (polygons) and soils (polygons) must cover each other
- Endpoints of electric lines (lines) must be capped by transformers (points)

A rule-based topology model can be built over both basic GIS structures: a simple feature model (ISO 19125-1) or an explicit topological model (ISO 13249-3).

If the rule-based topological model is built over an explicit topological model, then some spatial relationships are automatically maintained, and there is no need to design some topology rules: the detection of closed polygons, overlapped areas, dangles, undershoots, points inside polygons (Baars et al., 2004).

2. Topology software background

GIS software packages have worked with topology differently. In the present section, we offer a brief overview of how the theoretical aspects of planar spatial data topology have been transferred to GIS software. Table 1 shows the main characteristics related to the geometry structure and the rule-based topology system used by the most important GIS desktop and spatial databases.

To be able to justly compare all software, we consider that the ArcNode or SQL/MM systems implement some topology rules in an inherent way, although the number of topology rules is much lower (few or very few) than in a full rule-based topology system.

The rule-based topology system column has the following meaning:

- The software labelled "very basic and basic" shows that although they do not implement a pure rule-based topology system, they have an explicit topology system that allows certain SQL attribute queries to be built to check spatial relationships between features as stated before. If the software follows the SQL/MM standard, these SQL queries can be extended to use some relationships between layers because SQL/MM allows multiple levels of topological connectivity.

- The label “medium” shows that even if the software implements some type of topology rule system, the number of topology rules is restricted, and a dirty area system is especially not supported; therefore, every time a topology rule is validated, all features must be checked, which makes the system very inoperative.

- The only two software programs that support a full rule-based topology system (Section 3.5) are labelled as “advanced”.

2.1. ***Desktop GIS***

As mentioned before, ArcInfo coverages (arc-node model) maintain topologically structured data. However, some disadvantages including slowness when assembling features on the fly and constant validation after editing have pushed ESRI to implement another approach to topology to be applied on a simple feature model.

ArcGIS follows a rule-based topology model in which topology is handled via a set of rules that are applied to the feature classes that form the geodatabase.

The open GIS world has in GRASS its major counterpart. GRASS has usually defined itself as a topological GIS (Neteler et al., 2012). Its native format is based on arc-node representation. When importing spatial data stored in a format without topology, it is able to automatically build topology, although it also allows work with spaghetti data if the size of the dataset affects the system.

GvSIG and Kosmo have both implemented topology extensions with graphical user interfaces that provide a range of functionalities to clean and validate data. Like ArcGIS, they are based on a rule-based topology model. In both cases the JTS library (Erickson and County, 2009) provides the necessary algorithms to perform the basic geometry calculations.

QGIS follows the same approach as gvSIG or Kosmo, but the geometry library is not JTS based (Java Topology Suite) but OGR (OpenGIS simple features Reference implementation).

2.2. ***Moving on to DBMS***

As shown before, GIS vendors have conventionally lead the research of topology implementation for spatial data. Nevertheless, van Oosterom et al. (2002) argued why topology management should be carried out within the database: topology is a general and reusable aspect of the data, and thus it should be maintained at the database layer. Different reasons encourage the usage of databases instead of files systems: transactional integrity, multiple users, unified storage and solid SQL standards are the most prominent advantages.

Owing to the spread trend of maintaining spatial data in RDBMS, these products have started to support topology. Oracle Spatial 10 g pioneered a two-dimensional topological architecture in a DBMS environment, supporting primitive topologies including nodes, arcs and faces stored in persistent topology tables.

Radius topology developed by 1Spatial extends the Oracle Spatial

Database to manage topological structured data (Ellul, 2008). From Oracle 10 g, Radius topology was migrated to use the Oracle SQL/MM

Table 1. Topology model implemented by the main GIS desktop and spatial database software.

	Geometry structure[a]	Rule-based topology system[b]	Dirty areas[c]	Cluster tolerance[d]	Custom topology rules[e]	Topology rules number[f]	Spatial backend logic[g]
Arc/Info	ArcNode	Very basic	Y	Y	N	Very few	Y
ArcGIS 10.2	Very similar to SFA	Advanced	Y	Y	N	32	Y (ArcSDE)
Grass 7	OGC-SFA	Very basic	Y	Y	N	Very few	N
Kosmo 3	OGC-SFA	Medium	N	N	N	< 20	N
gvSIG 1.12*	OGC-SFA	Medium	N	N	N	< 20	N
QGIS 2.14	OGC-SFA	Medium	N	N	N	< 15	N
PostGIS 2.2	OGC-SFA	Basic	N	Y	N	few	Y
Oracle Spatial 11 g	OGC-SFA	Basic	N	Y	N	few	Y
Radius Topology (discontinued)	SQL/MM over spatial 10 g	Basic	Y	Y	N	few	Y
Developed System	OGC-SFA over Jaspa	Advanced	Y	Y	Y	67	Y

* The rule-based topology extension does not support gvSIG 1.12 newer versions.

[a] The vector model implemented by the software (Section 1.1).

[b] We consider a basic or very basic system to the software without an explicit topology rule system (an extended explanation is showed in this section).

[c] The software has a system to update the topology rules only in zones with updated or new features.

[d] The software is able to use snap rounding algorithms in order to adjust the vertices among different layers to improve the spatial analysis robustness.

[e] An advanced user can create new topology rules.

[f] The number of topology rules that implement the software (few and very few are used to mention the software without an explicit topology rule system thus the number of topology rules is very limited and unknown).

[g] The topology rules are developed in the spatial backend (with spatial stored procedures or other mechanism). These software ensure greater reliability and robustness in the integrityof the topology rules.

topology storage model and thus currently is mainly a graphical Oracle Spatial SQL/MM frontend, which allows features to be edited and fixed graphically.

On the open source side, PostGIS has begun to develop a topology schema model and accessory functions for topology editing. Once the schema are created, the user is able to store topological structured features such as nodes, edges and faces according to the generic topology model, define objects composed by those elements (PostGIS type “TopoGeometry”) and convert them to simple geometry objects. Creation, editing and validation is performed by SQL sentences (Picavent, 2010).

2.3. ***Rule-based topology***

A topologically structured database maintained in an RDBMS provides several benefits but also raises new problems. The positive consequences are that it ensures data integrity, the elimination of redundancy, consistent editing and updating or advanced querying (van Oosterom et al., 2002). However, topological structures pose some problems. Although there is no redundancy, Penninga (2004) claims that topological structures require almost five times more storage in worst case scenarios, largely due

to the need for indexes. Topological structures determine workflows, and the systems to ensure topology are fixed. There is little room to adapt it to the user's needs.

Topological structures also force higher complexity of updates because the topology must be rebuilt. In addition, the topological model must be capable of maintaining integrity across transactional boundaries, which can be problematic when exporting a huge volume of data into a topological model. Depending on the size and complexity of the dataset, the assembly of the spatial data from the topology primitives turns into a time-consuming task because it is far more complex than the geometric structure. Overall, a powerful system in terms of both software and hardware is required to support a topological structure.

Query performance varies depending on the type of query. The access to features is slower in the topology structure because they must be assembled. Conversely, queries involving relationships are faster when performed with topological structures. Penninga (2004) claimed that the larger the dataset, the better the relative topological query performance.

