

# Topological Error Correcting in GIS

**Thierry Ubeda**

Laboratoire d'Ingénierie des Systèmes d'Information  
Institut National des Sciences Appliquées de Lyon  
Bât 404, 20 avenue A. Einstein  
69621 Villeurbanne Cedex - France  
ubeda@if.insa-lyon.fr

and

**Max J. Egenhofer**

National Center for Geographic Information and Analysis  
and  
Department of Spatial Information Science and Engineering  
Department of Computer Science,  
University of Maine, Orono, ME 04469-5711, U.S.A.  
max@spatial.maine.edu

## Abstract

It is agreed upon that topological relations are of great importance regarding to GIS data sets consistency. A lot of errors that can be found in GIS data sets are coming from a lack of knowledge about topological relations between the geographical objects stored in the database. Consequently, topology can help to find errors in GIS data sets, and can help to correct them.

The topic of this paper is to present how topological relation can be used to detect and to correct errors in GIS data sets. Such an approach required three parts : the definition of errors, how to check the database and how to correct errors. This paper focus on the first and the third part.

Error will be described using topological integrity constraints. This method allowed one to define the constraints that fit its own data set, and then allowed to take the semantics of data into account. Correction will be made by applying transformations to the data. For each error, a set of possible corrections will be compute and the end-user will have to choose the appropriate one.

## 1. Introduction

Spatial Analysis in GIS is often hindered by erroneous information among data sets. Answers to spatial queries and spatial reasoning are then not reliable. Different kinds of spatial errors in GIS data sets can be defined. Structuring errors, like non polygon closure and self-intersecting lines, are coming from data structures. Two main origins can be drawn: the data structure is weak or the data do not respect the data structure requirements. Semantics errors, like a road within a lake and a building described as a line, are coming from the real world description. Such errors can not be found without using the semantics of real world entities.

Structuring errors depend on data models and data structures of GIS (Geographical Information Systems). Several attempts to define properties of geographical objects have been made (Ubeda & Servigne 1996a and 1996b, Plümer 1996). The goal of those properties was to detect and to avoid structuring errors. The list of properties defined in (Ubeda & Servigne 1996a and 1996b) is not complete for all kinds of data models, but can be used as a starting point for most of data structures. The list of properties given in (Plümer 1996) was complete for the data model he studied, namely planar graph. It appears that a classification of data model in GIS is required to design a set of properties suitable for all spatial databases. Unfortunately, such a classification doesn't exist yet. In (Franck 1984), Franck proposed a classification for model using only points

and arcs. Area objects have to be added to such a classification if we want to cover all kind of data models.

Semantics errors are defined using the meaning of geographical objects, that is to say the real world entities described by the object. Topological errors are a kind of semantic errors. Topological relations are based only on the shape of objects, but semantics of objects have to be taken into account to decide whether a topological scene is an error or not. Topological relations are of great importance in GIS (Cui & al 1993). A lot of errors contained in GIS came from erroneous topological relations among geographical objects (Laurini & Milleret-Raffort 1994). Most of GIS do not deal with topological relations, or consider only few relations such as adjacency and inclusion.

The goal of this paper is to define a correcting process for topological errors in GIS. A complete topological error correcting process should contain the three following parts:

- A topological error definition process
- A topological error checking process
- A topological error correcting process.

In this paper, topological errors will be defined as topological integrity constraints. Each time a constraint is not respected, an error is found. A scene in which a topological constraint is not respected is an error.

In a first part, a topological integrity constraint definition method will be introduced. It is based on topological relations defined by the 9-intersection model (Egenhofer 1991). It allows to define a constraint using all possible topological relations (and not only inclusion and adjacency).

Secondly, a topological error correcting method will be presented. The goal of such a process is to compute a set of possible corrections among which the end-user has to choose.

The way to present correcting scenarios will be discussed in the last part, then we will conclude.

### 3. Topological Constraints Definition

#### 3.1 *The 9-intersection model*

This topological model has been designed by Max J. Egenhofer in (Egenhofer & Herring 1990, Egenhofer 1991). In this model, binary topological relations between two objects A and B are defined in terms of the nine intersections of A's boundary (A), A's interior (A) and A's exterior (A-) with the boundary (B), interior (B) and exterior (B-) of B (see figure 1).

