Maintenance of Binary Topological Constraints through Active Databases

Claudia Bauzer Medeiros
cmbm@dcc.unicamp.br
DCC - IMECC - UNICAMP
13081-970 Campinas SP Brazil

Mariano Cilia
mcilia@dcc.unicamp.br
DCC - IMECC - UNICAMP
13081-970 Campinas SP Brazil

Abstract

This paper presents a system developed at UNICAMP for automatically maintaining topological constraints in a geographic database. This system is based on extending to spatial data the notion of standard integrity maintenance through active databases. Topological relations, defined by the user, are transformed into spatial integrity constraints, which are stored in the database as production rules. These rules are used to maintain the corresponding set of topological relations, for all applications that use the database. This extends previous work on rules and GIS by incorporating the rules into the DBMS rather than having them handled by a separate module.
Keywords: Active databases, topological relations, spatial integrity

1 Introduction

Active databases are systems that respond to events generated internally or externally to the system itself without user intervention. The active dimension is supported by production rule mechanisms, provided by the database management system (DBMS). *Production rules* are usually defined as clauses **If X then Y**, where X is a predicate to be tested, and Y is an action to be performed if the predicate is satisfied. Active databases have been suggested as an appropriate solution for constraint maintenance in the case of standard applications. No experiment has been made, however, of applying these ideas to georeferenced data.

The use of rules in GIS is usually treated either by means of an external mechanism which is coupled to the geographic database or in the context of deductive systems. The integration and management of spatial relations within an active DBMS provides the following advantages over the standard rule-based approaches:

- the specification and preservation of spatial relations is handled by the DBMS itself, regardless of any application code, allowing independent evolution of both data and application;
- the same underlying geographic database may be used by all applications that need to have the same view of the world, without requiring application rewriting for constraint checking;
- different applications can have distinct views of the same database, by enabling and disabling different sets of rules;
- spatial relations can be treated at the same architectural level as the data they refer to. This helps low level implementation issues, such as leaving the optimization of spatial operations to the DBMS.

The results presented in this paper are part of an ongoing research at UNICAMP, to extend the paradigm of active database systems to incorporate maintenance of spatial relations in the presence of updates. Given, however, the theoretical and implementation problems in specifying and maintaining general spatial constraints among georeferenced entities, we have so far restricted ourselves to the important problem of binary topological constraints among objects stored in vector format. This solution has been implemented at the Department of Computer Science, UNICAMP. The main results presented are the following:

- description of an extended active database architecture for managing georeferenced data;
- overview of the active DBMS prototype developed to enforce topological binary relations.

The theoretical background adopted here for specifying binary relations is based on [CevO93]; the active database system was implemented by combining the results reported in [CAM93] with the object oriented spatial data model of [CFS+94]. The prototype described in the paper is being tested on vector data collected for different kinds of phenomena in a specific region of the state of São Paulo, in the domain of telecommunications planning. Data samples are being provided by the R&D Telecommunication Center at Campinas, who are also responsible for the definition of the topological relations. The tests performed correspond therefore to current real world conditions.

This paper is organized as follows. Section 2 describes present research in the area of GIS and rule-based systems. Section 3 gives an overview of the theoretical basis for defining binary topological relations, and their inclusion in an object georeferenced data model. Section 4 describes the prototype developed. Section 5 shows how topological constraints were implemented as rules in this prototype. Finally, section 6 presents conclusions and future directions.

2 Active databases and GIS

Active object oriented DBMS have proved useful in the area of scientific database applications. The most common application for active databases is integrity maintenance. An *integrity constraint*, in a database environment, is a statement of a condition that must be met in order to maintain data consistency. DBMS have limited support for automatic maintenance of integrity constraints. Active databases are a solution to this problem, since rules can be activated upon update requests. Constraint specification and maintenance can then be kept independent from application development. The paradigm of active databases is useful for implementing or extending several database functions. Examples of present research in active databases are described in [Cha92, Buc94].

The basic model for active DBMS is by [DBM88], where a production rule is specified as a triple (E, C, A). This can be translated into the statement **When E If C Then A**. **E** stands for the *Event* upon which the rule is triggered, **C** the *Condition* to be tested, and **A** the *Action* to be performed if the condition is met. Events may be internal to the database (e.g., queries, updates), or external (e.g., hardware interrupts). Conditions are usually predicates over database states, which can be evaluated by performing a query in some language (usually the database native query language). Actions are expressed in the database programming language and may range from atomic actions to complete programs. From now on, the term *rule* in this paper will denote this triple.