The rule-based topology model may be considered a bridge between the geometrical (ISO 19125-1) and the explicit topological models (ISO 13249-3). It cleans the geometrical data by means of enforcing topology rules between layers. The result is not topologically structured data according to ISO 13249-3 but topologically correct data. In this case, the user chooses or creates a set of rules to handle the topology. These rules may be based on the attributes of the features or the cluster tolerance.

Baars et al. (2004) notes the main advantages and disadvantages of a topology rule system: flexibility when choosing constraints and error handling, quick validation processes and overall ease of usage. However, there are also negative aspects, such as data redundancy because of working with spaghetti data, lack of automatic correction of detected errors and, similarly to topological structure data, impossibility of extending the set of rules provided by the software.

3. Topology rule system developed

We have chosen to develop a topology rule-based system rather than a topologically structured data system because of the much better performance and the ease of use for end users.

Neither PostGIS nor Oracle Spatial or any other spatial DBMS currently implements a rule-based topology system. For GIS desktop applications, the most advanced software is ArcGIS, which has a topology rule-based system but it cannot be extended, has little variety of rules and cannot mix the feature attributes with the topology rules as a filter.

To implement an improved topology rule system, we have chosen the spatial extension Jaspa (Martinez-Llario and Gonzalez-Alcaide, 2011). Jaspa is a novel spatial extension for relational database management systems. It is an open source project developed by the authors of this paper, and the programming system is totally known by them. In addition, Jaspa is easy to extend thanks to the integrated structured developed in Java using stored procedures (Martinez-Llario and Gonzalez-Alcaide, 2011). Jaspa works with multiple RDBMSs including H2, PostgreSQL and Oracle, which makes it an

attractive option.

We have implemented in Jaspa a topology system based on rules, which is similar to the ArcGIS approach (ESRI, 2003) or the conceptual design of spatial constraints in SQL proposed in (Pelagatti et al., 2009). The enforcement of the topological rules between geometries is based on binary predicates, provided by the JTS library (Erickson and County, 2009) that implements the Dimensionally Extended Intersection Model (DE-9IM) (Clementini and Di Felice, 1995). The scope of Jaspa topology is planar topological structures.

Topological rules in Jaspa can be applied to a subselection of a layer rather than the full layer, to a single layer or between two layers. As claimed by Brown et al. (2005), topological rules impose conditions on the relationships between spatial layers to characterize the real world and prevent placement incoherencies and guarantee spatial consistency.

The rule-based topology engine has been programmed in Java and SQL. Topologic geometries are neither stored nor created on the fly. Jaspa topology can be perceived as a whole data model that relies on three key features: topology rules, cluster tolerance and dirty area management.

The topology rules can combine spatial and alphanumeric criteria without any restriction, and it is possible to validate the selected rules rather than the entire topology as in other systems.

The system allows the GIS user to define their own topology rules by using templates and SQL scripts. Currently, this task is only for advanced users because it requires some Java coding.

The large set of topology rules and the possibility to design new rules are crucial to validate and check the integrity of complex geographic database datasets.

3.1. ***Cluster tolerance***

The tolerance engine controls the distance at which the vertices or edges that make up the geometries are considered identical. The tolerance is set according to the accuracy used to store the coordinates. The engine snaps the vertices and segments between features from the layers that participate in the topology. The Jaspa function ST_Snap is its first approach.

This process adjusts the cartography vertices and avoids many of the problems that plague the current GIS software related to perform- ing vector-based spatial analysis (Belussi et al., 2015).

The ST_Snap algorithm follows the well-known family of snap rounding algorithms like (Halperin and Packer, 2002). Fig. 1 shows how the function ST_Snap (A, B) will snap the geometry A to the geometry B by moving the A vertices:

1. First, if a vertex of 'A' geometry is within the tolerance of a vertex of ‘B’ geometry, then this vertex from 'A' will be moved to the closest vertex of B. Vertex A1 will move to B2.

2. Second, if a segment of 'A' geometry is within the tolerance of a vertex of ‘B’ geometry, then this

segment will be split, and the newly added vertices from 'A' will be moved to the closest vertices of B. Segment A14 will be split, and the new vertex will move to B3.

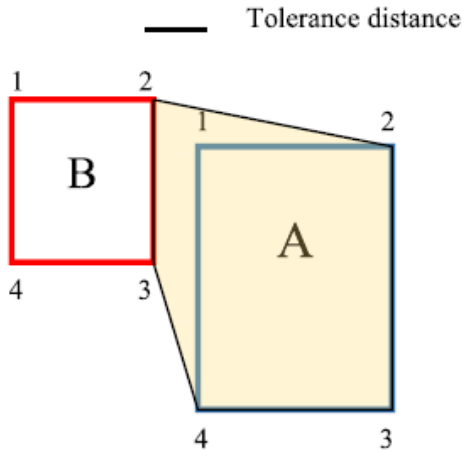

Fig. 1. Snapping two geometries with ST_Snap.

The full process is a recursive task that snaps all updated or new geometries. Fig. 2 shows the algorithm flowchart. This task is performed before any topology rule is validated.

The system of cluster tolerance is optional and must be explicitly specified by the GIS user.

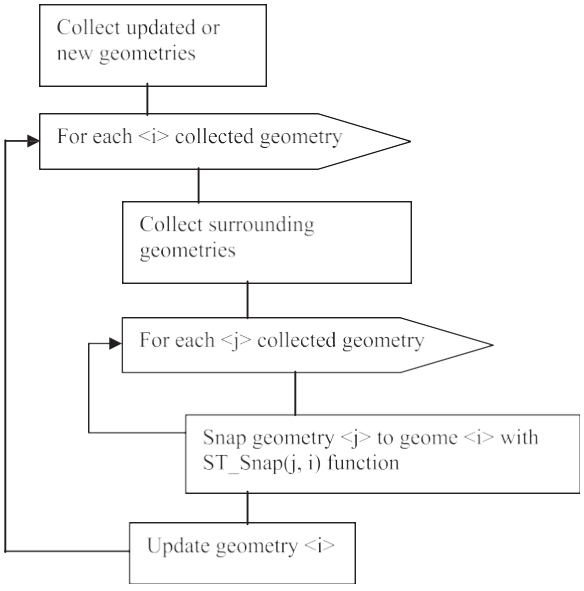

Fig. 2. Algorithm for snapping the geometries of a full layer.

The snapping process can produce invalid geometries according to the OGC-SFA, e.g., full or partial dimensional collapses, self-intersections. To fix this, the implemented topology system automatically detects these cases and fixes them using the *ST_MakeValid* Jaspa function. This function attempts to create a valid representation of a given invalid geometry: in the case of a geometry collapse, the output geometry may be a collection of lower-to-equal dimension geometries or a geometry of lower dimension, and single polygons may become multi-geometries in the case of self-intersections. If the topology system cannot fix the geometry automatically, an error is triggered, and the user must edit the geometry manually (rare cases).

3.2. ***Topology rule set***

Currently, Jaspa supports 67 topology rules; 30 of them are applied to features from only one layer, and 37 topology rules are applied between features from two layers.

