Published in: COSIT 2007, S. Winter et al. (Eds.)
LNCS 4736, pp. 285–302, 2007.

Reasoning on Spatial Semantic Integrity Constraints

Stephan Mäs

AGIS - Arbeitsgemeinschaft GIS,
Universität der Bundeswehr München,
Werner Heisenberg Weg 39,
85577 Neubiberg, Germany

{Stephan.Maes}@unibw.de

Abstract. Semantic integrity constraints specify relations between entity classes. These relations must hold to ensure that the data conforms to the semantics intended by the data model. For spatial data many semantic integrity constraints are based on spatial properties like topological or metric relations. Reasoning on such spatial relations and the corresponding derivation of implicit knowledge allow for many interesting applications. The paper investigates reasoning algorithms which can be used to check the internal consistency of a set of spatial semantic integrity constraints. Since integrity constraints are defined at the class level, the logical properties of spatial relations can not directly be applied. Therefore a set of 17 abstract class relations has been defined, which combined with the instance relations enables the specification of integrity constraints. The investigated logical properties of the class relations enable to discover conflicts and redundancies in sets of spatial semantic integrity constraints.

Keywords: Semantic Integrity Constraints, Spatial Relations, Class Level Relations, Reasoning, Consistency of Constraints, Constraint Networks

1 Introduction

Integrity, sometimes also called consistency, is a term originally used for the property of database systems of being free of logical contradictions within a model of reality. This model also contains defined integrity constraints that must hold on the database to grasp the semantics intended by the model [Egenhofer 1997]. Integrity constraints play a major role when the logical consistency of a data set has to be evaluated. For spatial data in particular constraints which comprehend the spatial peculiarities are of interest. While for database modelling a universally valid classification of integrity constraints is established and the constraint types are supported by most database systems, at present there is no sufficient integration of spatial integrity constraints and not even a theoretical basis for the formalisation of their contents and restrictions existing. This paper tries to contribute to a solution of these problems. It focuses on the formalisation of spatial semantic integrity constraints and the identification of conflicts and redundancies in sets of such constraints. Therefore we start with a categorisation of integrity constraints and point out how spatial integrity constraints integrate into this classification. Further the categorisation is used to outline the definition of spatial semantic integrity constraints. Since integrity constraints are defined at the level of entity classes the paper reviews the application of class relations in chapter 3. Based on that a set of 17 abstract class relations is defined which particularly supports the specification of spatial semantic integrity constraints. Another focus is on the investigation of the logical properties of the defined class relations. The reasoning algorithms investigated in chapter 4 enable to discover conflicts and redundancies in sets of spatial semantic integrity constraints. The practical value and the usability of the researched concepts are demonstrated in chapter 5, where a possible user interface for the definition of spatial semantic integrity constraints is designed.

2 Integrity Constraints

The restrictions defined by integrity constraints can be manifold. This paper focuses only on spatial semantic integrity constraints. The aim of this chapter is to categorize integrity constraints and to outline spatial semantic integrity constraints.

2.1 Categories of integrity constraints

In [Elmasri and Navathe 1994, pp. 638-643] the properties of integrity constraints for data modelling have been analysed. They propose the following classification of integrity constraints according to the type of the specified conditions:

1. **Domain constraints** restrict the allowed types of values of an attribute.
2. **Key and relationship constraints** refer to key values of entity classes, cardinalities of relationships between entity classes and participation requirements of relationships defined in the data model.
3. **General semantic integrity constraints** are explicitly specified and usually more complex. They refer to the semantics of the modelled entity classes. Therefore they can not be specified as domain or key and relationship constraints.

Spatial integrity constraints fit into two categories of this classification. Currently, a variety of database systems is already capable to handle the particular requirements of spatial data and provides predefined spatial data types. The constraints on this data types and the corresponding constraints on geometric and topological primitives are domain constraints. Such spatial integrity constraints are not discussed in this paper.

The second category of spatial constraints is semantic integrity constraints. This paper focuses on the definition of such spatial semantic integrity constraints. Therefore the following subchapter will give a closer look at the restrictions which are specified by these constraints.

2.2 What do Semantic Integrity Constraints Restrict?

Following the definition of Elmasri and Navathe semantic constraints are based on relations between the involved entities or on specific properties of a single entity. The validity of the relations is based on the semantics of the entities. Semantic integrity constraints are defined at the level of the entity classes and have to be explicitly defined. The restrictions defined by semantic integrity constraints can be manifold and complex, what makes a differentiation of the kinds of defined restrictions necessary. An approach to a categorisation of integrity constraints according to the restricted data model elements was made by [Ditt et al. 1997] and later extended by [Friis-Christensen el al. 2001]. They differentiated the following integrity constraint categories:

1. constraints referring to an attribute of a single entity
2. constraints referring to at least two attributes of a single entity
3. constraints referring to all entities of a single entity class
4. constraints referring to an entity and its associated entities of various classes
5. constraints referring to operations of entities.

All five categories include semantic integrity constraints. In this paper we only consider the categories three and four, leaving out constraints restricting single entity's attributes and operations of entities. We investigate integrity constraints on relations between the entities of a single or of two entity classes. As a further restriction we only consider binary relations which are not explicitly modelled. Relations which are explicitly defined in the data model are restricted by key and relationship constraints. Implicit relation can be deduced from the corresponding attributes of the involved entities. A typical example of such implied relations are topological relations (e.g. figure 1) between spatial entities [Egenhofer and Herring (1991)]. Usually they are not explicitly stored since they can be derived from the entity's geometries.

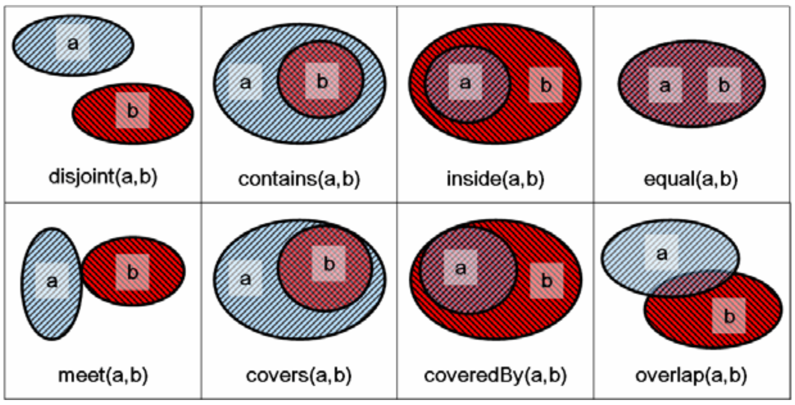


Fig. 1. Set of topological relations between areal entities

All examples of class relations and integrity constraints given throughout the paper are based on the binary topological relations between areal entities shown in figure 1. However, the defined class relations can also be combined with any other instance relation. To enable for reasoning on the integrity constraints (chapter 4) the instance relation must be part of a limited, jointly exhaustive and pairwise disjoint (JEPD) set of relations. This requirement is necessary when the consistency of the class relations shall be checked making use of reasoning algorithms. Most spatial relations are part of such a JEPD set of relations.

