An Integrity Constraints Driven System for Updating Spatial Databases

Alberto Belussi[1](contact author), Mauro Negri[2], Giuseppe Pelagatti[2], Fabio Spinazza[2]

[1] Dipartimento Scientifico e Tecnologico
Università degli Studi di Verona
Strada Le Grazie
37134 Verona, Italy
Fax.: +39-0458027929
belussi@sci.univr.it
[2] Dipartimento di Elettronica e Informazione
Politecnico di Milano
Piazza Leonardo da Vinci
20133 - Milano, Italy

Research Session.

Abstract. This paper describes a prototypal system which has been implemented in order to explore the possibility of using topological integrity constraints as interactive drivers to support spatial database updates. The idea of using constraints to drive updates is applied also in traditional (non-spatial) databases; for example, in order to preserve referential integrity, the user can be forced to select a value in a given set, instead of permitting him to write an arbitrary value and then checking that the value satisfies the constraint. This idea seems to be much more relevant in spatial databases, both because spatial data possesses a much richer set of constraints, and because spatial updates are more complex and error-prone than traditional, alphanumeric updates. The paper first defines formally a rather general spatial database environment with integrity constraints, then describes a prototypal system which has been built in order to explore the practical effectiveness of the general idea (the feasibility includes performance, because the constraints are used during the interaction with the user). The prototype which has been implemented is capable of driving updates on a restricted set of spatial data types (polygons only) and uses a restricted class of integrity constraints, with respect to the general definition; however, it is sufficiently powerful to express a wide range of constraints and to demonstrate the feasibility of the approach.

1 Introduction

Geographical information systems (GIS) are nowadays a widely used tool in many application fields that manipulate data describing phenomena or human activities that occur on the earth surface. The data sources that feed the spatial database of a GIS at the beginning of its life are mainly surveys from aerial photographs, topographic surveys, maps digitizing, etc. However, after the initial startup, other dataflows guarantee the update of the spatial and non-spatial information of a GIS.

As in traditional information systems also in GIS a frequent update activity is strategic for system availability and user confidence. Thus, it is important to design applications that support the user in this important task.

In traditional databases the update operations have to guarantee that the user's modifications/inputs satisfy the integrity constraints. Integrity constraints define the consistent states of the database. Notice that, some integrity constraints are not an obstacle for the update procedures, but, on the opposite, they can help the user to modify correctly the database. For example, in order to preserve referential integrity, the user can be forced to select a value in a given set, instead of permitting him to write an arbitrary value and then checking that the value satisfies the constraint. Here, we can say that the integrity constraint has somehow driven the update. This idea seems to be much more relevant in spatial databases, both because spatial data possesses a much richer set of constraints, and because spatial updates are more complex and error-prone than traditional, alphanumeric updates. In this paper we aim to investigate the possibility of applying integrity driven updates to geographical databases.

Spatial integrity constraints have been classifed in [2]. In this taxonomy Cockcroft identifies also the (sub)class of topological integrity constraints, emphasizing their role as basic constraints for the integrity of geographical data. Topology is a branch of geometry concerned with geometrical properties that remain invariant under topological transformations (rubber sheet transformation) [18]. In particular, relations with this property of invariance are called topological relations. Topological relations have been formally defined first in [3].

In this paper we focus on topological integrity constraints since they are able to express many meaningful characteristics of geographic data. In particular, we refer to the set of topological relations proposed in [1]. The following example shows the type of constraints we aim to deal with and the behaviour of the system, which uses them to drive updates.

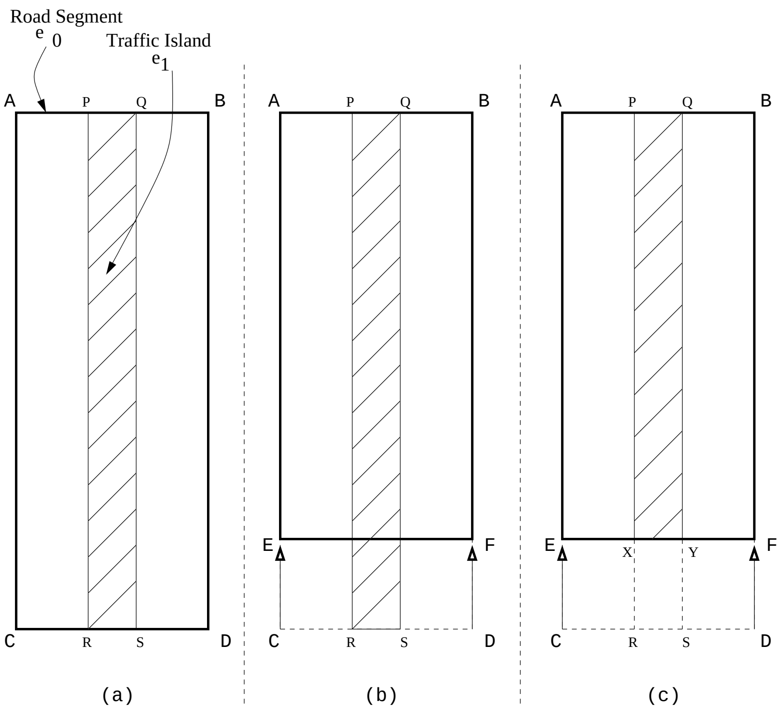

Fig. 1. *(a) An example of geographical database containing a road segment (e_0) and a traffic island (e_1). (b) An example of update without any integrity constraint checking during editing. (c) The same update in an integrity constraint driven system (see Example 1).*

Example 1 *Consider a geographical database containing road segments and traffic islands, both with polygonal representation. For each road segment the road name and the number of carriageways are stored. Moreoever, the following integrity constraint is imposed:* "every road segment with 2 carriageways must contain a traffic island, possibly sharing with it a part of boundary." *An admissible state of the database is shown in Figure 1(a). In Figure 1(b) an example of update in a traditional system with no support for integrity preservation is shown. The road segment e_0 is shortened by the user moving the segment $\overline{CD}$ to the position $\overline{EF}$. The traffic island now violates the above described integrity constraint. In Figure 1(c) the result of the same update in an integrity constraint driven environment is shown. In this case the system has moved automatically, during the update, the points R and S of the traffic island e_1 to X and Y respectively, in order to preserve the constraint. Other, more complex, cases will be shown in Section 3.* ◇

In the literature we find other works concerning spatial integrity constaints, in particular, in [10, 12] an active database mechanism was applied to adjust topological integrity constraint violations. Our work differs from these approaches since we try to drive user update in order to avoid constraint violation, while active database rules react only when violations occur. Hadzilacos & Tryfona in [7] proposed the Georelational Data Model language, that can be used also to express topological constraints. However, they did not study the issues of integrity preservation. Also Pizano et al. in [13] proposed the use of spatial integrity constraints to define consistent states of a spatial database, but their work referred to pictorial databases and they focus on depiction of unacceptable states rather than on updates. Other related works are [11, 9], which deal mainly with the design of interfaces to GIS for data retrieval.

The paper is organized as follows: in Section 2 we define formally a rather general spatial database environment with topological integrity constraints; in Section 3 we describe the prototypal system which has been built in order to explore the practical effectiveness of the general idea (this includes performance, because the constraints are used during the interaction with the user); moreover some aspects of implementation are described; in Section 4 we present conclusions and future works.