In an active DBMS, rules are stored and managed together with data. The activation and computation of rules are basically performed in three steps [MT94]:

- rule *triggering*: a given **E**vent is signalled, which demands checking the rule(s) to be executed (fired). This step requires going through the stored rules to select the appropriate ones.
- rule *evaluation*: the **C**ondition part of the chosen rule is checked, to verify if it is satisfied;
- rule *execution*: the **A**ction part of the rule is performed, if the condition is met.

The approach of combining rules and data handling is not new in the area of GIS research. In such a context, rules are used in a deductive role: they help process queries, perform data analysis or derive relationships among data. However, in most cases, rules are not integrated at the data management level (as opposed to the active database philosophy). Rather, rules and data are treated by different handlers as isolated components (the rule management system and the database management system), which need intermediate modules to allow their communication. Usually, GIS that contemplate rules rely in coupling modules for handling georeferenced data (e.g., for analysis or display) to an external rule management system. The integration of these components is provided by an expert system shell, which also supports spatial decision taking (see, for instance, [AD90]).

Coupling the rule base to the database then requires several levels of modelling and translation among modules, as well as demands that the user learn different languages (to access data and to specify rules). A typical example of such an approach appears in [KWB+93]. In this research, Arc Info provides the spatial data and a rule base is used to feed a reasoning program for classification, refinement and generalization.

Other examples of research in rules and georeferenced data are the use of expert system shells (e.g., [SR93, SRD+91, LL93]), spatial decision support systems built by coupling spatial databases and rule sets (e.g., [YS91, AYA+92]), expert systems for dense map name placement [DF92], rule based systems for querying [WCY89] or deductive systems based on PROLOG [Web90]. Such approaches are not satisfactory from a data management point of view. They make applications very sensitive to modifications in the relations among data, and leave to programmers the burden of having to know and check all relevant constraints at each step. These solutions are language-based and therefore require additional effort to integrate the rule programming language with the chosen DBMS.

The mismatch between the chosen rule language and the GIS database demand that users create complete new programs for each data analysis performed using the rules. Typically, users are required to apply successive sets of queries, store the results in intermediate files, and then apply the chosen set of analysis and correlation rules among these files (e.g., [LHM94]). The active database approach, on the other hand, allows the insertion of rules in the database which can then be used in combining results of queries without these intermediate steps.

Some steps towards bridging the gap between rules and data management can be found in, for instance, [SA93], [PMP93, AWP93]. [PMP93] discuss the importance of a rule management component as an integral part of a spatial database. One approach is to consider rules during the database modelling process: in [SA93], rules are modelled as a specific class in an object oriented design methodology for geographic systems and implemented in an object oriented language. Another approach is the use of deductive databases: [AWP93] analyze topological relationships expressed in a logic language within a deductive database, but the emphasis is on constraint specification and not maintenance.

3 Topological relations - adopted model

This section presents the model adopted for topological relations, and the algorithm for transforming them into (E,C,A) rules.

[Gut94] distinguishes between the following classes of spatial relationships:

- topological relations - those that are invariant under topological transformations like translation, scaling or rotation. Examples are *adjacent, inside, disjoint.*
- direction relations - those that establish relative positioning of elements within some positioning system (e.g., *north, south*).
- metric relations - those that can be expressed in scalar form defining measurement values (e.g., *distance*).

Our model for defining topological constraints is based on [CevO93]. This work originated with the definition of the *4-intersection model* (see [EF91]), which describes binary topological relations among area objects (connected regions without holes), later extended by [HT92] to include line and point objects.

The 4-intersection model considers all the binary combinations of *region intersection*: a region A is a 2D point set with a connected *interior* A^0 and *boundary* δA. If the $exterior A^{-1}$ is considered, one obtains the *9-intersection*

matrix. As stressed in [CSE94], the 9-intersection matrix forms the base set from which database users can construct more complex relationships that are appropriate to their application domain. The 4-intersection and the 9-intersection matrices are shown below, depicting all types of intersection between two areas (regions) A and B. For instance, element (2,2) of the matrix indicates whether the boundaries of A and B intersect. The third column and the third row belong to the 9-intersection matrix and correspond to including the *exterior* of A or B in the intersection evaluation.