To rigorously define the semantic meaning of the topology rules, we have used logical formulas (Praing and Schneider, 2008) based on mathematical notation similar to that used in the standard "Simple Feature Access" (OGC, 2011). For the spatial relational operators, we have used the document notation based on the matrix DE-9IM (Clementini et al., 1993).

Table 2 summarizes the notation and the spatial function used to describe the semantics of the topology rules. For instance, the "must be disjoint" topology rule is defined as $\forall a \in A, \nexists b \in B / a \cap b \ddagger \emptyset$, which means: For each geometry 'a' contained in layer A, there is not any geometry 'b' contained in layer B, such that the geometry 'a' intersects the geometry 'b'.

Table 3 shows the implemented topology rules on single layers. Some topology rules can be applied to any geometry type layer; for example, the rule "Must be single Part" supports point (P), line (L) or polygon (S) layers (the geometry type is called as 'P', 'L' or 'S' in the first column of the table), whereas other rules can be applied to just a geometry layer type including "Must not have dangles" (only line layers).

Table 4 shows the implemented topology rules between two layers. The fist column of the table shows the allowed geometry type layer combinations; e.g., the topology rule "Must be covered by boundary of" can be applied between punctual and lineal layers (P+L) and between punctual and polygonal layers (P+S).

For the implementation of the topology rules, we have used SQL plus some SQL Jaspa functions developed for these rules. All rules use a primary spatial filter plus some secondary spatial filers using the Jaspa spatial predicates following the standard OGC-SFA. However, the goal of this paper is not to show the implementation details, which are freely available on the Jaspa web site.

Some topology rules consider an extra tolerance parameter. This is beneficial in cases when the GIS user does not want to use a cluster tolerance (as explained in the previous chapter) with the layer.

All topology rules are performed in 2D except the "Must be connected 3D" topology rule, which works in 3D.

Table 2. Notation used to define the topology rules.

Symbol	Meaning
I(a), B(a), E(a)	Given a geometric object *a*, let *I(a)*, *B(a)* and *E(a)* represent the interior, boundary and exterior of "a", respectively.
F_dim(a)	Returns the dimension of a geometry according to the OGC-SFA. The *ST_Dim* Jaspa function implements it. This is performed in Jaspa through the JTS library
F_isValid(a)	Returns true is a geometry is valid according to the standard OGC_SFA. This is performed in Jaspa through the JTS library.
F_isSimple(a)	Returns true is a geometry is simple according to the standard OGC_SFA (do not cross by the same point twice). This is performed in Jaspa through the JTS library
F_distance(a, b)	Returns the minimum 2D cartesian distance between geometry 'a' and geometry 'b'
F_startPoint(a)	Returns the start or end point of a lineal geometry F_endPoint(a)
F_repeatedPoints (a)	Returns a point set with the repeated vertices (two identical followed vertices) F_segment (a), Returns the set of segments composing a geometry
F_zDistance(a, b)	Returns the minimum 3D cartesian distance across the Z axis between geometry 'a' and geometry 'b'
F_closestPoint (p0, A, tol),	Calculate the 2-dimensional point from the set of geometries A (layer A), which is closest to the point geometry 'p0'. If the distance between the calculate point and p0 is smaller than tolerance (tol) then returns the calculated point otherwise returns an empty point.
F_union (S)	Returns a geometry which is the union of the set of geometries S.

On the official Jaspa web site, we have created detailed documenta- tion of all 67 rules with graphical examples (Gonzalez-Alcaide and Martinez-Llario, 2011).

As stated before, ArcGIS has been the GIS software with the richest set of topology rules. It has 32 different topology rules (ESRI, 2012).

The proposed topology system implements 67 rules. Some of these rules can be applied to different geometry types (as in ArcGIS). If we do not consider the different geometry type versions (although the developed algorithm is different for each geometry type) to count the topology rule number, then the proposed system has 40 pure topology rules (ArcGIS has 27). We can conclude that the proposed system has a richer set of pure topology rules, and the rules can be applied to a wider set of geometry types.

The motivation to choose this set of rules is to give users the opportunity to apply geographic constraints to their datasets without any limitation. Some common spatial relations between layers, e.g., rivers

(lines) must not intersect buildings (polygons), buildings must be inside parcels (polygons), cannot be modelled with ArcGIS because of a lack of geometry type rule variations.

In other cases, some totally new topology rules are needed, such as those for checking network continuity (rivers, transportation, etc.) in 2D and in 3D using a threshold tolerance or checking whether boundaries of states are covered by boundaries of counties, etc. Some of these new topology rules are the following: “must be connected 2D with”, “must be connected 3D with”, “boundary must be covered by boundary of layer”, etc.

All ArcGIS (and other GIS software) topology rules have been included in the proposed topology system. A couple of ArcGIS topology rules require the use of two chained Jaspa topology rules: “Endpoint A must be covered by B” (“A must not have dangles” + “C must be covered by B”, where “C” is the returning error dataset from applying the first rule) and “must cover each other” (“A must be covered B” + “B must be covered A”).

3.3. ***Extending the rules***

The system allows GIS users to define their own topology rules by using templates and SQL scripts, but at this time, this task is only for advanced users because it requires some Java and SQL coding.

To add a new topology rule, we must register the rule in the Java code (defining a set of parameters, such as the geometry types and the number of layers involved).

The second step is to create the SQL sentence that inserts the geometries into the error table. This sentence uses a combination of Jaspa functions, data manipulation SQL commands and operators, and some parameters to set the table and columns names.

For example, Fig. 3 shows the SQL script for the rule “Must Be Disjoint” for polygons, which checks whether the polygons of a layer are disjoint. The output error table is a table (without any geometry constraint) that can store points, lines or polygons. This rule inserts into the error table the geometric intersections (rule violations).

The Jaspa API documentation shows the description of all of these parameters, and new topology rules can be added to the system simply by designing the SQL sentence and registering the new rule.

We want to mention that we have developed in Jaspa all spatial stored procedures used in the topology rule code. These stored procedures add the needed spatial functionality including ST_Intersects, ST_Boundary, ST_Intersection, ST_Extract, ST_Union, ST_Force_2D, ST_Length, and many more.

3.4. ***Semantic predicates***

Sometimes, it is convenient to apply a topology rule to a subset of a layer instead to the whole layer, e.g., to require street features to be connected to other street features at both ends, except in the case of streets belonging to the cul-de-sac or dead-end subtypes.

Table 3. Definition of Topology rules (one layer A).