3 Integrity Constraint Definition based on Class Level Relations

Integrity constraints are defined at the level of entity classes since they are always restricting entire classes or subsets of classes. When a database is checked against a spatial semantic integrity constraint, the checking procedure proves spatial relations between the involved instances. Thus a formalised description of such an integrity constraint must be linked to the instance relations the quality checking procedure applies. But as the following example illustrates instance relations are not suitable for integrity constraint formalisation. A natural language statement about two instances could be: “the watermill is overlapping the river”. Since *overlap* is a symmetric relation it also implies “the river is overlapping the watermill”. A corresponding semantic integrity constraint for the classes “watermill” and “stream” could be: “A watermill must overlap a stream”. Applying the symmetry of the instance relation again it becomes: “A stream must overlap a watermill”. These two statements about the classes obviously don’t have the same semantic and since not every stream is flowing through a watermill the second is not true. This example shows that instance relations can not clearly represent the semantic of statements about classes and that the formalisation of such statements requires specific class relations. [Donnelly and Bittner 2005] also identified this problem and provided an approach for the definition of class relations. The following subsections will review their solution with regard to an application for integrity constraint definition. Some class level relations don’t define violable restrictions on the involved classes and are therefore also not applicable as integrity constraints. Further on, some additional properties of class relations, which have not been considered by Donnelly and Bittner, are pointed out. Based on that a new set of class relations is defined which particularly supports the definition of integrity constraints.

3.1 Definition of Class Level Relations

Before the class relations can be applied it must be ensured that the classes conform to the following two requirements. First, the involved classes must have at least one instance, i.e. empty classes are not feasible. As stated before class relations are linked to individual relations. Thus the second condition specifies that if a class relation is defined, there must be at least one corresponding individual relation existent among the instances of the involved classes.

x,y,z	Denote variables for individuals / instances. Every instance must be associated to a class.
A, B, C	Denote variables for classes. Every class must have at least one instance.
Inst(x,A)	Means individual x is an instance of class A.
r(x,y)	Means individual x has the relation r to individual y; x and y are said to participate on the relationship instance r. The meta-variable r can stand for any relation of individuals (e.g. topological relations). Every relationship instance r can be associated to a class relation R.
R(A,B)	Denotes that R relates the classes A and B. The meta-variable R can stand for any class relationship. Every R is related to an individual relation r. If a class relation R(A,B) is defined at least one r must be existent between the instances of A and B.

Based on these variables and functions [Donnelly and Bittner 2005] define the following class relations:

$$R^{D\&B}_{some}(A,B) := \exists x \exists y (Inst(x,A) \cap Inst(y,B) \cap r(x,y)). \qquad \text{(D\&B1)}$$

$$R^{D\&B}_{all-1}(A,B) := \forall x (Inst(x,A) \rightarrow \exists y (Inst(y,B) \cap r(x,y))). \qquad \text{(D\&B2)}$$

$$R^{D\&B}_{all-2}(A,B) := \forall y (Inst(y,B) \rightarrow \exists x (Inst(x,A) \cap r(x,y))). \qquad \text{(D\&B3)}$$

$$R^{D\&B}_{all-12}(A,B) := R^{D\&B}_{all-1}(A,B) \cap R^{D\&B}_{all-2}(A,B). \qquad \text{(D\&B4)}$$

$$R^{D\&B}_{all-all}(A,B) := \forall x \forall y (Inst(x,A) \cap Inst(y,B) \rightarrow r(x,y)). \qquad \text{(D\&B5)}$$

$R^{D\&B}_{some}(A,B)$ holds if at least one instance of A stands in relation r to some instance of B. $R^{D\&B}_{some}(A,B)$ relations are very weak, but nevertheless useful for example when class relations are defined in an ontology. Integrity constraints which are only based on such relations are not expedient, since they only specify that a relation universally exists in reality without any concrete cardinalities. Within a data set the relation is in principle possible, but does not necessarily occur within the modelled part of reality. This means that a data set, which is usually representing parts of the reality, can either contain individuals that have the relation or it doesn't; both cases are conform to the integrity constraint. Since a violation against constraints which are only specifying $R^{D\&B}_{some}(A,B)$ relations is not possible, such constraints are not useful for quality assurance. This changes if $R^{D\&B}_{some}(A,B)$ relations are specified in conjunction with a defined set of entities, like for example a relation r holds for some entities of A and B within a certain area (possibly defined by an individual entity of C). Therewith the constraint is violable by the subsets of A and B and useful for quality assurance. Since the definition of such subsets can be manifold and complex the analysis in this paper is restricted to binary relations between entire entity classes, leaving out subsets of classes.

$\mathrm{R}_{\mathrm{all-1}}^{\mathrm{D\&B}}(\mathrm{A},\mathrm{B})$ holds if every instance of A has the relation r to some instance of B. In set theory such relations are called left-total. This class relation can be used to define the integrity constraint of the above mentioned windmill / stream example: $\mathrm{OVERLAPS}_{\mathrm{all-1}}^{\mathrm{D\&B}}(\mathrm{Windmill},\mathrm{Stream})$ specifies the *overlap* relation for all windmills but it doesn't include all streams.

$\mathrm{R}_{\mathrm{all-2}}^{\mathrm{D\&B}}(\mathrm{A},\mathrm{B})$ holds if for each instance of B there is some instance of A which stands in relation r to it. This means that every instance of B has the inverse relation of r to some instance of A. $\mathrm{R}_{\mathrm{all-2}}^{\mathrm{D\&B}}(\mathrm{A},\mathrm{B})$ is right-total / surjective.

$\mathrm{R}_{\mathrm{all-12}}^{\mathrm{D\&B}}(\mathrm{A},\mathrm{B})$ combines the definitions of (D&B2) and (D&B3). It holds if every instance of A stands in relation r to at least one instance of B and for each instance of B there is at least one instance of A which stands in relation r to it. R is left-total and right-total.