$$\left(\begin{array}{cc|c} A^0 \bigcap B^0 & A^0 \bigcap \delta B & A^0 \bigcap B^- \\ \delta A \bigcap B^0 & \delta A \bigcap \delta B & \delta A \bigcap B^- \\ \hline A^- \bigcap B^0 & A^- \bigcap \delta B & A^- \bigcap B^- \end{array}\right)$$

The 4-intersection matrices just indicate whether the intersection between two objects is empty or not, but not the nature of the intersection. [CevO93] extended the 4-intersection matrix to consider the dimension *dim* of the intersection between any two objects (the *dimension-extended method*) and showed formally that such an approach could be synthesized, for ease of handling by humans, into 5 mutually exclusive topological relationships. These relationships – *touch, in, cross, overlap, disjoint* – express when the dimension *dim* of the intersections between two objects can be *empty*, a *point* (0D), a *line* (1D) or an *area* (2D). In order to express all 52 valid binary relationships among these objects, [CevO93] also introduced three *boundary operators*, which allow returning the boundary of an object (a boundary line of an area or the two end points of a line).

All binary topological relations can be therefore checked against these five mutually exclusive relationships, being defined by the above expressions and boundary operators. The relationships between two objects A and B (of types line, point and area) can be expressed as follows:

- $(A\ in\ B) \Leftrightarrow (A \bigcap B \neq A) \wedge (A^0 \bigcap B^0 = \emptyset)$
- $(A\ touch\ B) \Leftrightarrow (A^0 \bigcap B^0 = \emptyset) \wedge (A \bigcap B \neq \emptyset)$
- $(A\ cross\ B) \Leftrightarrow (dim(A^0 \bigcap B^0) = max(dim(A^0), dim(B^0)-1) \wedge (A \bigcap B \neq A) \wedge (A \bigcap B \neq B)$
- $(A\ overlap\ B) \Leftrightarrow (dim(A^0) = dim(B^0) = dim(A^0 \bigcap B^0)) \wedge (A \bigcap B \neq A) \wedge (A \bigcap B \neq B)$
- $(A\ disjoint\ B) \Leftrightarrow (A \bigcap B = \emptyset)$

Not all relationships do apply to all object types. *Overlap* relationships, for instance, can only apply to area/area and line/line but not to other situations.

We implemented the dimension extended concept in our active system, described next.

4 System description

Our active spatial database prototype was implemented in Smalltalk and runs in UNIX workstations. Implementation details are omitted, since they are beyond the scope of this paper. The system is object-oriented. Since there is no standard definition for object-oriented models, we follow [Bee89]'s class-based framework. An object is an instance of a class and is characterized by its *state* (set of attribute values), and *behavior* (set of methods that can be applied to the object). An object o can be constructed out of other objects o_1,...,o_n, in which case o is called *complex* and o_1,...,o_n are called the *components* of o. If an object is not complex, then it is called *simple*. Classes can be structured into inheritance hierarchies. Objects communicate with each other via methods.

4.1 Basic architecture

The active architecture had to satisfy two main criteria: adequate support for specification and management of georeferenced data; and proper integration of rules and spatial data. The solution was to combine a georeferenced data model to an active database architecture.

The first criterion was met by designing an architecture that would automatically embed spatial modelling concepts into the database data model. We chose the georeferenced object oriented data model of [CFS+94] as the basis for such an architecture. It allows separating logical specification of entities from their implementation. This presents the advantage that user defined concepts can be directly mapped into database classes, thus minimizing the mismatch between modelling and implementation.

In this model, geographic reality is modelled according to four levels: the *real world level*, which concerns geographic reality; the *conceptual level*, which describes entities at a high level of abstraction; the *representation level*, where different representations are defined for a given conceptual entity; and the *implementation level*, where the actual database structures and access methods are defined.

The second criterion – integration of rules and data – was based on extending the architecture of the Sentinel project [CAM93] with the spatial classes of the [CFS+94] georeferenced data model. Sentinel is an active object oriented database system which was developed to support multimedia DBMS for scientific applications. It must be stressed that we did not use the actual Sentinel implementation. Rather, we just adapted its active architecture description to build our prototype.

We chose the Sentinel architecture for two main reasons:

- it treats rules as objects, which unifies their management from a database point of view; and, more important,
- it also treats events as objects, which allows associating and storing several properties inside events (e.g., temporal properties, spatial dependencies). This helps the specification and maintenance of control over different types of spatio-temporal relationships.

The definition of events as objects also has the advantage of fostering an open system, since event objects can be combined by additional operators, thus extending the event algebra according to the users' wishes.