2 A reference framework for the specification of topological integrity constraints in a spatial database schema

In this section we define a framework for the specification of topological integrity constraints in the schema of a spatial database. In the first subsection we present the basic characteristics that we expect to find in a spatial database schema, and in its correspondent instance. In the second subsection we define a predicate calculus for specifying formulas with spatial interpretation and we define a topological integrity constraint as a closed formula of this calculus.

2.1 Spatial database schema and instance

A formal definition of spatial database schema and instance is necessary for the definition of spatial constraints. The concepts presented in this section are not new and they are present in many models for GIS proposed in the literature (see for instance [14, 6, 7, 16, 8]). We adopt the feature-based approach, where features are the fundamental concept for the representation of geographical phenomena as described in [16]. The National Committee for Digital Cartographic Data Standards defines a feature as a real world entity representation [17]. Therefore, a feature can have an arbitrarily complex structure, however, three components are always present: the spatial location of the feature, the alphanumeric attributes of the feature and the relations of the feature with other features. A spatial database schema is modeled here as a set of feature types, where each feature type has exactly one spatial attribute representing the feature location and some alphanumeric attributes.

Definition 1. (Spatial Database Schema) *The schema of a spatial database is a triple* $S = (\mathcal{E}, n(), Dom_{\mathcal{E}}())$*:*

- $\mathcal{E} = \{E_1, ..., E_n\}$ *is a set of feature types.*
- $n : \mathcal{E} \rightarrow \mathbb{N}$ *is a function which defines the number of attributes of each feature type* $E_i \in \mathcal{E}$*. The j-th attribute of* E_i *is denoted by* $E_i.a_j$*,* $1 \leq j \leq n(E)$*. Each feature type* E_i *has a spatial attribute denoted as* $E_i.geo$ *and an identifier denoted as* $E_i.a_0$
- $Dom_{\mathcal{E}} : \mathcal{E} \times \mathbb{N} \rightarrow \{D_{number}, D_{string}\}$ *is a partial function which defines the domain of each attribute* $E_i.a_j$*. The domain of* $E_i.a_0$ *is* $\mathbb{N}$*. The domain of* $E_i.geo$ *is the spatial domain* $D_{spatial}$ *defined below.*

□

In this paper we consider only spatial objects embedded in the Euclidean plane $\mathbb{R}^2$ and we restrict our attention to three domains of discretized spatial objects [18]: a) the set of simple polygons having a simple looped polyline as boundary, b) the set of simple polylines and c) the Euclidean plane ($\mathbb{R}^2$), whose elements are single points. Formally, the spatial domain we consider is defined as follows.

Definition 2. (Spatial Domain) *The spatial domain* $D_{spatial}$ *is the union of the following three domains:*

- D_{point}*: it is the set* $\mathbb{R}^2$*.*

– *D_{line}: it is the set of the simple polylines of $\mathbb{R}^2$:*

$$\begin{aligned} D_{line} =&\{(P_0, ..., P_n) : n \geq 2 \wedge (\forall i \in \{0, ..., n\} : P_i \in \mathbb{R}^2) \wedge \\ &(\forall i, j \in \{0, .., (n-1)\} : i \neq j \Rightarrow ((P_i \neq P_j) \wedge (Seg(P_i, P_{(i+1)}) \cap Seg(P_j, P_{(j+1)}) = \emptyset))) \wedge \\ &(\forall i \in \{1..n-1)\} : P_n \neq P_i) \end{aligned}$$

where $Seg : \mathbb{R}^2 \times \mathbb{R}^2 \rightarrow 2^{\mathbb{R}^2}$ is a function that, given a pairs of points (P_1, P_2), returns the set of points of $\mathbb{R}^2$ that represents the line segment from P_1 to P_2 without endpoints. A polyline is looped if $P_0 = P_n$.

– *$D_{polygon}$: it is the set of simple polygons of $\mathbb{R}^2$ defined by the looped polylines of D_{line}.*

Therefore, $D_{spatial} = D_{point} \cup D_{line} \cup D_{polygon}$. □

Given the above definitions, we now define the instance of a spatial database schema.

Definition 3. (Instance of a Spatial Database Schema) *Given a spatial database schema $S = (\mathcal{E}, n(), Dom_{\mathcal{E}})$ an instance of S is a set I_S containing sets of instances of the feature types of S. In I_S there is one set S_{E_i} for each feature type $E_i \in \mathcal{E}$, where each element $e \in S_{E_i}$ is a tuple belonging to the domain $\mathbb{N} \times Dom_{\mathcal{E}}(E_i, 1) \times ... \times Dom_{\mathcal{E}}(E_i, n(E_i)) \times D_{spatial}$. Each set S_{E_i} must satisfy the following constraint: $(\forall e_1, e_2 \in S_{E_i} : e_1 \neq e_2 \Rightarrow e_1.a_0 \neq e_2.a_0)$. With $e.a_j$, $j \in \{0, ..., n(E_i)\}$ we indicate the j-th value of $e \in S_{E_i}$. With e.geo we indicate the $(n(E_i) + 1)$-value of $e \in S_{E_i}$.* □

2.2 A calculus for topological constraints specification

Spatial integrity constraints express the topological relations that must exist among the spatial values *e.geo* of a spatial database instance I_S in order to represent a consistent state of the database.

The following definitions 4-7 are based on the topological relations introduced first in [3], and refined in [1], which are valid also for the values belonging to $D_{spatial}$.

The concepts of boundary and interior of a spatial value are the basis for the definition of the topological relations. For each spatial value A of $D_{spatial}$ the boundary and the interior are defined as follows [18]:

Definition 4. (Boundary and Interior of a spatial value) *Given $A \in D_{spatial}$, the boundary of A (∂A) is:*

– *if A is a simple polygon, the simple looped polyline that defines it.*
– *if A is a simple polyline $p = (P_0, ..., P_n)$, the set $\{P_0, P_n\}$.*
– *if A is a point, the empty set.*

The interior (λ°) is defined as the difference: $\lambda^\circ = \lambda - \partial\lambda$. □

Definition 5. (Basic topological relations [1]) *Given two spatial values A and B of $D_{spatial}$, the set Rel_T of topological relations is the set of the following five relations:*

$$\begin{aligned} &(A\ Touch\ B) &&\Leftrightarrow (A^\circ \cap B^\circ = \emptyset) \ \wedge \ (A \cap B \neq \emptyset) \ \wedge \ (dim(A) \neq 0 \vee dim(B) \neq 0) \\ &(A\ In\ B) &&\Leftrightarrow (A \cap B = A) \ \wedge \ (A^\circ \cap B^\circ \neq \emptyset) \\ &(A\ Cross\ B) &&\Leftrightarrow (dim(A^\circ \cap B^\circ) = max(dim(A^\circ), dim(B^\circ)) - 1) \ \wedge \ (A \cap B \neq A) \\ & &&\wedge \ (A \cap B \neq B) \ \wedge \ ((dim(A) = 1 \wedge dim(B) \geq 1) \vee (dim(B) = 1 \wedge dim(A) \geq 1)) \\ &(A\ Overlap\ B) &&\Leftrightarrow (dim(A^\circ) = dim(B^\circ) = dim(A^\circ \cap B^\circ)) \ \wedge \ (A \cap B \neq A) \wedge (A \cap B \neq B) \\ & &&\wedge \ ((dim(A) = 1 \wedge dim(B) = 1) \vee (dim(A) = 2 \wedge dim(B) = 2)) \\ &(A\ Disjoint\ B) &&\Leftrightarrow (A \cap B = \emptyset) \end{aligned}$$

where $\cap$ denotes the intersection and the function $dim()$ computes the dimension of the intersection result and returns: 0: one or more points, 1: one or more polylines, 2: one or more polygons. □