This differentiation of class relations according to the totality of the involved individuals of the entity classes is very useful for the definition of integrity constraints. The class relations define constraints on all individuals of A (D&B2), all individuals of B (D&B3) or on all individuals of both arguments A and B (D&B4). In data modelling such definitions are called participation constraints on the relation. They specify whether the existence of an entity depends on its relation to another entity via the relationship type [Elmasri and Navathe 1994]. $\mathrm{R}_{\mathrm{all-1}}^{\mathrm{D\&B}}(\mathrm{A},\mathrm{B})$, $\mathrm{R}_{\mathrm{all-2}}^{\mathrm{D\&B}}(\mathrm{A},\mathrm{B})$ and $\mathrm{R}_{\mathrm{all-12}}^{\mathrm{D\&B}}(\mathrm{A},\mathrm{B})$ define total participation constraints on their relationship instances, since at least one of the classes is totally effected. $\mathrm{R}_{\mathrm{some}}^{\mathrm{D\&B}}(\mathrm{A},\mathrm{B})$ defines a partial participation constraint since not necessarily all instances of the classes A and B have the relationship instance.

A specific case of $\mathrm{R}_{\mathrm{all-12}}^{\mathrm{D\&B}}(\mathrm{A},\mathrm{B})$ is defined by the $\mathrm{R}_{\mathrm{all-all}}^{\mathrm{D\&B}}(\mathrm{A},\mathrm{B})$ relation, which holds if all instances of A have an relationship instance of R to all instances of B. This relation is very strong, since it defines restrictions on all relations between all individuals of the arguments A and B. Therewith the corresponding integrity constraints are very restrictive but for example useful when all instances of two classes are not allowed to intersect: $\mathrm{DISJOINT}_{\mathrm{all-all}}^{\mathrm{D\&B}}(\mathrm{Streets},\mathrm{Lakes})$.

Beside the total participation constraint the $\mathrm{R}_{\mathrm{all-all}}^{\mathrm{D\&B}}(\mathrm{A},\mathrm{B})$ relationship defines a so called cardinality ratio constraint, which specifies the number of relationship instances an entity can participate in [Elmasri and Navathe 1994]. In this case the number of B entities (i.e. "all" instances of B) defines in how many relationship instances each entity of A is participating and vice versa.

In data modelling total participation and cardinality ratio constraints are well established, for example when using the Entity-Relationship Model as a notation. In such models a total participation is represented by a double line for the relation and cardinality ratio for example by a N:1 next to the relation signature (see figure 2). In this example all buildings are restricted to be contained by only one parcel, while the parcels are allowed to contain an undefined number of buildings.

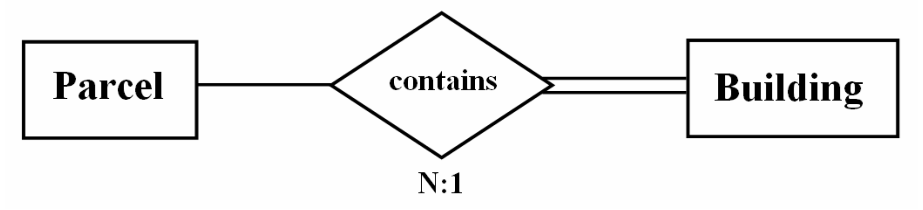


Fig. 2. Total Participation and Cardinality Ratio Constraints in an Entity-Relationship Model

The number of different cardinality ratio constraints of such a notation is infinite. Thus it is impossible to represent them all by separate class relations. But since some of them are indispensable for the definition of integrity constraints we decided to extend the framework of class relations of [Donnelly and Bittner 2005] by the concept of unambiguousness. Correspondingly the following class relations are defined:

$$\begin{aligned} R_{Left-D}(A,B) := \forall x,y,z(&Inst(x,A) \cap Inst(y,B) \cap Inst(z,A) \cap \\ &r(x,y) \cap r(z,y) \rightarrow x = z) \cap R^{D\&B}_{some}(A,B). \end{aligned} \tag{1}$$

$$\begin{aligned} R_{Right-D}(A,B) := \forall x,y,z(&Inst(x,A) \cap Inst(y,B) \cap Inst(z,B) \cap \\ &r(x,y) \cap r(x,z) \rightarrow y = z) \cap R^{D\&B}_{some}(A,B). \end{aligned} \tag{2}$$

R_{Left-D}(A,B) relations are left-definite / injective and specify that for no instance of B there is more than one instance of A which stands in relation r to it. This relation restricts the number of R relations an instance of B can participate; the instances of A are not restricted. The last term $R^{D\&B}_{some}(A,B)$ ensures that at least one instance relation r does exist between the instances of A and B.

The right-definite relations $R_{Right-D}$(A,B) specify that no instance of A participates in a relationship instance of R to more than one instance of B. When this relation is defined all instances of A are restricted while the instances of B are not affected.

Both relations are very useful for the definition of integrity constraints, since they restrict the number of possible relations of the involved individuals to a maximum of one.

3.2 Class Level Relations for Integrity Constraint Definition

The main properties of the class level relations used in the previous chapter are left-definite, right-definite, left-total and right-total. These properties are independent of each other, what means that no property implies or precludes one of the other properties. If a class relation is only defined as right-total there is no information about its left totality and the cardinality ratio available. For the definition of integrity constraints this situation is insufficient since the constraint relations should allow for combinations of properties as well as their negations. Therefore we define a new set of class relations which implies adjustments to achieve sets of pairwise disjoint relations.

$$\begin{aligned} R_{\mathrm{some}}(A,B) := R^{\mathrm{D\&B}}_{\mathrm{some}}(A,B) \cap \neg R^{\mathrm{D\&B}}_{\mathrm{all-1}}(A,B) \cap \neg R^{\mathrm{D\&B}}_{\mathrm{all-2}}(A,B) \cap \\ \neg R_{\mathrm{Left-D}}(A,B) \cap \neg R_{\mathrm{Right-D}}(A,B). \end{aligned} \tag{3}$$

$R_{\mathrm{some}}(A,B)$ is similar to the definition (D&B1) of [Donnelly and Bittner 2005] but while $R^{\mathrm{D\&B}}_{\mathrm{some}}(A,B)$ contains all other defined class relations these are now excluded. R_{some} is defined as not left-total and not right-total, what implies that some instances of A/B participate in a relation r to an instance of B/A and some don't. Furthermore the exclusions of $R_{\mathrm{Left\text{-}D}}(A,B)$ and $R_{\mathrm{Right\text{-}D}}(A,B)$ specify that some A/B participate in a relation r to at least two instances of B/A. Hence R_{some} is only valid for classes with more than two instances.