P	L	S	Rule / Algebra definition
X	X	X	Must be single Part $\forall a \in A$ / F_numGeometries (a) = 1
X	X	X	Must be disjoint $\forall a \in A, \nexists b \in A, b \neq a$ / $a \cap b \neq \varnothing$
X	X	X	Must be disjoint (with tolerance) $\forall a \in A, \nexists b \in A, b \neq a$ / *F_distance(a, b) < tol*
X	X	X	Must not be duplicated $\forall a \in A, \nexists b \in A, b \neq a$ / *F_equals(a, b)*
X	X	X	Must not have repeated points $\forall a \in A$ / F_repeatedPoints (a) = $\varnothing$
	X		Must not self intersect $\forall a \in A$ / F_isSimple(a)
	X		Must not self overlap $\forall a \in A$, SG = { *segment* $\in$ F_segment(a) } / $\forall sga \in SG, \nexists sgb \in SG, sga \neq sgb, sga \cap sgb \neq 0$
	X		Must not intersect or touch interior (except at endpoints) $\forall a \in A, \nexists b \in A, b \neq a$ / $I(a) \cap I(b) \neq \varnothing \vee I(a) \cap B(b) \neq \varnothing \vee B(a) \cap I(b) \neq \varnothing$.
	X		Must not touch interior $\forall a \in A, \nexists b \in A, b \neq a$ / $I(a) \cap B(b) \neq \varnothing \vee B(a) \cap I(b) \neq \varnothing$
	X		Must not intersect (except at endpoints) $\forall a \in A, \nexists b \in A, b \neq a$ / $I(a) \cap I(b) \neq \varnothing$
	X	X	Must not overlap $\forall a \in A, \nexists b \in A, b \neq a$ / F_dim ($I(a) \cap I(b)$) = F_dim (a)
	X		Must not have dangles $\forall a \in A$, $\exists b \in A, b \neq a$ / F_endPoint (a) $\cap b \neq \varnothing \vee$ $\exists c \in A, c \neq a$ / F_startPoint (a) $\cap c \neq \varnothing$
	X		Must not have dangles (with tolerance) $\forall a \in A$, $\exists b \in A, b \neq a$ / F_distance (F_endPoint (a), b) < tol $\vee$ $\exists c \in A, c \neq a$ / F_distance (F_startPoint (a), c) < tol
	X		Must not have pseudonodes $\forall a_i \in A$, S = { $b \in A, b \neq a_i$: F_startPoint (a_i) $\cap$ B(b) $\neq \varnothing$ }, R = { $b \in A, b \neq a_i$: F_endPoint (a_i) $\cap$ B(b) $\neq \varnothing$ } / (|S| = 1 $\vee$ |S| = 0) $\wedge$ (|R| = 1 $\vee$ |R| = 0)
	X		Must not have pseudonodes (with tolerance) $\forall a_i \in A$, S = { $b \in A, b \neq a_i$: F_distance (F_startPoint (a_i), B(b)) < tol }, R = { $b \in A, b \neq a_i$: F_distance (F_endPoint (a_i), B(b)) < tol } / / (|S| = 1 $\vee$ |S| = 0) $\wedge$ (|R| = 1 $\vee$ |R| = 0)
		X	Must be valid $\forall a \in A$ / F_isValid(a)
		X	Must not have gaps $\forall a_i \in A$, S = { $b \in A$: $a_i \cap b \neq \varnothing$ } / F_length (B(a_i)) = F_length (B(a_i) $\cap$ S)
	X		Must be connected 2d (with tolerance) $\forall a_i \in A$, p_O = F_startPoint(a_i), p_1 = F_endPoint(a_i), cp_O = F_closestPoint (p_O, A – {a_i}, *tol*), cp_1 = F_closestPoint (p_1, A – {a_i}, *tol*) / F_distance (p_O, cp_O) < *tol* $\vee$ F_distance (p_1, cp_1) < *tol*
	X		Must be connected 3d (with tolerance) $\forall a_i \in A$, p_O = F_startPoint(a_i), p_1 = F_endPoint(a_i), cp_O = F_closestPoint (p_O, A – {a_i}, *tol*), cp_1 = F_closestPoint (p_1, A – {a_i}, *tol*) / F_zDistance (p_O, cp_O) < *tol* $\vee$ F_zDistance (p_1, cp_1) < *tol*

Table 4. Topology rule definition (between two layers)

Geometry	Rule/Algebra definition
P+P, L+L, S+S	Must be disjoint with $\forall a \in A, \nexists b \in B$ / $a \cap b \neq \varnothing$
P+P, L+L, S+S	Must be disjoint with (with tolerance) $\forall a \in A, \nexists b \in B$ / *F_distance(a, b) < tol*
P+P	Must be coincident with $\forall a \in A, \exists b \in B$ / $a \cap b = a \wedge a \cap b = b$
P+P	Must be coincident with (with tolerance) $\forall a \in A, \exists b \in B$ / *F_distance(a, b) < tol*
P+L, P+S, L+L, L+S, S +S	Must be inside $\forall a \in A, \exists b \in B$ / $a \cap b = a$
P+L, P+S, L+L, L+S, S +S	Must be properly inside $\forall a \in A, \exists b \in B$ / $a \cap I(b) = a$
P+S,L+S	Must be covered by boundary of $\forall a \in A, \exists b \in B$ / $a \cap B(b) = a$
L+L	Must not intersect or touch interior with (except at endpoints) $\forall a \in A, \nexists b \in B$ / $I(a) \cap I(b) \neq \varnothing \vee I(a) \cap B(b) \neq \varnothing \vee B(a) \cap I(b) \neq \varnothing$
L+L	Must not touch interior with $\forall a \in A, \nexists b \in B$ / $I(a) \cap B(b) \neq \varnothing \vee B(a) \cap I(b) \neq \varnothing$
L+L	Must not intersect with (except at endpoints) $\forall a \in A, \nexists b \in B$ / $I(a) \cap I(b) \neq \varnothing$
L+L, S+S	Must not overlap with $\forall a \in A, \nexists b \in B$ / dim ($I(a) \cap I(b)$) = dim(a) = dim (b)
L+L, L+S, S+S	Must be covered by layer $\forall a_i \in A$, S = { $b \in B$: $a_i \cap b \neq \varnothing$ }, c = F_union(S) / $a_i \cap c = a_i$
L+S	Must be covered by boundary of layer $\forall a_i \in A$, S = { $b \in B$: $a_i \cap b \neq \varnothing$ }, c = F_union(S) / $a_i \cap$ B(c) = a_i
S+P	Must contain one point $\forall a_i \in A$, S = { $b \in B$: $a_i \cap b \neq \varnothing$ } / |S| = 1
S+P	Must contain one point properly $\forall a_i \in A$, S = { $b \in B$: I(a_i) $\cap b \neq \varnothing$ } / |S| = 1
S+P	Must contain points $\forall a \in A, \exists b \in B$ / $a \cap b = b$
S+P	Must contain properly points $\forall a \in A, \exists b \in B$ / $I(a) \cap b = b$
S+L	Boundary must be covered by layer $\forall a_i \in A$, S = { $b \in B$: $a_i \cap b \neq \varnothing$ }, c = F_union(S) / B(a_i) $\cap$ c = B(a_i)
S+S	Boundary must be covered by boundary of layer $\forall a_i \in A$, S = { $b \in B$: $a_i \cap b \neq \varnothing$ }, c = F_union(S) / B(a_i) $\cap$ B(c) = B(a_i)
L+L	Must be connected 2D with (with tolerance) $\forall a \in A$, p_O = F_startPoint(a), p_1 = F_endPoint(a), cp_O = F_closestPoint (p_O, B, *tol*), cp_1 = F_closestPoint (p_1, B, *tol*) / F_distance (p_O, cp_O) < *tol* $\vee$ F_distance (p_1, cp_1) < *tol*
L+L	Must be connected 3D with (with tolerance) $\forall a \in A$, p_O = F_startPoint(a), p_1 = F_endPoint(a), cp_O = F_closestPoint (p_O, B, *tol*), cp_1 = F_closestPoint (p_1, B, *tol*) / F_zDistance (p_O, cp_O) < *tol* $\vee$ F_ zDistance (p_1, cp_1) < *tol*

```
String rule =
"INSERT INTO [ERRORS_TABLE] ({DIM1} GIDGEOM1, GIDGEOM2, GEOM) "
       + "SELECT {DIM2} GIDGEOM1, GIDGEOM2, ST_FORCE_2D(GEOM) FROM "
       + "(SELECT A.GID AS GIDGEOM1, B.GID AS GIDGEOM2, "


        + "ST_DUMP (ST_INTERSECTION (A.[GEOM_COLUMN_TABLE1],
 B.[GEOM_COLUMN_TABLE1])) AS GEOM "
       + "FROM [TABLE1] A, [TABLE1] B " + "WHERE (A.GID = ?) AND "


                        + "A.GID < B.GID AND "


+ "ST_INTERSECTS (A.[GEOM_COLUMN_TABLE1], B.[GEOM_COLUMN_TABLE1]) "
+ " AND [ATTREXPAB]) AS FOO";
```