In order to permit the application of the above defined relations also to the boundary of a spatial value, Clementini et al. [1] defined also the following functions:

Definition 6. (Boundary of a polygon) *The function $b : D_{spatial} \rightarrow D_{spatial}$ returns, given a spatial value p, the polyline that defines p if p is a polygon, the empty set otherwise.* □

Definition 7. (End-points of a polyline) *The functions $f : D_{spatial} \rightarrow D_{spatial}$ and $t : D_{spatial} \rightarrow D_{spatial}$ return, given the polyline $p = (P_0, ..., P_n)$, the point P_0 and the point P_n respectively. If p is not a polyline both functions return the empty set.* □

We can now define a predicate calculus for specifing topological integrity constraints.

Definition 8. (Topological Calculus (TC(S)) - syntax) *Given a spatial database schema $S = (\mathcal{E}, n(), Dom_{\mathcal{E}})$ the syntax of the topological calculus is defined as follows:*

- *a set $V = \{e, e_0, e_1, ...\}$ of feature variables: they represent instances of a feature type; the i-th component of e is denoted by $e.a_i$; moreover each variable e has a component $e.geo$.*
- *given a feature variable e, a simple term is any component $e.a_i$;*
- *given a feature variable e, the component $e.geo$ and $b(e.geo)$, $f(e.geo)$, $t(e.geo)$ are spatial terms;*
- *an atomic formula can be:*
  - *a ground formula: $t_1 \ \theta \ t_2$ or $t_1 \ \theta \ c$, where t_1 and t_2 are simple terms, c is a constant and $\theta \in \{=, \neq, <, >, \leq, \geq\}$;*
  - *a spatial formula: $s_1 \ \varrho \ s_2$, where s_1 and s_2 are spatial terms and $\varrho \in Rel_T$;*
  - *range formula: $E_i(e)$ where $E_i \in \mathcal{E}$ and e is a feature variable;*
- *if A and B are a formulas, then $A \wedge B$, $A \vee B$, $(\forall e)A$ and $(\exists e)A$ are formulas, where e is an entity variable;*
- *every formula of TC(S) has to satisfy the following constraints (safeness conditions)*
  - *for each variable e there must exist a range formula $E_i(e)$.*
  - *for each subformula $(\forall e)A$ or $(\exists e)A$, a range formula $E_i(e)$ must be contained in A;*

□

Definition 9. (Topological Calculus (TC(S)) - semantics) *Given a spatial database instance I_S of a schema S, an interpretation of the TC(S) formulas in I_S is obtained by assigning:*

- *a mapping between the constants and the elements of the basic domains: D_{number}, D_{string} and $D_{spatial}$;*
- *a mapping between the symbols $E_i \in \mathcal{E}$ and the sets $S_{E_i} \in I_S$;*
- *the range formula $E_i(e)$ is to be interpreted as the membership relation to the set S_{E_i} where this set is obtained by the previously described mapping.*
- *the simple (spatial) term $e.a_j$ ($e.geo$) is to be interpreted as the component $e.a_j$ ($e.geo$) of the feature assigned to the variable e;*
- *the spatial terms $b(e.geo)$, $f(e.geo)$ and $t(e.geo)$ are to be interpreted applying the functions $b()$, $f()$ and $t()$ to the component $e.geo$ of the feature assigned to the variable e;*
- *the predicate symbols of ground formulas are to be interpreted as the usual equality and order relations on numbers and strings;*
- *the predicate symbols of spatial formulas are to be interpreted as the topological relations Rel_T defined in $D_{spatial}$.*

□

Given the schema $S = (\mathcal{E}, n(), Dom_{\mathcal{E}})$ any closed formula C of TC(S) represents a topological integrity constraint. The constraint C is satisfied by the instance I_S if it is true when it is interpreted in the instance I_S.

Definition 10. (Spatial Database Schema with Integrity Constraints) *The schema of a spatial database with integrity constraints $S_C = (S, C_S)$ is composed of:*

- *a spatial database schema $S = (\mathcal{E}, n(), Dom_{\mathcal{E}}())$;*
- *a set of topological integrity constraints $C_S = \{C_1, ..., C_n\}$ where each C_i $(1 \leq i \leq n)$ is a closed formula of the TC(S) calculus. Moreover, we suppose that the conjuction of the formulas $C_1, ..., C_n$ is satisfiable, i.e. there exists an instance of S where all the formulas $C_1, ..., C_n$ are true.*

□

3 Update Environment for Spatial Data

The introduction of spatial integrity constraints in a spatial database permits the system to check the database content for consistency, but it could also require a considerable computational overhead in order to perform such control. Since our aim is to use spatial integrity constraints as a driver for the user in spatial update operations, we had to limit the complexity of the constraints in order to:

– obtain a performance which is acceptable in interactive editing, and
– keep the number of entities which are involved in a single update sufficiently small.

Therefore, we define the *Simple Topological Integrity Constraint.* A Simple Topological Integrity Constraint is a topological integrity constraint with the following limitations:

– each constraint involves at most two feature types;
– a feature type can be involved in zero, one or more constraints; the constraints of a feature type can also be grouped to represent a disjunction of constraints;
– each constraint may contain filters that select the set of features of the two feature types that are involved in the constraint. These filters are limited to be selection on alphanumeric feature type attributes.

In the update environment we use only simple topological integrity constraints for driving the user in the update operations. Moreover, in this first version of the update environment we focus only on spatial objects of the domain $D_{polygon}$ and consequently also the spatial formula of TC(S) have been redefined to work only on polygons. In particular, the functional symbols $b(s)$, $f(s)$ and $t(s)$ (with s spatial term) have been removed and the topological relations used in the spatial formulas $s_1\ \varrho\ s_2$ have been substituted by the topological relations for areal objects defined in [1] with the Dimension Extended Method. We call this new set of relations Rel_T^{poly}.

The use of simple topological integrity constraints and polygons is sufficient for describing many real/world situations and to demonstrate the effectiveness of the approach proposed in this paper. Of course, the general constraints of TC(S) could be checked in a traditional way, but this is out of the scope of this paper.

Definition 11. (Topological relations for polygons [1]) *Considering the boundary and the interior of the polygons A and B, the intersections between them are computed and the dimension of the not empty result sets are considered using function $dim()$*[1]*. The possible cases are represented by the following vector:*

$$(dim(\partial A \cap \partial B), dim(\partial A \cap B^\circ), dim(A^\circ \cap \partial B), dim(A^\circ \cap B^\circ)) \in \{-,0,1\} \times \{-,1\} \times \{-,1\} \times \{-,2\}$$

The simbol "−" indicates that the result of the intersection is empty. Due to the general properties of the boundary and interior of areal objects of $\mathbb{R}^2$, the number of the admissible cases are reduced to the following ones:

$$\begin{aligned} Rel_T^{poly} = \{ & Disjoint(-,-,-,-), Meets_0(0,-,-,-), Meets_1(1,-,-,-), \\ & Overlaps_0(0,1,1,2), Overlaps_1(1,1,1,2), In(-,1,-,2), Contains(-,-,1,2), \\ & CoveredBy_0(0,1,-,2), CoveredBy_1(1,1,-,2), Covers_0(0,-,1,2), Covers_1(1,-,1,2)\} \end{aligned}$$

□

Notice that all these relations are mutually exclusive. Using a result of [1], it can be proved that the topological relations of Rel_T^{poly} between two polygons $e_i.geo$, $e_j.geo$ of an instance I_S can be expressed in the predicate calculus TC(S).

