Consistent Query Answering under Spatial Semantic Constraints

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

Leopoldo Bertossi[*]
Carleton University, Canada
bertossi@scs.carleton.ca

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

Abstract

Consistent query answering (CQA) is an inconsistency tolerant approach to obtaining semantically correct answers from a database that may be inconsistent with respect to a set of integrity constraints. In this work, we formalize the notion of *consistent query answer* for spatial databases with respect to a special but relevant class of spatial semantic integrity constraints (SICs). In order to do this, we first characterize conflicting spatial data, and next, define admissible instances that restore consistency while staying close to the original instance. In this way we obtain a *repair semantics*, which is used as an instrumental concept to define consistent answers as a set-theoretic and geometric aggregation of answers from all admissible repairs. After establishing the intractability of consistent query answering, we identify and investigate a class of denial SICs (IDSICs) and spatial queries for which it is possible to efficiently compute consistent query answers via core computation.

1 Introduction

Consistency in database systems is defined as the satisfaction by a database instance of a set of integrity constraints (ICs) that restricts the admissible database states. Although consistency is a desirable and usually enforced property of databases, it is common to find inconsistent spatial databases due to data integration, unforced integrity constraints, legacy data, or time lag updates. In the presence 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 computing the "consistent or correct" answers to queries [2]. For many reasons, the first two alternatives may not be appropriate, specially in the case of virtual data integration [6], where centralized and global changes to the data sources are not allowed. In this work, we follow and develop the latter approach, called *consistent query answering*, for the spatial domain.

Consistent query answering (CQA) is about characterizing and computing query answers from a database instance that are semantically correct, in spite of the possible violation of the integrity constraints by the database. CQA has been extensively investigated in the relational case (cf. [8, 4, 13, 5] for surveys of the

[*]Faculty Fellow of the IBM Center for Advanced Studies.

area). Extracting consistent data from inconsistent databases could be qualified as an "inconsistency tolerant" approach to querying databases [9]. The basic idea of this approach is that even though a database may violate its integrity constraints, it can still be used to compute consistent answers to queries. In this way, it shifts the goal from the consistency of a spatial database to the consistency of query answering.

In this paper, we develop CQA for spatial databases and certain classes of denial SICs. The spatial domain offers several new challenges in comparison with the relational case, which is specially due to the use of complex attributes to represent geometries, their combination with thematic attributes, and the nature of spatial (topological) relations.

We introduce this idea using the following informal and simple example.

Example 1 Consider a database instance with a relation *LandP*, denoting land parcels, with thematic attributes idl and $name$, and a spatial attribute $geometry$, of data type $polygon$. A SIC stating that geometries of two different land parcels must be disjoint or just touch is expected to be satisfied, i.e., land parcels cannot internally intersect. However, the instance in Figure 1 does not satisfy this SIC and, therefore, it is inconsistent: the land parcels with identifiers idl_2 and idl_3 overlap. Notice that these geometries partially intersect, and what is not intersecting can be considered as consistent data.

LandP		
idl	*name*	*geometry*
idl_1	n_1	g_1
idl_2	n_1	g_2
idl_3	n_1	g_3

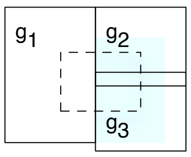

Figure 1: An inconsistent spatial database.

Suppose that a query requests the attribute idl of all land parcels whose geometries intersect with a query window, which represents the spatial region shown in Figure 1 as a rectangle with dashed boundaries. Although the database instance is inconsistent with respect to the SIC, we can still obtain useful and meaningful answers. In this case, only the intersection between g_2 and g_3 participates in the violation of the SIC, but what is left, after eliminating this intersection from g_2 or g_3, can be considered consistent and should be part of any "database repair" if we decide to restore consistency by means of minimal geometric changes. Thus, since the non-intersecting parts of geometries g_2 and g_3 intersect the query window, we would expect the following answers: $\langle dl_1\rangle$,$\langle idl_2\rangle$, and $\langle idl_3\rangle$.

For a query as above, that only requests the idl of land parcels, and the given database instance, the consistent answers coincide with the traditional answers (obtained ignoring the inconsistency). However, there is a difference between these two approaches when the query does not only request the identification, but also the geometries of land parcels that intersect the query window. In such case, the answers under the traditional approach, oblivious to inconsistencies, will return tuples with the original values in attributes idl and $geometry$, whereas consistent answers may return tuples with the original values in idl and modified values in the $geometry$ attribute of land parcels. These modifications are due to the repair and CQA semantics we use. For the new query, for example, the consistent answers would be three tuples: $\langle id_1, g_1\rangle$,$\langle id_2, g_2'\rangle$,$\langle id_3, g_3'\rangle$, where g_1 is the original geometry of land parcel id_1, and g_2' and g_3' are geometries derived from the intersection of geometries, grouped by thematic attributes, in answers from all admissible repairs. □

If we just concentrate on (in)consistency issues in databases (leaving aside consistent query answering for a moment), we can see that, in contrast to (in)consistency handling in relational databases, that has been largely investigated, not much research of this kind has been done for spatial databases. In particular, there is not much work around the formalization, satisfaction, checking or maintenance of SICs. However, some papers address the specification of some kinds of integrity constraints [10, 26], and check topological consistency at multiple representations and for data integration [18, 21, 36].

More recently, [16] proposes qualitative reasoning with description logic to describe consistency between geographic data sets. In [27], a set of abstract relations between entity classes is defined; and they could be used to discover redundancies and conflicts in sets of SICs. In [11], a formalization of a SICs is given and the study of the satisfiability problem of these constraints is analyzed. A proposal for fixing (changing) spatial database instances under different types of spatial inconsistencies is given in [34]. According to it, changes are applied over geometries in isolation; that is, they are not analyzed in combination with multiple SICs. In [32], some issues around query answering under violations of functional dependencies involving geometric attributes were raised. Despite de previous results, the problem of dealing with an inconsistent spatial database, while still obtaining meaningful answers, has not been systematically studied so far.

Consistent query answering from inconsistent databases was introduced and studied in the context of relational databases [2]. In that case, consistent answers to first-order queries are defined as 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. The set of tuples by which D and D' differ is minimal under set inclusion [2]. Other types of repair semantics have been studied in the relational case. For example, in [22, 37], repairs are obtained by allowing updates of attribute values in tuples. See [5] for more details and an extensive list of references.

In this work, we define a repair semantics for spatial databases with respect to a subset of SICs (also known as topo-semantic integrity constraints [34]), which impose semantic restrictions on topological relations and combinations thereof. In particular, we treat SICs that can be expressed by denials constraints. For example, they can specify that "two land parcels cannot internally intersect". These constraints are neither standardized nor integrated into current spatial database management systems (SDBMSs). They rather depend on the application, and must be defined and handled by the database developers. We concentrate on topological relations because they have spurred much research [19, 31, 15] and they are implemented in current Spatial SQL Languages (SSQLs). They are considered to capture the essence of a spatial configuration $-$topology matters, metric refines [20].

Other important spatial integrity constraints [14] are *domain (topological or geometric) constraints*, which refer to the geometry, topology, and spatial relations of the spatial data types. One of them could specify that "polygons must be closed". Many of these geometric constraints are now commonly integrated into SDBMSs [28].

The main contributions of this paper are:

- We formalize a spatio-relational schema and a subset of spatial semantic constraints, called *denial spatial integrity constraints* (DSICs).

- We formalize a repair semantics for spatial database instances under violations of DSICs. This is done through virtual changes of geometries that participate in violations of the DSICs. We provide a general definition of *database repair* that is based on shrinking geometries.

Unlike the preliminary work presented in [33], where database repairs are defined, inductively, by sequences of admissible geometric transformations applied over geometries, we rely now on a more general idea of shrinking geometries that may produce empty geometries or smaller geometries with respect to geometric inclusion. This makes it possible to define geometric operators for shrinking geometries that are more suitable than others, depending on an application; and also use global minimality as the principal criterion for selecting alternative repairs.

- We analyze the complexity of repair checking for spatial databases with respect to DSICs.
- Based on this formalization, we define the notion of consistent answer to a *range* and *join* query as an answer obtained by a *set-theoretic and geometric* aggregation of the answers obtained from all the admissible repairs.
- We show that computing consistent answers in the general case is intractable.
- In spite of the complexity results just mentioned, we show that CQA for a subclass of DSICs (IDSICs), and *basic* range and join queries, can be done efficiently via a *core computation*. This amounts to querying directly the intersection of all repairs of an inconsistent database instance, but without actually computing the repairs. We identify cases for which the core can be specified as a SSQL view of the original, inconsistent database.
- We present an experimental evaluation with real and synthetic data sets that compares the cost of CQA with the cost of evaluating queries directly over the inconsistent database (i.e., ignoring inconsistencies).

This work builds on, and extends, the results obtained in [33]. More precisely, we formalize consistent query answers on the basis of a repair semantics that shrink geometries. It assumes that we can shrink geometries in different ways, and considers global minimality of changes as the fundamental criterion to compare alternative repairs. Proceeding in this way gives us a semantics that does not provide only one possible transformation to solve a particular conflict, but makes possible alternative transformations that, when analyzed globally, define a repair that minimally differs from the original database instance. It also provides the theoretical foundations to define and investigate application-dependent geometric operators that shrink geometries. This work also extends [33] by considering, not only range queries, but also join queries. We also provide complexity results for repair checking of DSICs and CQA, and present experimental results, in particular, the evaluation of the proposed algorithms for CQA for a subset of DSICs and queries.

The remaining of the paper is organized as follows. In Section 2, we describe the spatial data model upon which we define the repair semantics and consistent query answers. A formal definition of a repair semantics for spatial inconsistent databases under DSICs is introduced in Section 3. In Section 4, we define consistent answers to conjunctive queries. We also analyze the computational properties of CQA. This leads us, in Section 5, to propose polynomial time algorithms (in data complexity) for consistent query answering with respect to a relevant class of DSICs and queries. An experimental evaluation of the cost of CQA is provided in Section 6. Final conclusions and future research directions are given in Section 7.

2 Preliminaries

Current models of spatial databases are typically seen as extensions of the relational data model (known as extended-relational or object-relational models) with the definition of abstract data types to specify spatial attributes. We now introduce a general spatio-relational database model that includes spatio-relational predicates (they could also be purely relational) and DSICs. It uses some of the definitions introduced in [30]. The model is independent of the geometric data model (e.g. Spaghetti [35], topological [24, 35], raster [25], or polynomial model [29]) 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 (relations) whose attributes belong to $\mathcal{A}$ or are spatial attributes. Spatial attributes take admissible values[1] in $\mathcal{P}(\mathbb{R}^m)$, the power set of $\mathbb{R}^m$, for a fixed m that depends on the dimension of the spatial attribute. (d) $\mathcal{T}$ is a fixed set of binary spatial predicates, with a built-in interpretation. (e) $\mathcal{O}$ is a fixed set of geometric operators that take spatial arguments, also with built-in interpretations. (f) $\mathcal{B}$ is a fixed set of built-in relational predicates, like comparison predicates, e.g. $<, >, =, \neq$, which apply to thematic attribute values.

We assume that each relation R has only one spatial attribute. Furthermore, each relation is subject to a *key constraint* of the general form (1), with a key formed by thematic attributes only.

$$\forall \bar{x}_1 \bar{x}_2 \bar{x}_3 s_1 s_2 \; (R(\bar{x}_1, \bar{x}_2; s_1) \wedge R(\bar{x}_1, \bar{x}_3; s_2) \rightarrow \bar{x}_2 = \bar{x}_3 \wedge \mathsf{Equals}(s_1, s_2). \quad (1)$$

Here, $\bar{x}_1$ is a non-empty sequence, $\bar{x}_2$ and $\bar{x}_3$ are possible empty sequences, of distinct variables representing values for thematic attributes of R, and s_i are variables for values of spatial attributes[2]. Furthermore, $\mathsf{Equals}(s_1, s_2)$ is a built-in spatial predicate (see Table 1).

A database instance D of a spatio-relational schema Σ is composed of relation instances, which are finite collections of tuples of the form $R(c_1, ..., c_n; g)$, where $R \in \mathcal{R}$, $\langle c_1, ..., c_n \rangle \in \mathcal{U}^n$ contains the thematic attribute values, and $g \in Ad \subseteq \mathcal{P}(\mathbb{R}^m)$, with Ad is the class of admissible geometries (cf. below). The extension in a particular instance of the relation R is a subset of $\mathcal{U}^n \times Ad$. To fix ideas, we concentrate in this work on the case where $m = 2$, however, we could generalize the work with geometries in a 3D space.

Among the different abstraction mechanisms for modelling single spatial objects, we concentrate on *regions* for modelling real objects that have an extent. They are useful in a broad class of applications in Geographic Information Systems (GISs). We will be interested in a finite representation of geometries, which is compatible with the specification of spatial data types and spatial relations as found in current SDBMSs [28]. Actually, in current implementations of SDBMSs, regions are defined as finite sets of polygons that, in their turn, are defined through a finite sequence of boundary points. We can call them *polygonal regions*. In consequence, an *admissible geometry* of the Euclidean plane will be either the empty geometry, $g_\oslash$, which corresponds to the empty subset of the plane, or a (closed and bounded) polygonal region with a positive area. An immediate consequence of this is that admissible geometries become finitely representable. By definition, these admissible geometries of $\mathbb{R}^2$ form the class Ad mentioned above. Notice that it holds $g_\oslash \cap g = g \cap g_\oslash = g_\oslash$ and $g_\oslash \cup g = g \cup g_\oslash = g$, for every region g. We introduce a built-in atom $\mathsf{NonEmpty}(s)$ that is true if region s is different from the empty region.

It is important to notice that, although the space is continuous and there exist infinitely many points between two given points, the representation of a geometry in a computer is discrete and finite. Spatial attributes are complex data types, and their manipulation may have an important effect on the computational complexity of certain decision problems and algorithms. In particular, the *size* of the relation instance is one of the parameters that contribute to the computational complexity. This size can be defined as a function of the number of tuples and the representation size of geometries in those tuples.

[1]We also refer to values of spatial attributes as geometries.

[2]When $\bar{x}_2$ and $\bar{x}_3$ are empty sequences, the key constraint takes the form $\forall \bar{x}_1 s_1 s_2 \; (R(\bar{x}_1; s_1) \wedge R(\bar{x}_1; s_2) \rightarrow \mathsf{Equals}(s_1, s_2))$.

Among different types of binary spatial relations (i.e., qualitative distance, orientation and topological relations), we concentrate on spatial predicates that represent *topological relations* between *regions*. Unlike orientation and qualitative distance relations, topological relations are currently implemented in SSQLs. Topological relations have a fixed semantics, and become the elements of $\mathcal{T}$. There are eight *base* binary relations over non-empty regions [19, 31].[3] For non-empty regions in $\mathbb{R}^2$, the definition of topological relations based on point-set theory [19] has its correspondence with topological relations defined by first-order logic, which uses a basic primitive relation of *connection* between regions [31]. In what follows, we use the definition based on point-set theory, because it is the one adopted by current SSQLs. According to [19], an atom $T(x, y)$ becomes true if four conditions are simultaneously true. Those conditions are expressed in terms of emptiness ($\emptyset$) and non-emptiness ($\neg\emptyset$) of the intersection of their boundaries (∂) and interiors ($\circ$). The definitions can be found in Table 1. For example, for non-empty regions x, y, $\mathsf{Touches}(x, y)$ is true if and only if all of $\delta(x) \cap \delta(y) \neq \emptyset$, $\circ(x) \cap \circ(y) = \emptyset$, $\delta(x) \cap \circ(y) = \emptyset$, and $\circ(x) \cap \delta(y) = \emptyset$ simultaneously hold.