$$R_{\mathrm{LD}}(A,B) := R_{\mathrm{Left-D}}(A,B) \cap \neg R_{\mathrm{Right-D}}(A,B) \cap \neg R^{\mathrm{D\&B}}_{\mathrm{all-1}}(A,B) \cap \neg R^{\mathrm{D\&B}}_{\mathrm{all-2}}(A,B). \tag{4}$$

$$R_{\mathrm{RD}}(A,B) := \neg R_{\mathrm{Left-D}}(A,B) \cap R_{\mathrm{Right-D}}(A,B) \cap \neg R^{\mathrm{D\&B}}_{\mathrm{all-1}}(A,B) \cap \neg R^{\mathrm{D\&B}}_{\mathrm{all-2}}(A,B). \tag{5}$$

$$R_{\mathrm{LT}}(A,B) := \neg R_{\mathrm{Left-D}}(A,B) \cap \neg R_{\mathrm{Right-D}}(A,B) \cap R^{\mathrm{D\&B}}_{\mathrm{all-1}}(A,B) \cap \neg R^{\mathrm{D\&B}}_{\mathrm{all-2}}(A,B). \tag{6}$$

$$R_{\mathrm{RT}}(A,B) := \neg R_{\mathrm{Left-D}}(A,B) \cap \neg R_{\mathrm{Right-D}}(A,B) \cap \neg R^{\mathrm{D\&B}}_{\mathrm{all-1}}(A,B) \cap R^{\mathrm{D\&B}}_{\mathrm{all-2}}(A,B). \tag{7}$$

The definitions (4) to (7) specify class relations which are either left-definite, right-definite, left-total or right-total. The corresponding other properties are excluded.

$$R_{\mathrm{LD.RD}}(A,B) := R_{\mathrm{Left-D}}(A,B) \cap R_{\mathrm{Right-D}}(A,B) \cap \neg R^{\mathrm{D\&B}}_{\mathrm{all-1}}(A,B) \cap \neg R^{\mathrm{D\&B}}_{\mathrm{all-2}}(A,B). \tag{8}$$

$$R_{\mathrm{LD.LT}}(A,B) := R_{\mathrm{Left-D}}(A,B) \cap \neg R_{\mathrm{Right-D}}(A,B) \cap R^{\mathrm{D\&B}}_{\mathrm{all-1}}(A,B) \cap \neg R^{\mathrm{D\&B}}_{\mathrm{all-2}}(A,B). \tag{9}$$

$$R_{\mathrm{LD.RT}}(A,B) := R_{\mathrm{Left-D}}(A,B) \cap \neg R_{\mathrm{Right-D}}(A,B) \cap \neg R^{\mathrm{D\&B}}_{\mathrm{all-1}}(A,B) \cap R^{\mathrm{D\&B}}_{\mathrm{all-2}}(A,B). \tag{10}$$

$$R_{\mathrm{RD.LT}}(A,B) := \neg R_{\mathrm{Left-D}}(A,B) \cap R_{\mathrm{Right-D}}(A,B) \cap R^{\mathrm{D\&B}}_{\mathrm{all-1}}(A,B) \cap \neg R^{\mathrm{D\&B}}_{\mathrm{all-2}}(A,B). \tag{11}$$

$$R_{\mathrm{RD.RT}}(A,B) := \neg R_{\mathrm{Left-D}}(A,B) \cap R_{\mathrm{Right-D}}(A,B) \cap \neg R^{\mathrm{D\&B}}_{\mathrm{all-1}}(A,B) \cap R^{\mathrm{D\&B}}_{\mathrm{all-2}}(A,B). \tag{12}$$

$$\begin{aligned} R_{\mathrm{LT.RT}}(A,B) := \neg R_{\mathrm{Left-D}}(A,B) \cap \neg R_{\mathrm{Right-D}}(A,B) \cap R^{\mathrm{D\&B}}_{\mathrm{all-1}}(A,B) \cap \\ R^{\mathrm{D\&B}}_{\mathrm{all-2}}(A,B) \cap \neg R^{\mathrm{D\&B}}_{\mathrm{all-all}}(A,B). \end{aligned} \tag{13}$$

The definitions (8) to (13) combine pairs of the four defined class relation properties and exclude the corresponding others. A special case is (13) which additionally excludes $R_{\text{all-all}}^{\text{D\&B}}(A, B)$.

$$R_{\text{LT.RT-all}} := R_{\text{all-all}}^{\text{D\&B}}(A, B). \tag{14}$$

Definition (14) is equivalent to (D&B5). $R_{\text{LT.RT-all}}$ is left-total and right-total and holds if all instances of A have a relationship instance of R to all instances of B.

$$R_{\text{LD.RD.LT}}(A, B) := R_{\text{Left-D}}(A, B) \cap R_{\text{Right-D}}(A, B) \cap R_{\text{all-1}}^{\text{D\&B}}(A, B) \cap \neg R_{\text{all-2}}^{\text{D\&B}}(A, B). \tag{15}$$

$$R_{\text{LD.RD.RT}}(A, B) := R_{\text{Left-D}}(A, B) \cap R_{\text{Right-D}}(A, B) \cap \neg R_{\text{all-1}}^{\text{D\&B}}(A, B) \cap R_{\text{all-2}}^{\text{D\&B}}(A, B). \tag{16}$$

$$\begin{aligned} R_{\text{LD.LT.RT}}(A, B) := R_{\text{Left-D}}(A, B) \cap \neg R_{\text{Right-D}}(A, B) \cap R_{\text{all-1}}^{\text{D\&B}}(A, B) \cap \\ R_{\text{all-2}}^{\text{D\&B}}(A, B) \cap \neg R_{\text{all-all}}^{\text{D\&B}}(A, B). \end{aligned} \tag{17}$$

$$\begin{aligned} R_{\text{RD.LT.RT}}(A, B) := \neg R_{\text{Left-D}}(A, B) \cap R_{\text{Right-D}}(A, B) \cap R_{\text{all-1}}^{\text{D\&B}}(A, B) \cap \\ R_{\text{all-2}}^{\text{D\&B}}(A, B) \cap \neg R_{\text{all-all}}^{\text{D\&B}}(A, B). \end{aligned} \tag{18}$$

The definitions (15) to (18) combine three of the four defined class relation properties respectively and exclude the corresponding fourth. Particular attention must be given to the class relations which are left-total and right-total ((17) and (18)). In case only one instance of A or B exists left-total and right-total class relations are always left-definite or right-definite, respectively. Furthermore they will also hold (D&B5). Thus it is necessary to separate the relations (17) and (18) from (14), which is done by the exclusion of $R_{\text{all-all}}^{\text{D\&B}}(A, B)$. Therewith the relations (17) and (18) are not possible if class A or class B has only one instance.

$$\begin{aligned} R_{\text{LD.RD.LT.RT}}(A, B) := R_{\text{Left-D}}(A, B) \cap R_{\text{Right-D}}(A, B) \cap R_{\text{all-1}}^{\text{D\&B}}(A, B) \cap \\ R_{\text{all-2}}^{\text{D\&B}}(A, B) \cap \neg R_{\text{all-all}}^{\text{D\&B}}(A, B). \end{aligned} \tag{19}$$

Definition (19) specifies class relations which are left-definite, right-definite, left-total and right-total. Similar to the definitions (17) and (18) $R_{\text{all-all}}^{\text{D\&B}}(A, B)$ is excluded to distinguish the relation from (14) for the case that A and B have only one instance. In this case the relation can’t occur.

