On the Consistency of Spatial Semantic Integrity Constraints

Stephan Mäs

Vollständiger Abdruck der von der Fakultät für Bauingenieur- und Vermessungswesen der Universität der Bundeswehr München zur Erlangung des akademischen Grades eines Doktor-Ingenieurs (Dr.-Ing.) genehmigten Dissertation.

Vorsitzender:	Prof. Dr.-Ing. habil. Wilhelm Caspary (i.R.), Universität der Bundeswehr München
1. Berichterstatter:	Prof. Dr.-Ing. Wolfgang Reinhardt, Universität der Bundeswehr München
2. Berichterstatter:	Prof. Dr.-Ing. Max J. Egenhofer, University of Maine, Orono, USA

Diese Dissertation wurde am 20.Oktober 2009 bei der Universität der Bundeswehr München eingereicht.

Tag der mündlichen Prüfung: 22.Dezember 2009

To Heike and Ronja

Abstract

Geographical data are the core of any Geographical Information System (GIS) and any Geographic Information (GI) application. Because of the increasing use of decentrally held data and networked services, detailed knowledge about the existing data (i.e., its origin, structure, formats, quality, availability and reference applications) becomes more and more important. The availability of such metadata and the evaluation of the fitness for use based on these metadata are vital.

With this thesis the author intents to contribute to the development of meaningful and machine-interpretable quality descriptions of GI. The work focuses on semantic integrity constraints (SIC). In general, integrity constraints define basic assumptions on the part of real world, which is represented by the data. They enable to detect inconsistencies, that is, unacceptable differences between the data and the data model. SICs are defined as specific integrity constraints, whose defined restrictions are based on the semantics of the modelled entities. They reflect business, legal and other required rules and regulations in the database. For spatial data, many SICs 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.

Currently the potential of SICs is far from being exploited and SICs are hardly supported by available GISs or spatial database systems. Their effective use mainly requires a formal description of the constraints that enables to transfer and compare the sets of SICs of different data sources. This thesis contributes to the second requirement. Currently, there is no solution for the comparison of SICs pairs and the detection of any conflicts or redundancies in sets of SICs. This also required the inference of implicit restrictions defined by the SICs. In consequence, the quality assurance of a data set is possibly more extensive than necessary, because sets of SICs might define redundant restrictions, the integration of SICs sets from multiple data sources is impossible and the assessment of the fitness for use based on the SICs cannot be supported. These are significant shortcomings for quality assurance and the knowledge sharing within the frame of spatial data infrastructures.

Three major contributions are elaborated in the thesis: (i) a detailed categorisation of SICs, (ii) a framework for the formal definition of SICs and (iii) a reasoning methodology for the detection of conflicting and redundant SICs.

(i) The classification distinguishes the SICs according to the involved types of spatial and non spatial relation and profoundly differentiates the properties and aspects restricted by spatio-temporal SICs.

(ii) The framework for formal definition of SICs is based on a set of 17 class-level relations. Such qualitative description of cardinality restrictions is novel. The definitions and reasoning rules of the class relations are described independently of concrete spatial or non-spatial relations, what makes them applicable for many types of SICs.

(iii) The introduced reasoning methodology enables for a detection of conflicts and redundancies in sets of SICs, which has hardly been a research topic before. The overall reasoning algorithm is based on the symmetry, composition and conceptual neighbourhood of class relations.

The feasibility of the proposed algorithm has been verified through a prototypical implementation as a plug-in extension of the ontology modelling and knowledge acquisition platform Protégé. Possible application areas are quality assurance of geodata, geodata integration and harmonisation, data modelling and ontology engineering, semantic similarity measurements and usability evaluation.

Contents

Chapter 1

Introduction

Geographical data are the core of any Geographic Information System (GIS) and any Geographic Information (GI) application. Because of the increasing use of decentrally held data and networked services, detailed knowledge about the existing data (i.e., its origin, structure, formats, quality, availability and reference applications) becomes more and more important. The availability of such metadata and the right evaluation of the fitness for use based on these metadata are vital.

In Europe these issues have already led to political consequences: in 2001 the European Commission initiated the Infrastructure for Spatial Information in Europe (INSPIRE) and in 2007 the INSPIRE directive became effective. It aims at the implementation of a European-wide Spatial Data Infrastructure (SDI). Some main principles of INSPIRE are the provision of access to relevant, harmonised and quality GI and the support to seamlessly combining spatial data from different sources for the formulation, implementation and evaluation of EU policies.[1] There is limited value in improving access to and sharing of geospatial data across the Web if the data quality is unknown or assumed to be assured (Sanderson et al., 2009). This makes obvious, that data quality and especially integrity are important aspects to enable such interoperable data exchange among different systems. Also the Open Geospatial Consortium (OGC), as a major standardisation body for GI, has paid attention to this by establishing a new Data Quality Working Group (DQWG) in 2007. Its mission is "to establish a forum for describing an interoperable framework or model for OGC Quality Assurance measures and Web Services to enable access and sharing of high quality geospatial information, improve data analysis and ultimately influence policy decisions."[2]

With this thesis the author intents to contribute to the development of meaningful and machine-interpretable quality descriptions of GI. The work focuses on semantic integrity constraints (SIC), which are sometimes also called user-defined integrity constraints (Codd, 1990). In general, integrity constraints (IC) define basic assumptions on the part of real world, which

[1] http://inspire.jrc.ec.europa.eu/ last visited at August 19th 2009
[2] http://www.opengeospatial.org/projects/groups/dqwg , last visited at August 13th 2009

is represented by the data. They enable to detect inconsistencies that is, unacceptable differences between the data and the data model (Egenhofer, 1997). For database systems ICs are commonly classified according to the conditions they specify into domain constraints, key and relationship constraints and SICs (Elmasri and Navathe, 1994). A SIC is defined as a specific IC, whose defined restrictions are based on the semantics of the modelled entities. SICs can also be seen as descriptions of how the semantics of the modelled concepts are enforced in the data. They reflect business, legal and other required rules and regulations in the database.

Currently the potential of SICs is far from being exploited and SICs are hardly supported by available GISs or spatial database systems (Louwsma et al., 2006). Their effective use mainly requires a formal description of the constraints that enables to transfer and compare the sets of SICs of different data sources. This thesis contributes to the second requirement. Currently there is no solution for the comparison of pairs of SICs and the detection of any conflicts or redundancies in sets of SICs. In consequence, the quality assurance of a data set is possibly more extensive than necessary, because sets of SICs might define redundant restrictions, the integration of sets of SICs from multiple data sources is impossible and the assessment of the fitness for use based on the SICs cannot be supported. These are significant shortcomings for quality assurance and the knowledge sharing within the frame of an SDI.

The following two scenarios illustrate two of the main application areas of the methodologies developed in this work. The resolution of the scenarios is given later on in the thesis.

1.1 Application Scenario 1: Quality Assurance

Efficient maintenance of data integrity has become a critical problem in large databases, since it is an expensive task to check the validity of a large number of ICs after each transaction (Doorn and Rivero, 2002). As part of a quality assurance check, data are proven against ICs to assure the logical consistency of the data set (Becker et al., 1999; Servigne et al., 2000; Pundt, 2002; Mostafavi et al., 2004; Mäs et al., 2005a; Vallieres et al., 2006). Before checking whether a dataset conforms to a set of ICs, it must be assured that the constraints themselves are not in conflict (Bravo and Rodríguez, 2009). If a set of ICs is internally inconsistent, there is no data set that complies with all ICs. Furthermore, a set of ICs might contain redundant restrictions, such that it is unnecessary to prove all constraints to assure consistency.

In the following application example three SICs between the entity classes airport, forest and airport tower are defined for quality assurance.

- All airports and forests are either disjoint or meet (i.e., their boundaries externally touch, otherwise the two geometries do not intersect).

- Every airport tower is contained in an airport (i.e., the airport tower is fully inside the airport and the boundaries of their geometries do not intersect). Every airport contains at least one airport tower.
- All forests are disjoint from all airport towers.

These conceptual definitions are shown in the left part of Figure 1.1 in an Entity Relationship Diagram (ERD). The right part of the figure contains the same diagram, but the third relation has been removed.

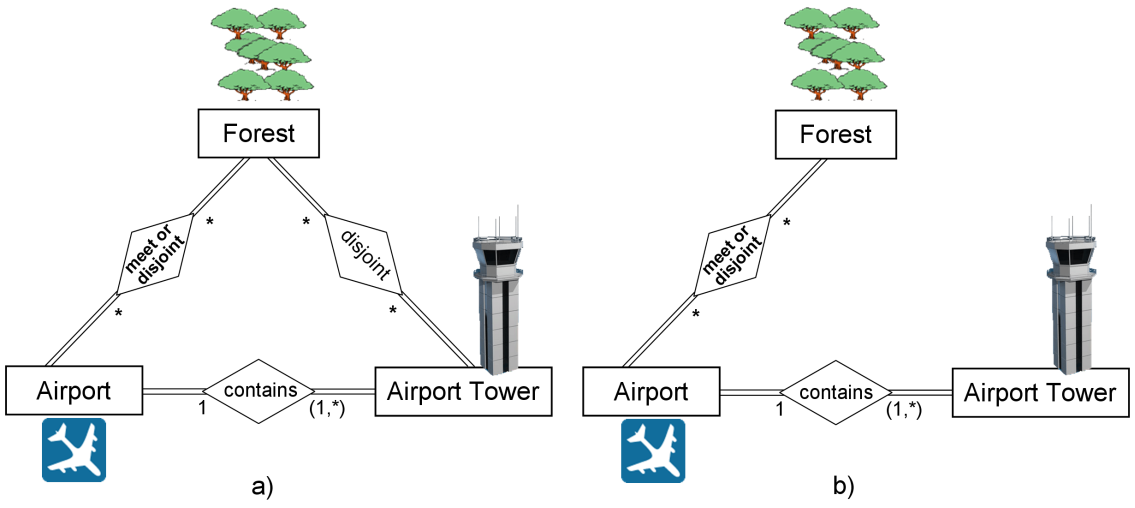


Figure 1.1: Example set of semantic integrity constraints in an ERD

An analysis of the three SICs reveals that the third constraint about the disjointness of forests from airport towers can be deduced from the other two. This means that the total amount of information of both diagrams of Figure 1.1 is the same, despite the removed relation in the right version. The left diagram contains redundant information, because two of the defined SICs implicitly contain the third. A generic and formal description of this deduction is derived in Chapter 5.

For quality assurance this means that the third SIC does not need to be proven if a data set is conform to the first two constraints. This shows that such a detection and removal of redundant constraints enables the reduction of the calculation costs of a quality check. In practice, this can be of great value, since the constraint sets used, for example, by utility companies or public agencies can easily contain hundreds of constraints.

As claimed above, the internal consistency of sets of SICs must be assured. Two constraints that apply to the same elements of a model can specify contradicting restrictions. For a triple of SICs, like in the example constraint set, it is easy to imagine that the third constraint can also be in contradiction to the restrictions implied by the other two.

The aim of this work is to develop a methodology for checking consistency and detecting redundancies in sets of SICs. As the application example shows, this requires not only the check of explicitly defined SICs, but also the disclosure of implicit restrictions.

1.2 Application Scenario 2: Data Harmonisation and Integration

At a recently held workshop on the challenges in geospatial data harmonisation the following definition was given:

> "Data harmonisation is about creating the possibility to combine data from heterogeneous sources into integrated, consistent and unambiguous information products, in a way that is of no concern to the end-user."[3]

Therewith data harmonisation concerns such issues as heterogeneities of spatial reference systems, semantics, terminology, application schemas, metadata and quality descriptions and geometric inconsistencies of adjacent or overlapping datasets. Heterogeneities and conflicts of SICs must also be considered, which is illustrated by the following scenario (Figure 1.2).

A user wants to analyse the flooding risk of a particular area. Therefore he or she needs corresponding information about floodplains and streams. A search in the Internet leads the user to two data providers that offer data for the required region. An integration of the data has many facets. Here the semantics of the entity classes are considered as similar and only the harmonisation of the SICs is taken into account. For this purpose, the user specifies a SIC, which the data must hold, in order to enable the analysis:

- All floodplains must overlap with at least one stream.

To check whether the two data sets conform with this requirement the user downloads information about the SICs, which have been checked during the quality assurance of the data sets. The first data source offers data for the three entity classes floodplain, stream and alluvial forest with the following SICs:

- All alluvial forests must be inside or covered by a floodplain.
- All alluvial forests must meet with at least one stream.

[3]http://www.esdi-humboldt.eu/events/agile2009.html (last visited at 12th June 2009)

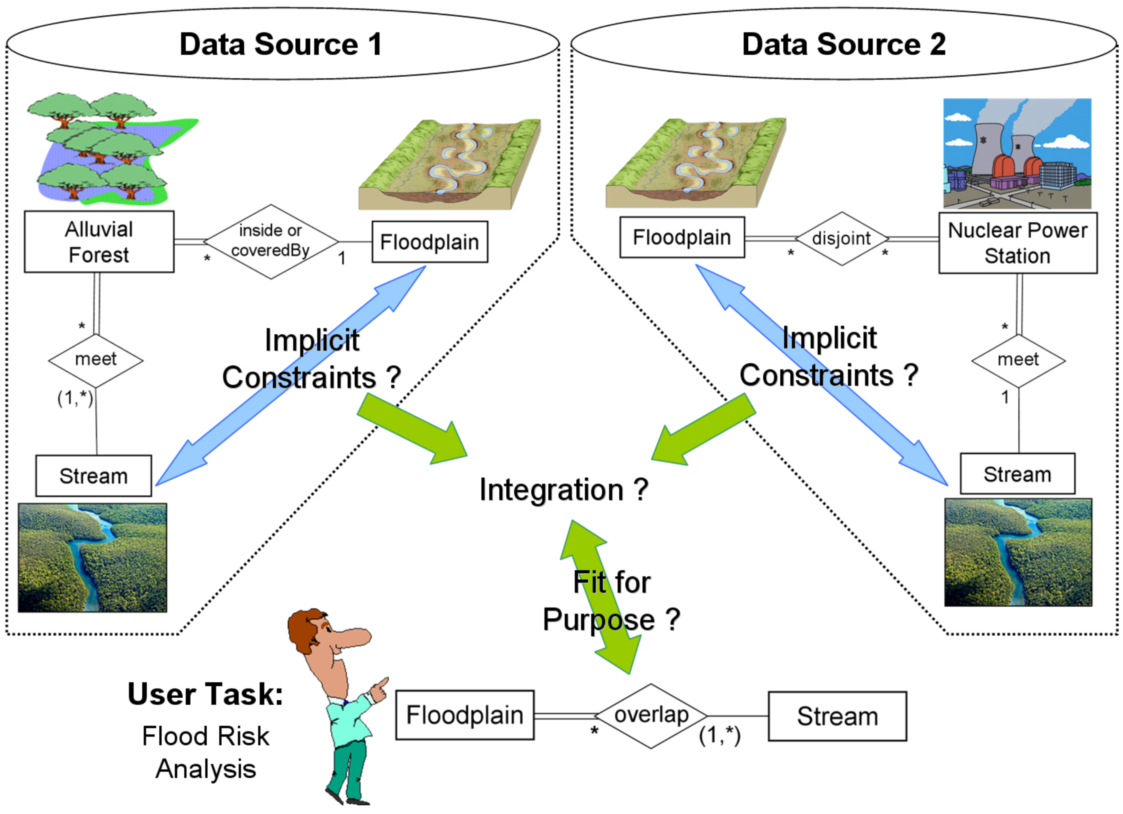


Figure 1.2: Scenario for the harmonisation of spatial semantic integrity constraints

Similar the second data source offers three entity classes with the constraints:

- All nuclear power stations must be disjoint from all floodplains.
- All nuclear power stations must meet with exactly one stream.