Relation	**Description**	$\partial(x) \cap \partial(y)$	$\circ(x) \cap \circ(y)$	$\partial(x) \cap \circ(y)$	$\circ(x) \cap \partial(y)$
$\mathsf{Disjoint}(x, y)$	disconnected	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$
$\mathsf{Touches}(x, y)$	externally connected	$\neg\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$
$\mathsf{Equals}(x, y)$	equal	$\neg\emptyset$	$\neg\emptyset$	$\emptyset$	$\emptyset$
$\mathsf{Inside}(x, y)$	non-tangential proper part	$\emptyset$	$\neg\emptyset$	$\neg\emptyset$	$\emptyset$
$\mathsf{Covered\_by}(x, y)$	tangential proper part	$\neg\emptyset$	$\neg\emptyset$	$\neg\emptyset$	$\emptyset$
$\mathsf{Includes}(x, y)$	non-tangential proper part inverse	$\emptyset$	$\neg\emptyset$	$\emptyset$	$\neg\emptyset$
$\mathsf{Covers}(x, y)$	tangential proper part inverse	$\neg\emptyset$	$\neg\emptyset$	$\emptyset$	$\neg\emptyset$
$\mathsf{Overlaps}(x, y)$	partially overlapping	$\neg\emptyset$	$\neg\emptyset$	$\neg\emptyset$	$\neg\emptyset$

Table 1: Definition of *base* topological relations between non-empty regions based on point-set theory.

In addition to the base topological relations, we consider three *derived relations* that exist in current SSQLs, and can be logically defined in terms of the other basic predicates: Intersects, Within, and Contains. We also introduce relation IIntersects, which holds when the interiors of two geometries intersect, and relation notEquals (NE), which holds when two geometries are not equal. IIntersects can be logically defined as the disjunction of Overlaps, Within and Contains, whereas notEquals is the disjunction of all base relations but Equals (cf. Figure 2). For every topological relations T in $\mathcal{T}$, its converse (inverse) relation, denoted by T^c, is in $\mathcal{T}$. Some of them are symmetric, like Equals, Touches, and Overlaps. For the non-symmetric relations, the converse relation of Covered_by is Covers, of Inside is Includes, and of Within is Contains.

As mentioned before, the formal definitions of topological relations [19, 31] do not consider the empty geometry as an argument. Indeed, at the best of our knowledge, no clear semantics for topological relations with empty geometries exists. However, in our case, we extent the definitions in order to deal with this case. This will allow us to use a classical two-valued logic, where atoms are always true or false, but never undefined. Accordingly, in our extended definition, for every $T \in \mathcal{T}$, and $g_1, g_2 \in Ad$: If $g_1 = g_\oslash$ or $g_2 = g_\oslash$, then $T(g_1, g_2)$ is false.

Given a database instance, additional spatial information is usually computed from the explicit geometric data by means of a set $\mathcal{O}$ of *geometric operators* associated with Σ. These operators are of different kinds, but all of them use at least one geometry as a parameter and return geometries or real numbers. We will be using the following spatial operators when defining a distance function to compare geometries and when

[3]The names of relations chosen here are in agreement with the names used in current SSQL [28], but differ slightly from the names found in the research literature. The relations found in SSQLs are represented in Figure 2 with thick boundaries.

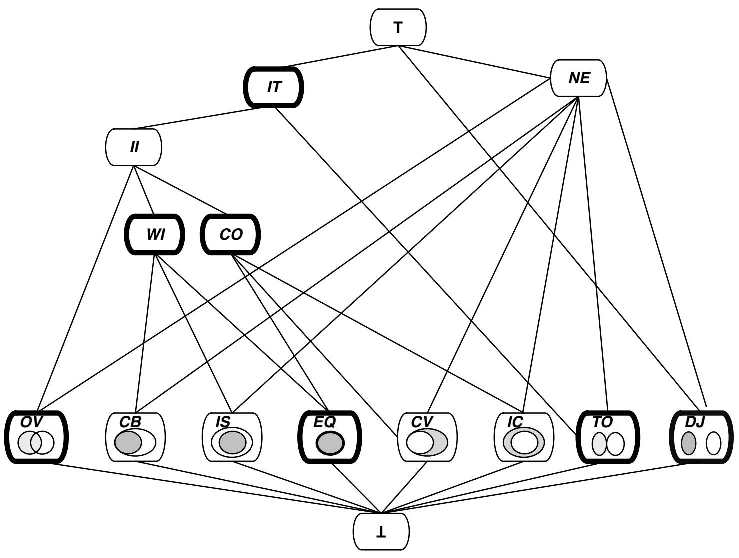

Figure 2: Subsumption lattice of topological relations between non-empty regions: OV (Overlaps), CB (Covered_by), IS (Inside), EQ (Equals), CV (Covers), IC (Includes), TO (Touches), DJ(Disjoint), IT(Intersects), II(IIntersects), WI(Within), CO(Contains), and NE(notEquals).

defining the core-based computation of CQA:[4]
(i) Intersection ($\cap$) returns the topological closure of the set intersection of two admissible geometries.
(ii) Difference ($\setminus$) returns the topological closure of the set difference between two admissible geometries.
(iii) GeomUnion ($\bigcup$) returns the topological closure of the union of a finite set of admissible geometries.
(iv) Area returns the area of an admissible region.

A schema Σ determines a first-order (FO) language $\mathcal{L}(\Sigma)$ of predicate logic. It can be used to syntactically characterize and express DSICs. For simplicity, we concentrate on *denial spatial integrity constraints* (DSICs),[5] which are sentences of the form:

$$\forall \bar{s}\bar{x}\, \neg(\bigwedge_{i=1}^{m} R_i(\bar{x}_i; s_i) \wedge \bigwedge_{i} \mathsf{NonEmpty}(s_i)\ \wedge \varphi \wedge \bigwedge_{j=1}^{n} T_j(v_j, w_j)), \tag{2}$$

where $\bar{s} = s_1 \cdots s_m,\ \bar{x} = \bar{x}_1 \cdots \bar{x}_m$ are finite sequences of geometric and thematic variables, respectively, and $0 < m, n \in \mathbb{N}$. Thus, each $\bar{x}_i$ is a finite tuple of thematic variables and will be treated as a set of attributes, such that $\bar{x}_i \subseteq \bar{x}_j$ means that the variables in $\bar{x}_i$ are also variables in $\bar{x}_j$. Also, $\forall \bar{x}$ stands for $\forall x_1 \cdots \forall x_m$; and $\forall \bar{s}$ stands for $\forall s_1 \cdots \forall s_m$, with the universal quantifiers ranging over all admissible geometries (i.e. regions). Here, $v_j, w_j \in \bar{s}$, $R_1, \ldots, R_m \in \mathcal{R}$, φ is an optional formula that is a conjunction of built-in atoms over thematic attributes, and $T_j, \ldots, T_n$ are predicates in $\mathcal{T}$.

A constraint of the form (2) prohibits certain combinations of database atoms. Since topological relations for empty geometries are always false, the explicit condition for non-empty geometries in the constraints

[4]Cf. [28] for the complete set of spatial predicates defined within the Open GIS Consortium. Although GeomUnion is part of SSQLs for several spatial databases (Postgres/PostGIS, Oracle), it is not explicitly defined in the OGC specification [28].

[5]Denial constraints are easier to handle in the relational case as consistency with respect to them is achieved by tuple deletions only [8].

LandP		
idl	*name*	*geometry*
idl_1	n_1	g_1
idl_2	n_2	g_2
idl_3	n_3	g_3

Building	
idb	*geometry*
idb_1	g_4
idb_2	g_5

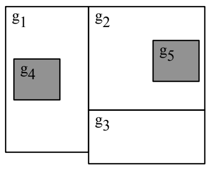

Figure 3: A spatial database instance.

could be eliminated. However, we do not want to make the satisfaction of the constraints relies on our particular definition of the topological relations for the empty region. In this way, our framework becomes more general and modular, in the sense that it would be possible to redefine the topological predicates for the empty region without affecting the semantics of the constraint.

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

The following sentences are DSICs:

$$\forall\, idl_1 \ldots s_2\, \neg(LandP(idl_1, n_1; s_1) \wedge LandP(idl_2, n_2; s_2) \ \wedge idl_1 \neq idl_2 \wedge \\ \mathsf{NonEmpty}(s_1) \wedge \mathsf{NonEmpty}(s_2) \wedge \mathsf{IIntersects}(s_1, s_2)). \quad (3)$$

$$\forall\, idb \ldots s_2\, \neg(Building(idb; s_1) \ \wedge \ LandP(idl, n; s_2) \ \wedge \mathsf{NonEmpty}(s_1) \wedge \\ \mathsf{NonEmpty}(s_2) \ \wedge \mathsf{Overlaps}(s_1, s_2)). \quad (4)$$

The DSIC (3) says that land parcels with different *ids* cannot internally intersect (i.e., they can only be disjoint or touch). The DSIC (4) establishes that building blocks cannot (partially) overlap land parcels. □

A database instance D for schema Σ can be seen as an interpretation structure for the language $\mathcal{L}(\Sigma)$. For a set Ψ of DSICs 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* with respect to Ψ, denoted $D \not\models \Psi$, when there is a $\psi \in \Psi$ that is *violated* by D, i.e., not satisfied by D. A *conflict* for a DSIC ψ of the form (2) is a pair of tuples $R_1(\bar{u}_1; g_1)$ and $R_2(\bar{u}_2; g_2)$ in D that violates ψ.

Given a schema Σ and a set of integrity constraints Ψ, a relevant problem is to determine if the set of constraints is satisfiable. The satisfiability problem for a database schema Σ and a set of spatial integrity constraints DSIC Ψ consists in determining if there exists a non-empty database instance D over Σ such that $D \models \Psi$. This problem restricted to schemas without empty regions, has been studied in [11] for a set of *topological dependency constraints* (TDs). This kind of dependencies can be transformed into DSICs of the form (2), excluding the $\mathsf{NonEmpty}(s)$ built-in atoms. The study in [11] shows that for a database schema with only one spatial attribute in its relations, satisfiability of a set of TDs can be checked in polynomial time with respect to the size of the schema, and the set of constraints. However, for a database schema with more than one spatial attribute in its relations, there exists a set of topological dependency constraints for which the satisfiability problem is NP-hard.

For a set Ψ of spatial integrity constraints of the form (2), it is easy to show that there always exists a non-empty instance D for an schema Σ, with empty regions, such that $D \models \Psi$. This is trivially proved since empty geometries always satisfy DSICs.

3 A Repair Semantics

In this section, we specify an update-based notion of repair semantics. For simplicity, in what follows we have considered denial constraints with at most two database relations and one topological relation. However, a denial constraint of the form (2) may have more spatio-relational predicates and topological relations.

A database D violates the constraint $\forall \bar{x}_1 \bar{x}_2 \forall s_1 s_2 \, \neg(R_1(\bar{x}_1; s_1) \wedge R_2(\bar{x}_2; s_2) \wedge \mathsf{NonEmpty}(s_1) \wedge \mathsf{NonEmpty}(s_2) \wedge \varphi \wedge T(s_1, s_2))$, when there are data values $\bar{a}_1, \bar{a}_2, g_1, g_2$, with g_1, g_2 non-empty geometries, for the variables in the constraint such that $(R_1(\bar{x}_1; s_1) \wedge R_2(\bar{x}_2; s_2) \wedge \varphi \wedge T(s_1, s_2))$ becomes true in the database under those values. This is denoted with $D \models (R_1(\bar{x}_1; s_1) \wedge R_2(\bar{x}_2; s_2) \wedge \mathsf{NonEmpty}(s_1) \wedge \mathsf{NonEmpty}(s_2) \wedge \varphi \wedge T(s_1, s_2))[\bar{a}_1, \bar{a}_2, g_1, g_2]$. When this is the case, it is possible to restore consistency of D by modifying g_1 or g_2, to make $T(g_1, g_2)$ false.

A preliminary discussion of update-based consistency restoration of spatial databases is in [33]. One of the key criteria to decide what update to apply is minimality of geometric changes. Another important element to consider is the semantics of spatial objects, which makes changes over the geometry of one type of object more appropriate than others. This work assumes that no previous knowledge about the quality and relevance of geometries exists and, therefore, it assumes that geometries are all equally important.

To define a repair semantics, we identify certain anti-monotonicity properties of topological relations, with respect to geometric inclusion of non-empty regions. Table 2 shows (anti-)monotonicity properties for topological relations over non-empty regions. For considerations of space, we have omitted the proof of these (anti-)monotonicity properties, which can be done by using composition of topological relations [17].

Relations T	**(Anti) Monotonicity Property**
Inside, Within	$T(s_1, s_2) \wedge \mathsf{Within}(s'_1, s_1) \Rightarrow T(s'_1, s_2)$
Intersects, IIntersects, Includes, Contains	$T(s_1, s_2) \wedge \mathsf{Within}(s_1, s'_1) \Rightarrow T(s'_1, s_2)$
Intersects, IIntersects, Inside, Within	$T(s_1, s_2) \wedge \mathsf{Within}(s_2, s'_2) \Rightarrow T(s_1, s'_2)$
Includes, Contains	$T(s_1, s_2) \wedge \mathsf{Within}(s'_2, s_2) \Rightarrow T(s_1, s'_2)$
Inside, Within	$\neg T(s_1, s_2) \wedge \mathsf{Within}(s_1, s'_1) \Rightarrow \neg T(s'_1, s_2)$
Disjoint, Intersects, IIntersects, Includes, Contains	$\neg T(s_1, s_2) \wedge \mathsf{Within}(s'_1, s_1) \Rightarrow \neg T(s'_1, s_2)$
Includes, Contains	$\neg T(s_1, s_2) \wedge \mathsf{Within}(s_2, s'_2) \Rightarrow \neg T(s_1, s'_2)$
Disjoint, Intersects, IIntersects, Inside, Within	$\neg T(s_1, s_2) \wedge \mathsf{Within}(s'_2, s_2) \Rightarrow \neg T(s_1, s'_2)$
Inside, Within	$\neg T(s_1, s_2) \wedge \mathsf{Within}(s_1, s'_1) \Rightarrow \neg T(s_1, s'_2)$

Table 2: Monotonicity of topological relations between non-empty regions.

The table shows that if atoms $\mathsf{Intersects}(g_1, g_2)$ and $\mathsf{IIntersects}(g_1, g_2)$ are true, and we enlarge geometries g_1 or g_2 to g'_1 or g'_2, respectively, $\mathsf{Intersects}(g'_1, g'_2)$ and $\mathsf{IIntersects}(g'_1, g'_2)$ will continue being true. In contrast, if we shrink geometries g_1 or g_2 to g'_1 or g'_2, respectively, $\mathsf{Intersects}(g'_1, g'_2)$ and $\mathsf{IIntersects}(g'_1, g'_2)$ may become false. Even more, when shrinking a geometry g to g', g' will not intersect other geometry g'', unless geometries g and g'' previously intersect. The latter is very important because it implies that by shrinking geometries, no new conflicts involving these topological relations will appear.