We call the above described refined calculus TC^{poly}(S). A simple topological integrity constraint is a closed formula of TC^{poly}.

[1] See definition 5

Definition 12. (Simple Topological Integrity Constraint (STIC) - syntax) *Given a spatial database schema $S = (\mathcal{E},\ n(), Dom_{\mathcal{E}}())$, a simple topological integrity constraint T_{E_i,E_j} between two feature types $E_i, E_j \in \mathcal{E}$ is composed of:*

- *a selection formula σ_0: it is a quantifier-free formula of $TC^{poly}(S)$ with only a free variable e_0 and containing only atomic ground formulas;*
- *a selection formula σ_1: it is a quantifier-free formula of $TC^{poly}(S)$ with one free variable e_1 or two free variables e_0, e_1; it contains only ground formulas involving e_1 and/or possibly ground formulas of the form $e_0.a_k \theta e_1.a_h$.*
- *a symbol $Q \in \{\forall, \exists\}$;*
- *a constraint formula ϕ: it is a quantifier-free conjuction containing a disjunction of atomic spatial formulas. ϕ has only two free variables e_0, e_1.*

□

The fact that ϕ can only contain a disjunction of atomic spatial formulas is not a limitation. Indeed, since all the topological relations of Rel_T^{poly} are mutually exclusive, any conjunction of spatial formulas is contraddictory. Notice that for safeness conditions (see Def. 8) σ_0 and σ_1 always contain the range formulas $E_i(e_0)$ and $E_j(e_1)$ (possibly $E_i(e_0)$) respectively.

Definition 13. (Simple Topological Integrity Constraint (STIC) - semantics) *The semantics of the STIC $T_{E_i,E_j} = (\sigma_0, \sigma_1, Q, \phi)$, denoted as $Sem(T_{E_i,E_j})$, is illustrated by the following two formulas of $TC^{poly}(S)$, the first one shows the semantics of T_{E_i,E_j} when $Q = \forall$, the second one when $Q = \exists$:*

$$(\forall e_0)(\forall e_1)(\sigma_0 \Rightarrow (\sigma_1 \Rightarrow \phi)) \tag{1}$$

$$(\forall e_0)(\exists e_1)(\sigma_0 \Rightarrow (\sigma_1 \wedge \phi)) \tag{2}$$

□

Given an instance I_S of a spatial database schema S, a STIC T_{E_i,E_j}:

- *is satisfied by I_S* if the formula $Sem(T_{E_i,E_j})$ is true when TC^{poly}(S) is interpreted in I_S;
- *is satisfied by a given feature $\overline{e_0} \in S_{E_i}$, $S_{E_i} \in I_S$*, if, considering the formula $Sem(T_{E_i,E_j})$ without the first quantifier, this formula is true when TC^{poly}(S) is interpreted in I_S and the variable e_0 is assigned the feature $\overline{e_0}$.
- *is active for $\overline{e_0}$*, if the formula σ_0 is true when the variable e_0 is assigned the feature $\overline{e_0}$.

Notice that, when $Q = \forall$, the STIC T_{E_i,E_j} constrains also all $e_1 \in E_j$, i.e. the STIC is somehow symmetrical. We extend the definition of spatial database schema including also a set T_S containing sets of simple topological integrity constraints. We denote the extended schema as $S_{CT} = (S, C_S, T_S)$. Each set of STIC represents a disjuction of constraints. The STICs of a set have all the same E_i, i.e. they represent alternative constraints for the same feature type E_i. An instance I_S satisfies the set T_S, if it satisfies each set of STICs it contains. A set of STICs $\{T_{E_i,E_{j_1}}, ..., T_{E_i,E_{j_n}}\}$ is satisfied by an instance I_S, only if it satisfies at least one STIC $T_{E_i,E_{j_k}}$ of the set.

Example 2 *Consider the database schema of the example 1:*

$$\begin{aligned} S =& (\{RoadSegment, TrafficIsland\}, n(), Dom_{\mathcal{E}}()). \\ n() =& \{(RoadSegment, 2), (TrafficIsland, 0)\} \\ Dom_{\mathcal{E}}() =& \{(RoadSegment, 1, D_{string}), (RoadSegment, 2, D_{number})\} \end{aligned}$$

We suppose that the first attribute of RoadSegment represents the name of the road and the second one represents the number of carriageways.

If we want to specify the integrity constraint: "every road segment with two carriageways must contain a traffic island, possibly sharing with it a part of the boundary." *we can write the following STIC for the RoadSegment feature type:*

$$\begin{aligned} T_{RoadSegment,TrafficIsland} =& (RoadSegment(e_0) \wedge (e_0.a_2 = 2), TrafficIsland(e_1), \exists, \\ & (e_0.geo\ Covers_0\ e_1.geo) \vee (e_0.geo\ Covers_1\ e_1.geo) \vee (e_0.geo\ Contains\ e_1.geo)) \end{aligned}$$

The semantics of $T_{RoadSegment,TrafficIsland}$ *is the following one:*

$$\begin{aligned}Sem(T_{RoadSegment,TrafficIsland}) = \ & (\forall e_0)(\exists e_1)((RoadSegment(e_0) \wedge e_0.a_2 = 2) \Rightarrow (TrafficIsland(e_1)\\ & \wedge ((e_0.geo\ Covers_0\ e_1.geo) \vee (e_0.geo\ Covers_1\ e_1.geo)\\ & \vee (e_0.geo\ Contains\ e_1.geo)))\end{aligned}$$

◇

3.1 Representing polygons and simple topological constraint instances

Let I_S be an instance of a spatial database schema with integrity constraints $S_{CT} = (S, C_S, T_S)$. In order to be able to manage the update of the spatial attribute $e.geo$ of all features $e \in S_{E_i}$ with $S_{E_i} \in I_S$, and to drive the user to preserve the STIC $\in T_S$, a possible solution is to store explicitly in a common data structure:

– the spatial database schema S together with the STICs of T_S;
– the spatial values $e.geo$ of the spatial database instance I_S together with a subset of the topological relation instances among $e.geo$ values of I_S that must be preserved to satisfy the STICs of T_S. A topological relation instance $e_0\ \varrho\ e_1$ represents the fact that $e_0.geo$ is in the topological relation ϱ with $e_1.geo$.

The satisfaction of a STIC by a feature e is materialized in a set of topological relation instances among e and the other features e_j of I_S. Notice that, the calculation on the fly of such relation instances is not computationally feasible. We call these topological relation instances *Simple Topological Integrity Constraint Instances* or simply STIC Instances. Our approach is based on the manipulation of this set of topological relation instances.

The following definition presents an abstract data structure for storing a set of STIC instances. It is used to describe our approach at high level; many different implementations are possible.