Fig. 3. SQL script definition of the 'Must be disjoint topology' rule.

In other GIS software programs such as ArcGIS, this is accomplished by using subtypes. Subtypes are a subset of features in a layer that share the same attributes. They are used as a method to categorize the data (Law and Collins, 2016). This is a prerequisite task before defining or performing the topology rules and can be considered as dividing the layers into different logical layers that affect our database model.

In the proposed system, the semantic filter is defined together with the topology rule, and there is no need to define new logical layers. In this way, the database model is not required to be adapted according to the semantic filters.

Chapter 4 shows an example of defining a semantic filter at the same time a new topology rule is added to our model.

3.5. ***Topology timestamp***

GIS workflow includes tasks such as loading new data into a layer or updating them on a regular basis. This means that it is necessary to continuously validate the topology rules in which a layer participates. To avoid validating topology rules with features that have already been validated, the proposed topology system provides a mechanism called "*TopoTime*". It ensures that the validation of the topology rules is applied only to registers that are not validated because they either contain new data or have been updated.

This mechanism consists of an SQL trigger that increments a sequential counter (version number) when a layer or a topology rule that participates in a topology model is somehow modified. The rule metadata table has a "*topotime*" field that contains the version number at the time the rule is validated.

Similarly, when a layer is added to a topology model, a "*topotime*" field is automatically added to the layer. This "*topotime*" layer field contains the version number from the last feature update for all features of the layer.

However, an additional parameter is required: when a feature is updated or deleted, its neighbouring geometries may change. The boxes of these neighbouring geometries are stored in a temporal table.

Thus, the only features that are considered when validating a rule are as follows:

1. Those with a higher "*topotime*" field than the one stored in the rule metadata table.

2. Those that intersect any geometry bounding box from the temporal table.

This temporal table is similar to the concept of "dirty areas" from ArcGIS (Law and Collins, 2016), but there is an important change: In ArcGIS, there is only one "dirty area" table for all layers participating in the topology, and in the designed system, there is one "dirty area" for each layer. The disadvantage of the designed system is that all of these temporal tables require more storage space than just one "dirty area" table for all layers. The benefit is that each topology rule can validate just the updated features participating in that rule and not in all defined rules, which is much more efficient.

We want to mention that the "dirty area" or "topotime" mechanism in the proposed system (as in ArcGIS) is completely valid for the full set of topology rules, in the sense that this mechanism depends only on the neighbouring geometries; this concept is correct and has been checked for all exposed rules. The only exception to this would be the case of designing a new customized topology rule that includes distance concepts; e.g., rivers must be closer than 1000 m from roads (in that case, the dirty area boxes from both rivers and roads should be expanded by at least the distance used as a parameter).

3.6. ***Topology management in Jaspa***

The process of topology management in Jaspa can be quickly summarized as follows: the creation of a topology model, addition of layers to the model and rules to be satisfied by the layers, and finally rule validation.

1) Creation of a topology model

Topology management in Jaspa starts by creating a topology model. If it is the first time, a topology schema is simultaneously created. This topology schema can host multiple topology models with different or equal rules or layers participating in each topology model. For each topology model, Jaspa automatically creates two metadata tables that will contain the layers and rules that will participate in the topology model:

1. Layers_TopologyModelName: It stores information about the tables that are part of a topological model (name, geometry type and the tolerance of each layer).

2. Rules_TopologyModelName: It stores information about the rules that are part of a topological model, such as the name of each rule, the name of the layers that participate in it with their geometry types and tolerances, and other fields with information about validation status.

2) Configuration of the topology model

First, the user must add the layers that participate in the topology model. At this step, the cluster tolerance for each layer is explicitly specified by the user. If not, the default tolerance is 0.001 units. The addition of a layer includes three events:

- The layer metadata table is updated.
- A temporal topotime table is added.
- A topotime field is added to the geometry table of each layer.

Second, the user adds to the topology model the rules that must be followed by the layers. It is possible to specify for each layer a filter to validate a set of features (rows). The same rule can be added to the topology model several times, with different selections of a layer participating in each instance of the rule. The consequence of adding a rule is the update of the rule metadata table.

3) Rules validation

Depending on the requirements of the user, a topology model can be fully or partially validated by the following criteria:

- Validate a full topological model.
- Validate a rule that participates in a specific topological model.
- Validate a layer that participates in a specific topological model.
- Validate a layer in every topological model in which it participates.

Jaspa validates by default only the features indicated by the "TopoTime" mechanism, although the entire layer can be validated if required. Regardless of the chosen validation criterion, the results of a validation process are as follows:

- Each rule metadata table is updated.
- If a rule is validated for the first time, an error table for each topology rule is created to insert the errors into it.

4. Practical case

The INSPIRE (Infrastructure for Spatial Information in Europe) European Directive (European Parliament, 2007) and subsequent regulations define the structure of spatial data and services for future European SDI. The INSPIRE policy of harmonization and interoperability of spatial data are consistent with international standards of the OGC (Open Geospatial Consortium) and the ISO/TC 211 (http://www.

isotc211.org).

The BTA (Base Topográfica Armonizada - Harmonized Spatial Database) (Martinez-Llario et al., 2012) is a complex geographic database model that attempts to follow the European specification data from INSPIRE in Spain.

The purpose of this chapter is to show how the new topology system can be applied to this cartographic model. The BTA model defines hundreds of spatial relationships. In this example, we show the use of just one topology rule to model a special conflictive case between two layers.

In this example, we have two datasets (layers) that model the transportation network. The layer "viarialin" contains the line strings defining the transportation network: edges and borders. The layer "viariapol" contains the polygons defining the road areas.

The zone of study covers a rectangle of 15×10 km2. The "viarialin" layer contains approximately 6200 lines, and "viariapol" contains approximately 800 polygons. The number of errors detected by this topology rule example can be extrapolated to any other zone.