In both available data sets the required SIC has not been proven. Obviously the data providers were more interested in the consistency of the alluvial forests or nuclear power stations, respectively, and did not constrain the relation between floodplains and streams. For the user in the scenario this raises the following questions:

1. Are there implicit SICs defined within the two constraint sets?
   Although not as obvious as in the example of the first application scenario, there might be implicit SICs defined between the entity classes floodplain and stream. Answering this question requires again a methodology to deduce the implicitly defined constraints.

2. Is it possible to integrate the two data sets or are there explicit or implicit SICs that contradict each other?

For an integration of the data sets also the two constraint sets must be integrated. Therefore, the implicitly and explicitly defined SICs between the common entity classes of both data sets have to be checked for contradictions and inequalities. Differing SICs between the common classes indicate different quality requirements of the two data sets so that the differing quality of the data sets has to be aligned before data integration. Contradicting SICs between common classes indicate that an integrated data set cannot hold all quality requirements of the two data sets.

3. How does the integrated data fit to the user's SIC and the planned analysis?
   The integrated SICs must be compared with the SIC, which is representing the requirements of the user. This proves whether the data can be used for the intended tasks and whether the data meet the user's quality requirements.

The methodologies for deducing implicitly defined SICs and checking sets of SICs for contradictions, which are developed in this work, support the alignment of the quality of data sets as part of the data integration and harmonisation. Moreover, they enable the evaluation of the integrated data set regarding the requirements, that a specific application demands.

A resolution of this application scenario is given in Section 7.3.

1.3 Objectives of Research

The two application scenarios demonstrate the need of a methodology to compare, manage and integrate SICs. The objective of this work is to develop such a methodology that supports the detection of explicit and implicit redundancies and contradictions and inequalities in sets of SICs. The actual evaluation of geodata against ICs is not part of this work, as this thesis investigates the internal consistency of the ICs. If ICs contradict each other, then the data model itself is inconsistent and it is impossible to acquire a consistent data set.

The restrictions defined by ICs and SICs are multifaceted and not all of them can be considered in this work. Spatio-temporal SICs constrain different aspects of geodata, like for example metric, topology or shape. It is obvious that conflicts and redundancies can particularly occur between constraints that restrict the same aspect. Therefore, this work analyses which spatio-temporal aspects are restricted by SICs. Based on the well-established classification of database ICs according to the involved types of conditions of Elmasri and Navathe (1994), an enhanced categorisation is developed that profoundly distinguishes SICs according to the restricted aspects and the involved spatio-temporal properties and relations.

For the consistency check of sets of SICs, the constraints have to be formally described. As the application scenarios show, such a check must include implicitly defined constraints, because the constraints do not necessarily directly contradict. Conflicts might, for example, result from other constraints defined within a triple of classes. Therefore the formalisation must enable for an inference of implicit constraints. Existing approaches of SIC formalisation neglect this requirement. A major concern of this work is to develop a logically sound formalisation of SICs and, based on this, investigate reasoning methodologies to infer implicit SICs. The clear and unique representation of the cardinality restrictions is a central issue of the required formalisation. The framework for SIC formalisation developed in this work defines SICs as relations among classes (Mäs, 2007b; Tarquini and Clementini, 2008). These class relations base on binary spatio-temporal relations (e.g., meet or disjoint) and a new qualitative description of cardinality restrictions. Both components have an influence on the logical properties of a class relation. The reasoning properties of spatio-temporal relations, such as their symmetry and compositions, are well researched (Cohn and Hazarika, 2001). This thesis investigates the reasoning properties of class relations (that is, SICs) and how these properties can be deduced from the reasoning properties of the spatio-temporal relations and the cardinality restrictions. For this purpose the symmetry, composition and conceptual neighbourhood of the cardinality restrictions are studied. The influence of both class relation components is analysed separately, such that the overall approach is generic and independent of specific spatio-temporal relations. It can be applied to sets of SICs that constrain different aspects of space and time.

A prototypical implementation shall verify the feasibility of the overall approach and demonstrate the use in the application scenarios. The results can help GISs to cope with inconsistencies between different data sources and also inconsistencies of the data with regard to user requirements.

1.4 Outline of the Thesis

This thesis is organised into eight chapters. Following this introduction, the second chapter reviews some fundamentals and the state-of-the-art in the area of spatial data modelling, quality assurance, formalisation of SICs and spatial relations and reasoning. Chapter 3 provides a detailed categorisation of ICs and SICs, which serves as a basis for the further work. Chapters 4 and 5 address the formal definition of spatial SICs, based on class relations and the analysis of their reasoning properties. The focus is here on the symmetry, composition and conceptual neighbourhood of class relations. The results are used for checking the consistency of spatial SICs, which is studied in Chapter 6, including the proof of a Constraint Satisfaction Problem (CSP) at the class level and the detection of restrictions, conflicts and redundancies in pairs of SICs. Chapter 7 illustrates the use of the developed concepts through an application scenario. Here the results of a prototypical implementation are used for demonstration. Chapter 8 concludes this work and gives an outlook on possible future research directions.

Chapter 2

Fundamentals of Spatial Integrity Constraints

This chapter summarises some fundamentals regarding the role of integrity constraints (ICs) in spatial data modelling and spatial data quality description and evaluation. It reviews current research that relates to ICs and their formalisation, spatial relations and spatial reasoning.

2.1 Levels of Data Modelling

Modelling is the process of selecting geographical entities and defining how they shall be represented in a GIS or in a geographical database (Bartelme, 2005). This selection is always depending on the application in mind. Since not all properties of the real world entities can be represented in an application neutral fashion, geodata modelling is always an abstraction process. Commonly, it is subdivided into three abstraction levels (Figure 2.1).

The **conceptual data model** expresses the meaning of required terms and concepts, to identify their characteristic properties and to find the correct interrelationships between different concepts. It defines which and how entities and their properties and relationships are to be acquired (Joos, 1999b). The conceptual schema should represent how domain experts understand the modelled entities, and express the requirements of users and application. Conceptual models are independent of any implementation and database management system (DBMS).

A **conceptual schema** is a formal description of a conceptual model (Elmasri and Navathe, 1994), usually using notations like ERDs or Unified Modeling Language (UML). The International Organization for Standardization (ISO) has specified conceptual schemas for describing the spatial and temporal characteristics of GI in corresponding standards (ISO/TC211, 2002a,b). The application of these standards to build a conceptual schema of a particular application, a so-called **application schema**, is described in (ISO/TC211, 2000). A gen-

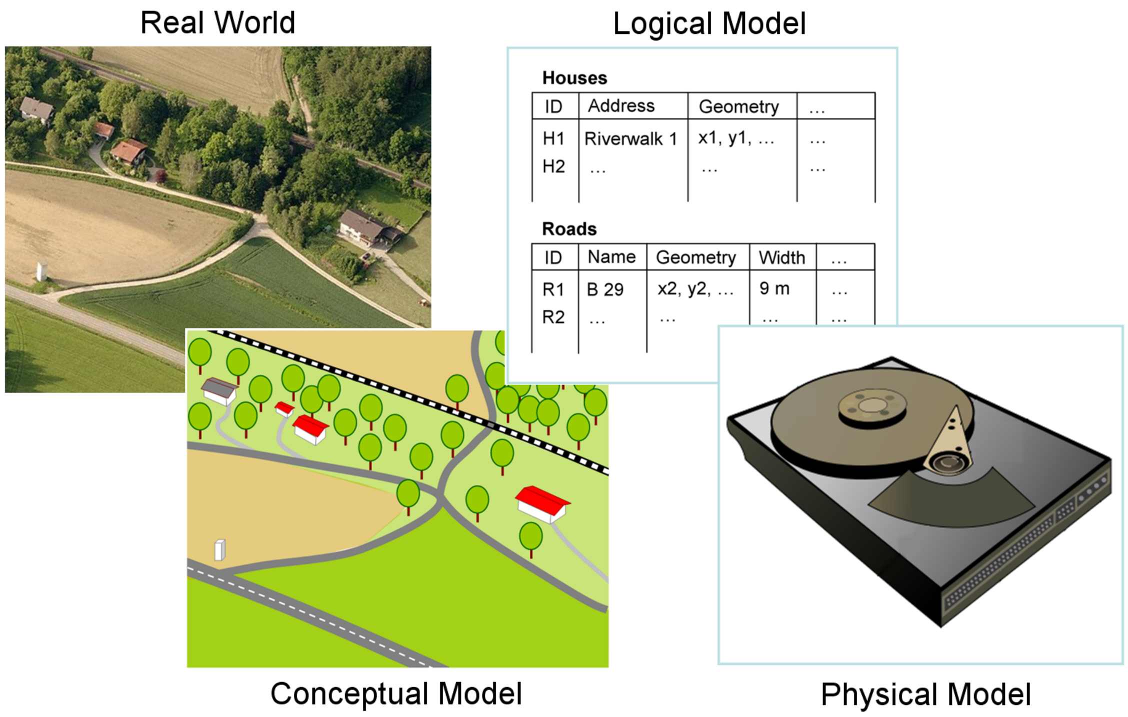

Real World
Logical Model
Houses
ID | Address | Geometry | ...
H1 | Riverwalk 1 | x1, y1, ... | ...
H2 | ... | ... | ...
Roads
ID | Name | Geometry | Width | ...
R1 | B 29 | x2, y2, ... | 9 m | ...
R2 | ... | ... | ... | ...
Conceptual Model
Physical Model

Figure 2.1: The three levels of data modelling

eral introduction into modelling based on the ISO 19xxx standards series has been given by Brodeur and Badard (2008).

The specification of ICs is part of the conceptual data modelling (Borges et al., 1999). ICs and SICs are part of the entity class definitions (Ditt et al., 1997; Friis-Christensen et al., 2001; Oosterom, 2006) and have a corresponding validity. In (Friis-Christensen et al., 2001) the specification of ICs has been pointed out as one of the requirements on notations used for geographic data modelling. Wang (2008) intensively studied the integration of the definition of spatial SICs into the conceptual modelling workflow and conceptual schema descriptions.

The conceptual model is mapped onto the **logical data model**, which provides the foundation for the design of the database. The logical data model describes the logical structure of the data, like, for example, the representation of the modelled relations in tables for relational databases. A logical data model bears on the entities defined in a conceptual model expressed in terms of a particular data management technology.

A **physical data model** as the last data modelling level describes the particular physical mechanisms used to represent data in a storage medium. A physical data model is a single logical model instantiated in a specific database management product. It includes detailed specification of data elements, data types, indexing, etc.

2.2 Quality of Geodata

The information obtained from a GIS rest upon data, which are representing parts of the real world. The reliability of the information and the reliability of decisions people make based on that information depends on the quality of these data. Quality is defined as "the totality of characteristics of a product that bear on its ability to satisfy stated or implied needs" (ISO/TC211, 2002c). The evaluation of data quality, and in particular the evaluation with regard to a particular application, can be very complex. The ISO defines the following five **data quality elements** (ISO/TC211, 2002c) to categorise quality information of GI:

- **completeness:** presence and absence of features, their attributes and relationships;
- **logical consistency:** degree of adherence to logical rules of data structure, attribution and relationships;
- **positional accuracy:** accuracy of the position of features;
- **temporal accuracy:** accuracy of the temporal attributes and temporal relationships of features and
- **thematic accuracy:** accuracy of quantitative attributes and the correctness of non-quantitative attributes and of the classifications of features and their relationships.

The ISO also defines corresponding quality evaluation procedures (ISO/TC211, 2003), quality measures (ISO/TC211, 2005) and metadata elements for the documentation and assessment of the quality of a GI resource (ISO/TC211, 2002d). This work focuses on logical consistency (sometimes also called integrity (Egenhofer, 1997)), which has according to the ISO standard the following subelements:

- conceptual consistency: adherence to rules of the conceptual schema;
- domain consistency: adherence of values to the value domains;
- format consistency: degree to which data is stored in accordance with the physical structure of the dataset and
- topological consistency: correctness of the explicitly encoded topological characteristics of a dataset.

Comparable, logical consistency comprises simple attribute ranges or value rules, basic geometric and topological constraints and specific consistency rules on spatial relationships (Kainz, 1995). ICs are such definitions or rules in data models. They must hold for individual data elements or among several data elements to unambiguously represent the intended semantics of the data model. Therefore, ICs enable the detection of inconsistencies, that is, unacceptable differences between the data and the data model (Egenhofer, 1997). They delimit correctness of a data set. Therefore, ICs are part of the data model and have to be considered during the development of a GIS and the data acquisition (Wang, 2008).

The actual evaluation of geodata against ICs is not part of this work, as this thesis investigates the internal consistency of the ICs. If ICs contradict each other, then the data model itself is inconsistent and it is impossible to acquire a consistent data set.

The consistency between the data model and the reality, that is the **model quality**, cannot be examined by ICs. Model quality describes how appropriately a model represents real-world phenomena (Joos, 1999a). Thus, a model with poor model quality also defines in many cases improper ICs. The evaluation of the model quality can only be made with regard to the specific requirements of an application.

Also the ISO - Technical Committee 211 (ISO-TC211) identifies ICs as one of the concepts used to define types of features. Constraints may be associated with feature types and properties of feature types. As part of an application schema (ISO/TC211, 2000) they are used to prevent users from the creation of erroneous data and thereby assure the integrity of the data. But the standard does not define a particular formal definition of ICs. It suggests that schema developers use a constraint language specific to their implementation environment. The following section reviews some approaches to IC formalisation.

2.3 Formalisation of Integrity Constraints

Nowadays, ICs which cannot be inherently or implicitly expressed in the data model, are usually hardcoded in the application software for the assurance of logical consistency. This particularly concerns SICs. The separation from the data model has significant disadvantages for their transfer, management, adaptability and reuse in different application components or systems (Louwsma et al., 2006; Oosterom, 2006).