Definition 14. (Simple Topological Integrity Constraint Instances) *Given an instance I_S of a spatial database schema with integrity constraints $S_{CT} = (S, C_S, T_S)$, the data structure $I_{STIC}(I_S, T_S)$ contains sets of 4-tuples $(e_0, e_1, r, T_{E_i,E_j})$. Each 4-tuple indicates that the feature e_0 is in the topological relation r with the feature e_1 and the preservation of this relation is sufficient ($Q = \exists$) or is necessary ($Q = \forall$) to guarantee the satisfaction of a STIC T_{E_i,E_j} belonging to a set of T_S. We consider sets of 4-tuples instead of single 4-tuples, for two reasons: a) for a STIC with existential quantifier $(\forall e_0)(\exists e_1)(P(e_0, e_1))$ there could be more than one feature e_1 that makes $P(e_0, e_1)$ true; b) a set of STICs in T_S represents a disjuction of constraints, thus more than one 4-tuple could exist in I_{STIC} when more that one constraint of the set is satisfied. Thus, I_{STIC} can be defined as follows:*

$$I_{STIC}(I_S, T_S) \subseteq 2^{U_\mathcal{E} \times U_\mathcal{E} \times Rel_T^{poly} \times U_{T_S}}$$

where U_{T_S} is the union of all sets of STICs of T_S. □

Example 3 *Considering the STIC of the previous example 2 and the spatial database instance shown in Figure 1(a), the content of the I_{STIC} substructure is:*

$$I_{STIC} = \{\{(e_0, e_1, Covers_1, T_{RoadSegment,TrafficIsland})\}\}$$

A more complex situation is shown in appendix A. ◇

The structure $I_{STIC}(I_S, T_S)$ stores a set of topological relation instances whose preservation guarantees the satisfaction of the STICs defined at schema level. By defining update operations that preserves such relations, we preserve also the satisfaction of the STICs defined in the schema. Notice that:

– not all topological relations among the $e_i.geo$ values of the spatial database instance I_S are stored in I_{STIC}, but only a subset that guarantees to preserve database integrity. This subset is not unique.
– Given a STIC T_{E_i,E_j} and an feature e_0 belonging to E_i that satisfies T_{E_i,E_j}, there always exists a set of 4-tuples $\{(e_0, e_{j_1}, r_1, T_{E_i,E_j}), ..., (e_0, e_{j_n}, r_n, T_{E_i,E_j})\}$ $(e_{j_1}, ..., e_{j_n} \in E_j)$ whose preservation guarantees that T_{E_i,E_j} is always satisfied by e_0. Indeed, this set in the worst case coincides with the set of all topological relation instances that e_0 has with the other features of the database.

Primitive Signature	**Description**	**Available in Operations**
Create_Poly(P, e_i)	It creates a polygon for the attribute $e_i.geo$ containing a single vertex P.	{I}
Insert_Segm(s, e_i)	It adds the segment s to the polygon $e_i.geo$.	{I}
Split_Segm(P, s, e_i)	It splits at the point P the segment s of the polygon $e_i.geo$.	{I,C,WC}
Merge_Segm(s_1, s_2, e_i)	It merges the segments s_1 and s_2 of the polygon $e_i.geo$.	{I,C,WC}
Move_Vert(V, P, e_i)	It moves the vertex V of the polygon $e_i.geo$ to the new position P.	{I,C,WC,SC}
Delete_Poly(e_i)	It removes the polygon of the attribute $e_i.geo$	{D}

Table 1. Available primitives during the update session. In the 3rd column the set indicates the operations in which the primitive can be invoked (I: insert, C: conservative update, WC: weak conservative update and SC: strong conservative update)

3.2 A framework for simple topological integrity constraints driven update

In this subsection we present an update environment that allows the user to modify an instance I_S of a spatial database schema with integrity constraints $S_{CT}(S, C_S, T_S)$. In particular, given a feature $e \in S_{E_i}$ with $S_{E_i} \in I_S$, we consider the modification of the attribute $e.geo$, the deletion of e, the update of the alphanumeric attributes $e.a_j$, and the insertion of a new feature e belonging to a feature type $E_i \in \mathcal{E}$.

The update environment requires the user to specify the feature $\overline{e}$ that he/she wants to update or delete, or the feature type $\overline{E_i}$ in case of insertion. Chosen the feature (or the feature type) to work with, the user has four update operations at disposal (for each operation Table 1 shows the available primitives):

– *INSERT*: It allows the addition of a new feature e_{ins} of the chosen feature type $\overline{E_i}$. It requires the input values for the alphanumeric attributes $e_{ins}.a_j$ $(1 \leq j \leq N(\overline{E_i})))$, and the specification of a polygon for the attribute $e_{ins}.geo$ that satisfies all the STICs of T_S. Before opening the update session, the system generates some temporary data structures:
  - The set $Temp_{relations}$: it contains the topological relation instances that the new feature e_{ins} must satisfy with the other features of the database according to the STICs of the schema; the system interacts with the user to define: a) the STICs to satisfy if ,for $\overline{E_i}$, there exist sets $\{T_{\overline{E_i},E_{j_1}}, ..., T_{\overline{E_i},E_{j_n}}\}$ representing a disjuction of STICs; at least one STIC must be chosen; b) the features $e_1 \in E_{j_k}$ to be considered for the satisfaction of the chosen STICs with existential quantifier; c) the spatial formula to satisfy for the STICs that have a disjunction of spatial formulas as constraint formula ϕ.
    If in a set the choosen STICs are all with existential quantifier and for all of them there does not exist any $e_1 \in E_{j_k}$ that makes true σ_1, then the insert operation is aborted.
    During the insert operation, when a topological relation instance is satified, it is deleted from $Temp_{relations}$ and added to the data structure I_{STIC}.
  - The set $Temp_{scene}$: it contains the topological relation instances of the database involving the features that appear in $Temp_{relations}$ together with the requested topological relation instances of $Temp_{relations}$. An analysis of satisfyability is performed on the set of topological relations $Temp_{scene}$ using the approach defined in [5]. If the test is negative, then the insert operation is aborted. Otherwise, the update session can be opened.
  - the set $Temp_{segments}$: it contains the topological relation instances that exist among each segment of e_{ins} and the other features that appear in $Temp_{relations}$.

  Considering the topological relation instances of $Temp_{relations}$ (they will have to be satisfied by e_{ins}), each new segment of e_{ins} is accepted only if it satisfies the conditions shown in the second column of Table 2. These conditions are necessary conditions to obtain, between $e_{ins}.geo$ and $e_j.geo$, the relation shown in the first column. At the end the polygon $e_{ins}.geo$ is accepted only if it satisfies the conditions show in the third column of Table 2. The conjunction of the conditions shown in the second and third columns of Table 2 represents a sufficient condition to obtain between $e_{ins}.geo$ and $e_j.geo$, the relation shown in the first column. The sufficiency and the necessity of these conditions follow trivially from the properties of simple polygons.

