An Inconsistency Tolerant Approach to Querying Spatial Databases

M. Andrea Rodríguez
Universidad de Concepción
Concepción, Chile
andrea@udec.cl

Leopoldo Bertossi
Carleton University
Ottawa, Canada
bertossi@scs.carleton.ca

Monica Caniupán
Universidad del Bío-Bío
Concepción, Chile
mcaniupa@ubiobio.cl

ABSTRACT

In order to deal with inconsistent databases, a repair semantics defines a set of admissible database instances that restore consistency, while staying close to the original instance. This set can be used to characterize consistent data and consistent query answers in inconsistent databases. In this work we present a repair semantics for spatial databases and spatial integrity constraints, i.e. constraints that combine semantic and topological aspects of spatial data. We also propose the notion of consistent answer to a spatial conjunctive query. This introduces the idea of inconsistency tolerance in the spatial domain, shifting the goal from the consistency of a spatial database to the consistency of query answers.

Categories and Subject Descriptors

H.2.8 [**Database Management**]: Database Applications—*Spatial databases and GIS*

General Terms

Theory

Keywords

Consistency, inconsistency tolerance, repair semantics

1. INTRODUCTION

The consistency of a database is defined as the satisfaction of a set of integrity constraints (ICs) that describes admissible states of the database. Although consistency is a desirable and usually enforced property of databases, it is non uncommon to find inconsistent spatial databases due to data integration, unforced integrity constraints, or time lag updates. In cases of inconsistencies, there are alternative courses of action: (a) ignore inconsistencies, (b) restore consistency via updates on the database, or (c) accept inconsistencies, without changing the database, but compute the "consistent or correct" answers to queries. For many reasons, the first two alternatives may not be appropriate [5], specially in the case of virtual data integration [4], where centralized and global changes to the data sources are not allowed. In this paper we explore the latter alternative in the spatial domain, i.e. in spatial databases and with respect to spatial semantic integrity constraints (SICs).

Extracting consistent data from inconsistent databases could be qualified as an "inconsistency tolerant" approach to querying databases. Intuitively, a piece of data will be part of a consistent answer if it is not logically related to the inconsistencies in the database with respect to its set of ICs. We introduce this idea using an informal and motivating example.

EXAMPLE 1. Consider a database instance with a relation *Landparcel*, with a thematic attribute (*idl*), and a spatial attribute, *geometry*, of data type *polygon*. An IC stating that geometries of two different land parcels must be disjoint or just touch, i.e. they cannot overlap, is expected to be satisfied. However, the instance in Figure 1 does not satisfy it, i.e. it is inconsistent: the land parcels with idls idl_2 and idl_3 overlap. Notice that these geometries are partially in conflict and what is not in conflict can be considered as consistent data.

Landparcel	*idl*	geometry
	idl_1	g_1
	idl_2	g_2
	idl_3	g_3

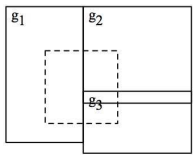

Figure 1: An inconsistent spatial database

Suppose that a query requests all land parcels whose geometries intersect with a query window, which represents the spatial region shown in Figure 1 as a rectangle with dashed borders. Although the database instance is inconsistent, we can still obtain useful and meaningful answers. In this case, only the intersection of g_2 and g_3 is in conflict, but the rest of both geometries can be considered consistent and should be part of a "database repair". Thus, since the no-conflicting parts of geometries g_2 and g_3 intersect the query window, we would expect an answer including land parcels with identities idl_1, idl_2 and idl_3. □

In contrast to (in)consistency handling in relational databases, not much research of this kind has been done for spatial databases. In the spatial domain, some related studies address the specification of integrity constraints [6, 16], and checking topological consistency at multiple representations

CM GIS '08 November 5-7, 2008. Irvine, CA, USA.

and for data integration [11, 24, 12, 17]. More recently, [10] proposes qualitative reasoning with description logics to describe consistency between geographic data sets. In [18] a set of abstract relations between entity classes was defined; and they could be used to discover redundancies and conflicts in sets of SICs. In [22] some issues around query answering under violations of functional dependencies involving geometric attributes were raised. However, the problem of handling an inconsistent spatial database has not been systematically addressed so far.

Consistent query answering (CQA) from inconsistent databases was introduced and studied in the context of relational database in [1] (cf. [5, 3, 8] for surveys). Consistent answers to queries are those that are invariant under all the minimal forms of restoring consistency of the original database. Thus, the notion of *repair* of an instance with respect to a set of ICs becomes a fundamental concept for defining consistent query answers. A *repair semantics* defines the admissible and consistent alternative instances to an inconsistent database at hand. More precisely, a *repair* of an inconsistent relational instance D is a consistent instance D' obtained from D by deleting or inserting whole tuples. This set of tuples by which D and D' differ is minimal under set inclusion [1]. Other repair semantics have been studied in the relational case. For example, in [25, 14] repairs are obtained by allowing updates of attribute values in tuples.

In this work we define a repair semantics for spatial databases with respect to spatial semantic integrity constraints (a.k.a. topo-semantic integrity constraints) [23], which impose semantics on topologies. For example, they can specify that "building blocks must be inside land parcels". This class of constraints is related to many, if not most, cases of inconsistency of spatial databases, and has not received enough attention from the research community. These constraints will be simply called *spatial integrity constraints* (SICs). Other spatial integrity constraints [9] are *domain (topological or geometric) constraints*, and refer to the geometry, topology, and spatial relations of the spatial data types. One of them could specify that "polygons must be closed". This kind of constraints are now commonly integrated into spatial DBMSs [19].

Our repair semantics considers restoring consistency of spatial databases through virtual changes of geometries that participate in violations of SICs. This requires the corresponding formalization of a repair of a spatial database instance. After doing that, we characterize consistent information in an inconsistent spatial database. A particular case is the notion of consistent answer to a spatial query, as an answer that can be obtained for the given query from all the admissible repairs.

Repair semantics and consistent query answers can be defined for a fairly broad class of SICs and queries. However, as it becomes clear soon, naive algorithms for computing consistent answers on the basis of the computation of repairs are of exponential time. For this reason, we investigate classes of SICs and queries for which CQA becomes tractable. Actually, CQA for a relevant subset of SICs and spatial range queries can be done via a *core computation*; that is, by querying directly the intersection of all repairs of an inconsistent database instance, but without actually computing the repairs. We show cases where this core can be specified as a view of the original, inconsistent database.

In comparison to the relational case, spatial databases offer new alternatives and challenges when defining a repair semantics. This is due, in particular, to the use of complex attributes to represent geometries, their combination with thematic attributes, and the nature of topological relations.

A repair semantics for spatial databases can be evaluated from different points of view. Of course, it has to be mathematically and logically sound. It also has to make sense from the point of view of modelling and manipulation of spatial data, i.e. from the point of view of its potential uses, intuitions around spatial data, and practical experience. These are our main two goals in this paper. However, there is still a third criterion that could be used to assess a proposed repair semantics: its possible computational implementation. Although we have left outside the scope of this work a full analysis of the repair semantics in terms of computability, complexity and implementation issues (which are all left for future work), some of the choices we make here for the semantics have been motivated by the possibility of implementing it.

This paper is organized as follows. In Section 2 we describe the spatial data model upon which we define the repair semantics. Section 3 analyzes alternative ways to restore consistency of spatial databases and their consequences. A formal definition of repair for spatial inconsistent databases under SICs is introduced in Section 4. In Section 5 we define consistent answers to spatial queries, and give a strategy to compute, in polynomial time (data complexity), consistent query answers for a class of SICs and range queries. Final conclusions and future research directions are given in Section 6.