We propose to solve inconsistencies with respect to DSICs of the form (2) by shrinking geometries. This repair semantics will be used as an instrumental concept to formalize the notion of consistent query answer. In particular, this way to solve inconsistencies of DSICs does not require, necessarily, making or materializing changes on the original database. We disregard translating geometries because there is no (anti-)monotonicity properties that could reduce the interaction of conflicts. We also disregard the creation of new objects (object splitting), because we would have to deal with null or unknown thematic attributes. We will see that even with our relatively simple repair semantics, we obtain hard cases of complexity in relation to determining repairs and CQA.

Proposition 1 Let $T(g_1, g_2)$ be a topological relation in $\mathcal{T}$ between admissible geometries g_1 and g_2 that is true. Then, $T(g'_1, g'_2)$ can be false if one the following condition occurs: (i) g'_1 or g'_2 are empty geometries or (exclusive) (ii) $\mathsf{Inside}(g'_1, g_1)$, $\mathsf{Covered\_by}(g'_1, g_1)$, $\mathsf{Inside}(g'_2, g_2)$, or $\mathsf{Covered\_by}(g'_2, g_2)$ is true.

Proof: When geometries are empty, the proof is trivial, since no topological relation with empty geometries is true. Then, we proof that a true atom $T(g_1, g_2)$ can be falsified by shrinking g_1 or g_2 to g'_1 and g'_2, respectively, using the notion of composition of topological relations. The composition of two topological relations $T_1(g_1, g_2)$ and $T_2(g_2, g_3)$, denoted by $T_1(g_1, g_2) \otimes T_2(g_2, g_3)$, defined over a common geometry g_2, enables to derive the set of possible topological relations that may hold between objects g_1 and g_3. For example, the composition $\mathsf{Touches}(g_1, g_2) \otimes \mathsf{Inside}(g_2, g_3)$ results in the set $\{\mathsf{Overlaps}(g_1, g_3), \mathsf{Covered\_by}(g_1, g_3), \mathsf{Inside}(g_1, g_3)\}$ of possible topological relations between g_1 and g_3. The composition of base topological relations has been studied previously and can be found in [17].

The composition of topological relations imposes constraints on the possible relations between geometries when they are shrunk. Let g'_1 be the corresponding shrunk geometry with respect to g_1 and $T(g_1, g_2)$ be the topological relation that must be falsify. It holds that $\mathsf{Inside}(g'_1, g_1)$ or $\mathsf{Covered\_by}(g'_1, g_1)$ must be true. By definition of the composition of topological relations and topological consistency [21], relation $T'(g'_1, g_2)$ can be true if $T'(g'_1, g_2) \in \mathsf{Inside}(g'_1, g_1) \otimes T(g_1, g_2) \cup \mathsf{Covered\_by}(g'_1, g_1) \otimes T(g_1, g_2)$, with $\cup$ denoting set union. Consequently, $T(g'_1, g_2)$ can be false, if $\mathsf{Inside}(g'_1, g_1) \otimes T(g_1, g_2) \cup \mathsf{Covered\_by}(g'_1, g_1) \otimes T(g_1, g_2) \setminus \{T(g'_1, g_2)\} \neq \emptyset$, with $\setminus$ denoting set difference.

As an illustration, let us consider the case of atom $\mathsf{Equals}(g_1, g_2)$ that is true and must be false. By shrinking g_1, we have a geometry g'_1 such that $\mathsf{Covered\_by}(g'_1, g_1)$ or $\mathsf{Inside}(g'_1, g_1)$ is true. By composition of topological relations, $\mathsf{Covered\_by}(g'_1, g_1) \otimes \mathsf{Equals}(g_1, g_2) \cup \mathsf{Inside}(g'_1, g_1) \otimes \mathsf{Equals}(g_1, g_2) = \{\mathsf{Covered\_by}(g'_1, g_2), \mathsf{Inside}(g'_1, g_2)\}$. Consequently, $\mathsf{Equals}(g'_1, g_2)$ is false.

The same analysis can be done when shrinking g_2 and when shrinking both g_1 and g_2. We can prove exhaustively, by using the compositions defined in [17], that for all topological relations, except relation $\mathsf{Disjoint}$, we can always falsify a topological atom by shrinking geometries. For relation $\mathsf{Disjoint}$, we can always falsify an atom by making one of the geometries empty. $\Box$

Notice that due to the interaction of different DSICs, even if in isolation a topological relation can be falsified by shrinking a geometry, this geometry must need to become empty to satisfy all constraints. This related to the satisfiability problem of a set of integrity constraints discussed at the end of the last section.

In what follows, we formalize our repair semantics based on shrinking geometries.

Given a database D, possibly inconsistent, the repairs of D will be among the instances D' that are consistent, i.e., $D' \models \Psi$, and also correlated to D.

Definition 1 Let D, D' be database instances of schema Σ that satisfy the key constraints (1). D and D' are (mutually) *correlated* if and only if, for every ground tuple of the form $R(c_1, \ldots, c_n; g)$, if $R(c_1, \ldots, c_n; g) \in D$, then there is a tuple $R(c_1, \ldots, c_n; g')$, with $R(c_1, \ldots, c_n; g') \in D'$; and the inverse also holds. If D is fixed, then we also say that D' is D-correlated. $\Box$

Notice that due to the satisfaction of the key constraints, two mutually correlated instances have the same cardinality and the same values for the thematic attributes in tuples. This correlation implicitly defines a *correlation function* f, such that $f(D) = D'$ and $f^{-1}(D') = D$. Typically, the fixed instance D will be the initial, inconsistent instance, and the repairs will be D-correlated (cf. Definition 4). In a D-correlated instance D', we can compare tuples one by one with their counterparts in instance D. In particular, we can see how the spatial-attribute values differ.

Example 3 (example 2 cont.) Consider the relational schema $LandP(idl,\ name\ ;$*geometry)*. For the instance D given in Example 2, the following instance D' is D-correlated.

LandP		
idl	*name*	*geometry*
idl_1	n_1	g_7
idl_2	n_2	g_8
idl_3	n_3	g_9

Here, for the correlation function, it holds $f(LandP(idl_1, n_1; g_1)) = LandP(idl_1, n_1; g_7)$, etc. □

Any two D-correlated instances D' and D'' can be compared tuple by tuple. This comparison between tuples will be done by means of a distance function that refers to the areas of geometries in tuples, since only geometries are modified.

Definition 2 For regions $g_1, g_2,\ \delta(g_1, g_2) = \mathsf{Area}((g_1 \smallsetminus g_2) \cup (g_2 \smallsetminus g_1))$. □

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

Definition 3 Let D, D' be spatial database instances over the same schema Σ, where D' is D-correlated. 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. □

For the repair semantics considered in this paper, the distance function can be simplified to $\delta(g_1, g_2) = \mathsf{Area}(g_1 \smallsetminus g_2)$, because we are always shrining geometries such that g_2 is geometrically included in g_1 or is empty.
Now it is possible to define a "repair semantics" as follows.

Definition 4 Let D be a spatial database instance over schema Σ and Ψ a set of DSICs. (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-correlated. (iii) For every tuple $R(c_1, \ldots, c_n; g) \in D$, if $f(R(c_1, \ldots, c_n; g)) = R(c_1, \ldots, c_n; g')$, with $R(c_1, \ldots, c_n; g') \in D'$, then $\mathsf{Within}(g', g)$ or g' is $g_\oslash$. (b) A *minimal repair* D' of D is a repair of D such that, for every repair D'' of D, it holds $\Delta(D, D'') \geq \Delta(D, D')$. $Rep(D, \Psi)$ denotes the set of minimal repairs. □

Example 4 Consider the database instance in Figure 4 that is inconsistent with respect to DSICs (3) and (4). Figure 5 shows three possible minimal repairs of this instance.

The original database instance contains fours conflicts, i.e., four pairs of tuples (geometries) that violate the integrity constraints. Geometries g_1 and g_2, and geometries g_2 and g_3, violate DSIC (3) since they internally intersect. Geometries g_2 and g_6, and g_3 and g_6, violate DSIC (4) since g_6 partially overlaps g_2 and g_3. Figure 5 shows only three of the possible minimal repairs using a shrinking-based semantics. Repair (a) considers that the whole intersection between g_1 and g_2 is eliminated from g_1. Repair (b) considers that the whole intersection between g_1 and g_2 is eliminated from g_2. Finally, repair (c) considers that we take part of the overlapping area from g_1 and part from g_2. We could have infinitive ways to solve this conflict, since there are infinite possibilities in between taking the whole intersection from g_1 and the whole intersection from g_2. Therefore, for an inconsistent database instance there will be, in general, an infinite number, actually, a "continuum", of repairs. In principle, the same applies with minimal repairs.

LandP		
idl	*name*	*geometry*
idl_1	n_1	g_1
idl_2	n_2	g_2
idl_3	n_3	g_3

Building	
idb	*geometry*
idb_1	g_4
idb_2	g_5
idb_2	g_6

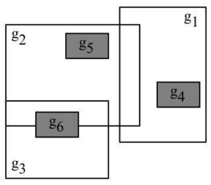

Figure 4: An inconsistent database instance.

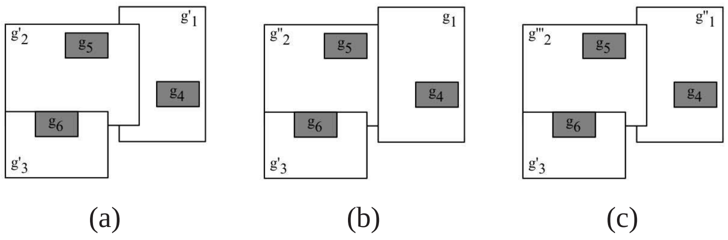

Figure 5: A subset of possible minimal repairs.

Notice that, based on minimality of geometric changes, there is only one way to repair conflict between g_2 and g_3 due to the interaction with the conflict between g_2 and g_6, and between g_3 and g_6. By taking half of the intersection between g_2 and g_3 from g_2 and the other half from g_3, we solve conflict between g_3 and g_6, and also between g_2 and g_6. These are the minimal geometric changes to repair these two conflicts. $\Box$

It is easy to show that for a database instance D and a set of DSICs Ψ, there is always a possible repair, since we could always make geometries that participate in the violation of an integrity constraint empty. Also, for a database instance D that is consistent with respect to Ψ, then D is its only minimal repair.

Notice that although there is always a repair of D with respect to Ψ, there is not always a minimal repair. To show a case when there is no minimal repairs, consider an atom $\mathsf{Touches}(g_1, g_2)$ that must be falsified. In such case, there are infinite ways to shrink geometry g_1 into g'_1 and make $\mathsf{Touches}(g'_1, g_2)$ false. Even more, since space is continuous, it is always possible to have g''_1 such that $\mathsf{Covered\_by}(g'_1, g''_1)$ is true and $\mathsf{Touches}(g''_1, g_2)$ is false. Thus, we do not have a lower bound of what must be eliminated from a geometry to obtain a minimal repair. Consequently, although we can restore consistency with respect to a topological predicate $\mathsf{Touches}$, it is not possible to have a minimal repair.

To show a case when there are minimal repairs, consider $\mathsf{IIntersects}(g_1, g_2)$ that must be falsified. Like in the previous case, there are infinite ways to shrink geometries g_1 or g_2 into g'_1 and g'_2, respectively, and make $\mathsf{IIntersects}(g'_1, g'_2)$ false. In all these cases, however, the intersection between g_1 and g_2 must be eliminated, which represents the lower bound of the area that has to be eliminated from g_1 or g_2. Consequently, we have minimal repairs.

We now introduce a first complexity result in terms of data complexity.

Proposition 2 For a set Ψ of DSICs, deciding if an instance D' is a minimal repair of an input database instance D is CO-NP-*complete* in data (complexity). That is, there is a schema Σ and a set Ψ of DSICs, such that the decision problem

$$\mathit{MinRep}(\Psi) := \{(D, D') \mid D \text{ is instance of } \Sigma \text{ and } D' \text{ is a minimaml repair of } D \text{ wrt } \Psi\}$$

is CO-NP-*complete*.

Proof:
(a) *Membership of* CO-NP: For input database instances D and D'. D' is *not* a minimal repair of D if there is a witnesses D'' for which it is possible to check in polynomial time in the size of D (and D') that: (i) D'' is D-*correlated*, for a certain function f; (ii) $D'' \models \Psi$; (iii) for every tuple $R(c_1, \ldots, c_n; g) \in D$, if $f(R(c_1, \ldots, c_n; g)) = R(c_1, \ldots, c_n; g')$, with $R(c_1, \ldots, c_n; g') \in D''$, then $\mathsf{Within}(g', g)$ or g' is $g_{\oslash}$; and (iv) $\Delta(D, D'') < \Delta(D, D')$.

(b) *Hardness*: For a fixed schema Σ and set Ψ of DSICs, we reduce the complement of 3-SAT to the problem of deciding if an instance D' is a minimal repair of an input instance D. More precisely, for an instance Φ of 3-SAT, we construct instances D and D', such that Φ is unsatisfiable if and only if D' is a minimal repair of D with respect to Ψ.

We will assume that each clause in a formula in CNF has exactly 3 literals. They are of the form $\Phi:\ c^1 \wedge \ldots \wedge c^n$, with $c^i = l^i_1 \vee l^i_2 \vee l^i_3$. A clause may have repeated literals, e.g. $c^1:\ x \vee \neg y \vee x$. We consider the following database predicates in Σ:

- $\mathit{Clauses}(I, K; G)$: It will contain tuples of the form $\langle I^i_j, l; g_{\oslash}\rangle$, where I^i_j identifies the position, j, of a literal in c^i, and l is the literal in that position. For example, for $c^1:\ x \vee \neg y \vee x$ we would have the tuples $\langle I^1_1, x; g_{\oslash}\rangle$, $\langle I^1_2, \neg y; g_{\oslash}\rangle$, and $\langle I^1_3, x; g_{\oslash}\rangle$. The key of $\mathit{Clauses}$ is attribute I.

- $\mathit{Lit}(K, K'; G)$: It will contain tuples of the form $\langle l, l'; g_{\oslash}\rangle$, with l and l' complementary literals that appear both in Φ. For example, $\langle x, \neg x; g_{\oslash}\rangle$ for the formula above. The key of Lit is $\langle K, K'\rangle$.

- $V(K; G), V_T(K; G), V_F(K; G)$: Each of them will contain literals that appear in Φ. That is, if literal l appears in Φ, then V_T will contain the tuple $\langle l; g_T\rangle$, with g_T a geometry to be defined below. Similarly for V_F and V, but with geometries g_F, g_I, respectively. Attribute K is always the key. The idea is that a literal l is considered to be true if geometries with key value l in V_T and V are equal. Likewise, it is considered to be false if the geometries with key value l in V_F and V are equal.

- $\mathit{Aux}(M; G)$ is an auxiliary predicate. It will be used to enforce particular repair transformations. The key of Aux is attribute M.

With these predicates we are in position to define the fixed set Ψ of DSICs. They are given in (5)-(8) below. They are independent from any propositional formula Φ or instance D associated with the former. However, to better understand the role of the DSICs, we will indicate how the predicates above are filled with tuples, obtaining an initial instance D.