A formal description of SICs would overcome these problems and enable the transfer of the integrity rules between different systems and system components. Approaches aiming at a formalisation are for instance based on:

- pictures of inconsistent scenes with corresponding interpretation rules (Pizano et al., 1989),
- first-order logic and topological relations (Hadzilacos and Tryfona, 1992),
- a generic structure based on two related entity classes, a topological relation and a cardinality specification (Ubeda and Egenhofer, 1997; Servigne et al., 2000),

- specifically defined rule languages (Joos, 1999a,b),
- an integration of ICs in a data modelling notation called OMT-G (Borges et al., 1999),
- UML and Object Constraint Language (OCL) with corresponding spatial extensions (Casanova et al., 2000, 2001; Duboisset et al., 2005; Louwsma et al., 2006; Oosterom, 2006; Belussi et al., 2006),
- UML in combination with so-called constraint decision tables (Wang and Reinhardt, 2007; Wang, 2008),
- semantic web languages like
  - Resource Description Framework (RDF) (Pundt, 2002),
  - formal ontology languages (Mostafavi et al., 2004),
  - rule languages such as Semantic Web Rule Language (SWRL) (Mäs et al., 2005b; Watson, 2007) or Rule Markup Language (RuleML) (Wang and Reinhardt, 2007),
- spatial relations between classes of individuals (so-called **class relations**) (Tarquini and Clementini, 2008),
- first-order logic predicates expressed in an Extensible Markup Language (XML) based rule language called Spatial QUality and Integration Rules Language (SQUIRL) (Sanderson et al., 2009) and
- an extended relational database model (Bravo and Rodríguez, 2009).

A more extensive overview of existing approaches to the formalisation of ICs has been given by Werder (2009). A general comparison of specification languages, which also includes natural languages, was made by Salehi et al. (2007).

These approaches to SIC formalisation aim at the definition and transfer of the constraints and an automatic setup of the quality assurance. For checking the internal consistency of SICs, their logical properties, and in particular the correlation with the logical properties of the spatial relations, have to be researched. This is not included in most of the approaches, although some of them mention the check of conflicts in sets of constraints as a future requirement (Louwsma et al., 2006; Oosterom, 2006). Only Bravo and Rodríguez (2009) investigated the consistency of spatial SICs. They found that for their formalisation the consistency check is computationally not tractable.

This work also requires a formalisation of SICs, but the focus is on the investigation of the logical properties of the constraints. It typically uses class relations, originally introduced by Donnelly and Bittner (2005), for SIC formalisation (Chapter 4).

2.4 Spatio-temporal Relations and Reasoning

A critical foundation of SICs is the formalisation of spatio-temporal relations and their logical inferences. This section provides an overview on spatio-temporal relations and reasoning techniques.

2.4.1 Spatial and Temporal Relations

The representation of space and time in a GIS can be manifold. A generic differentiation can be into quantitative and qualitative representation approaches (Hernández, 1994). The former rely on quantitative measurements or calculations, whereas qualitative representations are non-numerical descriptions of a situation. Typically, continuous properties are represented by discrete sets of relations of and between spatial entities. Qualitative representations are characterised by making only as many distinctions in the domain as necessary in a given context. Thus, qualitative descriptions are less precise than quantitative ones. For the definition of SICs this abstraction is in many cases sufficient and preferable, since verbal descriptions of SICs are usually also not numerically precise. Some of the advantages of qualitative over quantitative representations are their closeness to human cognition, communication and reasoning, their ability to deal with incomplete knowledge and, depending on the application, their computational efficiency (Hernández, 1994; Renz, 2002). This particularly holds for spatial relations of and between spatial entities (in this thesis also called **instance relations**). Typical examples are "*a* is inside *b*", "*c* is north of *d*" and "*e* is longer than *f*".

The different aspects of space, like topology, orientation, distance and shape, are usually represented by different spatial relations. A qualitative representation of one of these aspects is based on a jointly exhaustive and pairwise disjoint (JEPD) set of relations. This means that for all possible situations exactly one of the relations holds. Because of the qualitativeness, the number of possible relations must be finite.

The focus in this work is on **binary spatial relations**, that is, spatial relations between two entities, although other relations, such as unary or ternary relations, are also used for the definition of SICs. A typical example of a JEPD set of such relations is the topological relations between areal entities (Figure 2.2).

Further examples of spatial and temporal relations can be found for temporal intervals (Allen, 1983; Freksa, 1992a; Hornsby et al., 1999), for topological relations (Egenhofer and Herring, 1990; Kurata, 2008), for directional relations (Freksa, 1992b; Frank, 1992; Goyal and Egenhofer, 2001; Skiadopoulos and Koubarakis, 2001; Yan et al., 2006), metric relations (Frank,

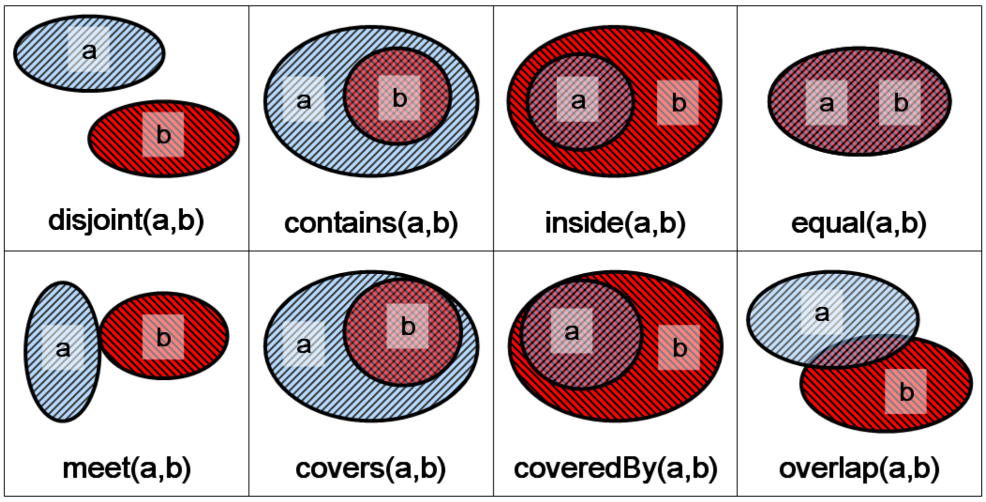


Figure 2.2: Set of topological relations between areal entities (Egenhofer and Herring, 1990)

1992), and shape relations (Cohn, 1995; Clementini and Felice, 1997). Cohn and Hazarika (2001) gave an extensive overview on spatial relations and their reasoning properties.

A qualitative set of relations only makes as many distinctions as necessary. This enables the coverage of the corresponding aspect of space with the desired granularity (Renz, 2002). Depending on the spatial situation and the acceptable precision, it is possible to use sets of relations with different resolutions. For example, in comparison to the set of relations shown in Figure 2.2, Grigni et al. (1995) defined sets of topological relations with different resolutions.

For the further work with spatial relations, the following operations are defined:

- r^i : converse/symmetric instance relation of r, for instance
  $inside^i(a,b) \Rightarrow contains(b,a)$.
- $r1 \cup r2$: union / disjunction of the two instance relations, for instance if
  $r1 := meet(a,b)$ and $r2 := equal(a,b)$ then
  $r1 \cup r2$ corresponds to either $meet(a,b)$ or $equal(a,b)$ (logical XOR).
- $r1 \cap r2$: intersection of the two instance relations, for instance if
  $r1 := meet(a,b)$ and $r2 := covers(b,c)$ then
  $r1 \cap r2$ corresponds to $meet(a,b)$ and $covers(b,c)$.

For further operations and mathematically precise definitions see Maddux (2006).

The formalisation framework for SICs, developed in this work (Chapter 4), bases on some specific properties of binary relations (Ebbinghaus, 1977; Maddux, 2006):

left-total:	for all x in set X there exists a y in set Y such that $r(x, y)$ $\forall x \in X : \exists y \in Y \cap r(x, y)$
right-total or surjective:	for all y in set Y there exists an x in set X such that $r(x, y)$ $\forall y \in Y : \exists x \in X \cap r(x, y)$
left-definite or injective:	for all x and z in set X and y in set Y it holds that if $r(x, y)$ and $r(z, y)$ then x = z $\forall x, z \in X, \forall y \in Y : (r(x, y) \cap r(z, y)) \Rightarrow x = z$
right-definite:	for all x in set X and y and z in set Y it holds that if $r(x, y)$ and $r(x, z)$ then $y = z$ $\forall x \in X, \forall y, z \in Y : (r(x, y) \cap r(x, z)) \Rightarrow y = z$
bijective:	left-total, right-total, left-definite and right-definite

The following subsections discuss reasoning techniques and properties of spatial and temporal relations.

2.4.2 Composition of Spatio-temporal Relations

The composition is an operation of relation algebra. It is also called the relative product (Tarski, 1941). The composition of binary relations enables the derivation of implicit knowledge about a triple of entities. If two binary relations are known, the corresponding third relation can potentially be inferred or some of the possible relations can be excluded. In this work, the following symbology is used for the composition:

$r1 \ ; r2 \Rightarrow r3$	The composition of $r1$ and $r2$ implies $r3$.

If the composition imposes no constraints, it results in a universal disjunction $\mathcal{U}$, which is the disjunction of all relations of the corresponding set.

$\mathcal{U}$	Universal disjunction of the corresponding set of relations.

Figure 2.3 provides a composition example for the topological relations between three regions. Two topological relations between the three regions can describe the scene shown on the left

of the figure: "*a meets b*" and "*b contains c*". Although the third relation is missing and the other two topological relations only provide an imprecise qualitative description of the scene, it is possible to deduce the third relation "*a* is *disjoint* from *c*".

$$meet\ ;\ contains \Rightarrow disjoint$$

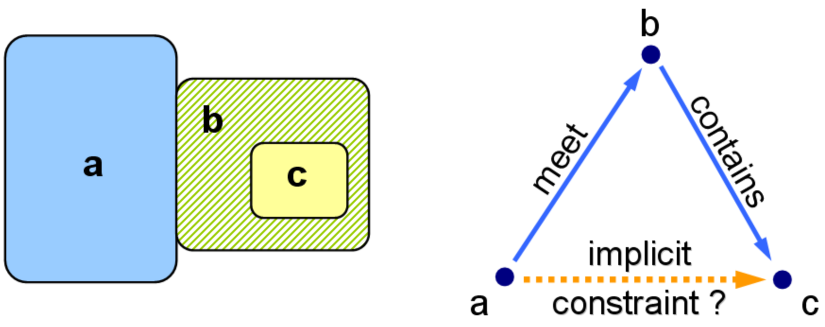


Figure 2.3: Example scene of three regions (left) and a corresponding composition of two topological relations (right)

Such a derivation can also be used to discover conflicts in case that all three relations are known. The composition rules of a set of relations are usually represented in a **composition table** (sometimes also called transitivity table) like the one compiled for the topological relations between areal entities in (Egenhofer, 1994)(Table 2.1). The examples throughout this work base on this composition table.

Many other sets of binary spatial relations also allow for such derivations. Further composition tables can be found for topological relations (Hernández, 1994; Grigni et al., 1995; Kurata and Egenhofer, 2006), directional/orientation relations (Frank, 1992; Freksa, 1992b; Hernández, 1994; Papadias and Egenhofer, 1997), distance relations (Hernández et al., 1995) and temporal relations (Allen, 1983; Freksa, 1992a).

2.4.3 Constraint Satisfaction Problems

The composition of relations can also be used to check the logical consistency in relation sets with more than three related entities. Such a network of binary relations can be expressed by a graph, in which the nodes represent the entities and the arcs the relations. Figure 2.4 shows an example graph of the topological relations of four regions. Such a graph does not need to be fully determined; it can also represent incomplete knowledge if some relations are unknown or not precisely known.

Table 2.1: Composition table of the eight topological relation between areal entities (d = disjoint, m = meet, e = equal, i = inside, cB = coveredBy, ct = contains, cv = covers and o = overlap)(Egenhofer, 1994)

	disjoint (b,c)	meet (b,c)	equal (b,c)	inside (b,c)	covered-By(b,c)	contains (b,c)	covers (b,c)	overlap (b,c)
disjoint (a,b)	$d \vee m \vee e \vee i \vee cB \vee ct \vee cv \vee o$	$d \vee m \vee i \vee cB \vee o$	d	$d \vee m \vee i \vee cB \vee o$	$d \vee m \vee i \vee cB \vee o$	d	d	$d \vee m \vee i \vee cB \vee o$
meet (a,b)	$d \vee m \vee ct \vee cv \vee o$	$d \vee m \vee e \vee cB \vee cv \vee o$	m	$i \vee cB \vee o$	$m \vee i \vee cB \vee o$	d	$d \vee m$	$d \vee m \vee i \vee cB \vee o$
equal (a,b)	d	m	e	i	cB	ct	cv	o
inside (a,b)	d	d	i	i	i	$d \vee m \vee e \vee i \vee cB \vee ct \vee cv \vee o$	$d \vee m \vee i \vee cB \vee o$	$d \vee m \vee i \vee cB \vee o$
covered-By(a,b)	d	$d \vee m$	cB	i	$i \vee cB$	$d \vee m \vee ct \vee cv \vee o$	$d \vee m \vee e \vee cB \vee cv \vee o$	$d \vee m \vee i \vee cB \vee o$
contains (a,b)	$d \vee m \vee ct \vee cv \vee o$	$ct \vee cv \vee o$	ct	$e \vee i \vee cB \vee ct \vee cv \vee o$	$ct \vee cv \vee o$	ct	ct	$ct \vee cv \vee o$
covers (a,b)	$d \vee m \vee ct \vee cv \vee o$	$m \vee ct \vee cv \vee o$	cv	$i \vee cB \vee o$	$e \vee cB \vee cv \vee o$	ct	$ct \vee cv$	$ct \vee cv \vee o$
overlap (a,b)	$d \vee m \vee ct \vee cv \vee o$	$d \vee m \vee ct \vee cv \vee o$	o	$i \vee cB \vee o$	$i \vee cB \vee o$	$d \vee m \vee ct \vee cv \vee o$	$d \vee m \vee ct \vee cv \vee o$	$d \vee m \vee e \vee i \vee cB \vee ct \vee cv \vee o$

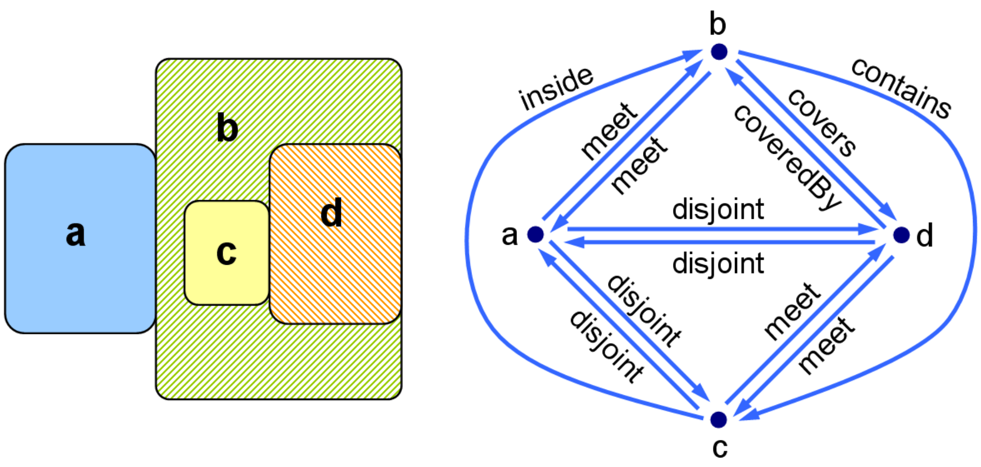


Figure 2.4: Example scene of four regions (left) and the corresponding topological relation graph (right)

The proof of consistency of such a graph is a Constraint Satisfaction Problem (CSP). CSPs are mathematical problems that are defined by those entities, that are restricted by a number of constraints. Formally, it is defined by a set of variables, the domain of values for these variables and a set of constraints, which are restricting a subset of the variables. These constraints define what combinations of values for the variables are valid. When reasoning on spatial relations, the variables represent the relations between the spatial entities, the domain of values is the corresponding set of relations (e.g., the eight topological relations between areal entities) and the constraints are defined by the logical properties of the relation set (symmetry and composition rules).