Each object A and B can be a point, a line or a polygon.

Definition of each part of each kind of geometric object is the following:

- P is a point :  $P = P = P$ .
- L is a line :  $L = \text{the two ending points of } L$ .  
 $L = L - L$ .
- Po is a polygon :  $Po = \text{the intersection of the closure of } Po \text{ and the closure of the exterior of } Po$ .  
 $Po = \text{the union of all open sets in } Po$ .

For each intersection, the value empty () or non-empty () is compute and store into a 9x9 matrix:

$$\begin{pmatrix} \partial A \cap \partial B & \partial A \cap B^\circ & \partial A \cap B^- \\ A^\circ \cap \partial B & A^\circ \cap B^\circ & A^\circ \cap B^- \\ A^- \cap \partial B & A^- \cap B^\circ & A^- \cap B^- \end{pmatrix} \begin{array}{l} \text{if the intersection is empty} \\ \text{if the intersection is non-empty} \end{array}$$

**CONSTRAINT = (Entity class1, Relation, Entity class2, Specification).**

**Figure 1.** The 9-intersection Matrix.

### 3.2 Group of relations

The 9-intersection model can be applied to all kinds of geographical objects. Considering points, lines and polygons, it leads to six groups of relations: point/point, point/line, point/polygon, line/line, line/polygon, polygon/polygon.

In (Egenhofer & Herring 1991), the authors gave the list of relations that can be realized in each group, if objects are embedded in 2-D (see Table 1).

Group of relations	Number of relations
point/point	2
point/line	3
point/polygon	3
line/line	23
line/polygon	19
polygon/polygon	6

**Table 1.** Number of relations per group.

They consider two converse relations as only one since it is possible to change A in B and B in A. Converse relations can only happen between two objects of the same kind, namely in point/point, line/line and polygon/polygon groups.

### 3.3 Design of new constraints

Topological integrity constraints will be defined between two geographical objects. The topological relation between two objects is the main part of the constraint. Considering the shape of objects, it is possible to compute all possible topological relations between two objects (according to the 9-intersection model). Considering the semantics of objects (their meaning), it is possible to define which topological relation is consistent and which one is inconsistent.

A topological constraint is defined as the association of two geographical objects, a topological relation between them and a specification (see Figure 2) which can be one of the following:

1. Forbidden
2. At least  $n$  times
3. At most  $n$  times
4. Exactly  $n$  times

**Figure 2.** The definition of a topological constraint.

The specification *forbidden* is the most interesting and usable one. Topological integrity constraints defined using this specification are a mean for end-users to describe topological situation they do not want to occur in their database.

The number of possible topological relations between two objects is an impediment to the design of such constraints. To provide an easy way to use interface, end-users must be given a set of topological relations described by names and not by mathematical definitions.

Unfortunately, it is almost impossible to give expressive names to all the relations. In addition, some topological relations are so closed that to avoid a situation to happen one will have to create several constraints (one for each topological relation matching some characteristics). For example, to forbid a river (a line) to cross a building (polygon) six constraints have to be defined.