Given an instance Φ for 3-SAT, of the form $\Phi:\ c^1 \wedge \ldots \wedge c^n$, with $c^i = l^i_1 \vee l^i_2 \vee l^i_3$, we construct an instance D for schema Σ as follows. $\mathit{Clauses}$ contains exactly the tuples $\langle I^i_j, l^i_j, g_{\oslash}\rangle$, with I^i_j being an identification symbol, i.e. $I^i_j \neq I^{i'}_{j'}$ for $j \neq j'$ or $i \neq i'$, and l^i_j is the literal in position j of clause i. In consequence, the number of tuples in the extension of $\mathit{Clauses}$ is tree times the number of clauses in Φ.

We insert a tuple $\mathit{Lit}(a, \neg a, g_{\oslash})$, with a an atom, whenever both a and $\neg a$ appear in Φ.

The extensions for V, V_T and V_F in D were described above. The extension of each of those predicates has as many tuples as different literals that appear in Φ. Let α be the number of different literals. It will be used below.

For predicate Aux, the only tuple in its extension is $\langle p; g_{I'} \rangle$, where p is a constant. e.g., an arbitrary propositional variable.

Now the geometries g_I, $g_{I'}$, g_T and g_F, to be used in the tuples above, can be arbitrary as long as they satisfy the following properties:

1. $g_{I'}$ is disjoint from g_I, g_T and g_F,
2. $g_I = g_T \cap g_F$,
3. $\mathsf{Area}(g_T) = \mathsf{Area}(g_F)$,
4. $\mathsf{Area}(g_T \setminus g_I) = \mathsf{Area}(g_F \setminus g_I)$, and
5. $\mathsf{Area}(g_I) > \mathsf{Area}(g_{I'}) > \alpha \times \mathsf{Area}(g_T \setminus g_I)$, where α is the number of different literals in Φ.

The idea is to create an instance D that is inconsistent with respect to the fixed set Ψ. The choice of these geometries will force specific repair transformations, subject to the requirement of minimally of shrinking geometries.

As an illustration of the construction so far, consider the formula $\Phi_0 = (x \vee \neg y \vee x) \wedge (z \vee \neg x \vee \neg x)$ that has x, y, z as propositional variables, and the four different literals $x, \neg x, \neg y, z$. Then $\alpha = 4$. Figure 6 shows an example of possible geometries for Φ_0. In this example, possible areas are: $\mathsf{Area}(g_T) = 5.5$, $\mathsf{Area}(g_F) = 5.5$, $\mathsf{Area}(g_I) = 5$, and $\mathsf{Area}(g_{I'}) = 4.5$. In this case, the relations in D_0 associated with formula Φ_0 are as shown in Figure 7.

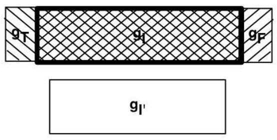

Figure 6: Geometries $g_I, g_{I'}, g_T, g_F$ for 3-SAT instance with four literals.

V_T

K	G
x	g_T
$\neg y$	g_T
z	g_T
$\neg x$	g_T

V_F

K	G
x	g_F
$\neg y$	g_F
z	g_F
$\neg x$	g_F

V

K	G
x	g_I
$\neg y$	g_I
z	g_I
$\neg x$	g_I

Lit

K	K'	G
x	$\neg x$	$g_\oslash$

Aux

K	G
p	$g_{I'}$

Clauses

C	I	K	G
c^1	I_1^1	x	$g_\oslash$
c^1	I_2^1	$\neg y$	$g_\oslash$
c^1	I_3^1	x	$g_\oslash$
c^2	I_1^2	z	$g_\oslash$
c^2	I_2^2	$\neg x$	$g_\oslash$
c^2	I_3^2	$\neg x$	$g_\oslash$

Figure 7: Database instance D_0 for formula Φ_0.

Now we define the set of DSICs. The first one ensures that each literal is true or false.

$$\begin{aligned} \forall l \forall x \forall s_1 \cdots \forall s_4 \neg (V(l; s_1) \wedge V_T(l; s_2) \wedge V_F(l; s_3) \wedge Aux(x; s_4) \wedge \\ \mathsf{NonEmpty}(s_1) \wedge \mathsf{NonEmpty}(s_2) \wedge \mathsf{NonEmpty}(s_3) \wedge \mathsf{NonEmpty}(s_4) \wedge \\ \mathsf{notEquals}(s_1, s_2) \wedge \mathsf{notEquals}(s_1, s_3) \wedge \mathsf{Disjoint}(s_1, s_4)). \end{aligned} \quad (5)$$

A second DSIC ensures that a literal cannot be true an false at the same time:

$$\begin{aligned}\forall l \forall x \forall s_1 \cdots \forall s_4 \neg (V(l; s_1) \wedge V_T(l; s_2) \wedge V_F(l; s_3) \wedge Aux(x; s_4) \wedge \\ \mathsf{NonEmpty}(s_1) \wedge \mathsf{NonEmpty}(s_2) \wedge \mathsf{NonEmpty}(s_3) \wedge \mathsf{NonEmpty}(s_4) \wedge \\ \mathsf{Equals}(s_1, s_2) \wedge \mathsf{Equals}(s_1, s_3) \wedge \mathsf{Disjoint}(s_1, s_4)).\end{aligned} \tag{6}$$

Using DSICs (5) and (6), the truth value of a literal l will be true if the geometries in V and V_T of literal l are equal. Likewise, the truth value of a literal l will be false if the geometries in V and V_F of literal l are equal.

A third constraint ensures that complementary literals cannot take the same truth value:

$$\begin{aligned}\forall l \forall l' \forall x \forall s_1 \cdots \forall s_7 \neg (Lit(l, l'; g_{\oslash}) \wedge V(l; s_1) \wedge V_T(l; s_2) \wedge V_F(l; s_3) \wedge \\ V(l'; s_4) \wedge V_T(l'; s_5) \wedge V_F(l'; s_6) \wedge Aux(x; s_7) \wedge \mathsf{NonEmpty}(s_1) \wedge \mathsf{NonEmpty}(s_2) \wedge \\ \mathsf{NonEmpty}(s_3) \wedge \mathsf{NonEmpty}(s_4) \wedge \mathsf{NonEmpty}(s_5) \wedge \mathsf{NonEmpty}(s_6) \wedge \mathsf{NonEmpty}(s_7) \wedge \\ \mathsf{Equals}(s_2, s_5) \wedge \mathsf{Equals}(s_3, s_6) \wedge \mathsf{Disjoint}(s_1, s_7) \wedge \mathsf{Disjoint}(s_4, s_7)).\end{aligned} \tag{7}$$

A last constraint states that, for each clause of the form $(l_1^i \vee l_2^i \vee l_3^i)$, at least one of the l_i^j must be true.

$$\begin{aligned}\forall c \forall l \forall l' \forall l'' \forall x \forall s_1 \cdots \forall s_7 \neg (Clauses(i, l; g_{\oslash}) \wedge Clauses(i', l'; g_{\oslash}) \wedge Clauses(i'', l''; g_{\oslash}) \wedge \\ V(l; s_1) \wedge V(l'; s_2) \wedge V(l''; s_3) \wedge V_F(l; s_4) \wedge V_F(l'; s_5) \wedge V_F(l''; s_6) \wedge Aux(x; s_7) \wedge \\ \wedge i \neq i' \wedge i \neq i'' \wedge \neq i' \neq i'' \wedge \mathsf{NonEmpty}(s_1) \wedge \mathsf{NonEmpty}(s_2) \wedge \mathsf{NonEmpty}(s_3) \wedge \\ \mathsf{NonEmpty}(s_4) \wedge \mathsf{NonEmpty}(s_5) \wedge \mathsf{NonEmpty}(s_6) \wedge \mathsf{NonEmpty}(s_7) \wedge \mathsf{Equals}(s_1, s_4) \wedge \\ \mathsf{Equals}(s_2, s_5) \wedge \mathsf{Equals}(s_3, s_6) \wedge \mathsf{Disjoint}(s_1, s_7) \wedge \mathsf{Disjoint}(s_2, s_7) \wedge \mathsf{Disjoint}(s_3, s_7)).\end{aligned} \tag{8}$$

The instance D constructed above for formula Φ is inconsistent with respect to Ψ: The DSIC (5) is false because, initially, neither g_T nor g_F are equal to g_I. Also, depending on the existence of complementary literals in Φ, DSIC (7) may also be false, since truth values of complementary literals are the same.

Now we construct an instance D', from D (and also Φ). It is the same as D, except for predicate Aux whose single tuple now becomes $\langle p; g_{\oslash} \rangle$. In this way, the topological atom Disjoint in each constraint is false. In consequence, $D' \models \Psi$, which makes D' a repair of D with respect to Ψ. Furthermore, $\Delta(D, D') = \mathsf{Area}(g_{I'})$. Its possible minimality as a repair will depend on Φ, as analyzed below.

To continue with our illustration, consider again the instance D_0 for the formula Φ_0. Instance D_0' is shown in Figure 8. In this case, $\Delta(D_0, D_0') = \mathsf{Area}(g_{I'}) = 4.5$.

It is clear that the reduction from Φ to (D, D') can be done in polynomial time in the size of Φ. Now we establish that it also answer preserving.

(i) *If* Φ *is* unsatisfiable, *then* D' *is the minimal repair of* D: When Φ is unsatisfiable, every truth assignment to literals in Φ will violate DSIC (8), even in the case that all others DSICs are satisfied. Therefore, for an unsatisfiable Φ, Ψ can be satisfied if and only if geometries of literals in V, V_T or V_F become empty, or if the geometry in the single tuple of Aux becomes empty. Due to the way geometries g_V, g_F, g_I, and $g_{I'}$ are defined, $\mathsf{Area}(g_T) = \mathsf{Area}(g_F) > \mathsf{Area}(g_I) > \mathsf{Area}(g_{I'}) > \alpha \mathsf{Area}(g_T \setminus g_I)$, with α the number of literals in clauses of Φ. Therefore, under consideration of minimality, there is no other instance D'' such that $D'' \models \Psi$ and $\Delta(D, D'') < \mathsf{Area}(g_I')$.

V_T	
K	G
x	g_T
$\neg y$	g_T
z	g_T
$\neg x$	g_T

V_F	
K	G
x	g_F
$\neg y$	g_F
z	g_F
$\neg x$	g_F

V	
K	G
x	g_I
$\neg y$	g_I
z	g_I
$\neg x$	g_I

Lit		
K	K'	G
x	$\neg x$	$g_\oslash$

Aux	
K	G
p	$g_\oslash$

Clauses			
C	I	K	G
c^1	I_1^1	x	$g_\oslash$
c^1	I_2^1	$\neg y$	$g_\oslash$
c^1	I_3^1	x	$g_\oslash$
c^2	I_1^2	z	$g_\oslash$
c^2	I_2^2	$\neg x$	$g_\oslash$
c^2	I_3^2	$\neg x$	$g_\oslash$

Figure 8: A repair D_0' of D_0 wrt Ψ.

(ii) *If* Φ *is* satisfiable, *then* D' *is not a minimal repair*: When Φ is *satisfiable*, there exists a truth assignment of the α literals in Φ that satisfies Ψ. Thus, it is possible to construct an instance D'' that shrinks the geometry in V_T or V_F of each literal l_k in Φ to become equal to the geometry of l_k in V. By making the right assignment, no conflicts with respect to DSICs (6), (7), and (8) occur, which is the minimal transformation over geometries. For each literal, the truth assignment has a cost equivalent to $\mathsf{Area}(g_T \setminus g_I) = \mathsf{Area}(g_F \setminus g_I)$, which sums up to $\alpha\mathsf{Area}(g_T \setminus g_I) < \mathsf{Area}(g_{I'})$. Thus, for a satisfiable Φ, there is an instance D'' such that $D'' \models \Psi$ and $\Delta(D, D'') = \alpha\mathsf{Area}(g_T \setminus g_I) < \Delta(D, D')$.

□

4 Consistent Query Answers

We can use the concept of minimal repairs as an auxiliary concept to define, and possibly compute, consistent answers to a relevant class of queries in $\mathcal{L}(\Sigma)$.

We study two conjunctive queries that are important in the spatial domain:

(a) *Range queries* are of the form

$$\mathcal{Q}(\bar{u}; s) : \exists\bar{z}(R(\bar{x}; s) \wedge T(s, w)), \tag{9}$$

where $\bar{u}$ are free thematic variables such that $\bar{u} = ((\bar{x}) \setminus \bar{z})$, s is a free spatial variable, w is a spatial constant that represents the spatial window of the query, and $\bar{z} \subseteq \bar{x}$.

(b) *Join queries* are of the form

$$\mathcal{Q}(\bar{u}; s_1, s_2) : \exists\bar{z}(R_1(\bar{x}_1; s_1) \wedge R_2(\bar{x}_2; s_2) \wedge T(s_1, s_2)), \tag{10}$$

with $T \in \mathcal{T}$, and $\bar{z} \subseteq \bar{x}_1 \cup \bar{x}_2$, $\bar{u}$ are free thematic variables such that $\bar{u} = ((\bar{x}_1 \cup \bar{x}_2) \setminus \bar{z})$, and $\{s_1, s_2\}$ are free spatial variables.

A *basic conjunctive query* (i.e., basic range or basic join query) is a query of the form (9) or (10) with $T = \mathsf{IIntersects}$. Basic conjunctive queries are relevant in the spatial domain, since they retrieve spatial features that are internally connected (i.e., they overlap, are equal, or are related by geometric inclusion). Notice that for range and join queries (and also for basic conjunctive queries) we project on all the spatial attributes to exploit the CQA semantics latter in this section.

LandP		
idl	*name*	*geometry*
idl_1	n_1	g_1
idl_2	n_2	g_2
idl_3	n_3	g_3

Building	
idb	*geometry*
idb_1	g_4
idb_2	g_5

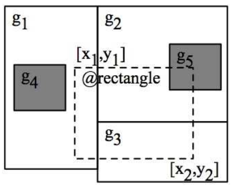

Figure 9: Example of a range query.

Remark 1 We will assume that the free variables correspond to a set of attributes of R with its key of the form (1). More precisely, for range queries, the attributes associated with $\bar{u}$ contain the key of R. For join queries, $\bar{u} \cap \bar{x_1}$ and $\bar{u} \cap \bar{x_2}$ contain the key for relations R_1, R_2, respectively. This is a common situation in spatial databases, where a geometry is retrieved together with its key value. $\Box$

A given query $\mathcal{Q}(\bar{x}; \bar{s})$, with free thematic variables $\bar{x}$ and free geometric variables $\bar{s}$, can be interpreted in an instance D for the schema. Accordingly, a sequence of thematic/spatial constants $\langle \bar{c}; \bar{g} \rangle$ is an answer to the query in D if and only if $D \models \mathcal{Q}(\bar{c}; \bar{g})$, that is, the query $\mathcal{Q}$ becomes true in D as a formula when its free variables $\bar{x}, \bar{s}$ are replaced by the constants in $\bar{c}, \bar{g}$, respectively. We denote with $\mathcal{Q}(D)$ the answer to $\mathcal{Q}$ in instance D.