In a consistent network of JEPD relations, the following three requirements are fulfilled (Rodríguez et al., 2004):

- **Node consistency** is ensured if every node has a self-loop arc, which corresponds to the identity relation (i.e., relation of an entity to itself).
- **Arc consistency** is ensured if every edge of the network has an edge in the reverse direction, that is, every relation has a converse binary relation.
- **Path consistency** is ensured if all relations are consistent with their induced relations, determined by the corresponding intersection of all possible composition paths of length two (n = number of nodes):

$$\forall_{i,j}\, r_{i,j} = \bigcap_{k=1}^{n} r_{ik}\,;\, r_{kj} \tag{2.1}$$

Since sets of binary spatial relation usually have a defined identity relation (for example *equal* for the topological relations of Figure 2.2) and converse relations, the proof of node consistency and arc consistency is relatively simple. Node and arc consistency are prerequisites for checking path consistency and the corresponding identity and converse relations must be added to the relation network.

Many algorithms that check path consistency have been proposed (Allen, 1983; Mackworth and Freuder, 1985; Papadias and Egenhofer, 1997). The following path consistency algorithm is a slightly adjusted version of the one published by Hernández (1994), which is based on Allen's algorithm. It consists of three parts:

```
To ADD $r_{ij}$
  begin
    Old $\leftarrow$ N(i,j);
    N(i,j) $\leftarrow$ N(i,j) $\cap$ $r_{ij}$;
    If (N(i,j) = $\varnothing$)
      then Inconsistency found;
    If (N(i,j) $\neq$ Old)
      then put pair <i,j> on Queue;
    Nodes $\leftarrow$ Nodes $\cup$ $\{$i,j$\}$;
  end;
```

The first part adds the new relation r_{ij} of the entities i and j to the set of relations N, which contains all known relations. If it is the first inserted relation, the set of relations contains universal disjunctions for all possible relations. If the intersection between the new relation and the known set of relations is empty a contradiction / inconsistency between the new and the known relations is detected. If the intersection results in a more constrained set of relations, the pair of nodes i, j (i.e., pair of entities) is put to the queue for propagation.

```
To CHECKCONSISTENCY
  While Queue is not empty do
    begin
      Get next <i,j> from Queue;
      PROPAGATE(i,j);
    end;
```

The consistency check of the relation set is then started by the CHECKCONSISTENCY procedure, which calls the following procedure as long as there are entries in the queue.

```
To PROPAGATE i,j
  begin
    For each node k do
    . begin
    .   New $\leftarrow$ N(i,k) $\cap$ (N(i,j);N(j,k));
    .   If (New = $\varnothing$)
    .     then Inconsistency found;
    .   If (New $\neq$ N(i,k))
    .     then add <i,k> to Queue;
    .   N(i,k) $\leftarrow$ New;
    . end;
    For each node k do
    . begin
    .   New $\leftarrow$ N(k,j) $\cap$ (N(k,i);N(i,j));
    .   If (New = $\varnothing$)
    .     then Inconsistency found;
    .   If (New $\neq$ N(k,j))
    .     then add <k,j> to Queue;
    .   N(k,j) $\leftarrow$ New;
    . end;
  end;
```

The PROPAGATE procedure determines, whether the new relation between i and j effects the results of the compositions in the network. It proves if the compositions of the new relation with known relations (and vice versa) constrain the relation between i and other nodes k, or relations between other nodes k and j. In these cases the constrained pair of nodes i, k or k, j is placed in the queue for further propagation. An inconsistency is detected if the intersection of the composition and the known relations of nodes i, k or k, j results in an empty set. Hernández (1994) also investigated algorithms for the resolution of such contradictions (so-called constraint relaxation algorithms) and for the deletion of relations from the network.

The elucidated algorithm can also be used if the knowledge is incomplete, that is, not all relations of the network are known. In general, CSP algorithms can be used to infer implicit knowledge and to check consistency in constraint networks, but also to detect redundancies allowing for data compression and minimisation of the number of constraints in database queries. The application of CSP algorithms for checking consistency in relation networks has been demonstrated for topological relations (Egenhofer and Sharma, 1993; Grigni et al., 1995; Hernández, 1994), directional/orientation relations (Papadias and Egenhofer, 1997; Hernández, 1994) and temporal relations (Allen, 1983; Rodríguez et al., 2004).

CSP algorithms are not infallible, however. Some relation networks are unsatisfiable and therefore inconsistent, even if its correctness has been proven with a CSP. This is due to the restricted expressive power and precision of qualitative relations. Grigni et al. (1995) showed a network of topological relations of areal entities that is consistent with respect to the composition inferences and the CSP, but there is no set of regions that realise it, because of restrictions related to planarity. However, the CSP satisfiability answer may be false positive, but never false negative.

2.4.4 Conceptual Neighbourhood

The notion of conceptual neighbourhood has been introduced by Freksa (1992a). In this work, the definition provided by Cohn et al. (1998) is used:

> "Two relations drawn from a pairwise exclusive and jointly exhaustive set can be called immediate conceptual neighbours if one can be transformed into the other by a process of gradual, continuous change that does not involve passage through any third relations."

In other words: "a pair of two relations $r1$ and $r2$ are conceptual neighbours if it is possible for $r1$ to hold at a certain time, and $r2$ to hold later, with no third relation holding in between"(Cohn, 2008).

The conceptual neighbourhoods of relations of a JEPD set are usually represented by a conceptual neighbourhood graph, like the one shown in Figure 2.5 for the set of topological relations between areal entities (Figure 2.2).

Further examples of conceptual neighbourhood networks of spatio-temporal relations can be found for temporal interval relations (Freksa, 1992a; Hornsby et al., 1999), for topological relations between regions (Egenhofer and Al-Taha, 1992), between regions and lines (Egenhofer and Mark, 1995), and between directed lines (Kurata and Egenhofer, 2006).

Knowledge about the conceptual neighbourhood of instance relations can be used to evaluate the possible changes of a relation or to measure the similarity among the relations between differing relation sets (Schwering, 2007).

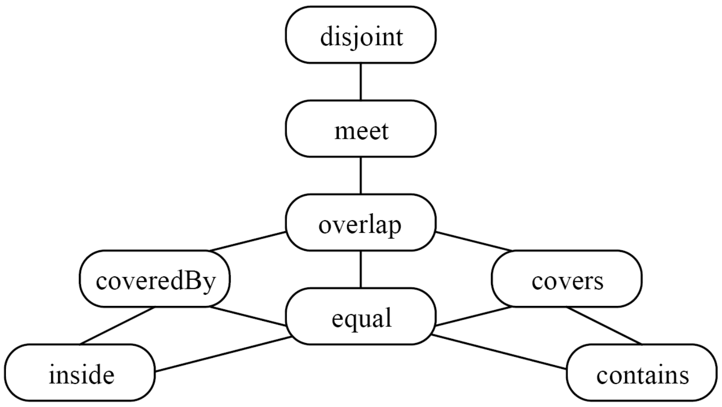


Figure 2.5: Conceptual neighbourhood graph of the eight topological relations between areal entities (Egenhofer and Al-Taha, 1992)

Chapter 3

Categories of Integrity Constraints

Integrity constraints (ICs) formalise basic assumptions on that part of the world that is represented by the data (Gröger and Plümer, 1997). ICs constitute the conceptual consistency, that is, the consistency of the data with respect to the conceptual data schema. Wang (2008) provides the following definition for spatial ICs, which is also used in this work:

Spatial integrity constraints are formal and accepted statements, definitions or qualifications for describing data consistency requirements in order to constrain spatial data to represent the reality in the context of a GIS application correctly.

This chapter reviews established approaches to IC classification and provides a definition of semantic integrity constraints (SICs). Furthermore a profound categorisation is proposed, which classifies SICs according to the semantic domains of the properties involved. This categorisation provides the preconditions for the formal definition of the constraints and the identification of conflicts and redundancies in sets of ICs.

3.1 Approaches to Categorisation of Integrity Constraints

The following section reviews existing approaches to categorising ICs, starting with general constraint categories in database systems and proceeding with the specific constraint types for spatio-temporal data.

3.1.1 General Integrity Constraints for Database Systems

Some basic categories of ICs for data modelling are fundamental to database systems (Elmasri and Navathe, 1994). They classify ICs according to their specification technique, the type of conditions they specify and the number of database states they constrain.

3.1.1.1 Specification Technique of Integrity Constraints

The specification of ICs in a database system can be inherent, implicit or explicit. **Inherent constraints** are directly associated with the constructs of the data model itself and do not need to be specified in the data schema. They specify rules that define the inherent properties of the data model constructs. Likewise as inherent constraints, **Implicit constraints** are contained in the database schema, but they are specified by the Data Definition Language (DDL) through the database schema definition. Implicit constraints restrict each entity type, attribute and relationship through the specification possibilities implied by the particular schema DDL. An example is the uniqueness constraint that is forced on an attribute when it is specified as a key of an entity type. More complex constraints, that are not expressed by the DDL and therefore have to be additionally specified, are called **explicit constraints**. Typical examples are the general semantic integrity constraints.

3.1.1.2 Specified Conditions of Integrity Constraints

A second classification identifies the following types of ICs applying to the specified conditions (Elmasri and Navathe, 1994):

- **domain constraints** restrict the allowed types of values of an attribute, for example Integer or Boolean.
- **key and relationship constraints** refer to the possibility to define key values (i.e., unique values) for entity classes, cardinality restrictions for relationships between entities and participation requirements.
- **general semantic integrity constraints** are explicitly specified and usually more complex. They refer to semantics of the modelled entity classes, which are not representable through the other two categories. They specify relations between the modelled concepts, that are usually not explicitly represented in the data.

3.1.1.3 Number of Constrained Database States

Elmasri and Navathe (1994) also distinguish ICs based on whether they restrict a single database state (**state constraints**) or multiple states (**transition constraints**). Thereby one database state includes all data of the database at a particular point in time. State constraints assure the consistency of a database before and after an update transaction. Examples are domain and key constraints. Transition constraints are restrictions on the changes between successive database states. The corresponding integrity checks analyse different versions of entities, their attributes and relations. For example, a transition constraint can assure that the value of an attribute is not decreased during a database update.

3.1.1.4 Data Model Elements Restricted by Integrity Constraints

A fourth approach was made by Ditt et al. (1997) and later on extended by Friis-Christensen et al. (2001). They classified ICs according to the involved elements of data models:

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

This classification is particularly useful to differentiate the restrictions of SICs.

3.1.2 Spatial Data Integrity Constraints

The categorisations listed so far do not address the particularities of spatial data. These specific properties allow for spatial analysis methods that can also be used for integrity checking and, therefore, enable the definition of spatial ICs. These constraints particularly deal with things like location, extent, shape or topology. Approaches to classify spatial ICs in order to comprehend the peculiarities of spatial data can be found in (Servigne et al. (2000); Cockcroft (1997, 2004); Borges et al. (1999); Frank (2001) and Louwsma et al. (2006)).

Servigne et al. (2000) define three kinds of spatial ICs that apply to structural, geometric and topo-semantic conditions:

Structural errors result from an insufficient implementation of the data model through inadequate data structures provided by the GIS. To overcome such shortcomings **structural constraints** have to be defined. Structural constraints are “programming tricks” used to handle entities that cannot be appropriately represented by the available data structures. In the categorisation following in the next section, structural errors are not considered, since it is assumed that the data structures sufficiently represent the data model.

Geometric constraints refer to the general geometric and topological assumptions of the geometry types of the data model. They define and restrict properties and relations of geometric and topological primitives independently of the semantics of specific entity classes. Common examples of geometric ICs are the closedness of polygons:

"All Polygons must be closed."

and the forbidden intersection of edges in a planar graph network:

"The edges of a graph network are not allowed to share any point, except their incident nodes."

or more general conditions for the primitives of a geometric complex according to ISO/TC211 (2002a):

"The interiors of the geometric primitives must be disjoint. If two primitives touch, the common boundary must be a primitive of lower dimension."

Such constraints can be used, for example, to assure a redundancy-free topology. Corresponding computable checking algorithms are based on axiomatic characterisations of the geometry types and apply graph theory. Plümer and Gröger (1997) define a set of geometric ICs for areal features. This example shows the optimal case, where a minimal and complete set of ICs can be defined for a specific geometry type. A set of ICs is minimal if no subset exists, that is also ensuring the integrity, that is, there is no redundancy in the set of constraints. A set of ICs is complete for a certain concept (in this case of the concept of a geometry type) if it logically implies this concept.

The **topo-semantic constraints** of Servigne et al. (2000) refer to topological relations between two entities. Since the validity of the topological relation depends on the semantic of the entities, topo-semantic constraints are a subtype of the general SICs (Elmasri and Navathe, 1994). Ubeda and Egenhofer (1997) gave a generic structure for the definition of topological ICs. They define a topological constraint as an association of two geographical entity classes, a binary relation, which is in this case a topological relation such as *touch* or *intersect*, and a specification, which can have one of the following values:

- forbidden (i.e., relation is not allowed),
- at least n times (i.e., relation is n times required),
- at most n times (i.e., relation is n times allowed), or
- exactly n times (i.e., relation is exactly n times required).