– *UPDATE OPERATIONS*: These operations permit the modification of the attribute $\overline{e}.geo$ of the chosen feature $\overline{e}$. When necessary these operations use the previously described temporary data structures. Three different versions of update are available: strong conservative update, conservative update and weak conservative update. In the first two versions, the modification of a segment is accepted only if it satifies the conditions of the second and third columns of Tables 2. In the third version, the same conditions are checked only at the end of the operation.
  1. *STRONG CONSERVATIVE UPDATE*: This version does not allow any geometric change that modifies the content of the temporary structure $Temp_{segments}$, containing the relations among the segments of $\overline{e}$ and the other features of the database, which are involved with $\overline{e}$ in any topological relation instance of I_{STIC}.
     The primitive $Move\_Vert(V, P, \overline{e})$ (see Table 1), if the vertex V is shared by $\overline{e}$ with another feature e_j, moves also the vertex of e_j only if this is necessary to preserve a relation between the segments of $\overline{e}$, which has V as end-point, and e_j.
     Figure 2(a) shows an example of strong conservative update.
  2. *CONSERVATIVE UPDATE*: This version does not allow any geometric change that produces the loss (even if temporary) of the satisfaction of a disjunction of STICs beloging to T_S.
     The primitive $Move\_Vert(V, P, \overline{e})$, if the vertex V is shared by $\overline{e}$ with another feature e_j, moves also the vertex of e_j only if this is necessary to preserve a disjunction of STICs of T_S.
     Figure 2(b) shows an example of conservative update.
  3. *WEAK CONSERVATIVE UPDATE*: This version allows temporary violations of the STICs of T_S. However, the final state of the polygon $\overline{e}.geo$ must satisfy the STICs of T_S.
     During a weak conservative update the user can also modify the choices he/she did concerning the topological relation instances that $\overline{e}$ has to satisfy (see INSERT); in this case an analysis of satisfyability is performed as for the insert operation.
     The primitive $Move\_Vert(V, P, \overline{e})$, if the vertex V is shared with another feature e_j, never moves also the vertex of e_j.
     Figure 2(c) shows an example of weak conservative update.

– *DELETE*: It allows the elimination of the chosen feature $\overline{e}$. Moreover, it checks the data structure I_{STIC} to identify the elements to be removed. If the deletion of a 4-tuple $(\overline{e}, e_j, r, T_{\overline{E_i}, E_j})$ in a set of I_{STIC} makes one STIC of T_S false the elimination of $\overline{e}$ is rejected. In this case the deletion can be performed only after a weak conservative update of the feature e_j that has blocked the deletion.

– *ALPHANUMERIC UPDATE*: It allows the update of the attributes $a_1, ..., a_{n(E_i)}$ of $\overline{e}$. Moreover, it checks if one of the following situations occurs:
  - The modification of the alphanumeric attribute $\overline{e}.a_k$ activates a STIC $T_{\overline{E_i}, E_j}$ of a set and this is the first STIC of that set to be activated by $\overline{e}$. In this case, interacting with the user, new sets of topological relation instances are added to the $Temp_{relations}$ structure, and a weak conservative update is performed.
  - The modification of the alphanumeric attribute $\overline{e}.a_k$ makes the $\sigma_1(\overline{e})$ of a STIC $T_{E_j, \overline{E_i}}$ with existential quantifier false. In this case, the 4-tuple $(e_j, \overline{e}, r, T_{E_j, \overline{E_i}})$, if present in a set of I_{STIC}, should be removed from the set. However, if this set becomes empty, the update of $\overline{e}.a_k$ is rejected. The update can be performed only after a weak conservative update of the feature e_j that has blocked the operation.
  - The modification of the alphanumeric attribute $\overline{e}.a_k$, disactivates a STIC $T_{\overline{E_i}, E_j}$. In this case, all the 4-tuples of I_{STIC} involving $\overline{e}$ are removed.

A system that adops this update environment should supply also a *transactional update* in order to allow the user to modify more istances $e_1, ..., e_n$ and performing the check of the STICs preservation at the end of the transaction.

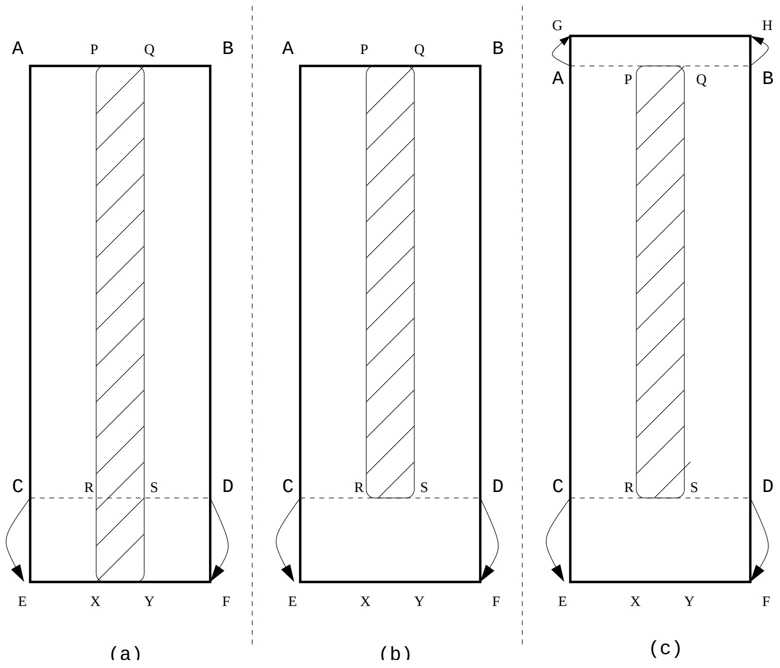

Fig. 2. *Examples of strong conservative (a), conservative (b), and weak conservative (c) update. Notice that, the conservative update (b) allows the segment $\overline{CD}$ to be moved to the new position $\overline{EF}$ without moving also the traffic island, since the satisfaction of the STICs instances of I_{STIC} (see example 3) is preserved by the segment $\overline{PQ}$. In the weak conservative update (c), the user before closing the operation must substitute in I_{STIC} the STIC instance representing the $Covers_1$ relation with a STIC instance representing a Contains relation as admitted by the STIC of the schema shown in example 2.*

3.3 Implementation

The system has been implemented using the development tools of Geomedia Professional^{TM} of Intergraph. The implementation details are described in [15].

The following two issues which have been encountered during system implementation are worth mentioning.

1. In order to reduce the cardinality of the set I_{STIC} the 4-tuples of the form $(e_0, e_1, Disjoint, R_{E_i,E_j})$ are not stored in the data structure implementing I_{STIC}. During the update operations this structure is completed with the subset of 4-tuples representing the disjoint relations that are necessary to check the satisfaction of the integrity rules for the modifing feature $\overline{e}$. The choice of the disjoint relation is not casual, since, considering a geographical database, the only topological relation that can produce sets of topological relation instances of high cardinality is the disjoint one as it does not require that the spatial objects have common points.
2. Before opening an update session, the user is requested to define a *working area*, which is a rectangle embedded in the reference space that contains the chosen feature e_i to be updated (deleted) or that will contains the new feature e_i in case of insertion. In this way, the size of the temporary structure $Temp_{segments}$ can be reduced.

4 Conclusion and future works

In this paper we have presented a framework for the specification of topological integrity constraints and have shown that a subset of these constraints (*Simple Topological Integrity Constraint*) can be used to drive the update activity of the user, thus maintaining database integrity. Moreover, the implemented system has demonstrated that the approach is feasible.