Example 5 Figure 9 shows an instance for the schema $\mathcal{R} = \{LandP(idl,$*name*$;\ geometry),\ Building(idb;\ geometry)\}$. Here, idl and idb are keys for relations $LandP$ and $Building$, respectively. Dark rectangles represent buildings, and white rectangles represent land parcels. The queries $\mathcal{Q}_1$ and $\mathcal{Q}_2$ below are a range and a join query, respectively. $\mathcal{Q}_1$ specifies the spatial window of the query by the list of constant points $([x_1, y_1], [x_2, y_1], [x_2, y_2], [x_1, y_2], [x_1, y_1])$, which are represented in Figure 9 by the rectangle with dashed boundary.

$$\begin{aligned} \mathcal{Q}_1(idb; g) \quad &: \quad Building(idb; g) \ \wedge \\ &\qquad \mathsf{Intersects}(g, ([x_1, y_1], [x_2, y_1], [x_2, y_2], [x_1, y_2], [x_1, y_1])). \\ \mathcal{Q}_2(idl, idl'; g, g') \quad &: \quad LandP(idb, n; g) \ \wedge \ LandP(idb', n'; g') \ \wedge \mathsf{Touches}(g, g'). \end{aligned}$$

The answers to $\mathcal{Q}_1$ are $\langle idb_2; g_5 \rangle$. The answer to $\mathcal{Q}_2$ is: $\{\langle idl_1, idl_2; g_1, g_2 \rangle, \langle idl_2, idl_3; g_2, g_3 \rangle, \langle idl_1, idl_3; g_1, g_3 \rangle, \langle idl_2, idl_1; g_2, g_1 \rangle, \langle idl_3, idl_2; g_3, g_2 \rangle, \langle idl_3, idl_1; g_3, g_1 \rangle\}$. Notice that the answers contain the key values. $\Box$

Now we define the notion of consistent answer to a conjunctive query.

Definition 5 Consider an instance D, a set Ψ of DSICs over D.

- Let $\mathcal{Q}(\bar{x}; s)$ be a query of the form (9). A tuple $\langle c_1, \ldots, c_m; g_1 \rangle$ is a *consistent answer* to $\mathcal{Q}$ from D with respect to Ψ if the following two conditions hold: (a) for every repair $D' \in Rep(D, \Psi)$, there is g_1' such that $D' \models \mathcal{Q}(c_1, \ldots, c_m; g_1')$, and (b) $g_1 = \bigcap\{g_1' \mid D' \models \mathcal{Q}(c_1, \ldots, c_m; g_1')$ for every $D' \in Rep(D, \Psi)\}$
- Let $\mathcal{Q}(\bar{x}; s_1, s_2)$ be a query of the form (10). A tuple $\langle c_1, \ldots, c_m; g_1, g_2 \rangle$ is a *consistent answer* to $\mathcal{Q}$ from D with respect to Ψ if the following two conditions hold: (a) for every repair $D' \in Rep(D, \Psi)$, there exist g_1' and g_2' such that $D' \models \mathcal{Q}(c_1, \ldots, c_m; g_1', g_2')$. (b) $g_1 = \bigcap\{g_1' \mid D' \models \mathcal{Q}(c_1, \ldots, c_m; g_1', g_2')$ for every $D' \in Rep(D, \Psi)\}$, and $g_2 = \bigcap\{g_2' \mid D' \models \mathcal{Q}(c_1, \ldots, c_m; g_1', g_2')$ for every $D' \in Rep(D, \Psi)\}$.

$Con(\mathcal{Q}, D, \Psi)$ denotes the set of consistent answers to a conjunctive query $\mathcal{Q}$, over D with respect to Ψ. $\Box$

Intuitively, a consistent answer to a range query (of the form (9)) is a tuple containing thematic values and a geometric value g_1, such that g_1 is the intersection over all regions $g'_1 \in \mathcal{Q}(D')$ that belong to each different repair $D' \in Rep(D, \Psi)$ and correlate[6] to the same tuple in D. Similarly, a consistent answer to a join query (of the form (10)) is a tuple with thematic values and two geometries values, such that these geometries are also defined by the intersection of all geometries, grouped by thematic attributes, in answers from all admissible repairs.

Notice that, since $\mathcal{Q}$ is operator free, the regions g'_i that appear in the repairs are regions obtained by shrinking original geometries stored in the database. Also, since we project key values in queries, f^{-1} can be applied. However, due to the intersection of geometries, the geometries in a consistent answer may not belong to the original instance or to any of its repairs (c.f. Example 6).

In contrast to the definition of consistent answer to a relational query [2], where a consistent answer is an answer in every repair, here we have an aggregation of query answers via the geometric intersection and grouped-by thematic attribute values. This definition is similar to that of consistent answers to aggregate relational queries with group-by [3, 12], in the sense that consistent answers are obtained by processing the collection of answers from individual repairs.

Definition 5 allows us to obtain more significative answers than in the relational case, because when shrinking geometries, we cannot expect to have, for a fixed tuple of thematic attribute values, the same geometry in every repair. If we do not use the intersection of geometries, we might lose or not have consistent answers due to the absence of geometries that are shared by all repairs.

Example 6 (example 4 cont.) Consider the set Ψ of DSICs, the instance D of Example 4, and the range query in Figure 10, which is expressed in logic as

$$\begin{aligned}\mathcal{Q}(idl; geometry):\ &\exists name\ owner(LandP(idl, name; geometry) \wedge \\ &\mathsf{Intersects}(geometry, ([x_1,y_1],[x_2,y_1],[x_2,y_2],[x_1,y_2],[x_1,y_1])),\end{aligned}$$

with $[x_i, y_i]$ constant points. This query can be expressed in the SSQL as:[7]

$$\begin{array}{ll}\text{SELECT} & idl, geometry \\ \text{FROM} & LandP \\ \text{WHERE} & \mathsf{Intersects}(geometry, ([x_1,y_1],[x_2,y_1],[x_2,y_2],[x_1,y_2],[x_1,y_1])).\end{array}$$

Although in Example 4 we showed only three of the possible minimal repairs, we could still derive the consistent answer to this query from the fact that this is obtained from the geometric intersection of the geometries that are an answer in each repair. As we showed in Example 4, all minimal repairs solve the conflict between geometries g_2 and g_3 by eliminating their intersection. However, the conflict between g_1 and g_2 can be solved in infinite ways, having in one extreme the whole intersection eliminated from g_1, and in the other extreme, the whole intersection eliminated from g_2. For any D' of D, however, we have tuples $LandP(idl_1, n_1; g'_1)$, $LandP(idl_2, n_2; g'_2)$, and $LandP(idl_3, n_3; g'_3)$ such that g'_1, g'_2, g'_3 intersect the spatial window of the query. Consequently, the answers to this query are tuples $\langle idl_1, g''_1\rangle, \langle idl_2, g''_2\rangle, \langle idl_3, g''_3\rangle$ (see

[6] D's are D-correlated, cf. Definition 1.

[7] For simplification, we omit the real definition of the query window as it is done in current SSQL, which uses constructors for spatial data types.

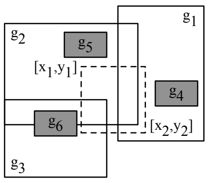

Figure 10: Querying an inconsistent database instance.

idl	*geometry*
idl_1	g''_1
idl_2	g''_2
idl_3	g''_3

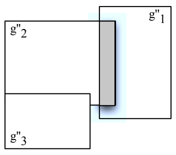

Figure 11: Consistent answers.

Figure 11), where g''_1, g''_2, g''_3 are defined as follows, and where $\bigcap$ represents geometric intersection over a *set of geometries*[8]:

$$g''_1 = \bigcap\{g'_1 \mid D' \models \mathcal{Q}(idl_1; g'_1) \text{ for every } D' \in Rep(D,\Psi)\}$$
$$g''_2 = \bigcap\{g'_2 \mid D' \models \mathcal{Q}(idl_2; g'_2) \text{ for every } D' \in Rep(D,\Psi)\}$$
$$g''_3 = \bigcap\{g'_3 \mid D' \models \mathcal{Q}(idl_3; g'_3) \text{ for every } D' \in Rep(D,\Psi)\}$$

□

Proposition 3 For a set Ψ of DSICs, CQA is in Π_2^P in data complexity.

Proof: Let Ψ be a set of DSICs and $\mathcal{Q}$ be a given query of the form (9) or (10). The complement of CQA is in NP^{coNP}: Given an instance D, nondeterministically choose an instance D' and check that $D' \not\models \mathcal{Q}$ and that D' is a minimal repair of D. The latter can be tested in $coNP$ by Proposition 2. But $NP^{coNP} = NP^{\Sigma_1^P} = \Sigma_2^P$. Therefore, CQA belongs to $co\Sigma_2^P = \Pi_2^P$. □

From a practical point of view, consistent query answers could include additional information about the degree in which geometries differ from their corresponding original geometries. For example, for the consistent answer $\langle idl_1; g''_1\rangle$ in Example 6, an additional information could be the relative difference between areas g_1 and g''_1, which could be calculated by $\delta(g_1, g''_1)/area(g_1)$.

5 Core-Based CQA

The definition of consistent query answer relies on the auxiliary notion of minimal repair. However, due to the intractability of deciding if a database instance is a minimal repair, and the potentially infinite number

[8]Do not confuse $\bigcap$ with the geometric operator $\cap$ in $\mathcal{O}$ over two geometries defined in Section 2.

LandP*			
idl_1	n_1	o_1	$g_1^\star$
idl_2	n_2	o_2	$g_2^\star$
idl_3	n_3	o_3	$g_3^\star$

Building*	
idb_1	$g_4^\star$
idb_2	$g_5^\star$
idb_3	$g_6^\star$

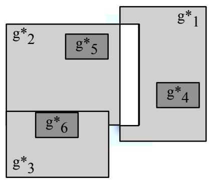

Figure 12: The core of an instance.

of repairs, determining consistent answers by computing, materializing, and finally querying all minimal repairs, must be avoided whenever there are more efficient mechanisms at hand. Along these lines, in this section, we present a methodology for computing consistent query answers for subclasses of conjunctive queries and DSICs. It works in polynomial time (in data complexity), and does not require the explicit computation of the database repairs.

We start by defining the *core*, which, intuitively, is the "geometric intersection" of the repairs. It is obtained by intersecting the geometries in the different repair instances that correlate to the same thematic tuple.

Definition 6 For an instance D and a set Ψ of DSICs, the *core* of D is the instance $D^\star$ given by $D^\star := \{R(\bar{a}; g^\star) \mid R \in \mathcal{R},$ 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 of D' with respect to D. □

Sometimes we will refer to $D^\star$ by $\bigcap^g Rep(D, \Psi)$. However, it cannot be understood as the set-theoretic intersection of the repairs of D. Rather it is a form of geometric intersection of geometries belonging to different repairs and grouped by common thematic attributes.

Example 7 (example 6 cont.) Figure 12 shows the *core* of the database instance in Figure 4, where gray rectangles represent geometries in the core that differ from their correlated geometries in the original inconsistent database instance. Here, $g_4^\star$, $g_5^\star$, and $g_6^\star$ are equivalent to the geometries g_4, g_5, and g_6 in the original database instance. Geometries $g_1^\star$, $g_2^\star$, and $g_3^\star$, in contrast, are obtained by considering the intersection of correlated geometries in minimal repairs. □

Notice the resemblance between the definitions of consistent answer and the core. Actually, it is easy to see that $D^\star = \bigcup_{R \in \mathcal{R}} Con(\mathcal{Q}_R, D, \Psi)$, where the query $\mathcal{Q}_R(\bar{x}; s) : R(\bar{x}; s)$ asks for the tuples in relation R.

The *core* is defined in terms of minimal repairs. However, as we will show, for a subset of DSICs, we can actually determine the *core* without computing those repairs. This is possible for DSICs of the form:

$$\forall \bar{x}_1 \bar{x}_2 s_1 s_2 \neg(R(\bar{x}_1; s_1) \wedge R(\bar{x}_2; s_2) \wedge \bar{x}'_1 \neq \bar{x}'_2 \wedge \mathsf{NonEmpty}(s_1) \wedge \mathsf{NonEmpty}(s_2) \wedge \mathsf{IIntersects}(s_1, s_2)), \quad (11)$$

where $\bar{x}'_1 \subseteq \bar{x}_1$, $\bar{x}'_2 \subseteq \bar{x}_2$, and both $\bar{x}'_1$ and $\bar{x}'_2$ are variables that capture the key of R. In these kind of DSICs, which will be called IDSICs, there are two occurrences of the same database predicate. Let us also denote by $\Psi(R)$ the IDSIC in Ψ over predicate R. Although this kind of DSICs uses only the topological relation IIntersects, it is of practical interest. By using this type of constraints, we are allowing regions to touch or be disjoint, which is a typical constraint for administrate boundaries or other geographic features (e.g. buildings, land parcels, and so on).

Remark 2 This class of IDSICs has the following properties hold, which will be useful when trying to compute the core:

County		
idl	*name*	*geometry*
idc_1	n_1	g_1
idc_2	n_2	g_2
idc_3	n_3	g_3

Reserve	
idr	*geometry*
idr_1	g_4
idr_2	g_5

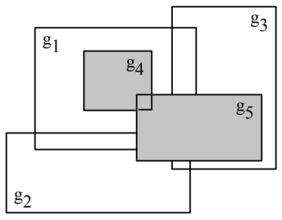

Figure 13: An inconsistent database with IDSICs of the form (11).

(i) To solve conflicts between tuples with respect to IDSICs, the intersection (internal intersection) of geometries must be eliminated so that geometries touch or become disjoint. Eliminating only the internal intersection represents a minimal way to restore consistency and, consequently, it leads to find minimal repairs. Notice however, that there may be infinite ways to eliminate this intersection.

(ii) Solving inconsistencies of IDSICs over different database predicates is independent. This is, solving conflicts between two tuples, with respect to a specific IDSIC φ_1 over a predicate R_1, is independent from solving a conflict with respect to any other another IDSIC φ_2 over a different database predicate.

(iii) Solving a conflict between two tuples with respect to a specific IDSIC does not introduce new conflicts. This is due to the anti-monotonicity property of predicates $\mathsf{IIntersects}$, which prevents a shrunk geometry from participating in a new conflict with an existing geometry in the database.

(iv) For any two geometries g_1 and g_2 in conflict with respect to a IDSIC, there always exists a *minimal version* of each geometry whose intersection with the original geometries in conflict has been eliminated (cf. Lemma 1). This can even means that the minimal version is the empty geometry. As a consequence, the core can be computed by eliminating from a geometry all its intersections with other geometries in conflict, disregarding the order in which these intersections are eliminated.

This property is not guaranteed for other kinds of DSICs. For instance, consider Example 6 with the instance in Figure 4 and its corresponding subset of minimal repairs in Figure 5. Although g_6 was originally in conflict with respect to g_2, there is no minimal repair where geometry g_6 has been shrunk. $\Box$

We illustrate some of these properties with the following example.

Example 8 Consider the schema $\mathcal{R} = \{County(idc,\ name;\ geometry),\ Reserve(idr;\ geometry)\}$, with idc the key of $County$ and idr the key of $Reserve$, and the following set Ψ of IDSICs:

$$\forall\, idc_1 \ldots s_2 \neg(County(idc_1, n_1; s_1) \wedge County(idc_2, n_2; s_2)\ \wedge idc_1 \neq idc_2 \wedge \\ \mathsf{NonEmpty}(s_1) \wedge \mathsf{NonEmpty}(s_2) \wedge \mathsf{IIntersects}(s_1, s_2)). \quad (12)$$

$$\forall\, idl_1 \ldots s_2 \neg(Reserve(idr_1; s_1) \wedge Reserve(idr_2; s_2)\ \wedge idr_1 \neq idr_2 \wedge \mathsf{NonEmpty}(s_1) \wedge \\ \mathsf{NonEmpty}(s_2) \wedge \mathsf{IIntersects}(s_1, s_2)). \quad (13)$$

County*		
idl	*name*	*geometry*
idc_1	n_1	$g_1^\star$
idc_2	n_2	$g_2^\star$
idc_3	n_3	$g_3^\star$

Reserve*	
idr	*geometry*
idr_1	$g_4^\star$
idr_2	$g_5^\star$

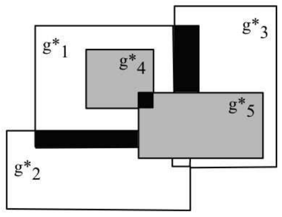

Figure 14: The core for the instance D in Figure 13.