Such a structure cannot cover all possible kinds of topological ICs. As the classification according to the involved data model elements (Section 3.1.1.4) demonstrates, ICs can refer to more than two entity classes as well as their attributes and operations.

Cockcroft (2004) subdivides topo-semantic constraints into semantic and user-defined constraints. The former are based on the nature and the physics of the objects, for instance “roads are not running through lakes”. User-defined ICs are of more artificial nature and describe social or business rules or laws defined by humans, for instance “a fuel station should not be within a certain distance of a school.” A more extensive classification of constraint background or contexts can be found in Frank (2001). He demonstrated how ICs are part of GIS ontology and showed that different constraints are appropriate to different tiers of ontology. This points out the importance to treat the constraints not independently of the context for which they are valid. Nevertheless, if a set of ICs is checked according to their internal consistency, all constraints have to be considered independently of their context. Thus the context is understood as metainformation of the constraints and not further considered in this work.

3.2 A Refined Categorisation of Spatio-temporal Integrity Constraints

These IC categorisations are either not practical for constraint formalisation and validation, as they leave out spatial aspects, or they do only cover some spatial aspects like topology. A refined categorisation (Figure 3.1), which particularly incorporates the different aspects of spatio-temporal data integrity, distinguishes the ICs according to the involved types of conditions and profoundly differentiates the aspects of spatial SICs (Mäs and Reinhardt, 2009).

In the scope of this work, this differentiation of SICs is necessary, because the developed reasoning concepts are based on the reasoning properties of the relation sets of each spatio-temporal aspect. The consistency of the constraints has to be separately checked for the constraints of each of these SIC classes. In general, the categorisation provides a basis for further work on SIC formalisation, management and validation. It points out the properties that can be constrained through the integrity rules and, thereby, identifies aspects, where presently the potential of the constraints is by far not yet exploited. The new concepts extend the well-established classification of database ICs (Elmasri and Navathe, 1994). The top level of Figure 3.1 contains these three basic types of ICs. The middle level distinguishes SICs according to the involved properties and relations. The bottom level regards to the spatial

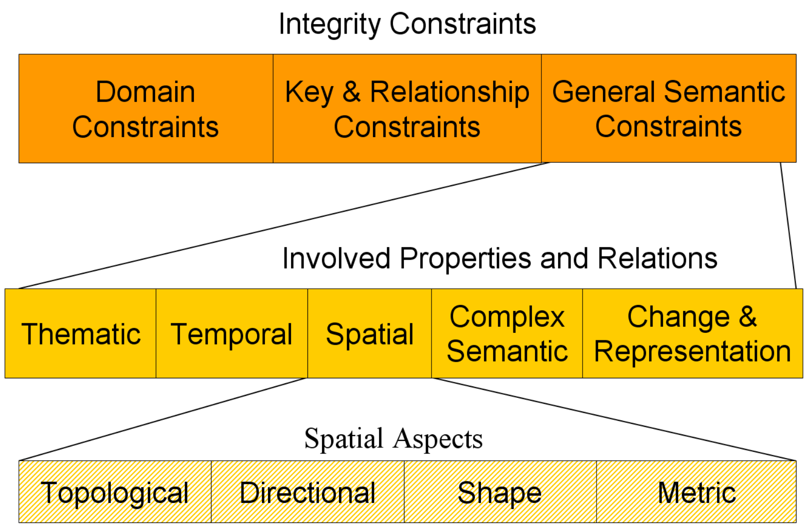


Figure 3.1: Categorisation of integrity constraints (Mäs and Reinhardt, 2009)

aspects that are restricted by spatial SICs. The following subsections provide definitions and examples of the defined IC classes, with a particular focus on ICs required by spatial data.

3.2.1 Domain Constraints

Domain constraints restrict the allowed types of values of an attribute. The value domain of an attribute is usually defined by a data type, such as a numerical data type (integer or real), character and string data type, Boolean data type, date and time data type or enumerated data type (Elmasri and Navathe, 1994). Database systems and schema description languages commonly include these data types and have the corresponding ICs inherently defined. Moreover, some applications require so-called user-defined data types, which are defined by the designers of the (database) schema.

Spatial and temporal information require data types and corresponding domain constraints, which restrict all kinds of defined primitives and complexes like for example geometric / topological and temporal primitives / complexes. The primitives and their corresponding ICs are defined independently of the semantics of entity classes. Currently, a variety of database systems is already capable of handling the particular requirements of spatial data and provides predefined spatio-temporal data types. For these systems, the corresponding ICs do not have to be explicitly specified. Domain constraints on geometric and topological primitives correspond to the geometric constraints defined by Servigne et al. (2000), which have been itemised in the previous section. International standards (e.g., ISO/TC211 (2002a, 2004)) specify names and geometric definitions for geometry types and geometric primitives. Since the constraints are part of the geometry type definition of the data model, they are also included in these standards.

Temporal primitives also define domain constraints, but since time is unidirectional, temporal primitives are less complex than the potential geometrical primitives of the two or three-dimensional space. For many applications, the interval type is the only temporal primitive beside the common date and time data types. A simple constraint like:

"the beginning of an event (i.e., interval) must be before the ending"

is the only internal assumption for time intervals and it sufficiently assures their integrity. Temporal primitives are also standardised by ISO/TC211 (2002b).

3.2.2 Key and Relationship Constraints

For the definition of key and relationship constraints Mäs and Reinhardt (2009) also refer to (Elmasri and Navathe, 1994). These constraints are mostly incorporated in the schema of the database and, therefore, do not need to be additionally specified. Examples are cardinality restrictions of associations. Since spatial and temporal information does not have particular requirements on key and relationship constraints they are not further researched in this work.

3.2.3 General Semantic Integrity Constraints

Following the definition of Elmasri and Navathe (1994) general SICs 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. SICs are defined on the level of the entity classes and have to be explicitly specified. A further subdivision of SICs can be made by grouping the restricted properties according to their semantic domain. Extending the approaches of Servigne et al. (2000) and Cockcroft (2004) the proposed categorisation (Figure 3.1) does not only consider topological relations. It includes thematic, temporal, spatial and change relations as well as relations between multiple representations of an entity. Additionally it defines complex SICs, which combine relations of more than one of the semantic domains, mereological relations and domain specific associations. Furthermore, constraints are not restricted to only two entity classes. SICs allow for the definition of restrictions affecting single or multiple entity classes in combination with their attributes and relations in a single constraint. Subsequently, descriptions and examples of the subtypes of SICs and references to relations and properties appropriate for the constraint definition are provided.

3.2.3.1 Thematic SICs

Thematic SICs secure the consistency of thematic attributes. They restrict the ranges of attributes of a single entity by specifying relations between the values of two or more attributes or one attribute value and a defined other value, for example, for a road entity class with the attributes *road_type* and *number_of_driving_lanes*:

"Roads of the type 'Autobahn' must have at least four driving lanes."

The applied comparison operators for the specification of SICs are well established and have been used in standards (OGC, 2005). Beside the common order relations $(=, <, >, \leq, \geq, \neq)$, they include operators for String and NULL comparison as well as fundamental arithmetic operators for addition, subtraction.

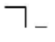
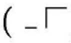
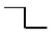
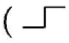
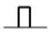
3.2.3.2 Temporal SICs

To assure the logical consistency of temporal information temporal SICs can be defined. These constraints substantiate rules that apply to temporal characteristics of the data. A simple example is the definition of a minimal duration of a particular time interval or event. More complex temporal SICs refer to the relation of particular points in time or time intervals like events, processes, states or actions. An example for an entity class bridge with the temporal attributes *end_of_construction_time* and *official_opening_date* is:

"The construction time of a bridge must end before the date of the official opening."

Allen (1983) defines a JEPD set of 13 binary relations (Table 3.1) between time intervals, which can be used for the definition of such constraints. The intervals represent events through an ordered pair of points in time, with the first point being earlier on the time scale than the second (domain constraint). Every relation of two intervals can then be determined by not more than two relations between the start- and endpoints (I_n- and I_n+) of the involved intervals (I_1 and I_2). This set of temporal relations is widely used in different domains and it is also incorporated into the ISO/TC211 (2002b) standard. Beside intervals, the standard also includes instants (temporal points) as primitives and, due to that, the validity of the relations is adequately adjusted. For cyclic time intervals, Hornsby et al. (1999) defined a corresponding JEPD set of 16 binary relations.

Table 3.1: Allens 13 binary relations between time intervals without gaps (taken from Rodríguez et al. (2004))

Relation	Symbol	Conditions	Converse
I_1 *before* I_2		$I_1+ < I_2-$	*after* ()
I_1 *meets* I_2		$I_1+ = I_2-$	*met_by* ()
I_1 *overlaps* I_2		$I_2- > I_1- \wedge I_1+ < I_2+ \wedge I_1+ > I_2-$	*overlapped_by* ( )
I_1 *starts* I_2		$I_1- = I_2- \wedge I_1+ < I_2+$	*started_by* ( )
I_1 *finishes* I_2		$I_1+ = I_2+ \wedge I_1- > I_2-$	*finished_by* ( )
I_1 *during* I_2		$I_1- > I_2- \wedge I_1+ < I_2+$	*contains* ( )
I_1 *equal* I_2		$I_1- = I_2- \wedge I_1+ = I_2+$	*equal*

3.2.3.3 Spatial SICs

For the definition of SICs, which restrict the spatial arrangement of entities and their spatial properties, relations specifically considering the spatiality have to be applied. Spatial relations between entities are usually not explicitly defined, but can be inferred from the geometries, shapes or extents of the entities and are, therefore, also used for spatial analysis. Such spatial relations can be subdivided into topological, directional, shape and metric relations. This differentiation has also been made at the lower level of the classification in Figure 3.1. The distinction of the different spatial aspects, the concepts of mereology and change is similar to the structure proposed by Cohn and Hazarika (2001) in their overview of qualitative spatial representation and reasoning.

Please remember that this subsection only considers ICs resulting of the entity's semantics. Constraints, which base on the definition of geometric and topological primitives belong to the class of domain constraints. In the following, the differentiated subclasses of spatial SICs are explained.

Topological SICs. Topology is probably the most fundamental perception of space (Cohn and Hazarika, 2001) and, as shown in empirical studies, the dominating factor when humans judge spatial configurations of entities (Knauff et al., 1997). Topology is a purely qualitative concept, independent of any quantitative measures and concerns the spatial connectedness of entities. Topological relations stay invariant under linear and affine transformations, such as rotation, translation and scaling. Topological relationships between two entities have been extensively studied in the literature. For an overview please see (Cohn and Hazarika, 2001).

In the GIS domain Egenhofer's Dimensionally Extended 9 Intersection Model (DI-9IM) is probably the most used method to analyse topological relations of two dimensional entities. It defines topological relations through matrices, which compare the point set intersections of the interiors, boundaries and exteriors of two entities (Egenhofer and Herring, 1990; Egenhofer and Franzosa, 1991). Based on the DI-9IM the ISO/TC211 (2004) defines topological relationship predicates, such as *intersect* or *overlap*, which can be conveniently used in topological SICs, for example:

"Lakes are not allowed to intersect with contour lines."

Since not all relations are possible for all geometry types, the standard also defines the validities of the relations[4]. Relations that are not expressible in natural language can be expressed through the *relate* relation, which directly refers to the DI-9IM matrix.

Depending on the application it might be necessary to use a different resolution of 'named' relations, that is, define a more or less detailed set of relations, as it has been done by Grigni et al. (1995). For example, the (ISO/TC211, 2004) relation *within* could be expressed through the two relations *inside* and *coveredBy*, to differentiate whether or not the inner entity is inside intersecting the boundary of the containing one. As shown by Knauff et al. (1997), sets of named topological relations with a higher abstraction seem to be cognitively irrelevant, that is, they do not conform to people's conceptual awareness of spatial relationships. However, for SICs coarse relations, such as *intersects*, are necessary to support a convenient definition.

Topological SICs can also restrict the topology of a single entity. Therefore, properties that specify things like the internal connection, the number of components and presence/absence of holes of a single entity are constrained, for example, for lake entities with polygon geometry:

"The inner rings of a lake (which represent the islands) are not allowed to intersect with each other or with the outer ring."[5]

Such restrictions extend the geometry type definitions with regard to the semantics of the concerned entity class.

Directional SICs. Directional SICs refer to orientation relations of entities. These relations base on the definition of a vector space. They are invariant under translation and uniform scaling. Generally, there are two groups of directional relations distinguished by their requirement of a fixed reference system.

[4]For example *crosses* is only valid for two entities with one dimensional geometries.

[5]According to ISO/TC211 (2004) two of the LinearRings that represent the boundary of a Polygon are allowed to intersect at a point (but only) as a tangent.

Qualitative cardinal directional relations, like *north*, *east* or *south-west*, describe the relative position of entities utilising the geographical coordinate system as reference. For the directional relation of two entities with point geometry a cone-shaped (or triangular; Figure 3.2a) and a projection-based (Figure 3.2b) approach have been investigated by Frank (1992).

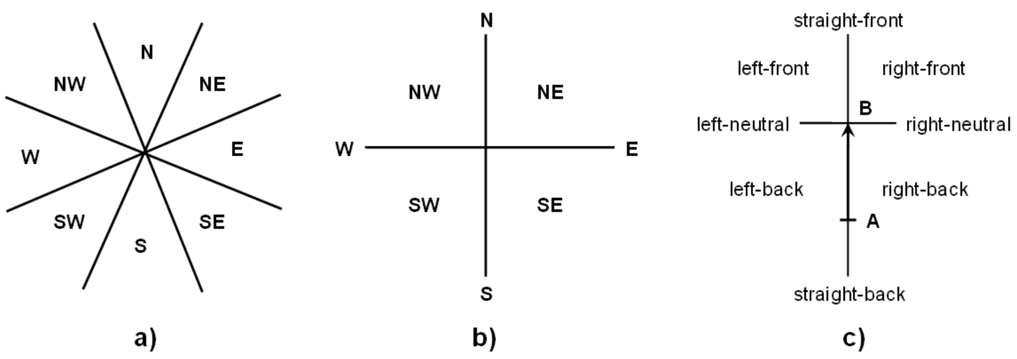


Figure 3.2: Examples of directional relations: a) cone-shaped and b) projection-based cardinal directional relations; c) left/right and front/back dichotomy for oriented entities

Cardinal directional relations for extended spatial objects have been researched by Goyal and Egenhofer (2001), Skiadopoulos and Koubarakis (2001) and Yan et al. (2006). Most of the proposed models are cognitively plausible, but as shown by Skiadopoulos and Koubarakis (2005) and Cicerone and Felice (2004) lead in some special cases to inconsistencies. The main problem is, that obviously a single model cannot handle all possible geometry types and their relations equally well. This might be the reason why up to now none of the proposed models has prevailed.