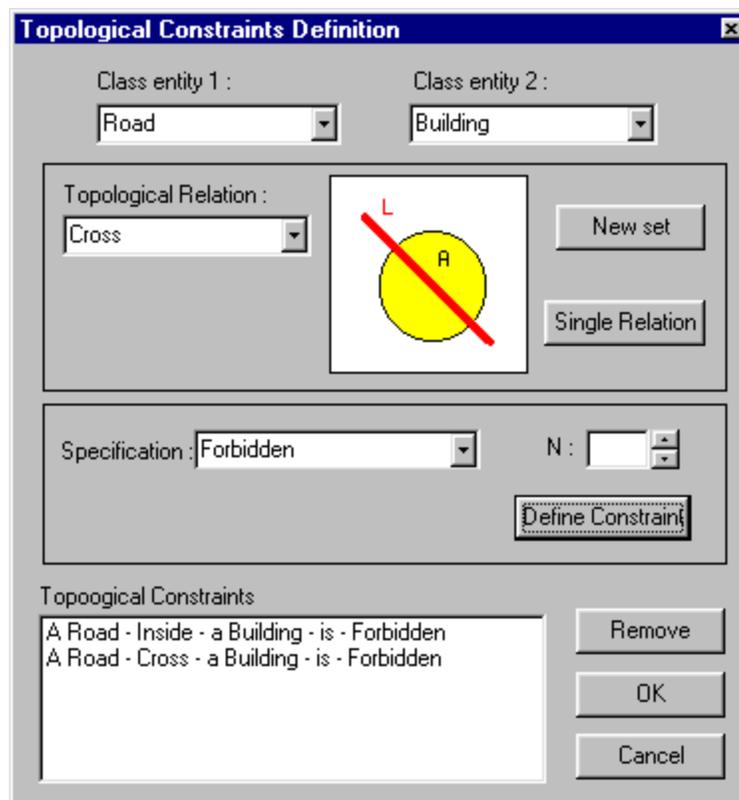
To provide a more usable interface, topological relations sharing common attributes have been grouped in subsets. Such subsets have been built in each group of relation (points/points, points/lines, etc.). For example, in the line/polygon group, a subset called *cross* have been created. It contains all topological relations where the line's interior intersect the polygon's interior and exterior :  $(L^{\circ}P^{\circ}=)(L^{\circ}P=)$ . For more details see (Ubeda & Servigne 1996b).

### 3.4 A visual interface to define topological constraints

In this part, we present a visual interface to define topological integrity constraints. Specifically, a dialogbox in which the user can choose a pair of entities, a topological relation or a set of topological relations, and a specification (see Figure 3).

Topological constraints are defined following the lists of operations given here:

1. Choose a first class of entities.
2. Choose a second class of entities.
3. Choose a relation or a set of relations among the list proposed.
4. Define the specification.



**Figure 3.** the definition interface of topological constraints.

In the case shown on Figure 3, the constraint defined is:  
(Road, Inside, Building, Forbidden)

The dialogbox shows a schema that illustrates the topological relation chosen in the constraint definition.

This interface has been designed using VisualC++.

### **Examples of topological constraints**

C1(Road, Cross, Building, Forbidden)

C2(Sluice, Joint, Waterpipe, Exactly 2 times)

This visual interface allows one to define constraints based on topological relations, as a first step. The next step is to translate them into a language capable of checking them. Such a language is out of the scope of this paper, but is not out of the scope of the whole study on topological consistency of spatial data.

## **4. Correcting Scenario Computation.**

The goal is to define a model to compute corrections to topological integrity constraint violations (topological errors). The model described in this part deals only with constraints defined using the *forbidden* specification. This specification is most useful, and the most common one. In addition, such constraints are very easy to describe. It is easier to define a case that should not happen than to define a case that must exist. A topological constraint is then defined as an inconsistent topological relation between two geographical objects.

Since an error is defined as a forbidden topological relation between two objects, the way to correct an error will be to change the topological relation between those objects. A set of correcting scenarios will be computed by applying several kinds of changes to both objects involved in the forbidden topological relation (together or one after each other). The changes proposed are the following :

- Objects modification :
  - Moving the objects.
  - Reshaping the objects.
- Deleting one object.
- Object splitting (so creating a new object).

### **4.1 Objects modification**

We present in this part two different kinds of modifications. Changes, as *moving an object*, ensure that the surface area of object remains unchanged. Other changes, as *reshaping*, have been designed to leave the topological relation between the reshaped object and the other objects of the databases unchanged (objects not involved in the forbidden relation).

#### **4.1.1 Moving**

One of the two objects involved in the forbidden relation will be moved according to *main directions*. We use the letter A for the moving object and the letter B for the other object.

The main directions are :

- X axis
- Y axis
- perpendicular to B (when possible)
- parallel to B (when possible)
- along A (when possible)

For the moves *perpendicular to*, *parallel to* and *along*, if the object is not a straight line, the direction has to be defined regarding to the boundary of the object :

- for a line : the two end points,
- for a region : the boundary segment that is the closest to the other object.

Such a correction does not change the length of a line, or the area of a region. The relative position of the two object is the only change. Consequently, determining all the possible moves of one object leads to all the possible new scenes.

For each direction, A can be moved according two ways. An ending condition to the move is when the topological relation between the two objects changes. The new scene is stored into a list of correcting scenarios and the object A is moved again until the relation become disjoint. Each time a new topological relation is reached, the scene is added to the list.