Events can be primitive or composite. Primitive events are generated by method activation, by the system clock or raised by the application. Composite events are generated by combining primitive and composite events by means of predefined operators (e.g., sequencing or disjunction).

Like in Sentinel, our objects are classified into three main groups:

- passive: those that can accept method execution but do not generate (raise) any event;
- reactive: those where events may be generated when methods are executed;
- notifiable: those that can be notified when events are generated by reactive objects.

Reactive classes are defined as standard object classes with an additional event specification interface. This allows connecting these classes to different types of events and rules. Rules are notifiable objects. Thus, when an event is

raised at a reactive object, the corresponding set of rules are notified and can be executed.

The notification mechanism is based on the idea of *subscription*. This allows connecting rules (notifiable objects) to the corresponding events (rules subscribe to events raised by reactive objects). Rules can dinamically subscribe and unsubscribe to events defined at reactive classes. This helps modelling the dynamics of the real world. Subscription is a means of optimizing rule activation upon event raising.

Rules can be defined over an object or all objects of a class. Whereas the original Sentinel architecture specifies that rules must be associated only to primitive events, we introduce the notion of subscription to composite events, speeding up rule execution.

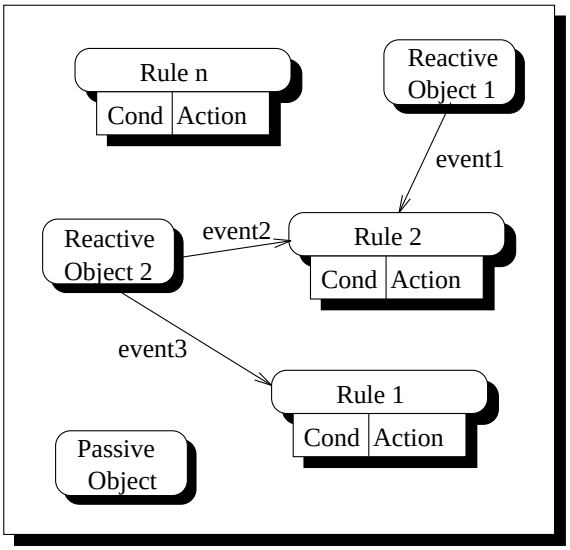

Figure 1: Association between rules and event objects

Figure 1 shows an example of the connection among different types of objects. Reactive Object2 notifies Rule2 when Event2 is raised, and Rule1 when Event3 is raised. Reactive Object1 notifies Rule2 when Event1 is raised. This notification corresponds to executing (firing) the corresponding rules; it is described as "Rule2 subscribes to < Event1 raised at Object1>", and "Rule2 subscribes to < Event2 raised at Object2>". Passive objects cannot notify any rule, though an object can change from active to passive during its lifetime, if the user so desires. RuleN does not subscribe to any event, though again this situation may change dinamically, according to user needs.

If Reactive Object2 were a Transformer, subject to integrity constraint "Every Transformer must be placed on a Pole", then $Event2$ might be an update of the Transformer's location, and $Rule2$ a rule to check this constraint. An execution of the *position_Transformer* method ($Event2$) would notify this rule, whose Condition field would contain "$\exists p \in Pole\ s.t. Transformer\ in\ p$".

4.2 Modelling and implementing geo-objects

The model of [CFS+94] was implemented in our prototype, with a single representation per modelled phenomenon. The world is assumed to be modelled in database classes – *geo-referencing classes* – whose objects describe regions of the Earth's surface, called *geo-regions*. At the conceptual level, the model supports both the field view and the object view of the world ([Cou92]), distinguishing between two basic classes of database entities: geographic fields – for manipulation of continuous variables – and geographic objects – for specification of identifiable entities.

As [CSE94], we also assume, for the purpose of topological constraints, the object view of the world, where a topological or vector data model represents spatial objects, and we ignore the specification of geo-fields.

Geographic objects (*geo-objects*) are identifiable entities in the real world and can be elementary, compound or weak. They have three main components: non-spatial attributes, other geo-objects, and a *Location* attribute, describing the object's spatial characteristics. The geometry of geo-objects, described at the *Representation Level* of the model, is described in terms of point, line and polygon features. These features are also objects belonging to the *Geometric Representation* class hierarchy. The location L of a geo-object is also specified as a complex object.