Cardinal directional relations are often used for verbal descriptions when people explain the relative position of entities in geographic space or when they reason about these entities. SICs that specify cardinal directional relations are hard to find, because usually these relations do not restrict the occurrence or the characteristics of entities and therefore no inconsistencies can occur. For example there is hardly any entity type that has to be north of another one (except the North Pole). More relevant might be relations in the three dimensional space like *above* and *below*, which also refer to a reference system.

The second group of directional relations deals with the order of entities in space, independently of a fixed reference system. The relations are typically used for entities with a well-defined front or back or a forward / backward orientation. Freksa (1992b) defines a set of relations based on a left/right and front/back dichotomy for oriented entities (Figure 3.2c).

Entities like houses, which usually have an intrinsic front side, separate the space into a front and back semi-plane. A resulting directional SIC could be:

"The backyard must be at the rear of a house."

Additionally to the relations defined by Freksa, relations like parallel or perpendicular, might be reasonable for the comparison of two directed entities. An example constraint could be for a 3D city model:

"Road signs have to face the direction opposite of the driving lane they are attached to."

So far only qualitative directional relations have been considered. In some contexts quantitative relations referring to discrete measures, like angles, distance relations or trigonometric calculations, might be useful as well. The disadvantage of such fixed discrete values is that in most cases similar angles cannot be intuitively distinguished by the user.

Shape SICs. Shape SICs restrict the geometry of an entity in terms of form, shape and stature. Since they ensure semantic integrity, they result from the entity's semantics, that is, the concepts of the entity classes which are represented in the data model.

Shape is a concept, that is difficult to describe qualitatively (Cohn and Hazarika, 2001) and for many entity classes no generally valid shape property is definable. Because of this, shape is rarely used for the definition of SICs. Nevertheless, shape constraints seem to be convenient, in particular for 3D data (e.g., to specify roof types of houses).

Some examples for qualitative shape descriptions of single entities rest on the sequence of the boundary extremes of areal features (Leyton, 1988; Schlieder, 1996), abstraction primitives, such as a convex hull or general convexity / concavity (Clementini and Felice, 1997), and the adjacency and relative position of concavities (Cohn, 1995). Clementini and Felice (1997) also investigated metric properties that describe the shape of an entity. They define a qualitative model for the definition of the symmetry of a single entity's boundary and sets of relations to describe the compactness, that is, the elongation of an areal entity. A qualitative characterisation of the straightness of polyline geometries has been studied by Gottfried (2007).

Metric SICs. Metric properties are based on distances between geospatial entities or their constituent parts. They change with scale, but stay invariant under rotation and translation. Corresponding metric SICs restrict distances and size.

Some quantitative metric properties like length, area size and radius are based on operations on the entity's geometry. Some of them are considered in international standards as methods of particular geometry types. For example, the ISO/TC211 (2002a) defines for Curve and LineString geometries a *Length* method and for Surface geometries an *Area* method, which return the length of the curve, respectively, the area of the surface. Such methods can be used to constrain things like the minimum length of a linear entity or the maximum size of an area. The ISO norm also provides the spatial analysis operators *Distance* and *Buffer*, which are also expedient for the analysis according to metric SICs. An example of such a constraint in context of a national law is:

"A petrol station must be in at least 300m distance from a school."

Qualitative systems for distances have been discussed by Frank (1992). He defines sets of distance relations between two entities like *far*, *medium* and *very_close* with varying granularity. Hernández et al. (1995) extend this approach by using not only equally spaced distance intervals. Furthermore, they capture the contextual information of the distance systems through the definition of reference frames. Since the metric terms of qualitative distance concepts are context-dependent, the context is important to consider for such SICs. Therefore, the use of numerical values seems to be more convenient and expressive for the definition of metric SICs.

3.2.3.4 Complex SICs

Many SICs combine relations of more than one of the semantic domains used in the categorisation above. They restrict relations or properties of several semantic domains in one constraint. An example of such a complex SIC is:

"A butterfly valve must not intersect a pipe if the diameter of the pipe is greater than 40cm."(Cockcroft, 2004)

This constraint specifies a restriction through a combination of the topological relation *intersect* between butterfly valve and pipe and the thematic relation *greater_than* of the pipe attribute diameter. The thematic relation is used to define a subset of pipe entities for which the constraint is valid.

For some of the concepts, itemised in the categorisation above, integrated algebras have been analysed, which are useful for the definition of complex SICs. Examples are spatio-temporal relationships between independent temporal regions (Claramunt and Jiang, 2001) or for combinations of topological and directional relations (Hernández, 1994; Sharma and Flewelling, 1995).

Beside the relations of the elucidated semantic domains, complex SICs can include mereological relations and domain specific associations that are explicitly defined in the data model. Mereology deals with relations between parts and their respective wholes. For spatial data the concepts of mereology and topology are usually not completely independent; the interrelations between the two notions have been investigated by Varzi (1994). The correspondence of spatial aggregation and the consistency of the thematic attributes of the sub-entities has been pointed out by Plümer and Gröger (1997). They define for example:

“The number of inhabitants of a country is the sum of the numbers of the inhabitants of its administrative districts.”

In this example administrative district entities are related to the country entity through a *partOf* relation. It also shows the strong connection between mereology and topology, since the districts are usually topologically contained by the country. In general, such dependencies between the properties of objects of different classes are called propagation (Egenhofer and Frank, 1992).

3.2.3.5 Change and Representation SICs

According to Hornsby and Egenhofer (1997) changes refer to operations performed on an object or a group of objects. Thereby the changes can either preserve the object’s identity or result in a change or a deletion of identity. Change SICs prevent forbidden modifications of the entities and their properties, that is, they restrict the change between the two versions or states of the entities.

In general there are two types of change SICs to consider. Firstly, there are transition constraints, which are restrictions on multiple states of a database (Elmasri and Navathe, 1994). They are the only type of IC that can only be checked during a database transaction. All other constraints can be proven independently of a transaction. Many GIS applications integrate temporal changes in their model and, therefore, store multiple states of an entity in one database state. Therefore constraints, that restrict the differences between multiple consecutive versions of a single entity or group of entities of one database state are included here as a second type of change SICs.

Possible relations that represent the change operation in SICs can be found in (Hornsby and Egenhofer, 1997), who classified operators for change actions of single entities (e.g., *create, deconstruct*), aggregates of objects (e.g., *combine, compound*) as well as their attributes and relations (e.g., *add, remove*). An example of a change SIC could apply to a numerical attribute, allowing the attribute value only to increase in case of change. The corresponding checking algorithm would have to compare the two versions of the entity or the attribute in the two database states.

Some geodatabases contain multiple representations of a property, for example, when an entity has a separate geometry stored for each level of detail. As shown by Egenhofer et al. (1994) and Tryfona and Egenhofer (1997), the consistency of those representations can also be evaluated by SICs. They described frameworks to assess the topological consistency of multiple representations. An example SIC is:

"The number of represented islands of a lake (e.g., holes of the lake's geometry) is not allowed to increase when the geometry is generalised."

Relations that represent the connection between multiple representations in SICs can be *generalisation*, *detailing* or for two specific levels of detail something like *LOD1toLOD2*. In GISs the change of representation mostly results in a generalisation of the entity's geometry, but thematic attributes can also be involved. The differences and restrictions between two representations are generally comparable to those resulting from change operations. Therefore, here a compound category for change and representation SICs is defined.

The restrictions of change and representation might refer to any relation or property of the semantic domains. For example, Egenhofer et al. (1994) restrict the changes of the internal topological properties of an entity and the topological relations between entities.

3.3 Summary

The categorisation of spatio-temporal ICs provides a basis for further work on SIC definition, formalisation, management and validation of internal consistency of SIC sets. The categorisation gives an overview of the different kinds of spatio-temporal relations, which can be used to define SICs. All relations must be linked to a corresponding mathematical definition, which is implemented in the quality evaluation procedures. The assignment of a spatial IC to one of the sub-classes of the categorisation is not always unambiguous, however. This is due to influences and overlaps between the domains of the constraint classes. In such cases, it is suggested to assign the constraint to the more abstract domain concept.

SICs can be manifold and complex and not all of them can be considered in this work. Thus the following investigations are confined to SICs referring to all entities of a single entity class and SICs referring to an entity and its associated entities of various classes (categories three and four of the classification according to the restricted data model elements, Subsection 3.1.1.4), leaving out constraints that restrict single entity's attributes and operations of entities. Only

SICs on relations between the entities of a single or of two entity classes are considered. A further limitation is that only **implicitly modelled binary relations** are considered. Relations that are explicitly specified in the data model are restricted by key and relationship constraints. An implicit relation can be deduced from the corresponding attributes of the involved entities. A typical example of such implied relations are topological relations (Figure 2.2) between spatial entities (Egenhofer and Herring, 1990). Usually they are not explicitly stored since they can be derived from the geometries of the involved entities. All examples of class relations and SICs given in the following chapters base on the binary topological relations between areal entities shown in this figure. However, the defined class relations (Chapter 4) can also be combined with any other instance relation.

The consistency of the constraints themselves has to be separately checked for the constraints of each of the SIC subclasses. Not all relations and constraint categories allow for the application of the reasoning methodology introduced in the following chapters. To enable for reasoning on the SICs (Chapter 5) the applied relations must be part of a limited JEPD set of relations. Most qualitative spatial relations meet this requirement. Some more detailed requirements on logic properties of the relations are analysed later. Clementini and Felice (2000) presented general requirements for spatial operators usable in database query languages. In particular requirements like expressiveness, a small number of operators in the set, consistency, generality, hierarchical structure, linguistic and cognitive soundness, qualitativeness and the support of uncertainty are valid for the relations used for the definition of SICs.

Chapter 4

A Framework for the Formal Definition of Semantic Integrity Constraints

Most existing approaches for a formal definition of semantic integrity constraints (SICs) (Werder, 2009) are either very specific, since they rely on a certain property such as the topological relation between two entities, or they make use of specifically defined checking procedures, and the SICs are, therefore, not interoperably exchangeable. None of the approaches investigates the logical properties of the SICs and considers the logical soundness as a requirement on the formalisation. For the examination of the internal consistency of sets of SICs this is a prerequisite. This chapter illustrates the importance of cardinality descriptions for such logically sound SIC definitions and introduces a corresponding set of 17 class relations (Mäs, 2007a,b), which can be used for SIC formalisation and reasoning.

4.1 Instance-Level vs. Class-Level Relations

Integrity constraints (ICs) are defined at the level of entity classes, since they always restrict entire classes or subsets of classes and not (exclusively) single instances. This means the ICs must hold for all instances of the restricted classes. When a database is checked against a spatial SIC, the checking procedure examines spatial relations between the involved instances. Thus a formalised description of such a SIC must be linked to the instance relation(s) which the quality checking procedure applies. The previous chapter provides examples of such relations for all proposed categories. In the literature, formal definitions of SICs are (following the natural language) mostly based on instance relations, whereas cardinality restrictions are often not or only incompletely considered (Ubeda and Egenhofer, 1997; Mäs et al., 2005b). However, as the following example illustrates, the very instance relations are not suitable for SIC formalisation. A natural language statement about two instances could be:

“house #12 is inside of parcel #1234”.

This is a simple statement about two entities that are related by the spatial relation *inside*. Since *inside* is the converse relation of *contains* (Section 2.4.1), the statement also implies:

“parcel #1234 contains house #12”.

A corresponding SIC for the entity classes *building* and *parcel* could be:

“buildings are inside of a parcels”.

Applying the converseness of the instance relation again, it becomes:

“parcels contain buildings”.

These statements can be mistaken, since they should be understood as “every building is inside of some parcel”, but NOT as “every parcel contains a building”. This example shows the influence of words like “all” or “some” on the semantic of a statement. They define cardinality restrictions on the applied relation. For a human reader it is often possible to grasp the correct semantics, but a formal description of such a statement must explicitly contain cardinality information. This example shows that instance relations alone cannot unambiguously represent the semantic of statements about classes. A relation among the classes is not subject to the same logical properties as a relation between instances and the cardinality information must be considered for reasoning. Thus, a logically sound formalisation of such statements requires specific **class relations**. Donnelly and Bittner (2005) also identified this problem and proposed an approach for the definition of relations between classes (in this work called class relations). Tarquini and Clementini (2008) confirm the suitability of these class relations for the SIC definition. The following section reviews this solution for an application for SIC definitions and, in particular, reasoning on them. Some class relations do not define violable restrictions on the involved classes and are, therefore, not applicable as SICs. Further, some additional cardinality 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, which particularly supports the logical reasoning on SICs, is defined in Section 4.3.

4.2 Cardinality Properties of Class Relations

Cardinalities express the number of elements of a set. Class relations define a cardinality restriction for a certain relation between the individuals of one or more classes. Therefore, it must be ensured that the classes conform to the following two requirements. Firstly, the involved classes must have at least one instance, that is, empty classes are not feasible. Since class relations are linked to individual relations, the second condition specifies that if a class relation is defined there must exist at least one corresponding individual relation among the instances of the classes involved. The investigations of this thesis are restricted to binary relations defined between entity classes. Relations between three or more classes, or between subsets of classes (e.g., blue houses as a subset of the class house), are not considered.

In the following definitions the lowercase letters x, y and z denote variables for instances or individuals. Every instance must belong to an entity class. For entity classes capital letters A, B and C are used as variables. $Inst(x, A)$ means individual x is an instance of class A. The function $r(x, y)$ means instance x has the relation r to instance y; x and y are said to participate in the relationship instance r. The meta-variable r can stand for any binary relation between instances (e.g., a topological relation) or for a disjunction of such relations. The validity of the binary relation depends on the properties of the instances (e.g., for spatial relations the geometry of the instances). Instance relationships can be associated with a class relation R. For class relation definitions $R_{<cp>}(A, B)$ 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 instance relation r or a disjunction of instance relations.[6] If a class relation $R_{<cp>}(A, B)$ is defined, at least one r must exist between the instances of A and B. The placeholder $< cp >$ stands for the cardinality properties of the class relation. Based on these variables and functions Donnelly and Bittner (2005) define the following class relations:

$$R^{D\&B}_{some}(A, B) := \exists x \exists y \left(Inst(x, A) \cap Inst(y, B) \cap r(x, y) \right). \tag{4.1}$$

$$R^{D\&B}_{all-1}(A, B) := \forall x \left(Inst(x, A) \Rightarrow \exists y (Inst(y, B) \cap r(x, y)) \right). \tag{4.2}$$

$$R^{D\&B}_{all-2}(A, B) := \forall y \left(Inst(y, B) \Rightarrow \exists x (Inst(x, A) \cap r(x, y)) \right). \tag{4.3}$$

$$R^{D\&B}_{all-12}(A, B) := R^{D\&B}_{all-1}(A, B) \cap R^{D\&B}_{all-2}(A, B). \tag{4.4}$$

$$R^{D\&B}_{all-all}(A, B) := \forall x \forall y \left(Inst(x, A) \cap Inst(y, B) \Rightarrow r(x, y) \right). \tag{4.5}$$

The relation $R^{D\&B}_{some}(A, B)$ (Equation 4.1) holds if at least one instance of A stands in relation r to some (at least one) 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. SICs, which are only based on such relations, are not expedient, since they only specify that a relation universally exists, but without 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 instances

[6] Disjunctions of all instance relations of an exhaustive set are invalid.

that have the relation or it does not contain such instances; both cases are conform to the SIC. Since a violation against constraints which are only specifying $R^{D\&B}_{some}(A,B)$ relations is impossible, such relations are not useful for quality assurance. A violation becomes possible 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). Therefore, the constraint is violable by the subsets of A and B and useful for quality assurance. However, since the definition of such subsets can be manifold and complex, the analysis is here restricted to binary relations between entire entity classes, leaving out subsets of classes.