As disjoint is as well a topological relation, the last scene is stored. Nevertheless, since a lot of positions are available for A (in this particular case), a minimum distance between the two objects is defined. This distance depends on the precision of point coordinates in the database. The stored scene based on the disjoint relation is the one in which the distance between the two objects is the minimal distance set for the database.

The following correction algorithm computes the correcting scenarios based on main directions:

```

Ri = the forbidden relation between n A and B
Er = the list of correcting scenarios
For both object, A and B
  A does not move (resp ectively B)
  For each direction
    For both way
      Rc (current relation between A and B) = Ri
      Repeat
        Move B until  $R(A,B) \neq R_c$ 
        If  $R(A,B) \neq R_c$  then
          Add R(A,B) to Er
        End If
        Rc = R(A,B)
      Until Rc = Disjoint
    End For
  End For
End For
    
```

**Figure 4.** Main direction correcting algorithm.

#### 4.1.2 Reshaping

Reshaping means moving a part of an object, leaving the other part unchanged. The goal of such a correction is to change the topological relation between the two objects without changing the relations with the other objects of the database. This kind of correction will affect the length of a line or the area of a region. Consequently, it will have to be applied to both objects (one after each other, always keeping one object unchanged) in order to determine all the possible new scenes. Such corrections will be used to adjust the borders of two closed regions, a line and the border of a region, or two lines. The adjustment will be made by a force-fitting algorithm that will snap characteristic points of A onto characteristic points of B. A characteristic point is a point used in the shape definition of the object (for example to describe the boundary of a region or a line). Homologue points are defined as a pair of very closed characteristic points, one belonging to A and the second to B, that will be snapped onto each other by the force-fitting algorithm.

There are two steps in the reshaping process :

- homologue points finding
- force-fitting algorithm

### Homologue points finding

The goal is to find which point of A and B will be matched. This process is defined by the two following steps :

1. Compute the distance between all points of A and all points of B. Store the results in a *distance matrix*.
2. The homologue points are each couple of points for which the distance is minimum and under a value  $d$ , set by the end-user.

### Force-fitting algorithm

The goal of this algorithm is to defined how to adjust very closed objects. It works on homologue points that have been computed by the previous process. The algorithm is described on Figure 5.