An *elementary geo-object* has no geo-objects as components. A *compound geo-object* is a geo-object constructed out of other geo-objects. A *weak geo-object* is a geo-object that contains only the location attribute and exists only as long as it is part of a (unique) compound geo-object.

In our prototype, the crucial part for expressing topological relations is the *Location* component of geo-objects. This was implemented as a class hierarchy, rooted at class *Location*, where objects are described as *Points, Lines* or *Polygons*. This hierarchy can be further specialized, according to the user's needs. Geo-objects can be either passive or reactive, depending on the type of checking the user wants to perform on them.

A compound geo-object O can thus be described as:

O :: tuple(tuple(NonSpatial ...), set(Geo-objects), L)

Assume we want to describe a telephone network. We may then define classes *CableSection* (where the *Location* component is based on *Line* segments), and *Connections* (*Location* based on *Point* elements). Individual cable section or connection objects may be weak or elementary. Using objects from these classes, we may construct the network description as nested lists of *CableSection* and *Connection* segments. This network uses data about street location for georeferencing purposes.

This type of modelling eliminates the need for associating real world objects to layers in order to insert them in the database. A layer is simply created by defining sets of interrelated objects (e.g., network layer), and geographic constraints among layers (e.g. telephone layer and street layer) are expressed as constraints among the respective object representations.

5 Expressing and Maintaining Constraints

Several algorithms have been reported in the literature for transforming integrity constraints expressed in a logic language into rules (e.g., [Mor84, WF90]). Since we used an object oriented database model, we adopted the mechanism of [MA94]. It consists of analyzing the constraint to identify classes and objects that may have to be checked whenever an update occur. The database schema is then analyzed to identify additional classes to be checked (due to constraint inheritance) and to eliminate superfluous entities (given method signature).

The previous section showed how we combined a georeferenced object-oriented data model and an active database

architecture in order to obtain an active object oriented spatial prototype. This section shows how this prototype is being used to enforce topological relations expressed as constraints on geo-objects.

A constraint mechanism based on active databases relies on two processes: (1) *transformation* of the constraint into (E,C,A) rules, which involves *event determination*; and (2) *maintenance* of the constraint thus expressed. We assume that each binary topological relation is expressed as a constraint in a logic language[1], using the topological relationships of [CevO93].

(1) *Transformation Step*

The *transformation step* we used is basically the same algorithm defined in [MA94]. This algorithm is based on generating a set of (E, C, A) rules using only constraint and database schema specification.

Initially, each constraint expression is analyzed together with the database schema and a spatial relationship table to determine which updates may violate it. Then, a set of events is generated, corresponding to pairs (u, o_i) or (u, C), where u is an update method (*New, Delete, Modify*), o_i is a specific reactive geo-object and C is the name of a reactive geo-object class. In other words, relevant events are those that update reactive geo-objects (either specific ones or any one in a given class). The concerned geo-objects must be reactive in order to notify the appropriate rules.

Event determination is divided into three categories according to the type of variables involved in the logically formulated constraint. Category1 involves *Named objects* (e.g., *Park Tower Bldg*) or constraints where all objects are bound by existential quantifiers (e.g., $\exists c1 \in Cables$, where $Cables$ is a geo-object class). Category2 involves constraints where all object variables are bound by universal quantifiers (e.g., $\forall\ c1, c2 \in\ Cables, (c1\ disjoint\ c2)$). Category3 concerns all other cases, and requires additional semantics analysis. The objects and classes involved in Category1 expressions can only be violated by *Modify* and *Delete* updates; those involved in Category2 can be violated by *Modify, Delete* and *Insert*.

(We stress that topologic relationships are checked against the *Location* components of geo-objects involved in a constraint. Thus, the constraint that forces a $Transformer$ to be placed on a $Pole$ is transformed into a constraint that checks the coincidence of the *Locations* of the two objects.)

The table below shows how update events are linked to the variables in a constraint. Existential quantifiers indicate that *modify* and *delete* methods may violate the constraint if applied to the *Location* component of the bound variable; universal quantifiers require checking of *insert* and *modify* methods; and named objects need checking upon *modify* or *delete* operations. Furthermore, additional checking may be required when deleting the last object of a class which is subject to an integrity constraint. For instance, Category2 constraint $(\forall\ c1, c2 \in\ Cables, (c1\ disjoint\ c2))$ needs to be checked if a new cable c is inserted in class $Cables$ or if the *Location* of an existing cable c is modified.