All together the definitions (3) to (19) specify 17 class relations which can be combined with any binary instance relations to associate classes. With the exception of $R_{some}(A,B)$ all of these relations specify restrictions which can be used as integrity constraints for quality assurance of the data.

Depending on the relations it is possible to define more than one class relation between two classes, even when the applied instance relations are part of the same JEPD set of relations. Figure 3 illustrates such an example, where two classes can be restricted by three topological class relations. In this scene every instance of A meets one and contains another instance of B. One instance of B meets two instances of A and some are disjoint from all instances of A. Other than those three relations are not occurring. The corresponding integrity constraints are the class relations defined in (11), (15) and (14):

$$\text{MEET}_{\text{RD.LT}}\,(\text{Entity Class A, Entity Class B})$$
$$\text{CONTAINS}_{\text{LD.RD.LT}}\,(\text{Entity Class A, Entity Class B})$$
$$\left[\text{MEET} \cup \text{CONTAINS} \cup \text{DISJOINT}\right]_{\text{LT.RT-all}}(\text{A}, \text{B})$$

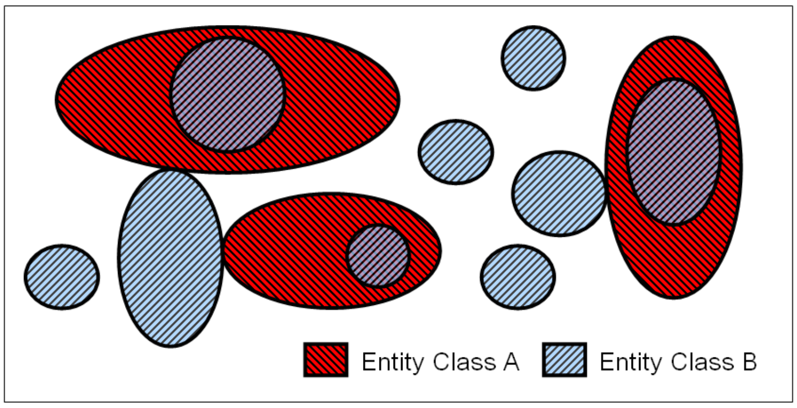


Fig. 3. Scene of two entity classes which could be defined by three class relations

The possibilities of such combinations of class relations are limited if they rest on instance relations which are part of a JEPD set of relations. For example if $R_{LT.RT\text{-}all}$ is defined no second $R_{LT.RT\text{-}all}$ relation and no class relation based other instance relations of that domain is possible between the two classes.

4 Reasoning on Semantic Integrity Constraints

This chapter investigates the logical properties of the class relations defined in the previous section. For the work with integrity constraints these properties can be very useful; for example they enable to discover inconsistencies and redundancies in a set of integrity constraints.

4.1 Reasoning on the Symmetry of the Class Relations

The transfer of logical properties of instance relations to class relations, like their symmetry and transitivity, has been researched by [Donnelly and Bittner 2005]. The purpose of this subchapter is to deepen the analysis of symmetry properties of the defined class relations. Spatial relations between instances are usually either symmetric or have a well defined inverse relation. Table 1 shows the correlation between symmetry properties of instance level relations and those of the corresponding class level relations.

r^i	Inverse instance relation.
R^i	Inverse class relation.

Table 1. Symmetry properties of the class relations

Individual Relation r is…	Class Relation R is…					
	left-definite	right-definite	left-total	right-total	R_{some}	$R_{LT.RT\text{-}all}$
symmetric	R right-definite	R left-definite	R right-total	R left-total	R_{some}	$R_{LT.RT\text{-}all}$
Not symmetric	R^i right-definite	R^i left-definite	R^i right-total	R^i left-total	R^i_{some}	$R^i_{LT.RT-all}$

The following examples illustrate the use of table 1 for class relations defined for the scene shown in figure 3. The class relations are based on the symmetric instance relation *meet* and the inverse relations *contains* and *inside*:

$$(\text{MEET}_{RD.LT}(A, B))^i = \text{MEET}_{LD.RT}(B,A).$$
$$(\text{CONTAINS}_{LD.RD.LT}(A, B))^i = \text{INSIDE}_{LD.RD.RT}(B,A).$$

The examples show that not all class relations are symmetric, even when they are based on symmetric instance relations. But it can be proven that if an instance relation is symmetric or has an inverse relation there exists also an inverse relation for each of the corresponding class relations.

4.2 Correlation between Class Relations and the Number of Individuals

For many entity classes the number of existing individuals is unknown or variable. For these classes the dependency between class relations and the number of individuals of a class is irrelevant. But for classes with a small and well defined number of individuals the designer of a data model is in many cases aware of these numbers. Such classes are for example earth surface or continents. Another example is the class “capital” which can only have one instance if the area of interest is restricted to a single “country”. The knowledge about these numbers and their correlation to the class relations should be included when reasoning on class relations.

As already stated in chapter 3.2 some class relations are not valid if one or both of the involved classes have less than three instances. The only class relation that is possible if both classes have only one instance is $R_{LT.RT\text{-}all}$. If class A has one instance the only possible class relations are $R_{LD.LT}$, $R_{LT.RT\text{-}all}$ and $R_{LD.RD.LT}$; if B has one instance only $R_{RD.RT}$, $R_{LT.RT\text{-}all}$ and $R_{LD.RD.RT}$.

The definition of class relations can also be restricted if the number of instances is more than one. If the number of instances of one of the classes is known, some class relations allow for conclusions about the number of instances of the other class. These reasoning properties are shown in the following list of theorems:

Count(A)	Denotes the number of individuals of the class A.
T1	$R_{LD.RD.LT}(A,B) \rightarrow \mathrm{Count}(A) < \mathrm{Count}(B).$
T2	$R_{LD.RD.RT}(A,B) \rightarrow \mathrm{Count}(A) > \mathrm{Count}(B).$
T3	$R_{LD.LT.RT}(A,B) \rightarrow \mathrm{Count}(A) < \mathrm{Count}(B).$
T4	$R_{RD.LT.RT}(A,B) \rightarrow \mathrm{Count}(A) > \mathrm{Count}(B).$
T5	$R_{LD.RD.LT.RT}(A,B) \rightarrow \mathrm{Count}(A) = \mathrm{Count}(B).$

Furthermore the number of instances can restrict the possible combinations of class relations between two classes. For example if exactly two instances of A exist, only a combination of two $R_{LD.RT}$(A,B) class relations can be defined for one set of JEPD instance relations. A third $R_{LD.RT}$(A,B) would require at least one more instance of A.