An inconsistent instance D of this schema $\mathcal{R}$ is shown in Figure 13. In it, counties with geometries g_1, g_2 and g_3 are inconsistent with respect to the IDSIC (12), because they internally intersect. Geometries g_4 and g_5 violate the IDSIC (13), because they also internally intersect.

Conflicts with respect to IDSICs (12) and (13) can be solved in an independent way, since they do not share predicates (cf. Remark 2(ii)). To obtain a minimal repair, consider first IDSIC (12) and the conflict between g_1 and g_2, which can be solved by eliminating the intersection $g_1 \cap g_2$ only from g_1, only from g_2 or partially from g_1 and g_2. Any of these alternative transformations does not produce geometries that could be in conflict with other geometries unless they were originally in conflict (cf. Remark 2(iii)). For instance, if we eliminate $g_1 \cap g_2$ from from g_1, we obtain a new geometry g_1' that will not be now in conflict with g_2, but will still be in conflict with geometry g_3. This conflict is not new, since g_1 was originally in conflict with g_3. By eliminating from g_1 its intersection with g_2, however, we also eliminate part of the original intersection between g_1 and g_3.

Notice that, like Example 4 also illustrates, there are infinitely many, actually, a continuum of, alternative transformations that eliminate the intersection $g_1 \cap g_2$ partially from both geometries. Consequently, there is potentially an infinite number of minimal repairs. However, we still can ensure that there exists a minimal repair where the whole intersecting area is eliminated from one of the geometries in conflict, which is used latter to obtain the core of D without actually computing each minimal repair. The core $D^\star$ is shown in Figure 14, where black areas have been eliminated from their corresponding original geometries. $\Box$

We introduce the set $\mathcal{G}_{R,\Psi,D}(\bar{a}, g)$ that contains, for a given tuple $R(\bar{a}; g)$ in a database instance D, all the possible versions of geometry g in the minimal repairs of D.

Definition 7 Let D be a database instance, Ψ a set of IDSICs and $R(\bar{a}; g) \in D$ a fixed tuple. Then, $\mathcal{G}_{R,\Psi,D}(\bar{a}; g) = \{g' | R(\bar{a}; g') \in D', D' \in Rep(D, \Psi), f^{-1}(R(\bar{a}; g')) = R(\bar{a}; g)\}$. $\Box$

To simplify the notation, we also introduce a logical formula that captures a conflict around a tuple of relation $R \in D$ and a IDSIC:

$$\forall \bar{x}_1 \bar{x}_2 s_1 s_2 (\mathit{Confl}_{D,R}(\bar{x}_1, s_1, \bar{x}_2, s_2) \;\Leftrightarrow\; (R(\bar{x}_1; s_1) \wedge R(\bar{x}_2; s_2) \wedge \bar{x}_1' \neq \bar{x}_2' \wedge \mathsf{NonEmpty}(s_1) \wedge \mathsf{NonEmpty}(s_2) \wedge \mathsf{IIntersects}(s_1, s_2))), \quad (14)$$

where $\bar{x}_1' \subseteq \bar{x}_1$, $\bar{x}_2' \subseteq \bar{x}_2$, and both $\bar{x}_1'$ and $\bar{x}_2'$ are a non-empty sequence of variables that capture the key of R. By imposing that $\bar{x}_1' \neq \bar{x}_2'$, only different tuples can be checked for topological relation $\mathsf{IIntersects}$.

The following lemma establishes that when a geometry g is involved in conflicts of IDSICs, there exists a version of g in the repairs that is $g_\emptyset$ or is minimum with respect to geometric inclusion. This result is useful to show that the minimum version of g is the one that is in the core.

Lemma 1 Consider D a database instance, a set Ψ of IDSICs and a fixed tuple $R(\bar{a}; g) \in D$. The set of geometries $\mathcal{G}_{R,\Psi,D}(\bar{a}, g)$ has a minimum element g_{min} that is the empty geometry $g_\oslash$ or a geometry defined under geometric inclusion.

Proof: By definition of a minimal repair, for every $D' \in \mathit{Rep}(D, \Psi)$ and $R(\bar{a}; g) \in D$, there must be $R(\bar{a}; g') \in D'$ and $f^{-1}(R(\bar{a}; g')) = R(\bar{a}; g)$. Consequently, $|\mathcal{G}_{R,\Psi}(\bar{a}; g)| \neq 0$. If $|\mathcal{G}_{R,\Psi}(\bar{a}; g)| = 1$, this means that g is the same in all minimal repairs and, therefore, it is the minimum element g_{min}. If $|\mathcal{G}_{R,\Psi}(\bar{a}; g)| > 1$, then g must undergo repair transformations.

Let us define a geometry $t = \bigcup\{g' |\ R(\bar{b}; g') \in D, D \models \mathit{Confl}_{D,R}(\bar{a}, g, \bar{b}, g')\}$. This geometry t defines the union of geometries in D that are in conflict with g with respect to a IDSIC in Ψ.

For a IDSIC, the intersection between t and g must be eliminated by using alternative geometric transformations that shrink geometries, where there is always one alternative that eliminates the whole intersection from g. Thus, we must have a sequence of transformations that includes eliminating, per each conflict in which $R(\bar{a}; g)$ participates, the whole intersection from geometry g. This produces a geometry $g^\star = g \setminus t$, with $g^\star \in \mathcal{G}_{R,\Psi,D}(\bar{a}; g)$. Also, due to anti-monotonicity property of topological relation $\mathsf{IIntersects}$, solving conflicts for IDSICs will not introduce new conflicts and, consequently, the intersection between t and g is the only part of g that must to be eliminated. Consequently, $g^\star = g \setminus t$ is the minimum element in $\mathcal{G}_{R,\Psi,D}(\bar{a}; g)$, which is $g_\oslash$ o is a minimum geometry in $\mathcal{G}_{R,\Psi,D}(\bar{a}; g)$ under geometric inclusion. □

Example 9 (example 8 cont.) The inconsistent instance D in Figure 13 has tuple $\mathit{County}(idc_1, n_1; g_1)$ that is in conflict with tuples $\mathit{County}(idc_2, n_2; g_2)$ and $\mathit{County}(idc_3, n_3; g_3)$ with respect to IDSIC (12). A possible repair transformation eliminates from g_1 its intersection with g_2, generating a resulting geometry g_1'. Then, to solve the conflict between g_1' and g_3, a possible repair transformation eliminates from g_1' its intersection with g_3, resulting a geometry g_1''. Notice that $\mathsf{Within}(g_1'', g_1')$ holds, and that g_1'' is the result of eliminating from g_1 all conflicting intersections with geometries. Then, it holds that there exists g_1'' in $\mathcal{G}_{County,\Psi,D}(idc_1, n_1; g_1)$. Even more, g_1'' is the minimal element in $\mathcal{G}_{County,\Psi,D}(idc_1, n_1; g_1)$(see Figure 14). □

Corollary 1 Consider a database instance D, a set Ψ of IDSICs, and a fixed tuple $R(\bar{a}; g) \in D$. For the minimum geometry g_{min} in $\mathcal{G}_{R,\Psi}(\bar{a}, g)$, it holds $R(\bar{a}; g_{min}) \in D^\star$.

Proof: Direct from Lemma 1 and the definition of the core as a geometric intersection. □

Corollary 2 Consider the core $D^\star$ of a database instance D with respect to a set Ψ of IDSICs. Then, $D^\star \models \Psi$.

Proof: By Corollary 1, for any $R(\bar{a}_1; g_{1_{min}}) \in D^\star$, $g_{1_{min}}$ is the minimum geometry in $\mathcal{G}_{R,\Psi}(\bar{a}_1; g_1)$. If $g_{1_{min}}$ is the empty geometry, then $g_{1_{min}}$ does not participate in any possible conflict. Let us consider now the case when $g_{1_{min}}$ is not empty. By definition of a minimal repair, if $R(\bar{a}_1; g_1') \in D'$ with $D' \in \mathit{Rep}(D, \Psi)$, there exists no $R(\bar{a}_2; g_2')$ in D' that is in conflict with $R(\bar{a}_1; g_1')$. By anti-monotonicity of topological relation $\mathsf{IIntersects}$, if $\mathsf{Within}(g_{1_{min}}, g_1')$ and $\mathsf{Within}(g_{2_{min}}, g_2')$, with $g_{2_{min}}$ the minimum geometry in $\mathcal{G}_{R,\Psi}(\bar{a}_2; g_2)$, then $R(\bar{a}_1; g_{1_{min}}) \in D^\star$ is not in conflict with $R(\bar{a}_2; g_{2_{min}}) \in D^\star$. Then, $D^\star \models \Psi$. □

Although $D^\star \models \Psi$ holds, $D^\star$ is not necessarily equivalent to a minimal repair in $\mathit{Rep}(D, \Psi)$.

5.1 Properties of the Core

In this section we establish that for the set of IDSICs and basic conjunctive queries, it is possible to compute consistent answers on the basis of the core of an inconsistent instance, avoiding the computation of queries in every minimal repair. This is established in Theorems 1 and 2, respectively.

Theorem 1 For an instance D, a set Ψ of IDSICs, and a basic range query $\mathcal{Q}(\bar{u}; s)$, then $Con(\mathcal{Q}, D, \Psi) = \mathcal{Q}(D^\star)$.

Proof: The projection of range queries always includes the key of the relation. Thus, if $\langle \bar{a}; g'' \rangle \in Con(\mathcal{Q}, D, \Psi)$, then for every $D' \in Rep(D, \Psi)$, there exists $R(\bar{b}; g')$, such that $\bar{a} \subseteq \bar{b}$, $f^{-1}(R(\bar{b}; g')) = R(\bar{b}; g)$ and $R(\bar{b}; g) \in D$, where $\mathsf{IIntersects}(g', w)$ is true for the spatial constant w of the basic range query and $g'' = \bigcap\{g' \mid \text{ for every } g' \in \mathcal{G}_{R,\Psi,D}(\bar{b}, g)\}$.

On one direction, by Lemma 1, there exists a tuple $R(\bar{b}; g_{min}) \in D' \in Rep(D, \Psi)$, with g_{min} the minimum geometry in $\mathcal{G}_{R,\Psi,D}(\bar{b}; g)$. By definition of consistent answer, if $\langle \bar{a}; g'' \rangle \in Con(\mathcal{Q}, D, \Psi)$, with $\bar{a} \subseteq \bar{b}$, then $\mathsf{Equals}(g'', g_{min})$. By Corollary 1, $R(\bar{b}; g_{min}) \in D^\star$. Consequently, $\langle \bar{a}; g_{min} \rangle \in \mathcal{Q}(D^\star)$.

In the other direction, if $\langle \bar{a}; g_{min} \rangle \in \mathcal{Q}(D^\star)$ (with $D^\star = \bigcap^g Rep(D, \Psi)$), then there exists a tuple $R(\bar{b}; g_{min}) \in D^\star$, with $\bar{a} \subseteq \bar{b}$ and $\mathsf{IIntersects}(g_{min}, w)$ true. By the monotonicity of $\mathsf{IIntersects}$, if $\mathsf{IIntersects}(g_{min}, w)$ is true, then for all geometries $g' \in \mathcal{G}_{R,\Psi,D}(\bar{b}, g)$, $\mathsf{IIntersects}(g', w)$ is also true. Then, by definition of consistent answer $\langle \bar{a}; g_{min} \rangle \in Con(\mathcal{Q}, D, \Psi)$. $\square$

A similar result can be obtained for basic join queries, i.e., queries that consider two database predicates (not necessarily different). The following example illustrates how to compute consistent answers to basic join queries. This example will be used to illustrate the proof of Theorem 2.

Example 10 (example 8 cont.) Consider the following basic join query posed to the instance D in Example 8. It is asking for the identifiers and geometries of counties and reserves that internally intersect.

$$\mathcal{Q}(idc, idr; g_1, g_2) : \exists n(County(idc, n; g_1) \wedge Reserve(idr; g_2) \wedge \mathsf{IIntersects}(g_1, g_2)).$$

Without using the core, the answers are obtained by intersecting all answers that result from every possible minimal repair.

Let $\mathcal{G}_{R,\Psi,D}(\bar{u}; g)$ be the set of correlated geometries in each minimal repair with tuple $R(\bar{u}, g)$ in D, then we have five different sets (one for each geometry in the original database instance), each of them with a minimum geometry that is equivalent to the geometry in the core (see Figure 14).

For the database relations $County$ and $Reserve$, there are two sets containing the possible extensions of these relations in the repairs: $\{County(D') | D' \in Rep(D, \Psi)\}$ and $\{Reserve(D') | D' \in Rep(D, \Psi)\}$, which are combined to produce different minimal repairs. Thus, for any two tuples $County(idc_i, n_i; g_i) \in D$ and $Reserve(idr_j; g_j) \in D$, theres exists a minimal repair D' in $Rep(D, \Psi)$ such that $County(idc_i, n_i; g_i^\star) \in D'$ and $Reserve(idr_j; g_j^\star) \in D'$, with $g_1^\star$ and $g_j^\star$ the minimum geometries in sets $\mathcal{G}_{County,\Psi,D}\ (idc_i, n_i; g_i)$ and $\mathcal{G}_{Reserve,\Psi,D}(idr_j; g_j)$, respectively.

Let $g_1^\star$, $g_2^\star$, $g_3^\star$, $g_4^\star$, and $g_5^\star$ be the minimum geometries in sets $\mathcal{G}_{County,\Psi,D}(idc_1, n_1; g_1)$, $\mathcal{G}_{County,\Psi,D}(idc_2, n_2; g_2)$, $\mathcal{G}_{County,\Psi,D}(idc_3, n_3; g_3)$, $\mathcal{G}_{Reserve,\Psi,D}(idr_1; g_4)$, and $\mathcal{G}_{Reserve,\Psi,D}(idr_2; g_5)$, respectively. Due to the monotonicity property of topological relation $\mathsf{IIntersects}$, if $\mathsf{IIntersects}(g_4^\star, g_1^\star)$, $\mathsf{IIntersects}(g_5^\star, g_1^\star)$, $\mathsf{IIntersects}(g_5^\star, g_2^\star)$ and $\mathsf{IIntersects}(g_5^\star, g_3^\star)$ are true, then $\mathsf{IIntersects}(g_4^\star, g_i)$ is true for any g_i such that $\mathsf{Within}(g_2^\star, g_i)$

holds. Likewise, $\mathsf{IIntersects}(g_5^\star, g_j)$ is true for any g_j such that $\mathsf{Within}(g_1^\star, g_j)$, $\mathsf{Within}(g_2^\star, g_j)$ or $\mathsf{Within}(g_3^\star, g_j)$ hold.