The "viarialin" layer has a feature attribute "componen1d", which categorizes the feature as follows:

- EJE: Line strings defining the centreline of the road area.
- BOR: Line strings defining the external boundary of the road area.
- BVI: Line strings defining the shared boundaries that divide two road polygons.
- ECO: Line strings defining the junction between centrelines (notice that these lines do not divide road polygons.

Fig. 4 shows that there is some duplicated data because the boundaries of the polygons of "viariapol" must overlap some lines from "viarialin" (only the 'BOR' and 'BVI' categories). These duplicated lines must be totally coincident. This duplication is not observed most of the time (see Fig. 5a) because the geometries have different numbers of vertices or segments or only because both layers come from different editing and capture processes and do not adjust exactly.

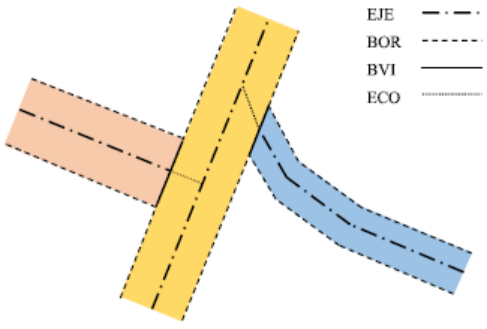

Fig. 4. Spatial relationships between two transportation network layers from BTA.

Fig. 5a shows the distance (‘d’) between the boundary of the polygon feature and the line feature. If ‘d’ is much less than the data accuracy (approximately one-tenth as a minimum), then we can assume that the line and/or the polygon can be moved to match each other exactly without distorting the map accuracy. This process is performed automatically by adjusting the vertices between both layers as we stated in chapter 3.1.

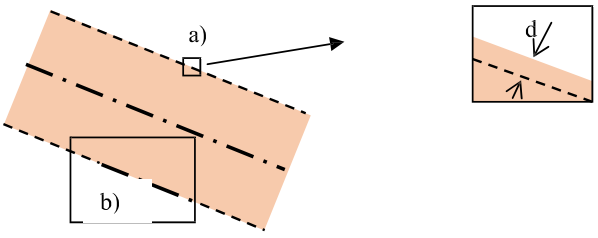

Fig. 5. Some topology errors.

In this example, we apply a topology rule to these two datasets, first without using this automatic adjustment and later with it. This way, the reader can check the importance of this procedure.

To model the explained spatial relationship between these two layers, we have developed a “line-polygon” topology rule called “Layer A must be covered by boundary of layer B”, where A is “viarialin” and B is “viariapol”. Any line from layer A must be covered by the boundaries of one or many polygons for layer B.

As shown in Fig. 4, the lines of type “EJE” and “ECO” are centrelines and must not participate in this rule. For this purpose, we have extended the system to use attribute filtering.

Fig. 5b shows a semantic inconsistency in our model: the boundary of the polygon is covered partially because there is a section of the line categorized as ‘EJE’ instead of ‘BVI’ or ‘BOR’.

The process is extremely easy. From an SQL client, we will run the followings SQL commands:

1) Create a new topology: SELECT command ('create topology t1')

2) Add the layer “viariapol” to the topology: SELECT command ('ADD layer viariapol TO t1')

3) Add the layer “viarialin” to the topology: SELECT command ('ADD layer viarialin TO t1')

4) Add the rule “must be covered by boundary of layer” (mbcbbol) to the topology:

SELECT command ('ADD RULE mbcbbol TO t1 using viarialin, viariapol attrexp1 {(Table 1).componen1d like "B%"}')

The “attrexp1” parameter passes a filter to the SQL script. In this case, only the attribute ‘component1d’ from “viarialin” that starts with ‘B’ (‘BOR’ and ‘BFI’ in Fig. 3) are required.

5) Validate de topology rule: SELECT command ('SHOW RULES ON t1')

The full process takes a few seconds for all layers, and by default, the process does not snap geometries between layers. The response is as follows:

Process finalized. Number of SQL executed: 3335 Number of affected rows: 872

The rule 1 in the topology t1 has been validated. Errors [Before:0 After validating:1313]

The validation shows that 1313 lines are not covered properly by the boundaries of the “viariapol” layer.

Although we have demonstrated how powerful the topology rule system is, the GIS user cannot manually fix all of these errors. To fix most of the errors, we have developed an adjusting method (see Section 3.1). By default, Jaspa does not adjust the geometries; to accomplish this, we must specify the property SNAP for each layer of the topology (the snapping tolerance is 0.001 m by default):

SELECT command ('ALTER LAYER viarialin ON t1 SNAP true');

SELECT command ('ALTER LAYER viariapol ON t1 SNAP true');

SELECT command ('validate rule 1 ON t1');

After validating the topology again, the number of features that violate the rule decreases to 103 (92% of the errors have been fixed automatically). The Jaspa topology system creates a new layer with these 103 problematics geometries (*r1_t1_mbcbbol_viariapol layer*) that the user can display with a GIS desktop (Fig. 6) to determine the problem of each feature.

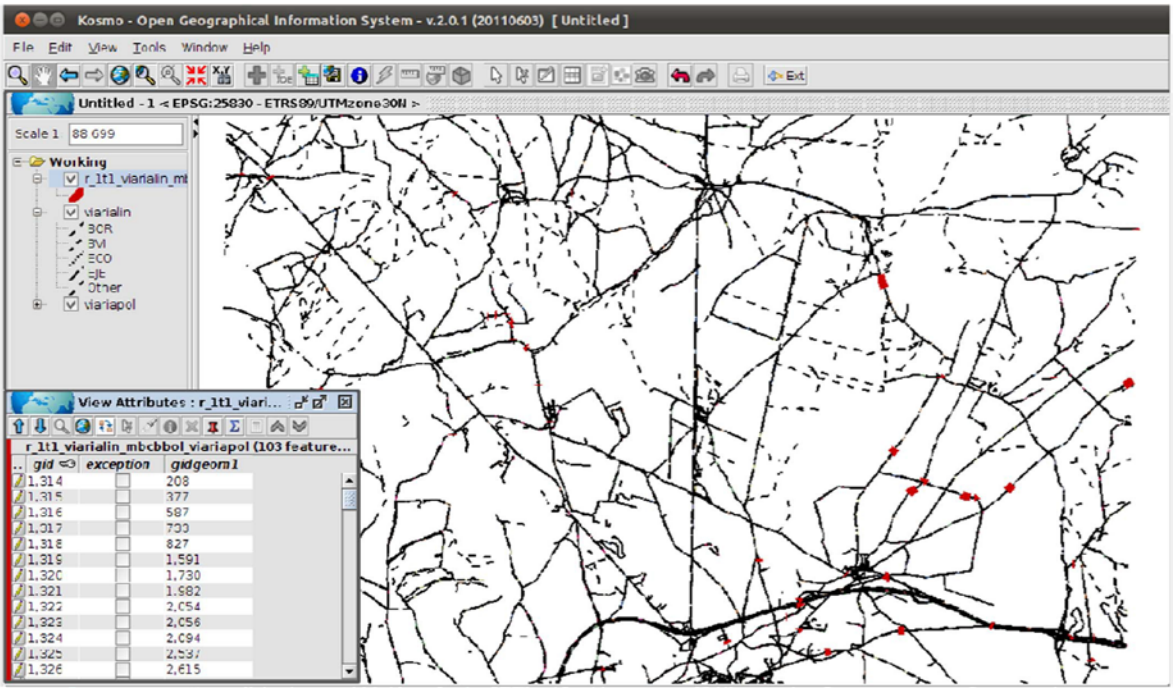

Fig. 6. Kosmo GIS with the Jaspa plugin showing the topology rule errors (red) layer.