Spatial formula	$e_0.segment$ **Accept Conditions**	$e_0.geo$ **Accept Conditions**
$e_0\ Disjoint\ e_1$	$e_0.segment\ Disjoint\ e_1.geo$	$\neg(e_1.geo\ In\ e_0.geo)$
$e_0\ Meets_0\ e_1$	$e_0.segment\ Disjoint\ e_1.geo\ \vee$ $(e_0.segment\ Touch\ e_1.geo\ \wedge$ $dim(e_0.segment\ \cap\ \partial(e_1.geo)) = 0)$	$(\exists segment_i \in e_0.geo)(segment_i\ Touch\ e_1.geo\ \wedge$ $dim(e_0.segment_i\ \cap\ \partial(e_1.geo)) = 0)$
$e_0\ Meets_1\ e_1$	$e_0.segment\ Disjoint\ e_1.geo\ \vee$ $e_0.segment\ Touch\ e_1.geo$	$(\exists segment_i \in e_0.geo)(segment_i\ Touch\ e_1.geo\ \wedge$ $dim(e_0.segment\ \cap\ \partial(e_1.geo)) = 1))$
$e_0\ Overlaps_0\ e_1$	$\neg(dim(e_0.segment\ \cap \partial(e_1.geo)) = 1)$	$(\exists segment_i \in e_0.geo)(segment_i\ Cross\ e_1.geo)\ \vee$ $(\exists segment_i, segment_j \in e_0.geo)$ $(segment_i\ In\ e_1.geo\ \wedge\ segment_j\ Touch\ e_1.geo$ $\wedge\ dim(e_0.segment\ \cap\ \partial(e_1.geo)) = 0)$
$e_0\ Overlaps_1\ e_1$	no conditions	$(\exists segment_i \in e_0.geo)$ $(dim(e_0.segment\ \cap\ \partial(e_1.geo)) = 1)\ \wedge$ $((\exists segment_i \in e_0.geo)(segment_i\ Cross\ e_1.geo)\ \vee$ $(\exists segment_i, segment_j \in e_0.geo)$ $(segment_i\ In\ e_1.geo\ \wedge\ segment_j\ Touch\ e_1.geo))$
$e_0\ In\ e_1$	$e_0.segment\ In\ e_1.geo\ \wedge$ $(e_0.segment\ \cap\ \partial(e_1.geo)) = \emptyset$	no conditions
$e_0\ Contains\ e_1$	$e_0.segment\ Disjoint\ e_1.geo$	$e_0.geo\ Contains\ e_1.geo$
$e_0\ CoveredBy_0\ e_1$	$e_0.segment\ In\ e_1.geo\ \wedge$ $(dim(e_0.segment\ \cap\ \partial(e_1.geo)) = 0\ \vee$ $(e_0.segment\ \cap\ \partial(e_1.geo)) = \emptyset$	$(\exists segment_i \in e_0.geo)$ $(dim(segment_i \cap \partial(e_i.geo)) = 0)$
$e_0\ CoveredBy_1\ e_1$	$e_0.segment\ In\ e_1.geo$	$(\exists segment_i \in e_0.geo)(segment_i\ Meet_1\ e_1.geo))$
$e_0\ Covers_0\ e_1$	$e_0.segment\ Disjoint\ e_1.geo\ \vee$ $e_0.segment\ Meets_0\ e_1.geo$	$(\exists segment_i \in e_0.geo)(segment_i\ Meet_0\ e_1.geo))$
$e_0\ Covers_1\ e_1$	$e_0.segment\ Disjoint\ e_1.geo\ \vee$ $e_0.segment\ Meets_0\ e_1.geo\ \vee$ $e_0.segment\ Meets_1\ e_1.geo$	$(\exists segment_i \in e_0.geo)(segment_i\ Meet_1\ e_1.geo))$

Table 2. Conditions to be checked during update sessions. $e_0.segment$ represents the segment that has been inserted or modified during the update session.

Future works includes: the removal of some of the limitations to STICs discussed in Section 3; the extension of the update environment to include also polylines and points. Moreover, the development and experimentation with the system has shown that the approach could be useful in preserving also topological relations, which are not defined as constraints at schema level, but which exist in the database instance and have been explicitely recognized during the update activity.

References

1. E. Clementini, P. Di Felice, and P. van Oosterom. A Small Set of Formal Topological Relationships Suitable for End-User Interaction. In *Proc. 3nd Symposium on Spatial Databases*, pages 277-295, 1993.
2. S. Cockcroft. A Taxonomy of Spatial Data Integrity Constraints *Geoinformatica* , 1(4):327-343, 1997.
3. M. J. Egenhofer, and R. D. Franzosa. Point-set Topological Spatial Relations. *International Journal of Geographical Information Systems*, 5(2):161-174, 1991.
4. M. J. Egenhofer. Reasoning about Binary Topological Relations. In *Proc. 2nd Symposium on Spatial Databases*, pages 143-160, 1991.
5. M. J. Egenhofer, and J. R. Herring. Categorizing Binary Topological Relationships between Regions, Lines, and Points in Geographic Databases. Technical report, Department of Surveying Engineering, University of Orono, ME, 1992.
6. R.H. Güting and M. Schneider. Realm-Based Spatial Data Types: The ROSE Algebra. *VLDB Journal*, 4:243-286, 1995.
7. T. Hadzilacos, and N. Tryfona. Logical Data Modelling for Geographical Applications. *International Journal of Geographical Information Systems*, 10(2):179-203, 1996.
8. T. Hadzilacos, and N. Tryfona. An Extended Entity-Relationship Model for Geographic Applications. *SIGMOD Record*, 26(3):24-29, 1997.

9. G. Koesters, B. U. Pagel, and H. W. Six, GIS-Application Development with GeoOOA. *International Journal of Geographical Information Science*, 11(4):307-335, 1997.
10. R. Laurini, and F. Milleret-Raffort. Topological Reorganization of Inconsistent Geographical Databases: A Step Towards their Certification. *Computers and Graphics*, 18(6):803-813, 1994.
11. J. Lopes de Oliveira, C. B. Mendeiros, and M. Cilia. Active Customization of GIS User Interfaces. In *Proc. ICDE'97*, pages 487-496, 1997.
12. C. B. Medeiros, and M. Cilia. Maintenance of Binary Topological Constraints through Active Database. In *Proc. 3rd ACM Workshop on Advances in GIS*, pages 127-134, 1995.
13. A. Pizano, A. Klinger, and A. Cardenas. Specification of Spatial Integrity Constraints in Pictorial Databases. *IEEE Computer*, pages 56-71, December 1989.
14. M. Scholl and A. Voisard. Thematic Map Modeling. In *LNCS 409: Proc. of the Int. Symp. on the Design and Implementation of Large Spatial Databases*, pages 167–190, 1989.
15. F. Spinazza. Il controllo di vincoli topologici in fase di editing di dati spaziali. .... thesis, Politencnico di Milano, 1999.
16. A. Y. Tang, T. M. Adams, and E. L. Usery. A Spatial Data Model Design for Feature-based Geographical Information Systems. *International Journal of Geographical Information Systems*, 10(5):643-659, 1996.
17. USGS. Spatial data transfer standard. Department of the Interior, U.S. Geographical Survey, National Mapping Division, 1990.
18. M. F. Worboys. GIS: A Computing Perspective. Tailor & Francis, London, 1995.