$\exists$	$\forall$	Named Object
delete modify	insert modify	delete modify
delete last	delete last	

The generation of the (E, C, A) rules, once events are determined, is described in [MA94] and beyond the scope of this paper.

[1] Formally specified in [Cil95]

(2) *Constraint Maintenance*

The *Constraint Maintenance* process requires three steps: (a) Event detection; (b) Condition evaluation; and (c) Action execution. The *event detection* step is automatic in our architecture, because of the subscription mechanism. Indeed, once the events are defined at the *transformation step*, it is enough to make the appropriate rules subscribe to these events. Then, as soon as an update method that may violate a constraint is executed on a reactive geo-object, this object notifies the corresponding rule. This, in turn, activates the execution of the condition evaluation.

Condition evaluation corresponds to querying the space of geo-objects involved in the constraint. In our implementation, we restricted ourselves to geo-objects whose geometries are *Points*, *Linesegments* and simple convex *Polygons*. Consider a topological constraint that involves some relation $\mathcal{R}$. The basic query is *find all geo-objects that have the topological relation $\mathcal{R}$ with the reactive geo-object which notified the rule.* Thus, the query is limited, on one side, to the reactive object which raised the event, but potentially, on the other side, to the whole *Location* space covered by the database.

This is therefore very costly for general cases (constraints involving only class names and no named objects). Constraints which relate existing individual (named) objects are easily checked since they just affect these objects and thus need not go through the whole database upon an update.

Finally, *action execution* corresponds to performing a method that is defined in the notified Rule object. In our implementation, this method is limited to the *abort* action, i.e., updates cannot be performed if they violate a constraint. More sophisticated mechanisms can be envisaged – e.g., corrective measures, but they involve more knowledge of an application semantics.

Example:

We provide a small example to give the general idea of the mechanism. Consider the constraint that specifies that all aerial telephone cable sections must be supported by a pair of poles – i.e., each cable section is bound by a pole in each extremity. In [CevO93], these extremities are obtained by applying the *boundary* operators f (from) and t (to) to a line. The constraint can be defined as[2]

$$\forall c \in\ AerCableSection, \exists\ p1, p2 \in\ Pole\ s.t.\\ f(c) = p1\ \wedge t(c) = p2\ \wedge p1\ \neq\ p2.$$

Event Definition. This is a Category3 constraint (and therefore requires semantics analysis to restrict the possible events). This constraint may be violated if a cable (universally bound variable) is inserted or changes position, or if a pole (existentially bound variable) is deleted or changes position. The rules $R1, R2$ that check this constraint are notified by the following complex events

$$R1 \text{ — } Event1 = < insert, c > \vee < modify, c.Loc >\\ R2 \text{ — } Event2 = < delete, p > \vee < modify, p.Loc >$$

where $c.Loc$ and $p.Loc$ are the *Location* components of the corresponding geo-objects.

Figure 2 shows the kinds of user actions that may violate this constraint: a *CableSection* may have its *Location* modified, or one *Pole* may be deleted or change *Location*. As well, the insertion of a new *CableSection* needs checking the existence of the two *Poles* (not shown in the figure).

[2] We stress that this is a simplified example, added to help understand the mechanism. The database schema is not included for lack of space. Details are provided in [Agu95].

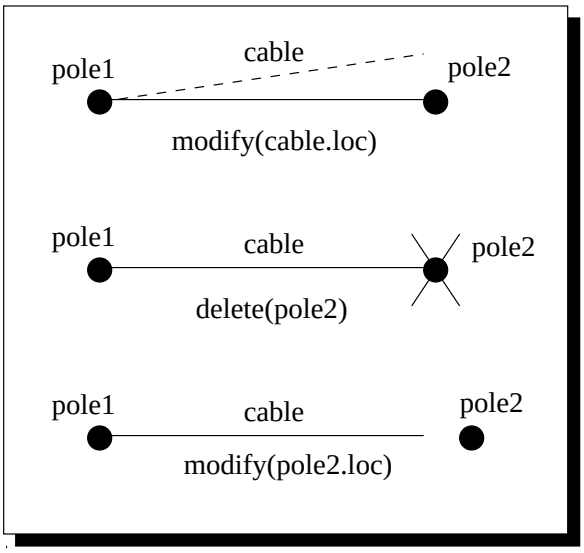

Figure 2: Example of Constraing Violation

Constraint Maintenance. When these events are raised, the rules are notified to check the conditions. For instance, if $Event1$ is raised, rule $R1$ is fired and condition "find $p1, p2$ s.t. $f(c) = p1.Loc \ \wedge \ t(c) = p2.Loc$" is checked. If the answer does not return two $Poles$, then the update is not allowed. Rule $R2$ is executed in a similar way.