The GIS user must check all of these remaining errors and attempt to fix them with a GIS editor. After this process, our datasets will be ready for any spatial analysis and hence to be delivered.

Fig. 7. shows two examples of the errors found and summarizes most of them. Fig. 7a shows a lack of

adjustment (approximately 3 mm) between *viarialin* and the boundary of *viariapol*. Fig. 7b shows many errors because the *viarialin* lines are marked as BVI type, which means that these lines should determine the boundaries of different polygons, but the figure shows only one large polygon.

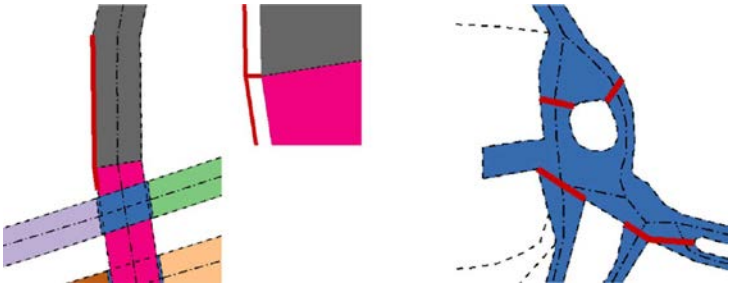

Fig. 7. Main errors in the case study.

4.1. Data specification INSPIRE rules

As we mentioned, the BTA model attempts to follow the European data specification from INSPIRE in Spain. Most of the designed topology rules can be used to validate the INSPIRE data specification, and this is one of the main goals of the presented software.

In the technical guidelines for the INSPIRE data specification, there are specific instructions about defining the geometrical spatial relation- ships among features.

In the data quality chapter from each data specification guideline, there are some explicit spatial constraints. To check these constraints, we can easily use the topology system presented; e.g., some of the spatial constraints that contain the guidelines from the transportation network data specification (data quality chapter) (INSPIRE, 2014) are as follows:

Completeness (commission) due to

- Number of duplicate feature instances.

Logical consistency (topological consistency) due to

- Number of invalid overlaps of surfaces. Total number of erroneous overlaps within the data.

- Number of missing connections due to undershoots or overshoots. The lack of connectivity exceeding the connectivity tolerance is considered to be an error if the real features are connected in the transport network.

- Number of invalid self-intersect errors. Count of all items in the data that illegally intersect with themselves.

- Number of invalid self-overlap errors. Count of all items in the data that illegally self-overlap.

The violation of the above constraints can be checked easily by applying the topology rules: “must

not be duplicated", "must not overlap", "must not overlap with", "must not self-intersect" and "must not self-overlap".

To check the lack of connectivity, we have developed the INSPIRE specific rules "must be connected 2D with" and "must be connected 3D with" (these rules are not supported by ArcGIS) that follow the INSPIRE definition. The INSPIRE technical guide, in its data capture chapter (ensuring network connectivity) defines the connectivity as follows: "link ends and nodes that are not connected shall always be separated by a distance that is greater than the connectivity tolerance", which is exactly the function of these topology rules.

The technical guidelines contain more examples about spatial constraints that can be checked using the implemented topology rules. To show all of these spatial constraints from each INSPIRE data specification is not the purpose of this paper but rather to show that the topology rule engine can be used to test most of them.

With this practical example, we wanted to show the following:

- The end user can use the predefined rules (an advanced user could design new topology rules) that consider the complex spatial relationships between layers.

- The layers are adjusted to avoid any geometrical precision matching problem between geometries.

- The attribute features ("componen1d" in this example) can be used to filter and apply the topology rule depending on the semantic meanings of the features.

- PostgreSQL has been used as the DBMS backend, but we could have used Jaspa with H2 or Oracle.

5. Conclusions

Jaspa is the first open source spatial extension for databases that implements a rule-based topology. Topology management is a method to determine the integrity of the data. To this end, Jaspa provides a highly flexible and fast procedure to enforce relationship integrity between layers. After validating and correcting, a high-quality dataset is obtained without the need to change to an arc-node model.

The Jaspa topology rule-based engine introduces five major ad- vantages over the topology rule systems provided in GIS desktops:

- By default, Jaspa covers more spatial relationships (67 topology rules) between layers than any other GIS system.

- An advanced user may freely define its own rules using SQL parametrized scripts in Jaspa code. As a result, each user will elaborate a set of rules that portrays its data model in the most suitable way. No other GIS software is able to do that.

- Topology rules can be applied to the whole layer or just a set of features from the layer. In ArcGIS, to

be able to apply the topology rule to a set of features, it is necessary to create subtypes. In other open source GIS, the topology rules area applied only to the whole layer.

- There is not a single cluster tolerance for all layers. This is particularly important in terms of scalability because some data have usually been collected with different accuracy levels, or the user may want to apply different tolerance depending on the layers.

- A topology scheme can be understood as a set of topology rules (and a cluster tolerance) that affect one or many datasets. A GIS user can define many topology schemes, and the same dataset can take part of several topology schemes as well. In other GIS such as ArcGIS, this is not allowed because each dataset can take part of just a topology schema.

- We showed that many spatial constraints defined in the INSPIRE data specification technical guides can be checked using these topology rules, and some rules have been implemented specifically for this INSPIRE purpose.

With this topology rule system, a user can design a rigorous quality control for the datasets. Users can design complex spatial relationships to model their geographic databases. The result is a much more reliable cartography to perform subsequent spatial analysis.

Most of the base maps of a country should follow complex GIS datasets such as the INSPIRE data specification in Europe. The proposed topology rule system offers a solution to be able to model the spatial constraints and check the data quality of these data specifications in a spatial database manager.

6. Future research

Despite the advantages, the topology treatment is not fully complete. Once the topology rules are applied, the software should be able to automatically fix some errors. The research is currently focused on improving the automatic spatial adjustment of features given a certain cluster tolerance and without producing invalid geometries.

As noted previously, a DBMS is the natural environment for storing data, so a full solution should be provided to satisfy users who require topological structures. Furthermore, the next step would be to implement a dual system in which the user could choose between storing spatial data in a compliance simple feature structure or a topological structure and offer mechanisms for conversion from one structure into another in addition to keeping the topology rules system to ensure high quality in spaghetti data. As stated by Baars et al. (2004), both types (topology rules and topological structures) may work together: first check the rules and subsequently migrate to a topologically structured model.

Software availability

JASPA Topology software: Free under the GNU GPL license and can be downloaded from http://jaspa.upv.es.

Developer: Jose Martinez-Llario, jomarlla@cgf.upv.es. First version: 2011.

Programing languages: Java, PL/Java and SQL.