A Another example of spatial database with STICs

In this appendix we show another example of spatial database with simple topological integrity constraints. We present the spatial database schema S, the set of STICs T_S, the spatial database instance I_S and the content of the structure I_{STIC}.

The schema S is defined as follows:

$$\begin{aligned}
S =&(\{River, Canal, UrbanArea, Waterworks\}, n(), Dom_{\mathcal{E}}())\\
n() =&\{(River, 1), (Canal, 1), (UrbanArea, 2), (Waterworks, 2)\}\\
Dom_{\mathcal{E}}() =&\{(River, 1, D_{string}), (Canal, 1, D_{string}), (UrbanArea, 1, D_{string}),\\
&(UrbanArea, 2, D_{number}), (Waterworks, 1, D_{string}), (Waterworks, 2, D_{number})\} \qquad (3)
\end{aligned}$$

The first alphanumeric attribute of all feature types represents the name of the feature. The second attribute of $UrbanArea$ stores the population of each urban area and the second attribute of $Waterworks$ stores the type of each plant (0: dam, 1: lock, etc.).

T_S contains the following STICs:

$$\begin{aligned}
T_S = \{&\{T_{UrbanArea,River}\}, \{T_{UrbanArea,Canal}\}\{T_{Canal,UrbanArea}\},\\
&\{T_{Waterworks,River}, T_{Waterworks,Canal}\}\}
\end{aligned}$$

where:

$$\begin{aligned}
T_{UrbanArea,River} = &(UrbanArea(e_0), River(e_1), \forall, (e_0.geo\ Disjoint\ e_1.geo) \vee\\
&(e_0.geo\ Meet_1\ e_1.geo) \vee (e_0.geo\ Meet_0\ e_1.geo))
\end{aligned}$$

$$\begin{aligned}
T_{UrbanArea,Canal} = &((UrbanArea(e_0), Canal(e_1), \forall, (e_0.geo\ Disjoint\ e_1.geo) \vee\\
&(e_0.geo\ Meet_1\ e_1.geo) \vee (e_0.geo\ Meet_0\ e_1.geo))
\end{aligned}$$

$$T_{Canal,UrbanArea} = (Canal(e_0), (UrbanArea(e_1), \exists, (e_0.geo\ Meet_1\ e_1.geo)$$

$$\begin{aligned}
T_{Waterworks,River} = &(Waterworks(e_0), River(e_1), \exists, (e_0.geo\ CoveredBy_1\ e_1.geo) \vee\\
&(e_0.geo\ CoveredBy_0\ e_1.geo) \vee (e_0.geo\ Meet_1\ e_1.geo))\\
&(e_0.geo\ Overlap_0\ e_1.geo) \vee (e_0.geo\ Overlap_1\ e_1.geo))
\end{aligned}$$

$$\begin{aligned} T_{Waterworks,Canal} = {} & (Waterworks(e_0), Canal(e_1), \exists, (e_0.geo\ CoveredBy_1\ e_1.geo) \vee \\ & (e_0.geo\ CoveredBy_0\ e_1.geo) \vee (e_0.geo\ Meet_1\ e_1.geo)) \\ & (e_0.geo\ Overlap_0\ e_1.geo) \vee (e_0.geo\ Overlap_1\ e_1.geo)) \end{aligned}$$

The instance I_S is shown in Figure 3 and the content of the I_{STIC} structure for the instance I_S is the following one:

$$\begin{aligned} I_{STIC} = \{ & \{(e_0, e_1, Meet_1, T_{UrbanArea,Canal}), (e_0, e_6, Disjoint, T_{UrbanArea,Canal})\}, \\ & \{(e_0, e_4, Disjoint, T_{UrbanArea,River})\}, \\ & \{(e_1, e_0, Meet_1, T_{Canal,UrbanArea}), (e_1, e_2, Meet_1, T_{Canal,UrbanArea})\} \\ & \{(e_2, e_1, Meet_1, T_{UrbanArea,Canal}), (e_2, e_6, Meet_1, T_{UrbanArea,Canal})\}, \\ & \{(e_2, e_4, Disjoint, T_{UrbanArea,River})\}, \\ & \{(e_3, e_6, CoveredBy_1, T_{Waterworks,Canal})\}, \\ & \{(e_5, e_4, Meet_1, T_{Waterworks,River})\}\} \end{aligned}$$

Notice that the first and the fourth set of I_{STIC} contains two 4-tuples, representing a STIC with universal quantifier; also the third set contains two 4-tuples, but in this case a STIC with existential quantifier is represented. This implies that the topological relation instances of the first and fourth set are necessary conditions, while the topological relation instances of the third set are sufficient conditions.

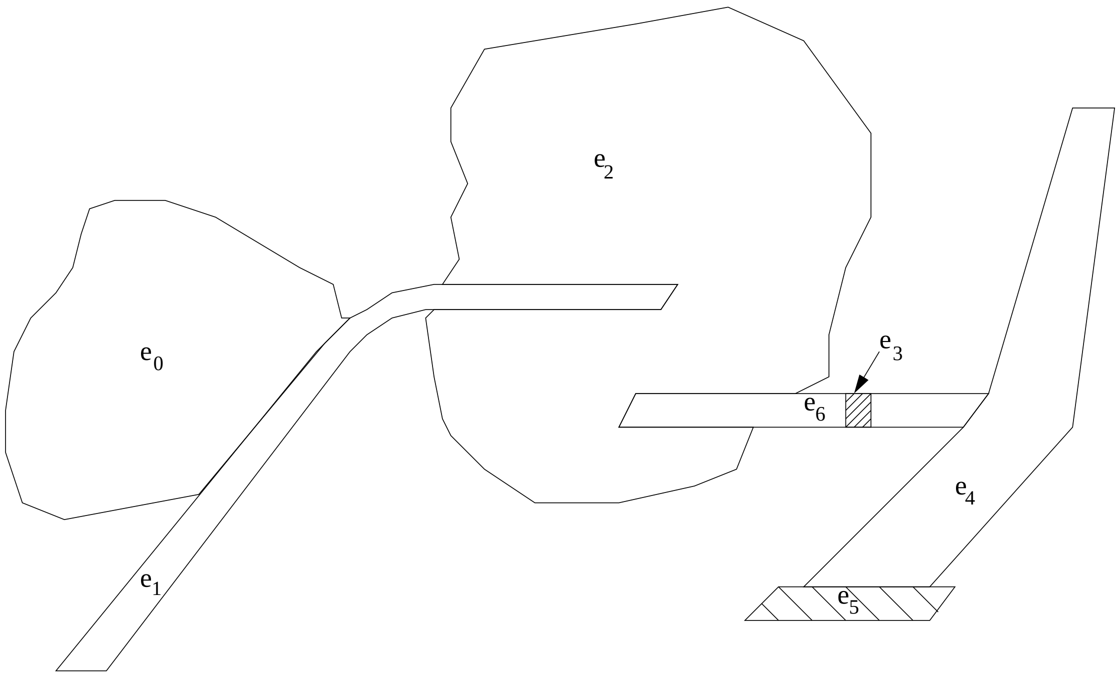

Fig. 3. *Example of spatial database instance on the schema (3). The features of the scene belong to the following types:* $e_0, e_2 \in S_{UrbanArea}$*,* $e_1, e_6 \in S_{Canal}$*,* $e_3, e_5 \in S_{Waterworks}$*,* $e_4 \in S_{River}$