4.3 Composition of Class Relations

The composition of binary relations enables for the derivation of implicit knowledge about a triple of entities. If two binary relations are known the corresponding third one can potentially be inferred or some relations can be excluded. Examples of composition tables of instance relations can be found in [Egenhofer 1994] and [Grigni et al. 1995] for topological relations between areal entities and in [Hernandez 1994] and [Freksa 1992] for directional/orientation relations. Many other sets of binary spatial relations also allow for such derivations.

A transfer of this reasoning formalism to the class level would be very useful for the work with integrity constraints and other applications of class relations. In general the composition of class relations is not independent of the composition of instance relations. The composition of class relations is possible if the applied instance relations belong to the same set of JEPD relations and this set allows for compositions at the instance level. Using for example the 17 class relations together with the 8 topological relations between regions (see figure 1) would result in 136 topological class relations and almost 18500 compositions. Since such an amount of compositions is hardly manageable we propose a two level reasoning formalism, which separates the compositions of the abstract class relations from those of the instance relations. For lack of space we don't derive all compositions in this paper, but the following three examples shall illustrate the general approach. For the compositions of the applied instance relations we refer to the composition table of binary topological relations between areal entities of [Egenhofer 1994].

The first example derives the composition from the two abstract class relations $R1_{LT.RT\text{-}all}(A,B)$ and $R2_{LT.RT\text{-}all}(B,C)$. Therewith all instances of A have a relationship instance of R1 to all instances of B and all instances of B have a relationship instance of R2 to all instances of C. Since all instances of A have the same kind of relation to all instances of B and all instances of B participate in same kind of relation to all instances of C, it is obvious that all instances of A must have the same relation to all instances of C. In other words every possible triple of instances of A, B and C is related by the same relations. Thus the composition of the abstract class relations must be:

$$R1_{LT.RT-all}(A,B) \cap R2_{LT.RT-all}(B,C) \Rightarrow R3_{LT.RT-all}(A,C).$$

For the combination of this result with the instance level compositions two cases have to be distinguished. If the composition of the instance relations is unique (i.e. it results in only one relation) the combined composition is also based on that single relation like in the following example:

$$\text{equal}(a,b) \cap \text{disjoint}(b,c) \Rightarrow \text{disjoint}(a,c).$$
$$\text{EQUAL}_{LT.RT-all}(A,B) \cap \text{DISJOINT}_{LT.RT-all}(B,C) \Rightarrow \text{DISJOINT}_{LT.RT-all}(A,C).$$

If the instance relation composition results in a disjunction of instance relations the combined composition also leads to a disjunction in the class relation, for example:

$$\text{meet}(a,b) \cap \text{covers}(b,c) \Rightarrow \text{disjoint}(a,c) \cup \text{meet}(a,c).$$
$$\text{MEET}_{LT.RT-all}(A,B) \cap \text{COVERS}_{LT.RT-all}(B,C) \Rightarrow [\text{DISJOINT} \cup \text{MEET}]_{LT.RT-all}(A,C).$$

In this example it is derived that all instances of A have one of the relations *disjoint* or *meet* to all instances of C.

For the second example the $R1_{LT.RT\text{-}all}$ relation between the classes A and B is kept and the relation between B and C is $R2_{RD}$. Therewith no instance of B participates in a relationship instance of R2 to more than one instance of C and some instances of B and C are not related by a relationship instance of R2. A possible scene that implements the two abstract class relations for the instance relations *meet* and *contains* is shown in figure 4. As the figure illustrates, for every possible triple of instances the relation between the instances of A and B is *meet* (instance of R1) but only triples which include the instances b1 and c1 have *contains* (instance of R2) as relation. In general R3 can only be derived for the triples of instances which have an R2 relation. In figure 4 this are only the relations a1 to c1 and a2 to c1 with the instance composition:

$$\text{meet}(a,b) \cap \text{contains}(b,c) \Rightarrow \text{disjoint}(a,c).$$

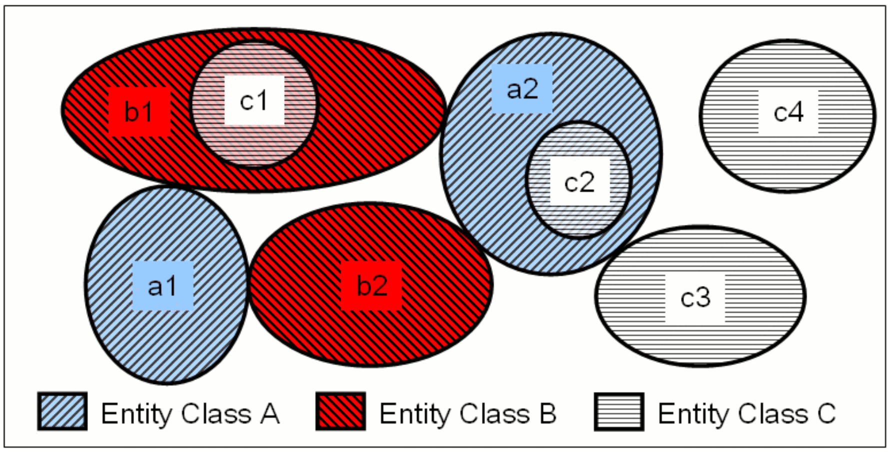


Fig. 4. Scene that implements $\text{MEET}_{\text{LT.RT-all}}$ (A, B) and $\text{CONTAINS}_{\text{RD}}$(B,C)

Figure 4 also shows that there are many other relations possible between instances of A and C. Thus it is not possible to derive a unique class relation; all of the abstract class relations (3) to (19) are possible. The only implication is that some (but not necessarily all) of the instances of A and C participate in the relation that results from the instance composition. Thus the composition of the abstract class relations is

$$\text{R1}_{\text{LT.RT-all}}(\text{A},\text{B}) \cap \text{R2}_{\text{RD}}(\text{B},\text{C}) \Rightarrow \text{R3}_{\mathcal{U}}(\text{A},\text{C}).$$

$\text{R}_{\mathcal{U}}$ denotes the universal disjunction of all class relations of (3) to (19) of the corresponding instance relation r.