2. PRELIMINARIES

Current models of spatial data are typically seen as extensions of the relational data model (object-relational models), with the definition of abstract data types to specify spatial or geometric attributes. We now introduce a general spatio-relational data model that includes spatio-relational predicates (that can be purely relational), spatial predicates, and also spatial ICs. It uses some of the definitions introduced in [20]. The model is independent of the geometric data model (e.g. Spaghetti, topological, raster, or constraint model) underlying the representation of spatial data types.

A spatio-relational database schema is of the form $\Sigma = (\mathcal{U}, \mathcal{A}, \mathcal{R}, \mathcal{T}, \mathcal{O}, \mathcal{B})$, where: (a) $\mathcal{U}$ is the possibly infinite database domain of atomic thematic values. (b) $\mathcal{A}$ is a set of thematic, non-spatial, attributes. (c) $\mathcal{R}$ is a finite set of spatio-relational predicates [1] whose attributes belong to $\mathcal{A}$ or are spatial attributes. The latter take admissible values in $\mathcal{P}(\mathbb{R}^m)$, the power set of $\mathbb{R}^m$, for an m that depends on the predicate. (d) $\mathcal{T}$ is a fixed set of binary spatial predicates. (e) $\mathcal{O}$ is a fixed set of geometric operators that take spatial arguments. (f) $\mathcal{B}$ is a fixed set of built-in relational predicates, like comparison predicates, e.g. $<, >, =, \neq$, that apply to thematic attribute values.

A database instance D of a spatio-relational schema Σ is a finite collection of ground atoms (or *spatial database tuples*) of the form $R(c_1, ..., c_n; s)$, where $R \in \mathcal{R}$, $\langle c_1, ..., c_n \rangle \in \mathcal{U}^n$ contains the thematic attribute values and $s \in \mathcal{P}(\mathbb{R}^m)$[2].

[1] spatio-relational predicates are relations in our database schema.

[2] For simplicity, we use one spatial attribute, but it is not difficult to consider a greater number of spatial attributes.

The extension in a particular instance of a spatio-relational predicate is a subset of $\mathcal{U}^n \times \mathcal{P}(\mathbb{R}^m)$. For simplicity, and to fix ideas, we will consider the case where the spatial type of predicates is $m = 2$.

Among the different abstraction mechanisms for modelling single spatial objects, we use regions for modelling real objects that have an extent. They are useful in a broad class of applications in Geographic Information Systems (GISs). More specifically, our model follows the specification of spatial operators (spatial relations or geometric operations) as found in current spatial database management systems [19], and considers closed regions in the topological space formed by point sets of the Euclidean space.

A region is either the empty region (or empty geometry), $g_\emptyset$, which corresponds to the empty subset of the plane, or is defined as a finite set of polygons of positive area. In this work, we represent a polygon as a ring or a simple polyline whose first and last points coincide. As expected, it holds $g_\emptyset \cap g = g \cap g_\emptyset = g_\emptyset$, for every region g. From now on, regions of $\mathbb{R}^2$ that conform to these spatial data types are called *admissible regions*.

Geometric attributes are complex data types, and their manipulation may have an important effect on the computational costs of certain algorithms and algorithmic problems. As usual, we are interested in *data complexity*, i.e. in terms of the size of the database. The *size* of a spatio-relational table (relation) can be defined in terms of the number of tuples and the number of points that are necessary to represent the geometry in a tuple.

We concentrate on binary (i.e. two-ary) spatial predicates that represent topological relations between regions. They have a fixed semantics, and become the elements of $\mathcal{T}$. There are eight basic binary relations over *non-empty* regions of $\mathbb{R}^2$ (dark boxes in Fig. 2) [13, 21]: *PO* (*PartialOverlaps*), *EQ* (*Equal*), *CB* (*CoveredBy*), *IS* (*Inside*), *CV* (*Covers*), *IC* (*Includes*), *TO* (*Touches*), *D* (*Disjoint*), *INT* (*Intersects*), *O* (*Overlaps*), *W* (*Within*), and *C* (*Contains*) [21, 13]. Some of them are symmetric, like *Disjoint*, *PartialOverlaps*, *Equal*, *Touches*. The others have their converse relation within the set.

In addition to the basic topological relations, we consider four *derived relations* (white boxes in Fig. 2) that are also included in query languages of current spatial database system [19]. These predicates can be logically defined in terms of the other basic predicates.

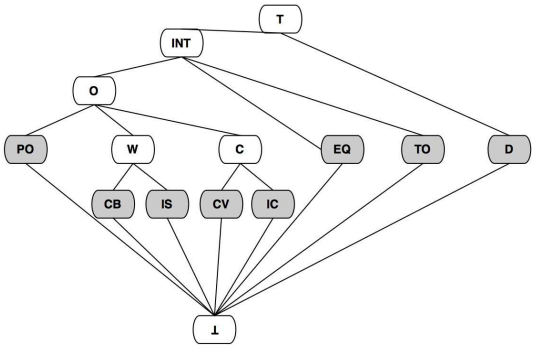

Figure 2: Subsumption lattice of relations

Given a database instance, additional spatial information is usually computed from the explicit geometric data by means of the spatial operators in $\mathcal{O}$ associated with Σ. Some relevant operators are: *Union* (binary), *Intersection*, *Difference*, *Buffer*, and *Union Aggregation* (*GeomUnion*)[3] (cf. [19] for the complete set of spatial predicates defined within the Open GIS Consortium). In this work, we will identify a particular subset $\mathcal{O}^a$ of spatial operators in $\mathcal{O}$, i.e., $\mathcal{O}^a \subseteq \mathcal{O}$. They will be used to shrink geometries with the purpose of restoring consistency, as we describe in Section 4.

Definition 1. The set $\mathcal{O}^a$ of *admissible transformation operations* contains the following geometric operations on closed and possibly empty geometries g and g':
(1) *Difference*(g, g') is the closure of the set difference between g and g'. More precisely: (a) *Difference*$(g, g) = g_\emptyset$. (b) *Difference*$(g, g_\emptyset) = g$. (c) *Difference* $(g_\emptyset, g) = g_\emptyset$. (d) For non-empty regions $g \neq g'$, *Difference*$(g, g') = (g \setminus g') \cup (boundary(g') \cap g)$.
(2) *Buffer*(g, d) is the geometry obtained by buffering a distance d around g, where d is a distance unit. *Buffer*(g, d) returns a closed region $\bar{g}$ containing geometry g, such that every point in the boundary of $\bar{g}$ is at a distance d from the boundary of g. However, *Buffer*$(g_\emptyset, d) = g_\emptyset$.
(3) *Identity*(g) = *Difference*$(g, g_\emptyset) = g$. □

Notice that all these operations, when applied to admissible geometric regions, produce closed, admissible regions.

A schema Σ determines a many-sorted, first-order language $\mathcal{L}(\Sigma)$ of predicate logic. It can be used to syntactically characterize and express SICs. For simplicity, we concentrate on *denial SICs*,[4] which are sentences of the form:

$$\forall^{\neq g_\emptyset}\ s_1 \cdots s_k \forall \bar{x} \neg (\bigwedge_{i=1}^{m} R_i(\bar{x}_i; s_i) \wedge \varphi \wedge \bigwedge_{j=1}^{n} T_j(v_j, w_j)), \quad (1)$$

where $0 \leq m, n \in \mathbb{N}$, $\bar{x} = \bigcup_1^m \bar{x}_i$, $v_k \in \bigcup_{j=1}^m \{s_j\}$, $w_k \in (\bigcup_{j=1}^m \{s_j\}) \cup (\bigcup_{j=1}^n \{\alpha_j\})$, α_j is a user defined query window, $R_1, \ldots, R_m \in \mathcal{R}$, φ is a formula containing built-in atoms over thematic attributes, $T_j \in \mathcal{T}$, and $\bar{\forall}$ denotes the universal closure. A constraint of the form (1) prohibits certain combinations of atoms for non-empty geometries.[5] The restriction to non-empty geometries is because predicates in $\mathcal{T}$ are undefined for empty geometries.