The relation $R^{D\&B}_{all-1}(A,B)$ (Equation 4.2) holds if every instance of A has the relation r to some (at least one) instance of B. In set theory, such relations are called **left-total**. This class relation can be used to define the SIC of the parcels and buildings example: $INSIDE^{D\&B}_{all-1}(Building, Parcel)$ specifies the inside relation for all buildings, but it does not include all parcels.

The relation $R^{D\&B}_{all-2}(A,B)$ (Equation 4.3) holds if for each instance of B there is some (at least one) instance of A that stands in relation r to it. This means that every instance of B has the converse relation of r to some instance of A. $R^{D\&B}_{all-2}(A,B)$ is **right-total** / surjective, like for example $CONTAINS^{D\&B}_{all-2}(Parcel, Building)$.

The relation $R^{D\&B}_{all-12}(A,B)$ (Equation 4.4) combines the definitions of the Equations (4.2) and (4.3). It holds, if every instance of A stands in relation r to at least one instance of B and if for each instance of B there is at least one instance of A, that stands in relation r to it. R is left-total and right-total. An example SIC is $CONTAINS^{D\&B}_{all-12}(Country, Capital)$, which specifies that every country has a capital and, conversely, every capital is contained in a country.

As these examples show, the differentiation of class relations according to the totality of the involved individuals of the entity classes is useful for the definition of SICs. The class relations define constraints on all individuals of A (Equation 4.2), all individuals of B (Equation 4.3) or on all individuals of both arguments A and B (Equation 4.4). In data modelling such definitions are called **participation constraints** on the relation (Elmasri and Navathe, 1994). They specify whether the existence of an entity depends on its relation to another entity via the relationship type. Equations 4.2 - 4.4 define total participation constraints on their relationship instances, since at least one of the classes is totally affected. $R^{D\&B}_{some}(A,B)$ defines a partial participation constraint, since not necessarily all instances of the classes A and B have the relationship instance.

A specific case of $R^{D\&B}_{all-12}(A,B)$ is defined by the $R^{D\&B}_{all-all}(A,B)$ relation (Equation 4.5), which holds if all instances of A have a relationship instance of R to all instances of B. This relation is very strong, since it defines restrictions on all relations between the individuals of the arguments A and B. The corresponding SICs are strongly restrictive, but useful when, for instance, no instances of two classes are allowed to intersect: $DISJOINT^{D\&B}_{all-all}(Streets, Lakes)$.

Beside the total participation constraint, the $R^{D\&B}_{all-12}(A, B)$ relationship defines a so-called **cardinality ratio constraint**, which specifies the number of relationship instances in which an entity can / has to participate (Elmasri and Navathe, 1994). In this case the number of B entities (i.e., "all" instances of B) defines in how many relationship instances r each entity of A participates, and vice versa.

In data modelling, total participation and cardinality ratio constraints are well established, for example when using an ERD 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 (Figure 4.1). In this example, all buildings are restricted to be contained by only one parcel, while the parcels are allowed to contain an imprecisely defined number (positive integer) of buildings.

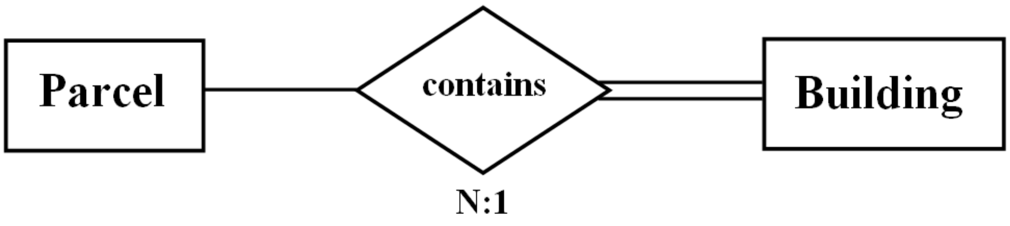


Figure 4.1: Total participation and cardinality ratio constraints in an ERD

Since this cardinality ratio cannot be specified with the class relations of Donnelly and Bittner (2005), the SIC between buildings and parcels is only imprecisely represented by $CONTAINS^{D\&B}_{all-2}(Parcel, Building)$. This shows that for a sufficient representation of SICs cardinality ratio constraints are indispensable.

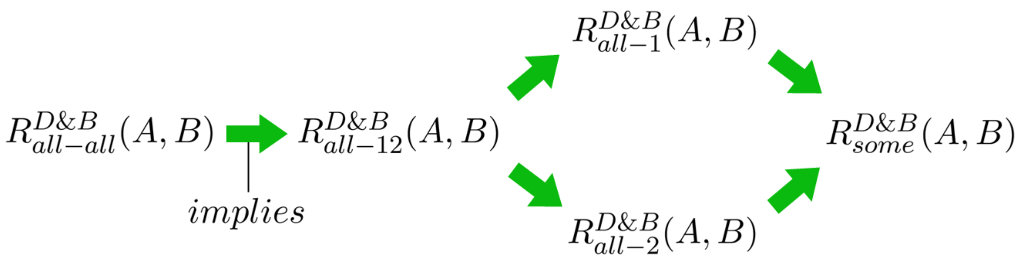

$$\begin{tikzcd}
 & & R^{D\&B}_{all-1}(A,B) \arrow[dr] & \\
R^{D\&B}_{all-all}(A,B) \arrow[r, "implies"'] & R^{D\&B}_{all-12}(A,B) \arrow[ur] \arrow[dr] & & R^{D\&B}_{some}(A,B) \\
 & & R^{D\&B}_{all-2}(A,B) \arrow[ur] &
\end{tikzcd}$$

Figure 4.2: Dependencies between Donnelly and Bittner's class relations

For the purpose of this work, another disadvantage of Donnelly and Bittner's approach is that their class relations are not pairwise disjoint. Since some of the class relations imply others (Figure 4.2) and none of them excludes one of the others, it is impossible to define SICs which are in conflict. Thus, a set of class relations with more expressive power is required.

The set of class relations that is used in this thesis is based on four basic **cardinality properties** (Equations 4.6 - 4.9), which refer to the specific properties of binary relations (Section 2.4.1). These cardinality properties define the intrinsic restrictions of the cardinalities (Mäs, 2007a,b).

$$LT(A,B,r) \;:=\; \forall x\,(Inst(x,A) \Rightarrow \exists y(Inst(y,B) \cap r(x,y)))\,. \tag{4.6}$$

$$RT(A,B,r) \;:=\; \forall y\,(Inst(y,B) \Rightarrow \exists x(Inst(x,A) \cap r(x,y)))\,. \tag{4.7}$$

$$\begin{aligned} LD(A,B,r) \;:=\; & \forall x,y,z(Inst(x,A) \cap Inst(y,B) \cap Inst(z,A) \cap \\ & r(x,y) \cap r(z,y) \Rightarrow x = z) \cap Ex(A,B,r). \end{aligned} \tag{4.8}$$

$$\begin{aligned} RD(A,B,r) \;:=\; & \forall x,y,z(Inst(x,A) \cap Inst(y,B) \cap Inst(z,B) \cap \\ & r(x,y) \cap r(x,z) \Rightarrow y = z) \cap Ex(A,B,r). \end{aligned} \tag{4.9}$$

$$Ex(A,B,r) \;:=\; \exists x \exists y\,(Inst(x,A) \cap Inst(y,B) \cap r(x,y))\,. \tag{4.10}$$

The cardinality properties $LT(A,B,r)$ and $RT(A,B,r)$ (Equations 4.6 and 4.7) define a totality for the class A and B, respectively. These properties correspond to the left-total and right-total class relations of Donnelly and Bittner (Equations 4.2 and 4.3).

$LD(A,B,r)$ and $RD(A,B,r)$ (Equations 4.8 and 4.9) define cardinality properties, that represent the concept of unambiguousness as a typical cardinality ratio constraint. Class relations that hold $LD(A,B,r)$ are **left-definite** and specify that for no instance of B there is more than one instance of A, that stands in relation r to it. This property restricts the number of r relations in which an instance of B can participate; the instances of A are not restricted. The last term ensures that at least one instance relation r exists between the instances of A and B (Equation 4.10). $RD(A,B,r)$ specifies that no instance of A participates in a relationship r to more than one instance of B. When this cardinality property is defined in a class relation, all instances of A are restricted, while the instances of B are not affected. The corresponding class relations are **right-definite**. Figure 4.3 shows the representation of the four cardinality properties in ERDs. In general, the number of different cardinality ratio constraints of such a notation is infinite. Thus, it is impossible to represent them all by separate cardinality properties. The unambiguousness cardinality ratio constraints have been chosen here, because they are commonly used for the definition of SICs and other cardinality restrictions such as (0..2) are relatively rare. Therefore, it can be assumed that this set of cardinality properties can precisely represent the majority of the SICs used in practice.

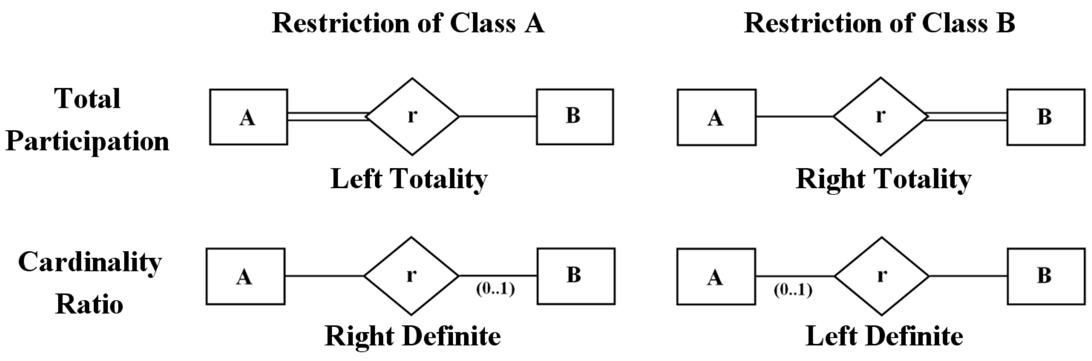


Figure 4.3: ERDs of the four cardinality properties

4.3 Definition of a Set of Class Relations

The four cardinality properties defined in the previous section are independent of each other so that no property implies or precludes one of the other properties. If a class relation is only defined as right-total (e.g., Equation 4.3) there is no information about its left-totality and the cardinality ratio. For the formal definition of SICs this situation is insufficient, since they base on combinations of the cardinality properties as well as their negations. For example the SIC in Figure 4.1 is based on the topological instance relation *contains* and the cardinality properties left-definite and right-total. The other two properties are invalid and should, therefore, be excluded. An investigation of all possible combinations of cardinality properties leads to the following categorisation:

- four class relations where one property is valid and the corresponding other three are excluded;
- six class relations where two properties are valid and the other two are excluded;
- four class relations that combine three of the four defined cardinality properties respectively and exclude the corresponding fourth;
- one class relation where all four properties are valid; and
- one class relation where none of the four properties is valid.

$$R_{LD}(A,B) \;:=\; LD(A,B,r) \cap \neg RD(A,B,r) \cap \neg LT(A,B,r) \cap \neg RT(A,B,r). \tag{4.11}$$

$$R_{RD}(A,B) \;:=\; \neg LD(A,B,r) \cap RD(A,B,r) \cap \neg LT(A,B,r) \cap \neg RT(A,B,r). \tag{4.12}$$

$$R_{LT}(A,B) \;:=\; \neg LD(A,B,r) \cap \neg RD(A,B,r) \cap LT(A,B,r) \cap \neg RT(A,B,r). \tag{4.13}$$

$$R_{RT}(A,B) \;:=\; \neg LD(A,B,r) \cap \neg RD(A,B,r) \cap \neg LT(A,B,r) \cap RT(A,B,r). \tag{4.14}$$

Equations 4.11 to 4.14 specify class relations that are either left-definite, right-definite, left-total or right-total. The corresponding other cardinality properties are excluded. An example SIC is "all floodplains overlap with at least one stream": $OVERLAP_{LT}(Floodplain, Stream)$ (Figure 4.4a).