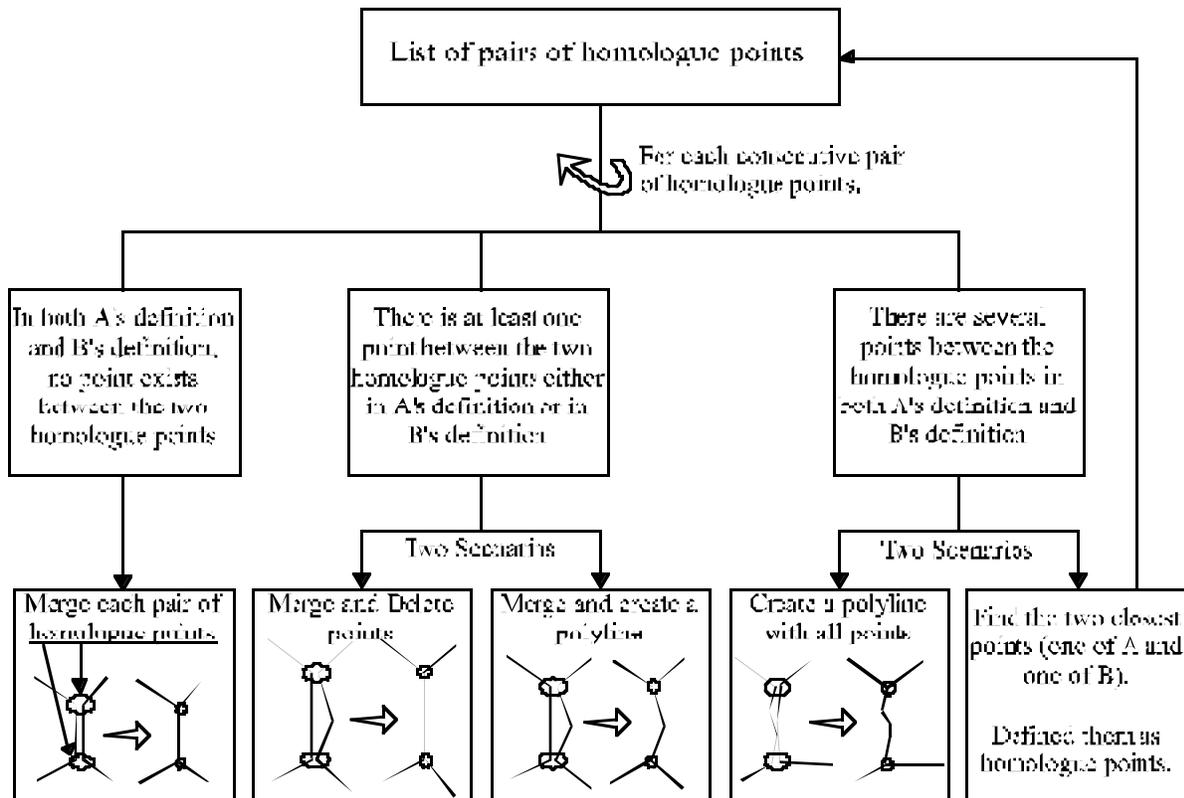


Figure 5. Force-fitting algorithm.

Applying these algorithm to all the *homologue point* provides all the possible scenarios of correction according to characteristic point matching.

For a line, only one of the ending points is allowed to move. Some other characteristic points can also be moved at the same time.

#### 4.2 Deleting one object.

This correction is useful when an object have been digitized twice. Two objects very closed to each other can then be found. Two corrections are possible :

- keeping A and removing B
- keeping B and removing A.

This leads to two correcting scenarios.

### 4.3 Object splitting (Creating an object).

The last way to change the topological relation between two objects is to split one of them into two different sub-objects. The only condition to check is that the two new topological relations are different from the previous one. This kind correction is useful to keep the planarity of a map. For example, when two lines crossing each other is a forbidden topological relation, a way to correct such an error is to create a point at the intersection of the lines and to split one of the line into two parts.

Such a correction can be proposed when the forbidden topological relation is such as one of the two objects shares a part of its interior with the interior or the boundary of the other object :

$$(O1^{\circ}O2 =) \quad (O1^{\circ}O2^{\circ} =)$$

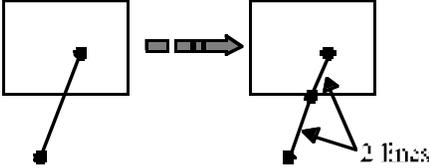
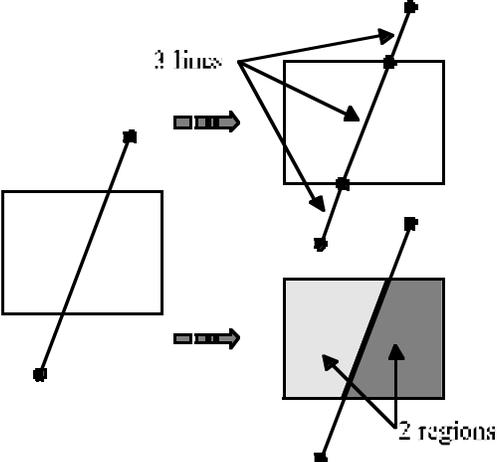
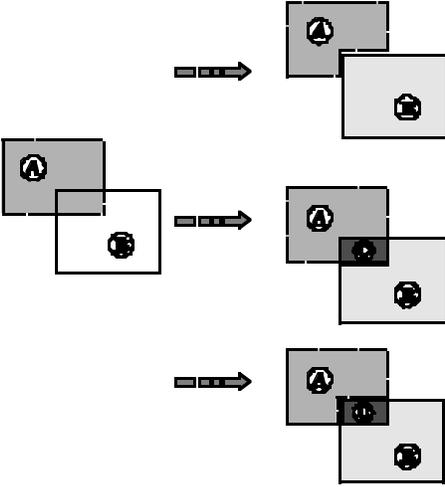
The corrections are :

- to split one of the two objects into several parts (2 or more)
- to create a new object based on the shared part and removing this part from each other object.

The following table gives the possible corrections for lines and regions.