Example 2. Figure 3 shows an instance for the schema $\mathcal{R} = \{Landparcel(idl, name, owner; geometry), Building(idb; geometry)\}$. Dark rectangles represent buildings and white rectangles represents land parcels. In *Landparcel*, the thematic attributes are *idl*, *name* and *owner*, whereas *geometry* is the spatial attribute of dimension 2. Similarly for *Building*, which has only *idl* as a thematic attribute.

Landparcel	*idl*	*name*	*owner*	*geometry*
	idl_1	n_1	o_1	g_1
	idl_2	n_2	o_2	g_2
	idl_3	n_3	o_3	g_3

[3]Operator *GeomUnion* returns the geometry that represents the point set union of all geometries in a given set. Although this function is part of SQL for several spatial databases (Postgres/PostGIS, Oracle), it is not explicitly defined in the OGC specification [19].
[4]Denial constraints are easier to handle in the relational case as consistency with respect to them is achieved by tuple deletions only [5].
[5](1) can be replaced by a FO sentence with relativized quantifiers. In the future, we may omit the $\neq g_\emptyset$ in (1) or leave quantifiers implicit.

Building	*idb*	*geometry*
	idb_1	g_4
	idb_2	g_5

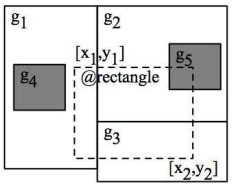

Figure 3: A spatial database instance

The following sentences are the denial SICs:

$$\neg(Landparcel(id_1, n_1, o_1; s_1) \wedge Landparcel(id_2, n_2, o_2; s_2) \wedge id_1 \neq id_2 \wedge Overlaps(s_1, s_2)). \quad (2)$$

$$\neg(Building(idb; s_1) \wedge Landparcel(idl, n_1, o_1; s_2) \wedge PartialOverlaps(s_1, s_2)). \quad (3)$$

They specify that geometries of land parcels with different *ids* do not overlap, and that building blocks cannot partially overlap land parcels. □

A database instance D for schema can be seen as a finite set of ground atoms from language $\mathcal{L}(\Sigma)$ that use predicates in $\mathcal{R}$. For a spatial database instance D for schema Σ and a set Ψ of SICs in $\mathcal{L}(\Sigma)$, $D \models \Psi$ denotes that each of the constraints in Ψ is true in (or satisfied by) D. In this case, we say that D is *consistent* with respect to Ψ. Correspondingly, D is *inconsistent* wrt Ψ, denoted $D \not\models \Psi$, when there is $\psi \in \Psi$ that is *violated* by D, i.e. not satisfied by D. The instance in Example 2 is consistent with respect to its SICs.

3. CONSISTENCY RESTORATION

Notice that D violates a constraint $\forall \bar{x}_1 \bar{x}_2 s_1 s_2 \neg(R_1(\bar{x}_1; s_1) \wedge R_2(\bar{x}_2; s_2) \wedge \varphi \wedge T(s_1, s_2))$ when there are data values $\bar{a}_1, \bar{a}_2$, g_1, g_2 for the variables for which $(R_1(\bar{x}_1; s_1) \wedge R_2(\bar{x}_2; s_2) \wedge \varphi \wedge T(s_1, s_2))$ becomes true in D, i.e. $D \models (R_1(\bar{x}_1; s_1) \wedge R_2(\bar{x}_2; s_2) \wedge \varphi \wedge T(s_1, s_2))$ $[\bar{a}_1, \bar{a}_2, g_1, g_2]$. When this is the case, it is possible to restore consistency of D by making $T(g_1, g_2)$ false by replacing g_1 or g_2 by other admissible geometries, which could be done in different ways. This suggests that if D is inconsistent with respect to a set Ψ of denial SICs, we may try to restore consistency by performing admissible updates on the geometries. These updates can be represented as sequences of spatial operators.

One of the key criteria to decide about the update to apply is minimality of geometric changes. Another important criteria may be the semantics of spatial objects, which makes changes over the geometry of one type of object more appropriate than others. In this work, however, we assume that no previous knowledge about the quality and relevance of geometries exists and, therefore, geometries are all equally important.

We start by identifying a suitable class of geometric transformations using some (anti) monotonicity properties of topological relations with respect to set-inclusion of regions, $\subseteq$. These properties would allow us to discard certain changes on geometries. For example, for $T = Intersects$, it holds

$$T(s_1, s_2) \wedge s_1 \subseteq s_1' \Rightarrow T(s_1', s_2) \quad (4)$$

$$T(s_1, s_2) \wedge s_2 \subseteq s_2' \Rightarrow T(s_1, s_2') \quad (5)$$

$$T(s_1, s_2) \wedge s_1 \subseteq s_1' \wedge s_2 \subseteq s_2' \Rightarrow T(s_1', s_2') \quad (6)$$

$$\neg T(s_1, s_2) \wedge s_1' \subseteq s_1 \Rightarrow \neg T(s_1', s_2) \quad (7)$$

$$\neg T(s_1, s_2) \wedge s_2' \subseteq s_2 \Rightarrow \neg T(s_1, s_2') \quad (8)$$

$$\neg T(s_1, s_2) \wedge s_1' \subseteq s_1 \wedge s_2' \subseteq s_2 \Rightarrow \neg T(s_1', s_2'). \quad (9)$$

From (4) to (6) we can see that the *Intersects* relation between two regions is kept if we enlarge one or both of them. So, if we have to make the topological predicate $Intersects(s_1, s_2)$ false, we cannot make changes that only enlarge geometries. (7)-(9) indicate that when two geometries do not intersect, shrinking them will keep them disjoint; and also that if we shrink a geometry to satisfy a SIC based on *Intersects*, no new inconsistencies will be added with respect to this predicate.

Relation T	(Anti) Monotonicity Property
Disjoint	$T(s_1, s_2) \wedge s_1' \subseteq s_1 \Rightarrow T(s_1', s_2)$ $\neg T(s_1, s_2) \wedge s_1 \subseteq s_1' \Rightarrow \neg T(s_1', s_2)$
Inside	$T(s_1, s_2) \wedge s_1' \subseteq s_1 \Rightarrow T(s_1', s_2)$ $T(s_1, s_2) \wedge s_2 \subseteq s_2' \Rightarrow T(s_1, s_2')$ $\neg T(s_1, s_2) \wedge s_1 \subseteq s_1' \Rightarrow \neg T(s_1', s_2)$ $\neg T(s_1, s_2) \wedge s_2' \subseteq s_2 \Rightarrow \neg T(s_1, s_2')$
Equal	$T(s_1, s_2) \wedge s_1' \subset s_1 \Rightarrow \neg T(s_1', s_2)$

Table 1: Monotonicity of topological relations

Table 1 shows (anti) monotonicity properties for other topological relations. We omit some of the monotonicity properties for the symmetric relations *Disjoint* and *Equal*. The table shows that enlarging a geometry makes an atom $T(s_1, s_2)$ false when T is *Disjoint* or *Inside*. In the latter case, when the enlarged geometry is inside the other.