6 Conclusions and future work

This paper described the use of an active spatial DBMS in the management of binary topological relations, for data in vector format. This DBMS was implemented in a prototypical form at the Computer Science Department, UNICAMP, using as a basis an extension of the architecture of the active DBMS Sentinel. The implementation, in Smalltalk, allows definition of georeferenced data according to an object-oriented model [CFS+94].

The main goal of the experiment reported here was to assess the feasibility of handling topological relations in an active database framework, with actual data. We have made some tests with simulated data and will now proceed with real test cases, which are restricted to point and line segment data.

We are also analyzing the suggestion of [CSE94] to optimize the computation of topological relations. This requires optimizing the evaluation of the condition component of the rule (i.e., spatial query optimization).

Topological relations are just an (important) subset of spatial constraints. As shown by [PS94], the 4-intersection model must be extended with orientation information in order to better express spatial relationships. Thus, an extension to this work would be to consider such a framework which, in turn, requires sophisticated orientation definitions and additional complexity in the relation checking algorithms.

At the present stage of the prototype, there is no appropriate user interface. Rather, users interact with data by means of the DBMS environment, which does not allow cartographic visualization. Thus, another extension to this work would be to provide an appropriate interface.

Acknowledgements

The work described in this paper was partially financed by grants from FAPESP and CNPq, as well as grants from the CNPq ProTeM-CC project GEOTEC and the European Community ITDC project GEOTOOLS number 116-82152. The authors thank Juliano Lopes de Oliveira for his comments on previous drafts of this paper.

References

[AD90] M. Armstrong and P. Densham. Database Organization Strategies for Spatial Decision Support Systems. *International Journal of Geographical Information Systems*, 4(1):3–20, 1990.

[Agu95] C. D. Aguiar. Integration of Heterogeneous Database Systems in Urban Planning Applications. Master's thesis, UNICAMP, march 1995. In portuguese.

[AWP93] A. Abdelmoty, M. Williams, and N. Paton. Deduction and Deductive Databases for Geographic Data Handling. In *Proc Third International Symposium Spatial Databases - SSD*, pages 443–464, 1993.

[AYA+92] D. Abel, S. Yap, R. Ackland, M. Cameron, D. Smith, and G. Walker. Environmental Decision Support Systems Project: an Exploration of Alternative Architectures for Geographical Information Systems. *International Journal of Geographical Information Systems*, 6(3):193–204, 1992.

[Bee89] C. Beeri. Formal Models for Object-oriented Databases. In *Proc. 1st International Conference on Deductive and Object-oriented Databases*, pages 370–395, 1989.

[Buc94] A. Buchmann. Active Object Systems. In A. Biliris A. Dogac, M. Oszu and T. Sellis, editors, *Advances in Object oriented Database Systems*, pages 201–224. Springer Verlag, 1994.

[CAM93] S. Chakravathy, E. Anwar, and L. Maugis. Design and Implementation of an Active Capability for an Object Oriented Database. Technical Report CIS-TR-93-001, University of Florida, 1993.

[CevO93] E. Clementini, P. DiF elice, and P. van Oosterom. A Small Set of Formal Topological Relationships Suitable for End-user Interaction. In *Proc Third International Symposium Spatial Databases - SSD*, pages 277–295, 1993.

[CFS+94] G. Camara, U. Freitas, R. Souza, M. Casanova, A. Hemerly, and C. B. Medeiros. A Model to Cultivate Objects and Manipulate Fields. In *Proc 2nd ACM Workshop on Advances in GIS*, pages 20–28, 1994.

[Cha92] S. Chakravarthy. (editor) Special Issue - Active Databases. In *IEEE Bulletin on Data Engineering*, december 1992.

[Cil95] M. Cilia. Maintenance of Topological Relations Using an Active OODBMS. Master's thesis, UNICAMP, 1995. In preparation, in Portuguese.

[Cou92] H. Couclelis. People Manipulate Objects (but Cultivate Fields): Beyond the Raster-Vector Debate in GIS. In *Proc International Conference on GIS - From Space to Territory: Theories and Methods of Spatial Reasoning*, Springer Verlag Lecture Notes in Computer Science 639, pages 65–77, 1992.