Then, by Corollary 1, $County(idc_1, n_1; g_1^\star)$, $County(idc_2, n_2; g_2^\star)$, $County(idc_3, n_3; g_3^\star)$, $Reserve(idr_1; g_4^\star)$, and $Reserve(idr_2; g_5^\star)$ are tuples in the core of D and, by definition of consistent answer, $\langle idc_1, idr_1, g_1^\star, g_4^\star\rangle$, $\langle idc_1, idr_2, g_1^\star, g_5^\star\rangle$, $\langle idc_2, idr_2, g_2^\star, g_5^\star\rangle$, $\langle idc_3, idr_2, g_3^\star, g_5^\star\rangle$ are answers to the query.

□

Theorem 2 For an instance D, a set Ψ of IDSICs, and a basic join query $\mathcal{Q}(\bar{x}_1, \bar{x}_2; s_1, s_2)$, then $Con(\mathcal{Q}, D, \Psi) = \mathcal{Q}(D^\star)$.

Proof: The projection of join queries also includes keys. Thus, if $\langle \bar{a}_1, \bar{a}_2; g_1'', g_2''\rangle \in Con(\mathcal{Q}, D, \Psi)$, then there exist tuples $R_1(\bar{b}_1; g_1') \in D'$, $R_2(\bar{b}_2; g_2') \in D'$, for every $D' \in Rep(D, \Psi)$ with $\bar{a}_1 \subseteq \bar{b}_1$, $\bar{a}_2 \subseteq \bar{b}_2$, $f^{-1}(R_1(\bar{b}_1; g_1')) = R_1(\bar{b}_1; g_1)$, $f^{-1}(R_2(\bar{b}_2; g_2')) = R_2(\bar{b}_2; g_2)$, where $\mathsf{IIntersects}(g_1', g_2')$ is true, $g_1'' = \bigcap\{g_1' \mid \text{ for every } g_1' \in \mathcal{G}_{R,\Psi,D}(\bar{b}, g_1)\}$, and $g_2'' = \bigcap\{g_2' \mid \text{ for every } g_2' \in \mathcal{G}_{R,\Psi,D}(\bar{b}, g_3)\}$.

We now analyze two cases:

(i) If $R = R_1 = R_2$, $\langle \bar{a}_1, \bar{a}_2; g_1'', g_2''\rangle \in Con(\mathcal{Q}, D, \Psi)$ is an answer to the query if and only if $\Psi(R) \notin \Psi$. This is because by solving conflicts with respect to $\Psi(R)$, all possible internal intersections between geometries in tuples of R will be eliminated. Consequently, after repairing, no geometries in R will internally intersect, which is the topological relation that geometries must satisfy to be an answer to a basic join query.

(ii) If $R_1 \neq R_2$, due to the independence of repairing R_1 and R_2 with respect to Ψ, there exists a $D'' \in Rep(D, \Psi)$, such that $R_1(\bar{b}_1; g_{1_{min}}) \in D''$, $R_2(\bar{b}_2; g_{2_{min}}) \in D''$, with $g_{1_{min}}$ and $g_{2_{min}}$ the minimum geometries in $\mathcal{G}_{R_1,\Psi,D}(\bar{b}_1, g_1)$ and $\mathcal{G}_{R_2,\Psi,D}(\bar{b}_2, g_2)$, respectively.

On one direction, by definition of consistent answer, if $\langle \bar{a}_1, \bar{a}_2; g_1'', g_2''\rangle \in Con(\mathcal{Q}, D, \Psi)$, with $\bar{a}_1 \subseteq \bar{b}_1$ and $\bar{a}_2 \subseteq \bar{b}_2$, then $\mathsf{Equals}(g_1'', g_{1_{min}})$ and $\mathsf{Equals}(g_2'', g_{2_{min}})$. By Corollary 1, $R(\bar{b}_1; g_{1_{min}}) \in D^\star$ and $R(\bar{b}_2; g_{2_{min}}) \in D^\star$. Consequently, $\langle \bar{a}_1, \bar{a}_2; g_{1_{min}}, g_{2_{min}}\rangle \in \mathcal{Q}(D^\star)$.

In the other direction, if $\langle \bar{a}_1, \bar{a}_2, g_{1_{min}}, g_{2_{min}}\rangle \in \mathcal{Q}(D^\star)$ (with $D^\star = \bigcap^g Rep(D, \Psi)$), then there exist tuples $R(\bar{b}_1; g_{1_{min}}) \in D^\star$ and $R(\bar{b}_2; g_{2_{min}}) \in D^\star$, with $\bar{a}_1 \subseteq \bar{b}_1$, $\bar{a}_2 \subseteq \bar{b}_2$, and $\mathsf{IIntersects}(g_{1_{min}}, g_{2_{min}})$ true. By monotonicity property of predicate IIntersects, if $\mathsf{IIntersects}(g_{1_{min}}, g_{2_{min}})$ is true, then, for any $D' \in Rep(D, \Psi)$, there exist $R_1(\bar{b}_1; g_1') \in D'$ and $R_2(\bar{b}_2; g_2') \in D'$ such that $\mathsf{Within}(g_{1_{min}}, g_1')$ and $\mathsf{Within}(g_{2_{min}}, g_2')$, and $\mathsf{IIntersects}(g_1', g_2')$ is also true. Therefore, by definition of consistent answer, $\langle \bar{a}_1, \bar{a}_2, g_{1_{min}}, g_{2_{min}}\rangle \in Con(\mathcal{Q}, D, \Psi)$.

□

The previous theorems tell us that we can obtain consistent answers to basic conjunctive queries by direct and usual query evaluation on the single instance $D^\star$, the *core* of D. This does not hold for non-basic conjunctive queries as the following example shows.

Example 11 Consider a database instance with a database predicate $R(id; geometry)$ whose spatial attribute values are shown in Figure 15(a). This database instance is inconsistent with respect to a IDSIC that specifies that geometries cannot internally intersect. Let us now consider a range query of the form $(R(x; g) \wedge Touches(g, w))$, where w is a user defined spatial window. Figure 15(b) shows the query over

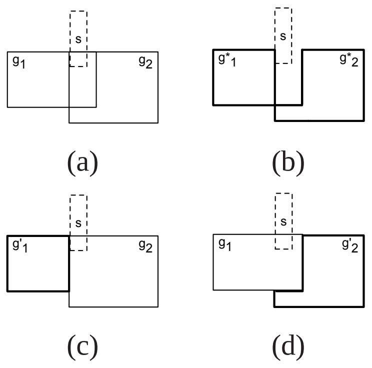

Figure 15: Core vs. consistent answers.

the intersection of all repairs (the *core*), obtaining geometries $g_1^\star$ and $g_2^\star$, where only $g_1^\star$ touches w. Figures 15(c) and (d) show the query over two minimal repairs, separately. The answer from the repair in (c) is g_1', and repair (d) does not return an answer because none of the geometries in this repair touches w. Consequently, the intersection of all minimal repairs, independently of other minimal repairs, is empty and differs from the answer obtained from the *core*. This difference is due to the fact that the query window w touches geometry g_1' in only one of the minimal repairs. $\Box$

The previous example shows the limitations of the core-based computation of CQA with respect to a subset of denial constraints and conjunctive queries. However, basic conjunctive queries are relevant queries in the spatial domain, and for IDSICs, we can provide an algorithm for efficiently computing consistent query answers.

5.2 Computing the core-based CQA

By Corollary 1, we can compute the core $D^\star$ with respect to a set Ψ of IDSICs without having to compute the minimal repairs. The basic idea is to determine the minimum geometry g_{min} in $\mathcal{G}_{R,\Psi,D}(\bar{a}, g)$, for every $R(\bar{a}, g) \in D$. By property of IDSICs, the minimum geometry g_{min} in $\mathcal{G}_{R,\Psi,D}(\bar{a}, g)$ is the one from which it has been eliminated the intersection of g with any other geometry in conflict. Then, by Theorems 1 and 2, we compute CQA over $D^\star$ for basic range and join queries.

We now show with the following example how our methodology to compute CQA could be implemented on top of current spatial database management systems by giving a specification of the core $D^\star$ as a view in SSQL.

Example 12 Consider a schema with the only relation *LandP(idl,name,owner;geometry)* with primary key idl, the IDSIC (3) of Example 2, and the instance in Figure 16. We want to consistently answer the query $\exists name\ owner(LandP\ (idl, name, owner; geometry)\ \wedge \mathsf{IIntersects}(geometry, ([x_1, y_1], [x_2, y_1], [x_2, y_2],$ $[x_1, y_2], [x_1, y_1]))$, where $[x_i, y_i]$ are constant points that define the query window in Figure 16 drawn as a rectangle with dashed boundary.

To answer this query, we generate a view of the $core$. That is, we eliminate from each geometry the union of its intersection with other land parcels. This is the definition of the core in SSQL:[9]

[9]In current SSQL $\mathsf{IIntersects}(g_1, g_2)$ is equivalent to $\mathsf{Intersects}(g_1, g_2)$ AND NOTTouches(g_1, g_2). Also, empty geometries are not evaluated in current SSQLs so that this built-in atom is omitted from the query.

LandP			
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

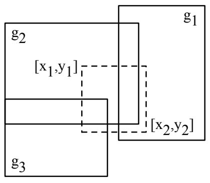

Figure 16: An example of an inconsistency database instance and basic conjunctive query.

```
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        LandP AS l1, LandP AS l2
  WHERE       l1.idl <> l2.idl AND Intersects(l1.geometry, l2.geometry) AND
              NOTTouches(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        LandP AS l1
  WHERE       NOT EXISTS(SELECT l2.idl, l2.geometry
              FROM LandP AS l2
              WHERE l1.idl <> l2.idl AND Intersects(l1.geometry, l2.geometry) AND
              NOTTouches(l1.geometry, l2.geometry)))
```

We now can evaluate the following query on the view $Core$ to compute the consistent answer to the original query:

$$\begin{array}{ll} \text{SELECT} & idl, name, owner, geometry \\ \text{FROM} & Core \\ \text{WHERE} & \mathsf{Intersects}(geometry, ([x_1, y_1], [x_2, y_1], [x_2, y_2], [x_1, y_2], [x_1, y_1])) \end{array} \quad (15)$$

The core-based answers to the query are $\langle idl_1, n_1, o_1, g'_1\rangle$, $\langle idl_2, n_2, o_2, g'_2\rangle$ and $\langle idl_3, n_3, o_3, g'_3\rangle$, where g'_1, g'_2, and g'_3 are shown in Figure 17.

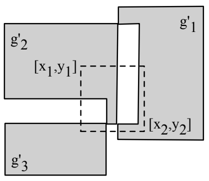

Figure 17: Geometries in the core-based computation of CQA.

□

This core-based method allows us to compute consistent answers in polynomial (quadratic) time (in data complexity) in cases where there can be an infinite number of repairs. This corresponds to a polynomial time algorithm of order polynomial with respect to the size of the database instance.

As opposed to the previous work in [33], where DSICs used for core-based computation of CQA can also include topological relations $\mathsf{Intersects}$ and Equals, we define now IDSICs with the only possible topological relation $\mathsf{IIntersects}$. This comes from the more general definition of repair semantics, where there is no specific geometric operators that limit the way we shrink geometries for solving conflicts. Indeed, by defining admissible transformations based on specific geometric operators to solve conflicts, we may apply the core-based computation of CQA for a wider type of DSICs, this at the cost of approximating the minimality of repairs. For example, the work in [33] proposes to solve repair conflicts with respect to topological relation Equals by eliminating the smallest geometry between the two geometries in conflict. By doing so, we may not have a minimal form of repair since we do not need to eliminate the whole geometry to falsify a predicate Equals, but, as we showed in [33], we can use the core-based computation of CQA.

6 Experimental Evaluation

In this section, we analyze the results of the experimental evaluation we have done of the core-based computation of CQA using synthetic and real data sets. The experiment includes a scalability analysis that compares the cost of CQA with increasing numbers of conflicting tuples and increasing sizes of database instances. We compare these results with respect to the direct evaluation of basic conjunctive queries over the inconsistent database (i.e., ignoring inconsistencies). The latter reflects the additional cost of computing consistent answers against computing queries that ignore inconsistencies.

6.1 Experimental Setup

We create synthetic databases to control the size of the database instance and the number of conflicting tuples. We use a database schema consisting of a single predicate $R(id; geometry)$, where id is the numeric key and $geometry$ is a spatial attribute of type polygon, and the following IDSIC:

$$\forall x_1 s_1 x_2 s_2\, \neg(R(x_1; s_1) \ \wedge\ R(x_2; s_2) \ \wedge\ \mathsf{NonEmpty}(s_1) \wedge \mathsf{NonEmpty}(s_2) \ \wedge \\ x_1 \neq x_2 \ \wedge\ \mathsf{IIntersects}(s_1, s_2)) \qquad (16)$$

We create five consistent instances including 5,000, 10,000, 20,000, 30,000, and 40,000 tuples of homogeneously distributed spatial objects whose geometries are rectangles (i.e., 5 points per geometric representation of rectangles). Then, we create inconsistent instances with respect to IDSIC (16) with 5%, 10%, 20%, 30%, and 40% of tuples in conflict. These instances were created by making geometries, chosen at random, internally intersect. Due to the spatial distribution of rectangles, the core of a database instance has the same size that the size of its corresponding original instance. Thus, we are not introducing additional storage costs in our experiments.

To have a better understanding of the computational cost of CQA, we also evaluate the cost of CQA over real and free available data of administrative boundaries of Chile [1]. Chilean administrative boundaries have complex shapes with many islands, specially, in the South of Chile (e.g., a region can have 891 islands). For the real database, we have two predicates $Counties$ and $Provinces$. Notice that, at the conceptual label,

$Provinces$ are aggregations of $Counties$. In this experiment, however, we have used the source data as it is, creating separated tables for $Counties$ and $Provinces$ with independent spatial attributes. For this real database, we consider IDSIC of the form:

$$\forall x_1 x_2 s_1 s_2 \, \neg(R(x_1; s_1) \ \wedge \ R(x_2; s_2) \ \wedge \mathsf{NonEmpty}(s_1) \wedge \mathsf{NonEmpty}(s_2) \wedge \\ x_1 \neq x_2 \ \wedge \ \mathsf{IIntersects}(s_1, s_2)), \tag{17}$$

with R being $Counties$ or $Provinces$.

Table 3 summaries the data sets for the experimental evaluation. The percentage of inconsistency is calculated as the number of tuple in any conflict over the total number of tuples. The geometric representation size is calculated as the number of points in the boundaries of geometries in a database instance.

Source	Name	Tuples	Inconsistency (%)	Geometric representation size
Synthetic	Synthetic	5,000-40,000	5-40	25,000-200,000
Real	Provinces	52	59	35,436
	Counties	307	12.7	72,009

Table 3: Data sets of the experimental evaluation.

We measure the computational cost in terms of seconds needed to compute the SSQL statement on a Quad Core Xeon X3220 of 2.4 GHz, 1066 MHz, and 4 GB in RAM. We use as spatial DBMS PostgreSQL 8.3.5 with PostGIS 1.3.5.

6.2 Experimental Results

Figure 18 shows the cost of the core computation for the different synthetic database instances. To make this experimental evaluation easier and faster, we used materialized views so that we computed only once the core and applied queries on this core's view. However, we added the computational cost of the core to each individual query result to have a better understanding of the cost of applying CQA.