Shrinking geometries will solve inconsistencies with respect to SICs involving any topological relations, except *Disjoint* and a specific case of *Intersects*. For instance, when the geometries in the database satisfy the topological relation *Equal* (a particular case of *Intersects*), in which case shrinking any of the inconsistent geometries will solve the inconsistency with respect to *Equal*, but not with respect to *Intersects*. In these particular cases, we can still solve inconsistency by shrinking geometries but allowing a geometry g to become *empty*, i.e. $g_\emptyset$. This can be interpreted as cancelling the geometry without deleting the whole tuple. This is possible because we are not considering constraints over metric measures of the geometric representation, such as a minimum area.

Other geometric changes may be combinations of shrinking and enlarging a geometry. Actually, the translation of a geometry could be conceived as first shrinking it and then enlarging it at a different place. Although it is easy to solve a particular inconsistency with respect to a particular SIC, we might generate new inconsistencies with respect to the same or other SICs. Figure 4(a) shows an inconsistent instance with respect to (2): geometries g_1 and g_2 overlap. Shrinking g_2 restores consistency (Figure 4(b)). However, translating g_2 down eliminates the inconsistency with respect to g_1, but introduces a new inconsistency with respect to g_3 (Figure 4(c)).

On the basis of this analysis, we propose to solve inconsistencies with respect to SICs of the form (1) through shrink-

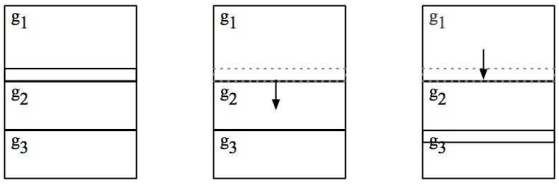

Figure 4: Effect of geometric changes

ing or cancellation of geometries. From an ontological point of view, by shrinking we eliminate conflicting parts of geometries, instead of adding new uncertain geometries by enlargement. We have to define how much to shrink a geometry in order to restore consistency. Actually, we may have a continuum of possibilities for shrinking geometries.

In Figure 1, we could shrink both g_2 and g_3, so that they touch just at the border, but this border could be placed anywhere in the region where they originally overlap. Having infinitely many possible repairs carries obvious problems when thinking of practical implementations. In particular, it may not be possible to compute them if needed. Even if we do not want to compute them, representing them and possibly querying them, even implicitly, offer practical difficulties. In Figure 1 (and in general), we will consider only the "extreme" repairs (only two in Figure 1), but not the continuum in between.

In summary, consistency restoration will be achieved by shrinking or cancelling geometries, and only a discrete number of extreme repairs will be considered. In Section 4, this *update based* methodology will be expressed in terms of sequences of *admissible operations* in $\mathcal{O}^a$.

4. A REPAIR SEMANTICS

We can compare geometries and their transformations by means of a distance function that refers to their areas. We assume that $area \in \mathcal{O}$ is an operator that computes the area of a geometry.

DEFINITION 2. For regions g_1, g_2, $\delta(g_1, g_2) = area(g_1) - area(g_2)$. □

The amount of geometric change could be measured in different ways, in particular if we allow geometric translations. In such case, the area may remain the same, but not the spatial position. However, as we will modify geometries by shrinking them, using the just introduced distance makes sense. We will be comparing a region g_1 with a region g_2 that is obtained by shrinking g_1, and then it will hold $\delta(g_1, g_2) \geq 0$. Since we need the distance between two instances via the geometric attributes, we will assume that it is possible to compare geometries by correlating their (relational-topological) tuples, one by one. This requires a correspondence between instances. In the following definition, $D|\mathcal{A}$ stands for the restriction of instance D to a subset $\mathcal{A}$ of its attributes.

DEFINITION 3. Let D be a fixed database instance over schema Σ. (a) An instance D' over Σ is (D, f)-*indexed* if f is a bijective function from $D|\mathcal{A}$ to $D'|\mathcal{A}$, such that $f(R(c_1, \ldots, c_n; s)) = R(c_1, \ldots, c_n; s')$ for some region s'. (b) For a (D, f)-*indexed* instance D' and tuple $R(c_1, \ldots, c_n; s) \in D'$, $f^{-1}(s)$ denotes $f^{-1}(R(c_1, \ldots, c_n; s))$. □

In an (D, f)-indexed instance D' we can compare tuples one by one with their counterparts in instance D. In particular, we can see how the geometric attribute values differ. In some cases there is an obvious function f (in which case we simply use the notion of D-indexed), for example when there is a key from a subset of $\mathcal{A}$ to the spatial attribute S.

EXAMPLE 3. (example 2 cont.) Consider the relational schema *Landparcel*(*idl*, *name*, *owner*; *geometry)*, with $\mathcal{A} = \{idl, name, owner\}$. For the given instance D in Example 2, the following are the $D|\mathcal{A}$-instance, and (D, f)-indexed instance, resp.

Landparcel$\vert\mathcal{A}$	*idl*	*name*	*owner*
	idl_1	n_1	o_1
	idl_2	n_2	o_2
	idl_3	n_3	o_3

Landparcel	*idl*	*name*	*owner*	*geometry*
	idl_1	n_1	o_1	g_7
	idl_2	n_2	o_2	g_8
	idl_3	n_3	o_3	g_9

Here, $f(Landparcel(idl_1, n_1, a_1; g_1)) = Landparcel(idl_1, n_1, a_1; g_7)$, etc. □

When restoring consistency, it may be necessary to consider complex combinations of tuples and SICs. Eventually, we should obtain a new instance, hopefully consistent, that we have to compare to the original instance in terms of their distance.

DEFINITION 4. Let D, D' be spatial database instances over the same schema Σ, with D' (D, f)-indexed. The *distance* $\Delta(D, D')$ between D and D' is the numerical value $\Delta(D, D') = \Sigma_{\bar{t} \in D} \delta(\Pi_S(\bar{t}), \Pi_S(f(\bar{t})))$, where $\Pi_S(\bar{t})$ is the projection of tuple $\bar{t}$ on its spatial attribute S. □

Now it is possible to define a "repair semantics", which is independent of the geometric operators used to shrink geometries.

DEFINITION 5. Let D be a spatial database instance over schema Σ, Ψ a set of SICs, such that $D \not\models \Psi$. (a) An *s-repair* of D with respect to Ψ is a database instance D' over Σ, such that: (i) $D' \models \Psi$. (ii) D' is (D, f)-indexed. (iii) For every tuple $R(c_1, \ldots, c_n; g) \in D$, if $f(R(c_1, \ldots, c_n; g)) = R(c_1, \ldots, c_n; g')$, then $g' \subseteq g$. (b) A *minimal s-repair* D' of D is a repair of D such that, for every repair D'' of D, it holds $\Delta(D, D'') \geq \Delta(D, D')$. □

This is an "ideal and natural" repair semantics that defines a collections of *semantic repairs*. The definition is purely set-theoretic and topological in essence. It is worth exploring the properties of this semantics and its impacts on properties of consistent query answers (as invariant under minimal s-repairs) and reasoning from or with them. However, in this work we will use an alternative repair semantics that is more operational in nature. Under this alternative notion of repair, we will have a finite number of them for a given instance, instead of a continuum, as may be the case, for s-repairs (cf. Section 3).

The *operational* definition of repair (cf. Definition 6) will make it possible to deal with repairs in current spatial database management systems and in terms of standard geometric operators. Consistency will be restored by applying

a finite sequence of admissible transformation operations to conflicting geometries. Now we have to analyze and propose those operations.