Operating System requirements: MS Windows XP or newer, recent Mac OSX, GNU/Linux or a UNIX variant.

Acknowledgments

This work has been partially supported by the research project “Creation and cartographic feeding of spatial data Infrastructures in the local government by means of a data model that integrates cadastre, planning and cultural heritage”, CSO2008-04808 from the Spanish Government (CICYT) and the European Union Funds.

References

Baars, M., Stoter, J., Oosterom, P., Verbree, E., 2004. Rule-based or explicit storage of topology structure: a comparison case study. In: Proceedings of the 7th AGILE Conference on Geographic Information Science 2004, Heraklion, Greece.

Belussi, A., Migliorini, S., Negri, M., Pelagatti, G., 2015. Impact of data representation rules on the robustness of topological relation evaluation. Geoinformatica 19 (2), 1–44.

Brown, D., Riolo, R., Robinson, D., North, M., Rand, W., 2005. Spatial process and data models: toward integration of agent-based models and GIS. J. Geogr. Syst. 7 (1), 25–47. http://dx.doi.org/10.1007/s10109-005-0148-5.

Clementini, E., Di Felice, P., 1995. A comparison of methods for representing topological relationships. Inf. Sci. - Appl. 3 (3), 149–1178. http://dx.doi.org/10.1016/1069- 0115(94)00033-X.

Clementini, E., Di Felice, P., van Oosterom, P., 1993. A Small Set of Formal Topological Relationships Suitable for End-User Interaction. Lecture Notes in Computer Science 692. Springer, 277–295. http://dx.doi.org/10.1007/3-540-56869-7_16.

De Hoop, S., van Oosterom, P., 1992. Storage and Manipulation of Topology in Postgres. Proceedings EGIS '92, München (1992) 1324–1336 01/1992.

Ellul, C., 2008. Functionality and Performance - Two Important Considerations when Implementing Topology in 3D. Thesis submitted for the Degree of Doctor of Philosophy (PhD). University of London.

Erickson, J., County, P., 2009. Java Topology Suite in Action. Washington URISA GIS Conferences. 2009. Available online at 〈 http://www.waurisa.org/conferences/2009/

presentations/Weds/OpenSource_Weds_Erickson_PierceCounty.pdf 〉

ESRI, 2003. ArcGIS Working With Geodatabase Topology. An ESRI White Paper. Available online at〈 http://downloads.esri.com/support/whitepapers/ao_/ geodatabase-topology.pdf〉.

ESRI, 2012. ArcGIS 10.2 Geodatabase Topology Rules. Available online at 〈 http:// resources.arcgis.com/en/help/main/10.2/01mm/pdf/topology_rules_poster.pdf 〉

European Parliament, 2007. Directive 2007/2/EC of the European Parliament and of the council of 14 March 2007 establishing an Infrastructure for Spatial Information in the European Community (INSPIRE). Available online at 〈 http://eur-lex.europa.eu/LexUriServ/LexUriServ.do*Uri=OJ:L:2007:108:0001:0014:EN:PDF 〉

Gonzalez-Alcaide, M., Martinez-Llario, J., 2011. Jaspa oficial documentation. Topology rules chapter. Available online at: 〈 http://jaspa.upv.es/jaspa/v0.2.0/manual/html/ topology.html#topo.rules 〉

Halperin, D., Packer, E., 2002. Iterated snap rounding. Comput. Geom. 23 (2), 209–225. http://dx.doi.org/10.1016/S0925-7721(01)00064-5.

INSPIRE (Thematic Working Group), 2014. D 2.8.I.7: INSPIRE Data Specification on Transport Networks Technical Guidelines. INSPIRE Thematic Working Group Transport Networks, 2014-04-17.

INSPIRE (Official Journal of the European Union), 2010. COMMISSION REGULATION (EU) No. 1089/2010 of 23 November 2010 implementing Directive 2007/2/EC of the European Parliament and of the Council as regards interoperability of spatial data sets and services.

ISO (International Standard Organization), 2004. ISO 19125-1:2004. Geographic information. Simple feature access – Part 1: Common architecture.

ISO (International Standard Organization), 2012. ISO 19152:2012. Geographic information. Land Administration Domain Model (LADM).

ISO (International Standard Organization), 2016. ISO/IEC 13249-3:2016: Information technology. Database languages. SQL multimedia and application packages – Part 3: Spatial.

Law, M., Collins, A., 2016. Getting to know ArcGIS Pro. Esri Press, New York.

Martinez-Llario, J., Gonzalez-Alcaide, M., 2011. Design of a Java spatial extension for relational databases. J. Syst. Softw. 84 (12), 2314–2323. http://dx.doi.org/10.1016/ j.jss.2011.06.072.

Martinez-Llario, J., Ruiz-Lopez, F., Coll, E., 2012. Materialization of BTA database using open source software. Procedia Technol. 1, 15–18. http://dx.doi.org/10.1016/ j.protcy.2012.02.005.

Murray, C., 2009. Oracle Spatial Topology and Network Data Models Developer's Guide.

Oracle Corporation, California.

Neteler, M., Bowman, M.H., Landa, M., Metz, M., 2012. GRASS GIS: a multi-purpose open source GIS. Environ. Model Soft. 31, 124–130. http://dx.doi.org/10.1016/ j.envsoft.2011.11.014.

OGC (Open Geospatial Consortium Inc.,), 2011. OpenGIS Implementation Standard for Geographic information - Simple feature access - Part 1: Common architecture.

Pelagatti, G., Negri, M., Belussi, A., Migliorini, S., 2009. From the conceptual design of spatial constraints to their implementation in real systems. In: Proceedings of the 17th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems (GIS '09). ACM, New York, NY, USA, 448–451. Doi: ⟨http://dx.doi.org/10.1145/1653771.1653841⟩.

Penninga, F., 2004. Testing Oracle 10g Topology using cadastral data. OTB Research Institute for Housing, Urban and Mobility Studies.

Picavent, V., 2010. State of the art of foss4g for topology and network analysis. Free and Open Source Software for Geospatial (FOSS4G). Barcelona.

Praing, R., Schneider, M., 2008. Efficient implementation techniques for topological predicates on complex spatial objects. Geoinformatica 12 (3), 313–356.

Van Oosterom, P., Stoter, J., Quak, W., Zlatanova, S., 2002. The balance between geometry and Topology, In: Richardson, D., Oosterom, P., (Eds.), Advances in Spatial Data Handling, In: Proceedings of the 10th International Symposium on Spatial Data Handling, 209–224. Doi: ⟨ http://dx.doi.org/10.1.1.107.9743⟩ .

Worboys, M.F., Bofakos, P., 1993. A canonical model for a class of areal spatial object. Lecture Notes in Computer Science. Adv. Spat. Databases 692, 36–52. http:// dx.doi.org/10.1007/3-540-56869-7.

Yan, Y., Li, G., Xie, J., Guo, L., 2011. A comparative study of spatial data harvest standards. International Archives of the photogrammetry. Remote Sens. Spat. Inf. Sci. XXXVIII-4/W25. http://dx.doi.org/10.5194/isprsarchives-XXXVIII-4-W25-49- 2011.