It might be possible that the number of instances of the classes allow for an exclusion of some of the class relations from the disjunction (see section 4.2). For the example in figure 4 the combined composition is:

$$\text{MEET}_{\text{LT.RT-all}}(\text{A},\text{B}) \cap \text{CONTAINS}_{\text{RD}}(\text{B},\text{C}) \Rightarrow \text{DISJOINT}_{\mathcal{U}}(\text{A},\text{C}).$$

Instance compositions which don't result in a unique relation are treated in analogy to the first example.

In the third example R1_{some} relates the classes A and B and R2_{RD} the classes B and C. This implies that some (not all) instances of A and B are related by a relationship instance of R1 and some (not all) instances of B and C are related by a relationship instance of R2. This doesn't mean that a triple of A, B and C instances exists, which includes both instance relations r1 and r2. Thus the instance composition is not possible and also the composition of the abstract class relations leads to no restriction of possible relations. The combined composition is undetermined:

$$\text{R1}_{\text{some}}(\text{A},\text{B}) \cap \text{R2}_{\text{RD}}(\text{B},\text{C}) \Rightarrow \mathcal{U}_{\mathcal{U}}(\text{A},\text{C}).$$

$\mathcal{U}$	denotes the universal disjunction of instance relations of the corresponding set of relations, for example for topological relations the disjunction of all 8 relations shown in figure 1.

These examples show that the composition of the defined class relations is possible. It can be extended with similar derivations to all possible compositions. The two levels of compositions can be separately analysed and therewith the reasoning formalism can be used with any spatial or non-spatial set of instance relations. In general the composition of class relations is not independent of the composition of instance relations. The class level composition is only possible if the corresponding instance relation can be derived.

4.4 Consistency of Class Relation Networks

The application of reasoning algorithms for checking consistency and discovering redundancies in networks of instance relations has for example been demonstrated in [Egenhofer and Sharma 1993] and [Rodríguez 2004]. The proof of consistency of a network of binary relations is a constraint satisfaction problem. In a consistent network of JEPD relations the following three constraints are fulfilled: node consistency, arc consistency and path consistency. The reasoning properties of class relations investigated in the previous subchapters provide the basis to check these three consistency requirements in networks of class relations.

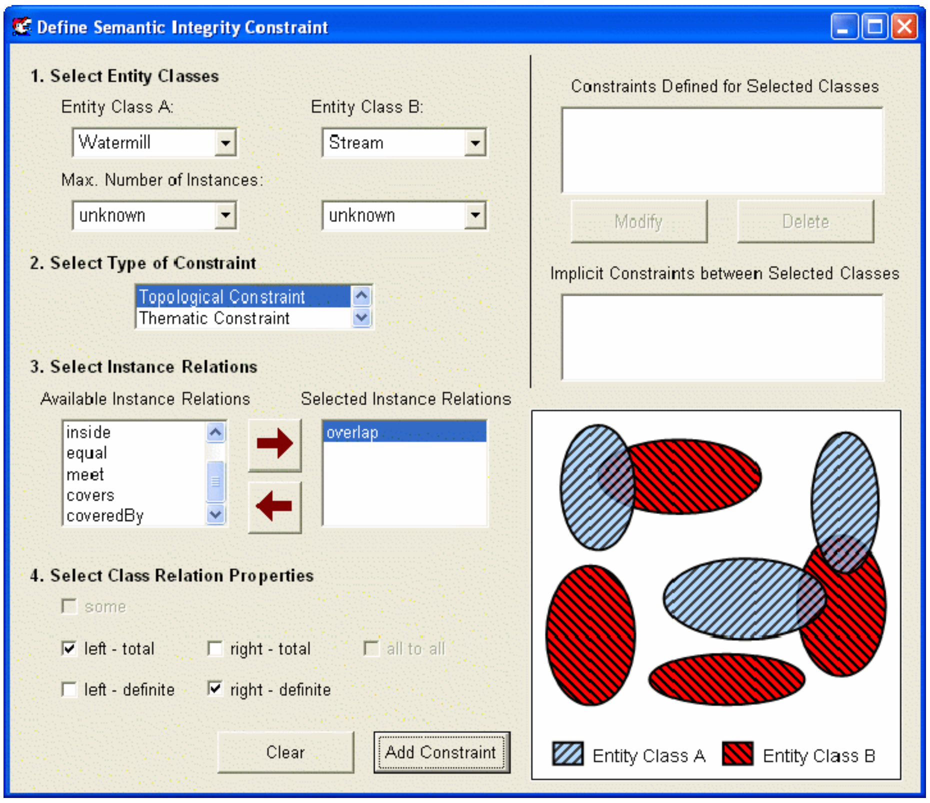
Node consistency is ensured if every node has an identity relation. For the class relation networks this means that every class must have a relation to itself. If a corresponding identity instance relation is available the identity class relation is in general $\mathrm{R}_{\mathrm{LD.RD.LT.RT}}$; for example $\mathrm{EQUAL}_{\mathrm{LD.RD.LT.RT}}(\mathrm{A},\mathrm{A})$ when using the topological relations of figure 1.

A network of relations is arc consistent if every edge of the network has an edge in the reverse direction, i.e. every relation has an inverse relation. It has been shown in section 4.1 that if an instance relation is symmetric or has an inverse relation there is also an inverse relation for each of the corresponding class relations. For instance relations this is the only requirement to proof the arc consistency. As exemplified in section 3.2 it is possible to define more than one class relation between two classes, even when the applied instance relations are part of the same JEPD set of relations. This is a fundamental difference to the instance relations and has to be considered when checking the arc consistency at the class level. If there are combinations of class relations defined their consistency has to be proven, because not all class relations can be combined (see section 4.2). This also includes the available knowledge about the number of instances. If the number of instances of more than one class is known also the theorems (T1) – (T5) have to be checked for those classes. This shows that the arc consistency of networks of class relations is more complex to prove than for networks of instance relations, but it is possible to exclude inconsistencies.

For the proof of path consistency the compositions of all possible node triples must be checked. Therefore the composition of the class relations investigated in the previous subchapter can be used.

5 Stepwise Definition of Semantic Integrity Constraints

The logical properties of class relations investigated in the previous section can be used to check the consistency and to find redundancies in a set of integrity constraints. Thus the corresponding reasoning algorithms should be applied when the semantic integrity constraints are defined to discover inconsistencies as early as possible. Figure 5 shows a possible user interface for the definition of semantic integrity constraints between two classes.