It is easy to see that each true relationship (atom) of the form $T(g_1, g_2)$, with $T \in \mathcal{T}$, can be falsified by applying a sequence of operations in $\mathcal{O}^a$ to g_1 and/or g_2. Important considerations at the moment of defining this sequence of operators are: (a) the minimality condition; that is, we only eliminate conflicting parts of geometries. (b) obtaining only extreme repairs and not the whole continuum in between.

In the following, we indicate, for each relation $T \in \mathcal{T}$, alternative sequences of operators that falsify a true atom of the form $T(g_1, g_2)$. Table 2 shows our *list of admissible transformations*. This table prescribes particular and legal ways of applying the admissible operators of Definition 1. In some case, those operations have to be applied in complete sequences. Only those operations or sequences thereof will be accepted in our operational definition of repair (cf. Definition 6). Each entry in the table of the form $g_1' = \cdots$, $g_2' = \cdots$. under a same topological predicate T is called an *admissible transformation*.

Table 2 shows that, for *PartialOverlaps* (PO), we have two ways of transforming each of the geometries g_1, g_2 into g_1', g_2' resp., in such a way that when $PartialOverlaps(g_1, g_2)$ is true, $PartialOverlaps(g_1', g_2')$ becomes false. The selection will depend on the relative size between overlapping and non-overlapping areas: (1) the overlapping area between g_1 and g_2 using $Difference(g_1, g_2)$ or $Difference(g_2, g_1)$ (first example for PO in Table 3); and (2) the non-overlapping part of g_1 and g_2 using $Difference(g_1, Difference(g_1, g_2))$ or $Difference\ (g_2, Difference(g_2, g_1))$ (second example for PO in Table 3) (cf. Example 4).

Op. T	$T(g_1, g_2)$ becomes false by applying...
PO	1. $g_1' = Difference(g_1, g_2)$, $g_2' = Identity(g_2)$. 2. $g_1' = Identity(g_1), g_2' = Difference(g_2, g_1)$. 3. $g_1' = Difference(g_1, Difference(g_1, g_2))$, $g_2' = Identity(g_2)$. 4. $g_1' = Identity(g_1)$, $g_2' = Difference(g_2, Difference(g_2, g_1))$.
IC, CV, IS, CB, O, W, C	1. $g_1' = Difference(g_1, g_2)$, $g_2' = Identity(g_2)$. 2. $g_1' = Identity(g_1)$, $g_2' = Difference(g_2, g_1)$.
TO, INT	1. $g_1' = Difference(g_1, buffer(g_2, d))$, $g_2' = Identity(g_2)$. 2. $g_1' = Identity(g_1)$, $g_2' = Difference(g_2, buffer(g_1, d))$.
EQ, D	1. $g_1' = Difference(g_1, g_1)$, $g_2' = Identity(g_2)$. 2. $g_1' = Identity(g_1)$, $g_2' = Difference(g_2, g_2)$.

Table 2: Admissible transformations

Table 2 also shows that *Touches* and *Intersects* are predicates for which the eliminated area is not completely delimited by the real boundary of objects. Actually, we need to separate the touching boundaries, and we do so by buffering a distance d around one of the geometries and taking the overlapping part from the other one. Here, we consider d to be a fixed value associated with the minimum size of a geometry in the cartographic scale of the database instance.

Table 3 graphically illustrates the application of the admissible transformation to restore consistency of predicates $T = PartialOverlaps$, $T = Disjoint$, and $T = Touches$. The dashed boundary is the result of applying $Buffer(g, d)$.

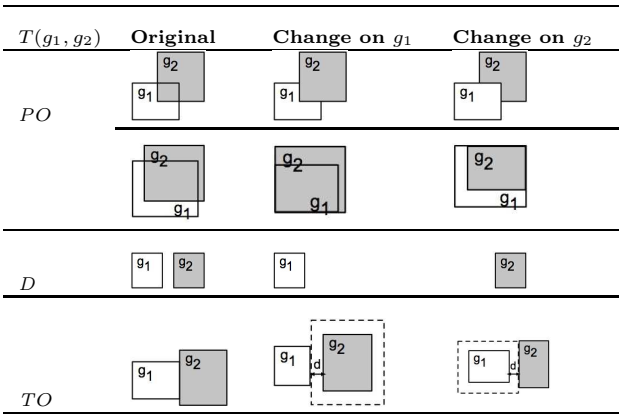

Table 3: Examples of admissible transformations

We now introduce the formal definition of repair semantics that is used in the definition of consistent query answers.

Definition 6. Let D be an instance over schema Σ, Ψ a set of SICs, such that $D \not\models \Psi$. (a) A *repair* of D with respect to Ψ is a database instance D' over Σ, such that: (i) $D' \models \Psi$. (ii) D' is (D, f)-indexed. (iii) For every tuple $R(c_1, \ldots, c_n; g) \in D$, if $f(R(c_1, \ldots, c_n; g)) = R(c_1, \ldots, c_n; g')$, then $g' \subseteq g$ and g' is the result of applying a finite sequence of admissible transformations (cf. Table 2) to the initial geometries in D. (b) A *minimal repair* D' of D is a repair of D such that, for every repair D'' of D, $\Delta(D, D'') \geq \Delta(D, D')$. (c) $Rep(D, \Psi)$ denotes the set of minimal repairs of D with respect to Ψ. □

The use of the admissible transformation operations to restore consistency enforces that: (a) regions in conflict are shrunk, (b) the whole conflicting areas are eliminated, (c) a finite number of minimal repairs is associated to a given instance (instead of infinitely many, even a continuum, under the s-repair semantics), (d) repairs can be computed (if needed) using existing spatial DBMS. These useful properties are not necessarily shared by the class of minimal s-repairs, which may be more natural from a theoretical point of view. As we will now discuss, it is not necessarily the case that a minimal repair in the sense of Definition 6 is a minimal s-repair.

As consequence of removing the whole areas in conflict in a repair, in addition to the cases of relations $Disjoint$ and $Intersects$ as discussed in Section 3, we may also have to make a geometry empty ($g_\emptyset$) for SICs involving relations $Equal$, $Inside$ and $ConveredBy$. For example, for topological predicate $Equal$, an s-repair will be obtained by shrinking, as much as needed, one of the geometries, but without making the geometries empty. However, by using the admissible transformations, we restore consistency with respect to $Equal$ by making empty one of the geometries in conflict. Clearly, what we obtain is not a minimal s-repair. Nevertheless, this solution has also practical sense, since having two

geometries that are topologically equal could, in most cases, be the result of duplicate data, and one of them should be eliminated. In this case, we should consider if not only a geometry should be eliminated, but the whole tuple containing it.

EXAMPLE 4. (example 2 cont.) The instance D in Figure 5 is inconsistent with respect to (2) and (3), because the land parcels with geometries g_2 and g_3 overlap, and also the land parcels with geometries g_2 and g_4. Likewise, buildings with geometry g_5 and g_6 overlap land parcels with geometries g_1 and g_2, respectively.

Figure 6 shows the two minimal repairs of D. In them, the regions with thicker boundaries are the regions that had their geometries changed.