The cost of computing the core is largely due to the join given by the topological relation of a IDSIC, which could decrease using more efficient algorithms and spatial indexing structures.

For the synthetic database instance, Figures 19 and 20 show the cost rate between computing a CQA with respect to simple range or join queries (with the spatial predicate $\mathsf{IIntersects}$) that ignore inconsistencies. Range queries use a random query window created by a rectangle whose side is equivalent to 1% of the total length in each dimension. Notice that the time cost of computing a range query, for a database instance with 10,000, was approximately 15 ms was 900 times less than computing a join query on a database instance of the same size. These reference values exhibit linear and quadratic growth for range and join queries, respectively, as we consider increasing sizes of database instances. The computational cost of CQA to join queries includes the computation of the core; however, this cost could be amortized if we use a materialized view of the core for computing more than one join query. In the time cost of CQA for range queries, we have optimized the computation by applying the core-computation over a subset of tuples previously selected by the query range. This optimization is not possible for join queries, since no spatial window can constrain the possible geometries in the answer.

The results indicate that CQA to a range query costs around 100 times the cost of a simple query for a database instance with 40000 tuples. This grows quadratically, since it primarily due to the join computation of the core. Indeed, when comparing the CQA to a join query, we only duplicate the relative cost, and in the

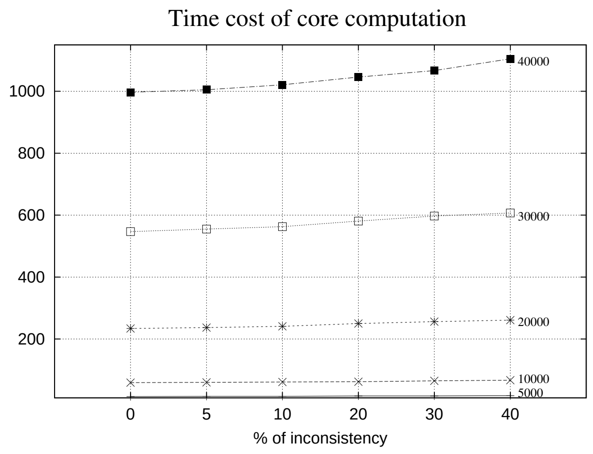

Figure 18: Time cost of the core computation for different IDSICs, different levels of inconsistency, and different sizes of databases instances.

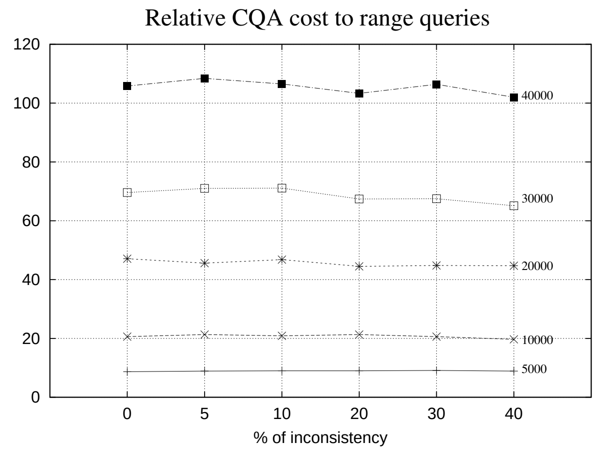

Figure 19: Relative cost of CQA to range queries.

best case, keep the same cost. However, join queries have a significant larger computational cost. Notice that the computation cost for a CQA to range query is around 60s in the worst case (40,000 tuples).

We also evaluate the scalability of the CQA cost to range queries in function of the size of the query window (i.e., spatial window). In Figure 21 we show the relative CQA cost to range queries on a synthetic database instance with 10,000 tuples and range queries whose random spatial windows varied from 1% to 5% of the size in each dimension. The results indicate that the relative cost increases logarithmical as we increase the size of the query window.

Finally, we applied the core-based computation of CQA to the real database instances in Table 3. Table 4 summaries the results obtained with these data, which were in agreement with the results obtained with the synthetic database instances. In this table, $\Delta Points$ represents the relative difference in the size of the geometric representation between the core and the original database. Notice that computing the core increased the geometric representation of $Provinces$ up to 5.0%, which is bounded by the shape of geometries in

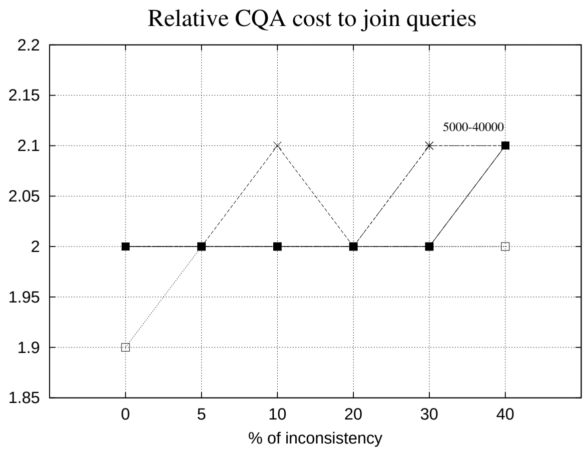

Figure 20: Relative cost of CQA to join queries.

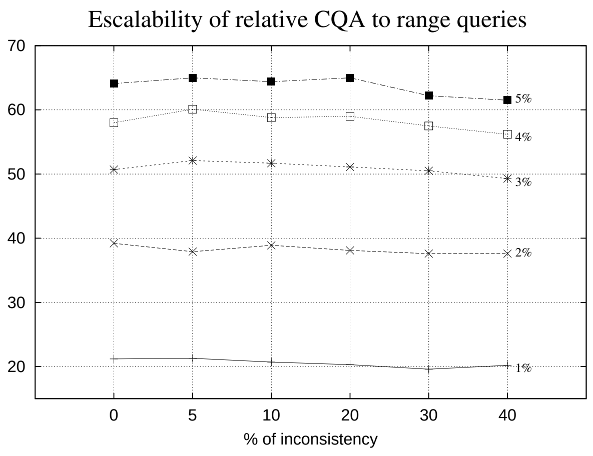

Figure 21: Relative cost of CQA to range queries and different sizes of the query window (using a database instance with 10,000 tuples).

conflict (i.e., the size of the original geometric representation). In the case of $Counties$, however, the size of the geometric representation of the core decreases down to -0.03%. Since the geometry of provinces should be the geometric aggregation of counties, we could expect to have a relationship between $\Delta Points$ for $Provinces$ and $Counties$. However, the source data set uses independent geometries for $Provinces$ and $Counties$ and no comparison can be made.

7 Conclusions

We have formalized a repair semantics and consistency query answers for spatial databases with respect to DSICs. The repair semantics is used as an auxiliary concept for handling inconsistency tolerance and computing consistent answers to spatial queries. It is based on updates that shrink geometries of objects, even at the point of deleting geometries for some exceptional cases, as for predicate Disjoint.

			Range		Join	
Data	Δ Points	Core	Simple	CQA	Simple	CQA
Provinces	+5.0%	17.7	0.04	0.25	29.8	63.4
Counties	-0.03%0	18.1	0.1	2.1	40.6	55.7

Table 4: CQA cost with real data (costs of core and queries in seconds).

Complexity analysis shows that CQA is intractable. With the purpose of avoiding to compute and query all repairs, we have identified cases of DSICs (IDSICs) and conjunctive (basic range and join) queries where the consistent answers can be obtained by posing a standard query to a single view of the original instance. This view is equivalent to the intersection of all possible minimal repairs, what we called the *core* of a database instance, which for IDSICs can be computed in polynomial time without determining each repair.

An experimental evaluation of the core-based computation of CQA reveals that answering range queries has a cost that varies quadratically with the number of tuples in the databases. This is mainly due to the spatial join involved in computing the core. These results do not use optimizations with spatial indexing, which has been left for future work. Even more, they assume that we have to compute the core for each query, which could be optimized by using materialized views.

This work leaves many problems open. We have considered only regions to represent spatial objects. A natural extension of this work would be to define a repair semantics for other spatial abstractions, such as polylines, points, networks, and so on. We would also like to explore not only DSICs, but also other classes of semantic ICs. This includes also the possibility of considering combinations of spatial with relational constraints, e.g. functional dependencies and referential ICs.

Acknowledgments

We would like to thank Loreto Bravo for her contribution to improve this paper. We also thank funding by FONDECYT, Chile, grant number 1080138. Part of this research was done when L. Bertossi was invited to the Universidad de Concepcion and Universidad del Bío-Bío. Leopoldo Bertossi has also been partially funded by an NSERC Discovery Grant (#315682).

References

[1] ALBERS, C. Gisdata chile, administrative boundaries since 2007. http://www.rulamahue.cl/mapoteca/catalogos/chile.html, 2009.

[2] ARENAS, M., BERTOSSI, L. AND CHOMICKI, J. Consistent query answers in inconsistent databases. In *18th ACM Symposium on Principles of Database Systems PODS'99* (1999), ACM Press, pp. 68–79.

[3] ARENAS, M., BERTOSSI, L. AND CHOMICKI, J. Scalar aggregation in fd-inconsistent databases. In *Proc. International Conference on Database Theory* (2001), Springer LNCS 1973, pp. 39–53.

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

[5] BERTOSSI, L. *Database Repairing and Consistent Query Answering*, Morgan & Claypool, Synthesis Lectures on Data Management, 2011.

[6] BERTOSSI, L. AND BRAVO, L. Consistent query answers in virtual data integration systems. In *Inconsistency Tolerance* (2005), vol. 3300 of *Lecture Notes in Computer Science*, Springer, pp. 42–83.

[7] BERTOSSI, L., BRAVO, L., FRANCONI, E. AND LOPATENKO, A. The complexity and approximation of fixing numerical attributes in databases under integrity constraints. In *Information Systems 33* 4-5 (2008), 407–434.

[8] BERTOSSI, L. AND CHOMICKI, J. Query Answering in Inconsistent Databases. Chapter in book *Logics for Emerging Applications of Databases* (2003), J. Chomicki, G. Saake and R. van der Meyden (eds.), Springer, pp. 43–83.

[9] BERTOSSI, L., HUNTER, A. AND SCHAUB, T., Eds. *Inconsistency Tolerance* (2005), vol. 3300 of *Lecture Notes in Computer Science*, Springer.

[10] BORGES, K., LAENDER, A. AND DAVIS, C. Spatial integrity constraints in object oriented geographic data modeling. In *ACM International Symposium on Advances in GIS* (1999), ACM Press, pp. 1–6.

[11] BRAVO, L. AND RODRÍGUEZ, M. A. Formalization and Reasoning about Spatial Semantic Integrity Constraints. In *Data & Knowledge Engineering* 72 (2012), 63–82.

[12] CANIUPAN, M. *Optimizing and Implementing Repair Programs for Consistent Query Answering in Databases*. PhD thesis, Carleton University, Department of Computer Science, 2007.

[13] CHOMICKI, J. Consistent query answering: Five easy pieces. In *International Conference on Database Theory ICDT'07* (2007), vol. LNCS 4353, Springer-Verlag, pp. 1–17.

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

[15] COHN, A., BENNETT, B., GOODAY, J. AND GOTTS, N. M. Qualitative Spatial Representation and Reasoning with the Region Connection Calculus. *GeoInformatica*, 1 (1997), 275–316.

[16] DUCKHAM, M., LINGHAM, J., MASON, K. T. AND WORBOYS, M. F. Qualitative reasoning about consistency in geographic information. *Inf. Sci. 176*, 6 (2006), 601–627.

[17] EGENHOFER, M. Deriving the Composition of Binary Topological Relations. In *J. Vis. Lang. Comput. 5* 2 (1994), pp. 133–149.

[18] EGENHOFER, M., CLEMENTINE, E. AND FELICE, P. D. Evaluating inconsistency among multiple representations. In *Spatial Data Handling* (1995), pp. 901–920.

[19] EGENHOFER, M. J. AND FRANZOSA, R. D. Point set topological relations. *International Journal of Geographical Information Systems 5* (1991), 161–174.

[20] EGENHOFER, M. AND MARK, D. Naive Geography. *International Conference on Spatial Information Theory (COSIT)* (1995), 1–15.

[21] EGENHOFER, M. AND SHARMA, J. Assessing the consistency of complete and incomplete topological information. *Geographical Systems 1* (1993), 47–68.

[22] FRANCONI, E., PALMA, A. L., LEONE, N., PERRI, S. AND SCARCELLO, F. Census data repair: a challenging application of disjunctive logic programming. In *LPAR* (2001), R. Nieuwenhuis and A. Voronkov, Eds., vol. 2250 of *Lecture Notes in Computer Science*, Springer, pp. 561–578.

[23] GRIGNI, M., PAPADIAS, D. AND PAPADIMITRIOU, C. Topological Inference. In *IJCAI*, pages 901–907, 1995.

[24] GÜTING, R. H. Graphdb: Modeling and querying graphs in databases. In *VLDB* (1994), J. B. Bocca, M. Jarke and C. Zaniolo, Eds., Morgan Kaufmann, pp. 297–308.

[25] GÜTING, R. H. AND SCHNEIDER, M. Realm-based spatial data types: The rose algebra. *VLDB J. 4*, 2 (1995), 243–286.

[26] HADZILACOS, T. AND TRYFONA, N. A model for expressing topological integrity constraints in geographic databases. In *Spatio-Temporal Reasoning* (1992), A. U. Frank, I. Campari, and U. Formentini, Eds., vol. 639 of *Lecture Notes in Computer Science*, Springer, pp. 252–268.

[27] MÄS, S. Reasoning on spatial semantic integrity constraints. In *COSIT* (2007), S. Winter, M. Duckham, L. Kulik and B. Kuipers, Eds., vol. 4736 of *Lecture Notes in Computer Science*, Springer, pp. 285–302.

[28] OPENGIS. OpenGIS simple features specification for SQL. Tech. rep., Open GIS Consortium, 1999.

[29] PAREDAENS, J., DEN BUSSCHE, J. V. AND GUCHT, D. V. Towards a theory of spatial database queries. In *PODS* (1994), ACM Press, pp. 279–288.

[30] PAREDAENS, J. AND KUIJPERS, B. Data models and query languages for spatial databases. *Data Knowl. Eng. 25*, 1-2 (1998), 29–53.

[31] RANDELL, D. A., CUI, Z. AND COHN, A. G. A spatial logic based on regions and connection. In *KR* (1992), pp. 165–176.

[32] RODRÍGUEZ, M. A. Inconsistency issues in spatial databases. In *Inconsistency Tolerance* (2005), vol. 3300 of *Lecture Notes in Computer Science*, Springer, pp. 237–269.

[33] RODRÍGUEZ, M. A., BERTOSSI, L. AND CANIUPAN, M. An inconsistency tolerant approach to querying spatial databases. In *ACM-GIS* (2008), W. G. Aref, M. Mokbel and M. Schneider, Eds., ACM.

[34] SERVIGE, S., PURICELLI, T. AND LAURINI, R. A methodology for spatial consistency improvement of geographic databases. *GeoInformatica 4* (2000), 7–24.

[35] THOMPSON, D. AND LAURINI, R. *Fundamentals of Spatial Information Systems*. No. 37 in APIC. Academic Press, 1992.

[36] TRYFONA, N. AND EGENHOFER, M. Consistency among parts and aggregates: A computational model. *Transactions on GIS 1*, 1 (1997), 189–206.

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