Topological relation to correct	Correction schema	New relations
$L_1 \ L_2 =$		$L_1 \ L_2 =$ $(L_1 \ L_2 =)$ $(L_1 \ L_2 =)$
		$L_1 \ L_2 =$ $L_1 \ L_2 =$
$L_1 \ L_2 =)$ $L_1 \ L_2 =)$		$(L_1 \ L_2 =)$ $(L_1 \ L_2 =)$ $L_1 \ L_2 = 0$

**Table 2.** Line-line splitting corrections.

<p><u>Line - Region</u></p> <p>(L R =) (L R° =)</p>		<p>L R =</p> <p>L R =</p>
<p>L R =) (L R° =)</p> <p>L R =</p>		<p>L R =</p> <p>L R =</p> <p>L R =</p> <p>L R =</p>
<p><u>Region - Region</u></p> <p>R1 R2 =</p>		<p>R1 R2 =</p> <p>R1 R2 =</p>

**Table 3.** Line-polygon and polygon-polygon splitting corrections.

### 5. Correction Scenarios Presentation and Application

For each topological error, a list of correcting scenarios will be computed. The last step of the correcting process is then to choose which one to apply. To help the end-user to select the appropriate correction, the list of correcting scenarios will be presented using filtering and sorting process.

### **Filtering process**

1. All the correcting scenarios in which the topological relation is used in a topological constraint are removed from the list of corrections. This will be applied when there is more than one constraint defined for a given pair of geographical objects.
2. For correcting scenarios obtained by moving one object, a maximum range is defined. All corrections for which the moving distance is over the threshold are removed from the list of corrections.

### **Sorting process**

The end-user can specify one parameter that will help him to find the appropriate correction. Correcting scenarios where this parameter is verified are proposed first. The possible values for this parameter are :

1. keeping the area of the object unchanged
2. minimum distance move
3. border adjustment (result of force-fitting first)
4. keeping two objects
5. specifying the new topological relation

Those two process will facilitate the choice of the correcting scenario to apply. If the end-user cannot find an appropriate correction among the list proposed, a set of tools will be provided in order to let the end-user modify the geographical object.

## **6. Bibliography**

- Cui Z., Cohn A.G., et Randell D.A., 1993 Qualitative and Topological Relationships in Spatial Databases, in *Proceedings of the Third Symposium on Large Spatial Databases*, Singapore, June 23-25, 1993 (SSD'93). Lecture Notes in Computer Science n°692. pages 296-315.
- Egenhofer M. J. & Franzosa R. D., 1991 Point-set topological spatial relations, in *International Journal of Geographical information Systems*, volume 5, number 2, April-June 1991. pages 161-174.
- Egenhofer M. J. et Herring J. R., 1990 A Mathematical Framework for the Definition of Topological Relationships, in *Proceedings of the 4th International Symposium on Spatial Data Handling*, Zurich, 1990 (SDH 90). pages 803-813.
- Egenhofer M. J. et Herring J. R., 1991, Categorizing Binary Topological Relations Between Regions, Lines, and Points in Geographic Databases. Technical Report, Department of Surveying Engineering, University of Maine, Orono.
- Egenhofer M. J., 1991 Reasoning about Binary Topological Relations, in *Proceedings of the Second Symposium on Large Spatial Databases*, Zurich, 1991 (SSD'91). Lecture Notes in Computer Science n°525. pages 143-159.
- Laurini R. et Milleret-Raffort F., 1994, Topological Reorganization of Inconsistent Geographic Databases : a Step Toward their certification, in *Computer and Graphics*, volume 18, number 6, nov-dec 94, p. 803-813.
- Plümer L., 1996, Achieving Integrity and Topology in Geographical Information Systems, In: *First International Conference on Geographic Information Systems in Urban Regional and Environmental Planning*, Samos, Greece - April 1996, pages 45-60.
- Ubeda T. & Servigne S., 1996a, Capturing Spatial Object Characteristics for Correcting and Reasoning, In: *Proceeding of Joint European Conference and Exhibition on Geographical Information*, Barcelona, Spain, March 1996 (JEC-GI 96), pages 24-33.
- Ubeda T. & Servigne S., 1996b, Geometric and Topological Consistency of Spatial Data, in the Proceedings of the First International Conference on Geocomputation, Leeds, UK, 17-19 September 1996, pages 830-842.