Landparcel	*idl*	*name*	*owner*	*geometry*
	idl_1	n_1	o_1	g_1
	idl_2	n_2	o_2	g_2
	idl_3	n_3	o_3	g_3
	idl_4	n_4	o_4	g_4

Building	*idb*	*geometry*
	idb_1	g_5
	idb_2	g_6

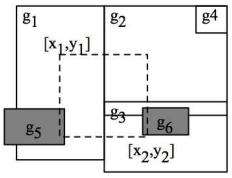

Figure 5: Inconsistent spatial database instance

For example, the inconsistency involving geometries g_2 and g_3 is repaired by applying *Difference*(g_2, g_3) to g_2, i.e. removing from g_2 the whole overlapping geometry, and keeping the geometry of g_3 as originally. The latter is obtained by applying *Identity*(g_3) to g_3, which is shown in Figure 6(a), and (b). Notice that if we apply *Difference*(g_3, g_2) to g_3, i.e. we remove the whole overlapping area from g_3, we still have an inconsistency, because the building with geometry g_6 will continue partially overlapping geometries g_2 and g_3. Then, this change will require another transformation, to ensure that g_6 is completely covered or inside of g_3.

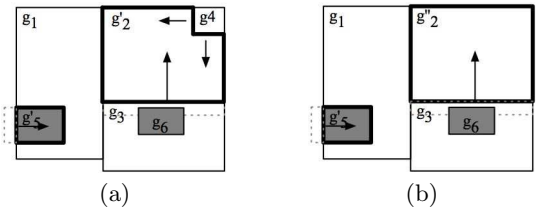

Figure 6: Spatial database repairs

The inconsistency between g_2 and g_4 is repaired by shrinking g_2, eliminating the overlapping area with respect to g_4 (cf. Figure 6(a)). This change is obtained by applying *Difference*(g_2, g_4) to g_2; or by eliminating geometry g_4, i.e. applying *Difference* $(g_4, g_2) = g_\emptyset$, as illustrated in Figure 6(b). Finally, the inconsistency between g_1 and g_5 is repaired by taking the non-overlapping part of geometry g_5 with respect to geometry g_1. Notice that we could have repaired by eliminating the overlapping region between g_1 and g_5, but this is not a minimal change. □

Since spatial operators remove the whole conflicting geometries associated with objects in conflict, we obtain a finite number of repairs, which is important if we need to compute them. However, the following example shows that, even applying the admissible transformations, there may be exponentially many minimal repairs in the size of the database, a phenomenon already observed with relational repairs with respect to functional dependencies [5].

EXAMPLE 5. Consider the schema in Example 2, and the SIC (2). The database instance contains n spatial tuples, as shown in Figure 7. There are $n-1$ overlapping geometries.

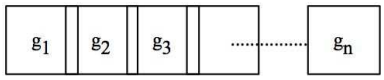

Figure 7: Exponential number of repairs

In order to solve each of those overlaps, we have the options of shrinking either one of the two regions involved. We have 2^{n-1} possible minimal repairs. □

5. CONSISTENT QUERY ANSWERS

We can use the minimal repairs as an auxiliary concept to define, and possibly compute, consistent answers to a relevant class of queries in $\mathcal{L}(\Sigma)$. We concentrate here on *operator free* conjunctive queries, i.e. conjunctive queries of the form

$$\mathcal{Q}(\bar{u}): \ \exists \bar{y}(R_1(\bar{x}_1; s_1) \wedge \cdots \wedge R_n(\bar{x}_n; s_n) \wedge \varphi), \qquad (10)$$

where φ is a conjunction of built-in atoms over thematic attributes and over spatial attributes of the form $T(t_1, t_2)$, where $T \in \mathcal{T}$, and t_1, t_2 are spatial terms (geometric variables or query windows α). Also, $\bar{y} \cup \bar{u} \subseteq \bigcup_i (\bar{x}_i \cup \{s_i\}) \cup Var(\varphi)$, and $\bar{u} \cap \bar{y} = \emptyset$. φ does not contain operators from $\mathcal{O}$. We also add the *safety condition* that all variables appear in some of the R_i.

The classical *spatial range queries* form a particular class of conjunctive queries:

$$\exists \bar{y}(R(\bar{x}; s) \wedge T(s, w)), \qquad (11)$$

with T either *Intersects* or *Overlaps*, and w a spatial constant, e.g. a query window. The free variables in the query are those in $(\bar{x} \cup \{s\}) \smallsetminus \bar{y}$.

EXAMPLE 6. (example 2 cont.) The following is a range query for the instance in Figure 8:
$\mathcal{Q}(idb; geometry): \ Building(idb, geometry) \wedge$
$Overlaps(geometry, [x_1, y_1], [x_2, y_1], [x_2, y_2], [x_1, y_2], [x_1, y_1])$.
Here, the spatial window is the (closed) polygon obtained by joining the four points in the indicated order. The answer to this query is $\langle idb_2, g_5 \rangle$. □

Given a query $\mathcal{Q}(\bar{x}; \bar{s})$, a sequence of thematic/spatial constants $\langle \bar{c}; \bar{g} \rangle$ is an answer to the query in instance D if and

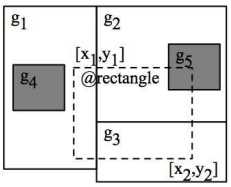

Figure 8: Query example

only if $D \models \mathcal{Q}(\bar{c};\bar{g})$. The formula for $\mathcal{Q}$ becomes true in D when its free variables are replaced by (interpreted as) the constants in $\langle \bar{c};\bar{g}\rangle$. We denote with $\mathcal{Q}(D)$ the set of answers to $\mathcal{Q}$ in instance D. Now we can define the notion of consistent answer to a conjunctive spatial range query.

Definition 7. Consider an instance D, a set Ψ of SICs, and a conjunctive query $\mathcal{Q}(\bar{x};\bar{s})$. A tuple of thematic/geometric constants $\langle c_1,\ldots,c_m;g_1,\ldots,g_l\rangle$ is a *consistent answer* to $\mathcal{Q}$ with respect to Ψ if: (a) For every $D' \in Rep(D,\Psi)$, there exist $g'_1,\ldots,g'_l$ such that $D' \models \mathcal{Q}(c_1,\ldots,c_m;g'_1,\ldots,g'_l)$. (b) g_i is the intersection over all regions g'_i that satisfy (a) and are correlated to the same tuple in D.[6] $Con(\mathcal{Q},D,\Psi)$ denotes the set of consistent answers to $\mathcal{Q}$ in instance D with respect to Ψ. □

Since $\mathcal{Q}$ is operator free, the regions g'_i in repairs appear in relations, and then f^{-1} can be applied. However, due to the intersection, the geometries in a consistent answer may not belong to the original instance or its repairs.

In contrast to the definition of consistent answer to queries in relational databases [1], where a consistent answer is an answer in every repair, here we have an aggregation of query answers via the intersection, similar in spirit to consistent answers to aggregate queries with group-by [5, 15, 7]. This definition of consistent answer allows us to obtain more significant answers: Since we are shrinking geometries, we cannot expect to have, for a fixed tuple of thematic attribute values, the same geometry in every repair. If we did not use the intersection of geometries, we might lose or not have consistent answers due to the lack of geometries in common among repairs.

Example 7. (example 4 cont.) Consider the spatial range query $\mathcal{Q}(idl, geometry)$:
$\exists name\ owner(Landparcel(idl,name,owner,geometry) \wedge Overlaps(geometry,([x_1,y_1],[x_2,y_1],[x_2,y_2],[x_1,y_2],[x_1,y_1]))$.

Consider the two minimal repairs in Figure 6. In them, objects idl_1 and idl_3 do not change geometries, whereas object idl_2 does, from g_2 to g'_2 (cf. Figure 6(a), (b), resp.). Likewise, object idl_4 changes from g_4 to g'_4 ($g_\emptyset$).

idl	$geometry$
idl_1	g_1
idl_2	g'_2
idl_3	g_3

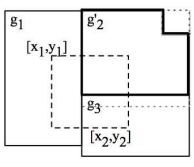

Figure 9: Consistent answers

[6]Via the correlation function f, cf. Definition 3.