[CSE94] E. Clementini, J. Sharma, and M. Egenhofer. Modelling Topological Spatial Relations: Strategies for Query Processing. *Computer & Graphics*, 18(6):815–822, 1994.

[DBM88] U. Dayal, A. Buchmann, and D. McCarthy. Rules are objects too: a knowledge model for an active, object oriented database system. In *Lecture Notes in Computer Science*, volume 334, pages 129–143. Springer Verlag, 1988. 2nd Workshop in OODBS.

[DF92] J. Doerschler and H. Freeman. A Rule-based System for Dense Map Name Placement. *Communications of the ACM*, 35(1):68–79, 1992.

[EF91] M. Egenhofer and R. Franzosa. Point-set Topological Spatial Relations. *International Journal of Geographical Information Systems*, 5(2):161–174, 1991.

[Gut94] R. H. Guting. An Introduction to Spatial Database Systems. *The VLDB Journal*, 3(4):357–400, 1994.

[HT92] T. Hadzilacos and N. Tryfona. A model for expressing topological integrity constraints in geographic databases. In *Proc International Conference on GIS - From Space to Territory: Theories and Methods of Spatial Reasoning*, Springer Verlag Lecture Notes in Computer Science 639, pages 251–268, 1992.

[KWB+93] C. Kontoes, G. Wilkinson, A. Burrill, S. Goffredo, and J.Megier. An Experimental System for the Integration of GIS Data in Knowledge-based Image Analysis for Remote Sensing of Agriculture. *International Journal of Geographical Information Systems*, 7(3):247–263, 1993.

[LHM94] P. Longley, G. Higgs, and D. Martin. The Predictive use of GIS to Model Property Valuations. *International Journal of Geographical Information Systems*, 8(2):217–236, 1994.

[LL93] Y. Leung and K. S. Leung. An Intelligent Expert System Shell for Knowledge-based Geographical Information Systems: 1 - the tools. *International Journal of Geographical Information Systems*, 7(1):189–200, 1993.

[MA94] C. B. Medeiros and M. J. Andrade. Implementing Integrity Control in Active Databases . *The Journal of Systems and Software*, pages 171–181, december 1994.

[Mor84] M. Morgenstern. Constraint equations: Declarative expression of constraints with automatic enforcement. In *Proc. 10th VLDB*, pages 291–300, 1984.

[MT94] D. Montesi and R. Torlone. A Rewriting Technique for Implementing Active Object Systems. In *Proc. International Symposium Object oriented Methodologies and Systems*, pages 171–188, 1994.

[PMP93] N. Pissinou, K. Makki, and E. Park. Towards the Design and Development of a New Architecture for Geographic Information Systems. In *Proc. 2nd International Conference on Information and Knowledge Management- CIKM*, pages 565–573, 1993.

[PS94] D. Papadias and T. Sellis. Quality Representation of Spatial Knowledge in Two-Dimensional Space. *The VLDB Journal*, 3(4), 1994.

[SA93] R. Subramanian and N. Adam. *Advanced Database Systems*, volume LNCS 159, chapter Applying OOAD in the Design and Implementation of an Intelligent Geographic Information System, pages 127–150. Springer Verlag, 1993.

[SR93] A. Srinivasan and J. Richards. Analysis of GIS Spatial Data Using Knowledge Based Methods. *International Journal of Geographical Information Systems*, 7(6):479–500, 1993.

[SRD+91] A. Skidmore, P. Ryan, W. Dawes, D. Short, and E. O'Loughlin. Use of an Expert System to Map Forest Soils from a Geographical Information System. *International Journal of Geographical Information Systems*, 5(4):431–446, 1991.

[WCY89] J. Wu, T. Chen, and L. Yang. QPF: a Versatile Query Language for a Knowledge-based Geographical Information System. *International Journal of Geographical Information Systems*, 3(1):51–58, 1989.

[Web90] C. Webster. Rule-based Spatial Search. *International Journal of Geographical Information Systems*, 4(3):241–260, 1990.

[WF90] J. Widom and S. Finkelstein. Set Oriented Production Rules in Relational Database Systems. In *Proc. ACM SIGMOD*, pages 259–270, 1990.

[YS91] Z. Yang and D. Sharpe. Design of Buffer Zones for Conservation Areas and a Prototype Spatial Decision Support System (SDSS). In *Proc GIS/LIS'91*, volume 1, pages 60–70, 1991.