Define Semantic Integrity Constraint
Constraints Defined for Selected Classes
Modify
Delete
Implicit Constraints between Selected Classes
Clear
Add Constraint
▨ Entity Class A
▨ Entity Class B

1. Select Entity Classes

Entity Class A: Watermill

Entity Class B: Stream

Max. Number of Instances: unknown / unknown

2. Select Type of Constraint

Topological Constraint (selected)
Thematic Constraint

3. Select Instance Relations

Available Instance Relations: inside, equal, meet, covers, coveredBy

Selected Instance Relations: overlap

4. Select Class Relation Properties

☐ some

☑ left - total ☐ right - total ☐ all to all

☐ left - definite ☑ right - definite

Fig. 5. User interface for the definition of semantic integrity constraints between two classes

Therewith the definition of an integrity constraint based on the defined class relations is compiled in four main steps:

Firstly the user selects the entity classes which shall be restricted by the constraint. After the selection previously defined and implied semantic integrity constraints between these two classes are displayed on the right side of the window. For the definition of spatial integrity constraints the geometry types of the entity classes must be known, because some spatial relations are only valid for certain geometry types. Hence the geometry types should be either known by the system or can be read from available data model or schema information like UML models (Unified Modeling Language) encoded in XMI (XML Metadata Interchange) or GML application schema (Geography Markup Language) documents. If there is no information about the geometry types of the entity classes available or if some entity classes have more than one geometry a corresponding listbox for each entity class should be added to the interface. If the user is aware of the number of existent instances he can enter them after the selection of entity classes.

Secondly, the user selects the type of semantic constraint he wants to define for example topological, metric, directional or other non-spatial constraints like temporal. These types are classified according to the semantic domains of the individual relations.

As a third step the user selects one or more instance relations which the class relation of the desired integrity constraint is based on. The assortment of instance relations made available by the interface is adjusted to geometry types of the entity classes and the type of semantic integrity constraint selected in the previous steps. As stated before the geometry types of the entity classes must be known, because for example the valid topological relations between line entities differ from those between areal entities. Here a categorisation of topological relations like the one given in [Egenhofer and Herring 1991] for the region, line and point geometries is necessary.

The fourth step is the selection of the desired properties of the class relation which have been introduced in the previous sections of the paper. The four class relation properties can be separately activated which is much more convenient for the user than selecting one of the 17 defined class relations. To ensure that the user inputs conform with the 17 class relations only check boxes which lead to valid class relations are enabled. For example the “all to all instances” check box is only enabled, when left-total and right-total are checked while the other three aren’t. The final integrity constraint relation results out of the combination of the selected instance relation(s) and the activated class relation properties. The interface shown in figure 5 contains the settings of the previous example integrity constraint which defines an $\text{OVERLAP}_{\text{RD.LT}}$ relation between the classes “watermill” and “stream”.

6 Conclusion

The paper defines abstract 17 class level relations which enable a formalised specification of semantic integrity constraints. The investigated reasoning concepts can be used to find conflicts and redundancies in sets of spatial semantic integrity constraints. The definitions and reasoning rules of the class relations are described independently of a concrete set of instance relations, what makes them applicable for many spatial and non-spatial relations. The only requirements on the instance relations are that they are part of a JEPD set of relations and have defined inverse relations and compositions. Further work will be on the implementation of the introduced approach for checking consistency of sets of integrity constraints to prove the introduced reasoning concepts.

The formalised specification of integrity constraints is improving their management and usability, which will finally result in an improvement of data quality. If an integrity constraints can be composed of other integrity constraints a dataset automatically complies with this constraint if it has been checked against the composing constraints. Hence the exclusion of redundant integrity constraints minimises the number of integrity constraints which have to be verified during a quality check and therewith it is reducing calculation costs.

The investigation of the categories of integrity constraints revealed that there are many different kinds of semantic integrity constraints, but not all of them can be covered by this approach. Nevertheless this framework provides a basis that can be extended by other, possibly more complex types of semantic integrity constraints. As one possible next step semantic constraints on attributes could be included.

The defined class relations are not restricted to applications as integrity constraints. As originally suggested by [Donnelly and Bittner 2005] they can also be useful for the definition of relations between classes in an ontology. Moreover the reasoning concepts can be used to check the consistency of the relations in such ontology or to discover conflicts in the concepts defined in different ontologies.

The use of the introduced concepts is currently restricted by the unavailability of composition tables for many of the spatial or non-spatial relations. For a broader application at least composition tables for topological relations between entities with simple geometries like points or linestrings must be available. The application of other spatial relations is mostly hampered by the lack of a common understanding of their concepts.

References

Ditt, H.; Becker, L.; Voigtmann, A.; Hinrichs, K. H. (1997): Constraints and Triggers in an Object-Oriented Geo Database Kernel. In: 8th International Workshop on Database and Expert Systems Applications (DEXA '97), p. 508-513

Donnelly, M.; Bittner, T. (2005): Spatial Relations Between Classes of Individuals. In: Lecture Notes in Computer Science, Volume 3693, Sep. 2005, Pages 182–199

Egenhofer, M.; Herring, J. (1991): Categorizing Binary Topological Relationships Between Regions, Lines, and Points in Geographic Databases. Technical Report, Department of Surveying Engineering, University of Maine, Orono, ME.

Egenhofer, M.; Sharma, J. (1993): Assessing the Consistency of Complete and Incomplete Topological Information. In: Geographical Systems 1 (1): 47-68, 1993

Egenhofer, M. (1994): Deriving the Composition of Binary Topological Relations. In: Journal of Visual Languages and Computing 5 (2): 133-149, 1994.

Egenhofer, M.J. (1997): Consistency Revisited. Editorial in: GeoInformatica, 1, pp. 323 – 325

Elmasri, R.; Navathe, S. B. (1994): Fundamentals of Database Systems, 2nd Edition (Addison-Wesley), The Benjamin/Cummings Publishing Company Inc.

Freksa, Christian (1992): Using Orientation Information for Qualitative Spatial Reasoning. In: Theories and Methods of Spatio-Temporal Reasoning in Geographic Space. A. U. Frank, I. Campari, and U. Formentini, (Eds.), Lecture Notes In Computer Science, vol. 639. Springer-Verlag, London, 162-178.

Friis-Christensen, A.; Tryfona, N.; Jensen, C.S. (2001): Requirements and Research Issues in Geographic Data Modeling. In: Proceedings of the 9th ACM international symposium on Advances in geographic information systems, Atlanta, Georgia, USA, pp: 2 – 8

Grigni, M., Papadias, D., and Papadimitriou, C. (1995): Topological inference. In Proceedings of the International Joint Conference of Artificial Intelligence (IJCAI), pp. 901-906, 1995

Hernandez, D. (1994): Qualitative Representation of Spatial Knowledge. In: Lecture Notes in Artifical Intelligence, Vol. 804, Springer-Verlag New York, Inc.

Rodríguez, A.; Van de Weghe, N.; De Maeyer, P. (2004): Simplifying Sets of Events by Selecting Temporal Relations. In: Lecture Notes in Computer Science, Volume 3234, Jan 2004, Pages 269 – 284