The consistent answers are the intersections of geometries in the minimal repairs that also overlap the query window and are associated to the same object. Here, $g'_2 \cap g''_2 = g'_2$, so g_1, g'_2, and g_3 overlap the query window and are part of the consistent answer to this query, cf. Figure 9. Geometries with thicker lines undergo changes. □

Now we establish that for some queries, we can compute consistent answers by focusing on the *core*, i.e. the intersection of all the database repairs of a given instance D with respect to a set Ψ of SICs.

Definition 8. For an instance D and a set Ψ of SICs, the *core* of D is the instance $D^\star$ given by $D^* = \bigcap Rep(D,\Psi) := \{R(\bar{a},g^\star) \mid R \in \mathcal{R}$, there is $R(\bar{a};g) \in D$ and $g^\star = \bigcap\{g' \mid R(\bar{a};g') \in D'$ for some $D' \in Rep(D,\Psi)$ and $R(\bar{a};g') = f(R(\bar{a};g))\}\}$. Here, f is the correlation function. □

Theorem 1. For an instance D, a set Ψ of SICs, and a spatial range query of the form (11), $\langle \bar{a},g\rangle$ is a consistent answer to $\mathcal{Q}$ if and only if $\langle \bar{a},g\rangle \in \mathcal{Q}(D^*)$, where D^* is the core of D. □

The theorem tells us that we can obtain consistent answer by direct and usual query evaluation on the single instance $D^\star$, the core of D.

Example 8. (example 4 cont.) Figure 10(a) represents the *core* for the inconsistent instance in Figure 5, and Figure 10(b), the window for the query in Example 7. □

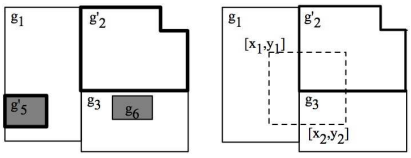

Figure 10: The repairs' core

Computing consistent answers to spatial range queries on the *core* carries computational benefit, since, as we will show, we can compute these answers with respect to a relevant class of SICs without actually explicitly computing repairs. More specifically, this is possible when the SICs are formulas of the form (12):

$$\forall \bar{x_1}\bar{x_2}s_1s_2\neg(\ R(\bar{x_1},s_1)\ \wedge\ R(\bar{x_2},s_2)\ \wedge\ \bar{x_1}\neq\bar{x_2} \wedge\ T^*(s_1,s_2)), \qquad (12)$$

where $T^* \in \{Overlaps, Intersects, Equal\}$. This class of SICs constraints the interrelations between spatial objects that are instances of the same entity.

Notice that there are at most two occurrences of the same spatio-relational predicate in the same SIC, but there could be more than one SIC for the same spatio-relational predicate. Moreover, the topological predicates in this set, when falsified by applying transformations in Table 2, do not produce new inconsistencies with respect to SICs of the form (12).

An additional good property of this class of SICs is that, in the intersection of minimal repairs (i.e. the core), the conflicting parts of each original geometry in conflict is always eliminated. This is not only due to the fact that they use the spatial predicate T^*, but also to the fact that they use it on geometries of the same spatio-relational predicate (i.e. only one spatio-relational predicate per SIC).

The importance of eliminating the parts from each geometry in conflict, as the SICs of the form (12) allow, is that we can give an algorithm to be applied uniformly over each geometry in conflict. In this way we can avoid considering alternative repairs.

In the following we concentrate only on SICs of the form (12) that contain the predicate *Overlaps*. We still allow several SICs with different spatio-relational predicates. We will show how to obtain the core without computing any repair and use it for CQA. The other predicates allowed in (12) can be treated similarly, using other admissible spatial operators than those used for *Overlaps* to compute the *core*.

In order to simplify the notation, in what follows we introduce a logical formula as an abbreviation that captures a conflict around a spatio-relational R with respect to a SIC:

$$\forall \bar{x}_1 \bar{x}_2 s_1 s_2 (Confl_R(\bar{x}_1, s_1, \bar{x}_2, s_2) \equiv (R(\bar{x}_1; s_1) \wedge R(\bar{x}_2; s_2) \wedge \bar{x}_1 \neq \bar{x}_2 \wedge Overlaps(s_1, s_2))).$$

PROPOSITION 1. Let D be an instance and Φ a set of SICs of the form (12) with $T^* = Overlaps$. For the *core* $D^\star$ of D with respect to Φ, it holds
$D^* = \{R(\bar{a}, Difference(g, t)) \mid R \in \mathcal{R}, R(\bar{a}, g) \in D, t = \bigcup\{g' \mid \text{there is } \bar{b}, \text{ such that } D \models Confl_R(\bar{a}, g, \bar{b}, g')\}\}$,
where $\bigcup$ is the *geomUnion* operator. □

Notice that t is the union of all the geometries that are in conflict with a given geometry g. It is obtained by using the aggregation operator *geomUnion*. This alternative characterization of the core allows us to compute it avoiding the explicit computation of all repairs. As we will see below, we can also use this representation of the core for computing consistent answers. This holds for spatial range queries $\mathcal{Q}$ of the form (11) and sets Φ of SICs of the form (12) with topological predicate *Overlaps*.

We now give an example of this approach. In order to show that our methodologies could be implemented on top of current spatial database management systems, we use spatial SQL. In it we basically specify the core as a view, on top of which the query expecting for consistent answers is called.

EXAMPLE 9. (example 7 cont.) The example considers only the relation *Landparcel* and the SIC (2) in Example 2. We want to consistently answer the query of Example 8, i.e. $\exists name\ owner(Landparcel(idl, name, owner, geometry) \wedge Overlaps(geometry, ([x_1, y_1], [x_2, y_1], [x_2, y_2], [x_1, y_2], [x_1, y_1]))$.

To answer this query, we apply Proposition 1, generating a view that represents the core of the repairs. That is, we eliminate from each geometry the union of conflicting regions with respect to each land parcel. In this case, the conflicting geometries for g_2 are g_3 and g_4; for geometry g_3 is g_2; and for geometry g_4 is g_2. This is the definition of the core in SQL:

```
CREATE VIEW   Core
  AS (SELECT  l1.idl AS idl,
              l1.name AS name, l1.owner AS owner,
              difference(l1.geometry,
              geomunion(l2.geometry)) AS geometry
      FROM    Landparcel AS l1, Landparcel AS l2
      WHERE   l1.idl <> l2.idl AND
              Overlaps(l1.geometry, l2.geometry)
      GROUP BY l1.idl, l1.name, l1.owner,
              l1.geometry)
    UNION
    (SELECT   l1.idl AS idl, l1.name AS name,
              l1.owner AS owner, l1.geometry AS geometry
      FROM    Landparcel AS l1
      WHERE   NOT EXISTS(SELECT l2.idl, l2.geometry
              FROM Landparcel AS l2
              WHERE l1.idl <> l2.idl AND
              Overlaps(l1.geometry, l2.geometry)))
```

Now we can pose the following query to compute the consistent answer to the original query:

```
SELECT   idl, name, owner, geometry FROM Core
WHERE    Overlaps(geometry, ([x1, y1], [x2, y1], [x2, y2],
                 [x1, y2], [x1, y1]))
```

The answer is shown in Figure 9. This query is a classic selection from the *Core* view. □

This core-based method allows us to compute consistent answers in polynomial time (quadratic) for cases where there can be exponentially many repairs. In Example 5, where we have 2^{n-1} minimal repairs, we can apply the query $\mathcal{Q}$ over the core, and we only have to compute the difference of a geometry with respect to the union of all other geometries in conflict. This corresponds to a polynomial time algorithm of order $O(n^2)$, with n the number of tuples.

Applying similar ideas to those used to compute consistent answers to range queries under SICs of the form (12) with predicate *Overlaps*, we can derive consistent answers under SICs including predicates T^* in $\{Intersects, Disjoint, Equal\}$. To do so, we also have to specify the *core*, without actually obtaining the repairs. For space limitation, this is left for an extended version of this work.

6. CONCLUSIONS

We have presented a repair semantics for spatial databases with respect to SICs that makes precise the idea of inconsistency tolerance: Even when a database is inconsistent it should return consistent answers to queries. Repairs are based on updates that shrink geometries of objects, even at the point of deleting geometries for some exceptional cases, as for predicate *Disjoint*. Geometries are virtually updated applying admissible geometric operators, which are available in most spatial DBMS.

Repairs are used as an auxiliary concept for characterizing consistent information in a database and defining consistent answers to spatial range queries. By restricting ourselves to the application of the admissible transformations, we obtain a finite number of possible repairs. However, there may still be exponentially many of them.

To avoid computing and querying all repairs, we have identified cases where the consistent answers to a query can be obtained by posing a query to a single view of the original instance. The standard answers so obtained are the consistent answers to the original query. This view can be specified in logical terms, but we may need a language more

expressive than the one of conjunctive queries. This situation is reminiscent of the use of logic programs to specify repairs of relational databases, on top of which the queries expecting for consistent answers are posed [2]: The original query is rewritten into a (more expressive) logic program.

In an ideal situation, the original conjunctive query would be rewritten into a new simple query, hopefully a conjunctive one too. We know that, in the relational case, this is only sometimes possible, for complexity reasons [3]. A similar situation is expected in the spatial domain. Identifying tight classes of queries and SICs for which low complexity rewriting can be obtained requires further investigation.

This paper leaves many problems for ongoing and future work, most prominently, computability and complexity issues. For example, interesting decision problems are deciding (a) the existence of non-trivial repairs (i.e. not obtained by cancelation of geometries), (b) whether an instance is a repair of another, and (c) whether a spatio-relational tuple is a consistent answer to a query. As in the relational case, we expect to find hard cases for all these problems. For them, it would be interesting to obtain lower complexity approximation algorithms.

We have considered only regions to represent spatial objects. As a natural extension would could define a repair semantics for other spatial abstractions, such as polylines, points, networks, etc. We would also like to explore not only denial SICs, but also other classes of semantic ICs. This includes also the possibility of considering combinations of spatial with relational constraints, e.g. functional dependencies, referential ICs.

Since repairs are query independent, our repair semantics could also provide the logical and operational foundations for spatial data cleaning and spatial data quality assessment.

Acknowledgments

This project is partially funded by FONDECYT, Chile, grant number 1080138. Part of this research was done when L. Bertossi was invited to the University of Concepcion. He has also been partially funded by NSERC (DG#315682).

7. REFERENCES

[1] M. Arenas, L. Bertossi, and J. Chomicki. Consistent query answers in inconsistent databases. In *Proc. PODS'99*, pages 68–79. ACM Press, 1999.

[2] P. Barcelo, L. Bertossi, and L. Bravo. Characterizing and computing semantically correct answers from databases with annotated logic and answer sets. In *Semantics of Databases*, LNCS 2582, pages 1–27. Springer-Verlag, 2003.

[3] L. Bertossi. Consistent query answering in databases. *ACM Sigmod Record*, 35(2):68–76, 2006.

[4] L. Bertossi, and L. Bravo. Consistent query answers in virtual data integration systems. In *Inconsistency Tolerance*, LNCS 3300, pages 42–83. Springer-Verlag, 2004.

[5] L. Bertossi and J. Chomicki. Query answering in inconsistent databases. In *Logics for emerging applications of databases*, pages 43–83. Springer, 2003.

[6] K. Borges, A. Laender, and C. Davis. Spatial integrity constraints in object oriented geographic data modeling. In *Proc. ACM GIS'99*, pages 1–6. ACM Press, 1999.

[7] M. Caniupan. Optimizing and implementing repair programs for consistent query answering in databases. PhD thesis, Carleton University, Department of Computer Science, 2007.

[8] J. Chomicki. Consistent query answering: Five easy pieces. In *Proc. ICDT'07*, LNCS 4353, pages 1–17. Springer-Verlag, 2007.

[9] S. Cockcroft. A taxonomy of spatial integrity constraints. *GeoInfomatica*, 1(4):327–343, 1997.

[10] M. Duckham, J. Lingham, K. T. Mason, and M. F. Worboys. Qualitative reasoning about consistency in geographic information. *Inf. Sci.*, 176(6):601–627, 2006.

[11] M. Egenhofer, E. Clementine, and P. D. Felice. Evaluating inconsistency among multiple representations. In *Spatial Data Handling*, pages 901–920, 1995.

[12] M. Egenhofer and J. Sharma. Assessing the consistency of complete and incomplete topological information. *Geographical Systems*, 1:47–68, 1993.

[13] M. Egenhofer and R. D. Franzosa. Point set topological relations. *International Journal of Geographical Information Systems*, 5:161–174, 1991.

[14] E. Franconi, A. L. Palma, N. Leone, S. Perri, and F. Scarcello. Census data repair: a challenging application of disjunctive logic programming. In *Proc. LPAR'01*, LNCS 2250, pages 561–578. Springer, 2001.

[15] A. Fuxman, E. Fazli, and R. J. Miller. Conquer: Efficient management of inconsistent databases. In *Proc. SIGMOD'05*, pages 155–166. ACM Press, 2005.

[16] T. Hadzilacos and N. Tryfona. A model for expressing topological integrity constraints in geographic databases. In *Proc. COSIT'92*, LNCS 639, pages 252–268. Springer-Verlag, 1992.

[17] B. Kuijpers, J. Paredaens, and J. V. den Bussche. On topological elementary equivalence of spatial databases. In *Proc. ICDT'97*, LNCS 1186, pages 432–446. Springer-Verlag, 1997.

[18] S. Mäs. Reasoning on spatial semantic integrity constraints. In *Spatial Information Theory*, LNCS 4736, pages 285–302. Springer-Verlag, 2007.

[19] OpenGis. Opengis simple features specification for SQL. Technical report, Open GIS Consortium, 1999.

[20] J. Paredaens and B. Kuijpers. Data models and query languages for spatial databases. *Data Knowl. Eng.*, 25(1-2):29–53, 1998.

[21] D. A. Randell, Z. Cui, and A. G. Cohn. A spatial logic based on regions and connection. In *Proc. KR'92*, pages 165–176. Morgan Kaufmann, 1992.

[22] A. Rodríguez. Inconsistency issues in spatial databases. In *Inconsistency Tolerance*, LNCS 3300, pages 237–269. Springer-Verlag, 2005.

[23] S. Servige, T. Puricelli, and R. Laurini. A methodology for spatial consistency improvement of geographic databases. *GeoInformatica*, 4:7–24, 2000.

[24] N. Tryfona and M. Egenhofer. Consistency among parts and aggregates: A computational model. *Transactions on GIS*, 1(1):189–206, 1997.

[25] J. Wijsen. Database repairing using updates. *ACM Trans. Database Syst.*, 30(3):722–768, 2005.