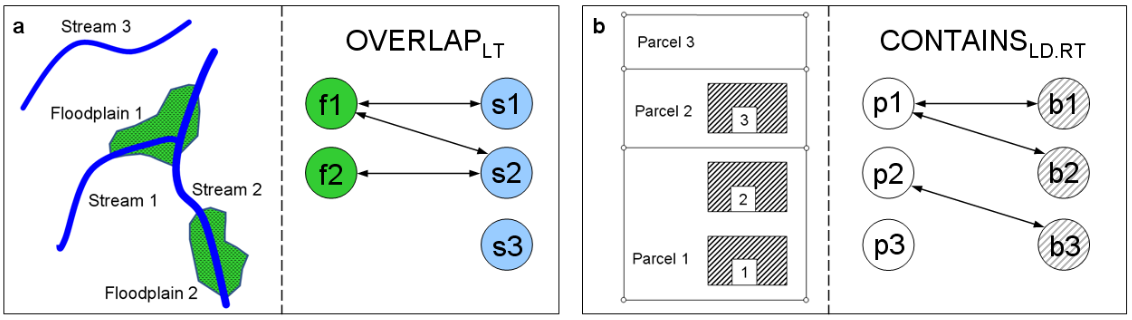


Figure 4.4: Example scenes for the SICs between the classes a) *floodplain* and *stream* and b) *parcel* and *building*

As shown in the figure the constraint is left-total, because *all* floodplains are overlapping a stream. Right-definite, left-definite and right-total must be excluded, because a floodplain can overlap with more than one stream (e.g., floodplain 1), a stream can have more than one floodplain (e.g., stream 2) and streams (e.g., stream 3) do not necessarily have a floodplain.

$$R_{LD.RD}(A,B) := LD(A,B,r) \cap RD(A,B,r) \cap \neg LT(A,B,r) \cap \neg RT(A,B,r). \tag{4.15}$$

$$R_{LD.LT}(A,B) := LD(A,B,r) \cap \neg RD(A,B,r) \cap LT(A,B,r) \cap \neg RT(A,B,r). \tag{4.16}$$

$$R_{LD.RT}(A,B) := LD(A,B,r) \cap \neg RD(A,B,r) \cap \neg LT(A,B,r) \cap RT(A,B,r). \tag{4.17}$$

$$R_{RD.LT}(A,B) := \neg LD(A,B,r) \cap RD(A,B,r) \cap LT(A,B,r) \cap \neg RT(A,B,r). \tag{4.18}$$

$$R_{RD.RT}(A,B) := \neg LD(A,B,r) \cap RD(A,B,r) \cap \neg LT(A,B,r) \cap RT(A,B,r). \tag{4.19}$$

$$R_{LT.RT}(A,B) := \neg LD(A,B,r) \cap \neg RD(A,B,r) \cap LT(A,B,r) \cap RT(A,B,r) \cap \neg R_{LT.RT-all}(A,B). \tag{4.20}$$

Equations 4.15 to 4.20 combine pairs of the four defined cardinality properties and exclude the corresponding others. For example, Equation 4.17 can be used to define the SIC between parcels and buildings: $CONTAINS_{LD.RT}(Parcel, Building)$ (Figure 4.4b). The constraint is left-definite and right-total, since all buildings are within exactly one parcel. The other cardinality properties are excluded, because parcels are not restricted. A special case in this group is Equation 4.20, which also excludes $R_{LT.RT-all}(A,B)$.

$$R_{LT.RT-all}(A,B) \quad := \quad \forall x \forall y \left(Inst(x,A) \cap Inst(y,B) \Rightarrow r(x,y) \right). \tag{4.21}$$

Equation 4.21 is equivalent to Donnelly and Bittners relation $R^{D\&B}_{all-all}(A,B)$ (Equation 4.5). $R_{LT.RT-all}(A,B)$ is left-total and right-total and holds if all instances of A have a relationship instance of R to all instances of B. This constraint is very restrictive and, therefore, highly useful for the definition of SICs. $R_{LT.RT-all}(A,B)$ is a specific case of $R_{LT.RT}(A,B)$. Thus it has been excluded in Equation 4.20 to achieve a set of pairwise disjoint relations.

$$R_{LD.RD.LT}(A,B) \quad := \quad LD(A,B,r) \cap RD(A,B,r) \cap LT(A,B,r) \cap \neg RT(A,B,r). \tag{4.22}$$

$$R_{LD.RD.RT}(A,B) \quad := \quad LD(A,B,r) \cap RD(A,B,r) \cap \neg LT(A,B,r) \cap RT(A,B,r). \tag{4.23}$$

$$\begin{aligned} R_{LD.LT.RT}(A,B) \quad := \quad & LD(A,B,r) \cap \neg RD(A,B,r) \cap LT(A,B,r) \cap RT(A,B,r) \cap \\ & \neg R_{LT.RT-all}(A,B). \end{aligned} \tag{4.24}$$

$$\begin{aligned} R_{RD.LT.RT}(A,B) \quad := \quad & \neg LD(A,B,r) \cap RD(A,B,r) \cap LT(A,B,r) \cap RT(A,B,r) \cap \\ & \neg R_{LT.RT-all}(A,B). \end{aligned} \tag{4.25}$$

Equations 4.22 to 4.25 combine three of the cardinality properties and exclude the corresponding fourth. Particular attention must be given to class relations, that are left-total and right-total (Equations 4.24 and 4.25). 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 also hold for Equation 4.21. Thus, it is necessary to separate the relations 4.24 and 4.25 from 4.21, which is done by the exclusion of $R_{LT.RT-all}(A,B)$. Consequently, the relations $R_{LD.LT.RT}(A,B)$ and $R_{RD.LT.RT}(A,B)$ are impossible if class A or class B has only one instance. An example SIC is $CONTAINS_{LD.LT.RT}(Airport, AirportTower)$, which specifies that every airport contains at least one airport tower and every airport tower is contained in exactly one airport (Figure 4.5a). Right-definite is excluded, because some airports (e.g., airport 1) have more than one airport tower.

$$\begin{aligned} R_{LD.RD.LT.RT}(A,B) \quad := \quad & LD(A,B,r) \cap RD(A,B,r) \cap LT(A,B,r) \cap RT(A,B,r) \cap \\ & \neg R_{LT.RT-all}(A,B). \end{aligned} \tag{4.26}$$

Equation 4.26 specifies class relations, that are left-definite, right-definite, left-total and right-total (bijective, Section 2.4.1). Again $R_{LT.RT-all}(A,B)$ is excluded to distinguish the relation from 4.21 for the case that A and B have only one instance. In this case the relation cannot occur. $R_{LD.RD.LT.RT}(A,B)$ can for example be used to specify that every country

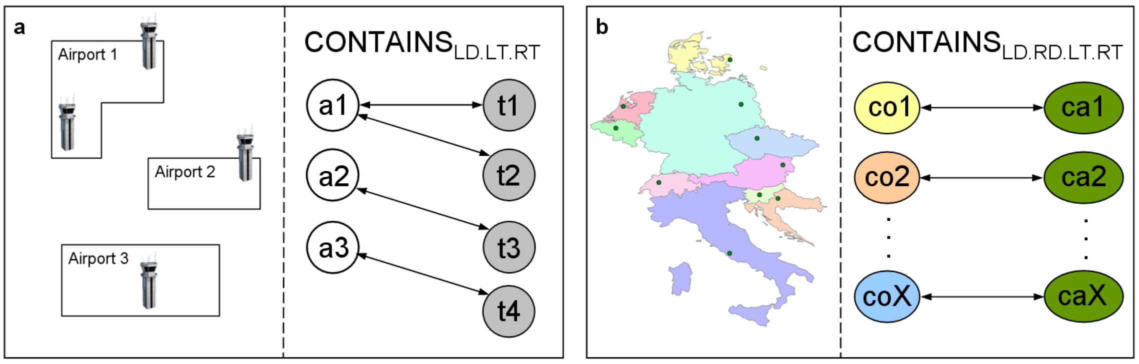


Figure 4.5: Example scenes for the SICs between the classes a) *airport* and *airport tower* and b) *country* and *capital*

contains exactly one capital and conversely: $CONTAINS_{LD.RD.LT.RT}(Country, Capital)$ (Figure 4.5b).

$$R_{some}(A,B) \quad := \quad \neg LD(A,B,r) \cap \neg RD(A,B,r) \cap \neg LT(A,B,r) \cap \neg RT(A,B,r) \cap Ex(A,B,r). \tag{4.27}$$

Equation 4.27 is defined for the situation that none of the four cardinality properties is valid, but nevertheless some instances of A stand in relation r to some instances of B[7]. $R_{some}(A,B)$ is similar to Equation 4.5 of Donnelly and Bittner, but while their definition contains all other defined class relations, these are now excluded. $R_{some}(A,B)$ is defined as not left-total and not right-total, which implies that some instances of A and B participate in a relation r to an instance of B and A and some do not. Furthermore the exclusions of $LD(A,B,r)$ and $RD(A,B,r)$ specify that some A and some B participate in a relation r to at least two instances of B and A. Hence $R_{some}(A,B)$ is only valid for two classes with more than two instances each. Since $R_{some}(A,B)$ does not specify concrete cardinalities, it is relatively weak. All cardinalities from "2" till "$all - 1$" are valid for both classes. Therefore, it is a relatively imprecise representation of all SICs that the other 16 class relations do not cover.

Together, Equations 4.11 to 4.27 specify 17 class relations that can be combined with any binary instance relation to associate classes. All relations are pairwise disjoint and the set is jointly exhaustive if the applied instance relations are part of a JEPD set. With the exception of $R_{some}(A,B)$, all of these relations specify restrictions, that can be used as SICs for quality assurance of the data. Subsequently relations, that are not linked to a particular instance relation are referred to as **abstract class relations** (e.g., $R_{LD.RD.LT}(A,B)$). Only relations that incorporate a concrete instance relation are called **class relations** (e.g.,

[7]This was one of the preconditions for class relations, mentioned earlier in this chapter.

$DISJOINT_{LD.RD.LT}(A, B)$). An example for each of the abstract class relations is given in Figure 4.6[8].

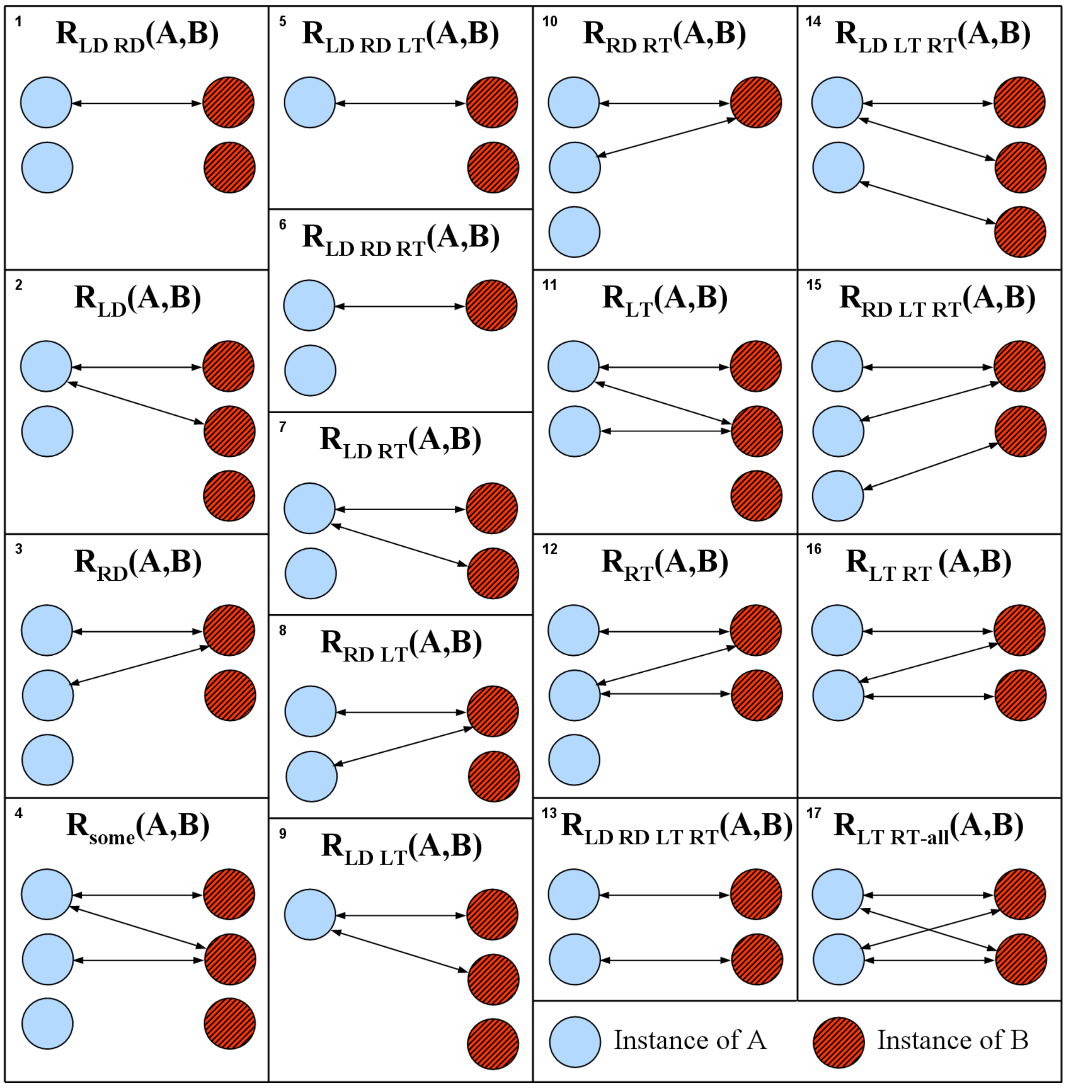


Figure 4.6: Examples of the 17 abstract class relations

Figure 4.6 shows example constellations for the instances of A and B and the instance relation r. Class relations neither specify the exact number of instances of the classes nor the exact arrangement of the instance relations.

[8]The order of the relations in this figure results from their conceptual neighbourhood (Chapter 5). The numbers of the relations are used for their identification in this work.

All 17 class relations can also be based on disjunctions of instance relations, for example:

$$[MEET \cup DISJOINT]_{LT.RT-all}(Airport, Forest).$$

This class relation specifies that all airports either meet or are disjoint from each forest.[9]

Similar to the instance relations it is also possible to define **disjunctions of class relations** if the knowledge about the relation of two classes is not unique. Such disjunctions combine multiple class relations, while only one of them is valid (logical XOR), for example:

$$DISJOINT_{LT.RT-all}(A, B) \cup MEET_{LT.RT}(A, B).$$

Depending on the relations, it is possible to define more than one class relation between two classes, even if the applied instance relations are part of the same JEPD set of relations. For example, for the classes watermill and stream it could be useful to define that every watermill must overlap with a stream:

$$OVERLAP_{RD.LT}(Watermill, Stream).$$

A second SIC between these classes could supplement that all watermills and streams either overlap or are disjoint (this excludes all other instance relations of the set).

$$[OVERLAP \cup DISJOINT]_{LT.RT-all}(Watermill, Stream).$$

Nevertheless the possibilities of such combinations of class relations are limited if they are based on instance relations that are part of a JEPD set of relations. For example, if $R_{LT.RT-all}(A, B)$ (relation #17) is defined, no second $R_{LT.RT-all}(A, B)$ relation and no class relation based on other instance relations of that domain are possible between the two classes. However, this points out a major difference to the instance relations, where only one relation of a JEPD set can be valid for two instances.

A **negation** of class relations is infeasible with this formal description and it does not seem practically relevant. Some approaches enable the definition of "forbidden" relations between instances (Ubeda and Egenhofer, 1997) in SICs, for example:

"roads and lakes are not allowed to intersect".

[9] A disjunction of instance relations like for example $meet(a, b) \cup equal(a, b)$ corresponds to the logical XOR, that means either $meet(a, b)$ or $equal(a, b)$ is valid. If a class relation is based on a disjunction of instance relations this disjunction corresponds to the logical OR. In the example *meet* and/or *disjoint* relations are valid between airport and forest entities.

Such forms of negation are also infeasible with this approach. Nevertheless it is possible to express these constraints if the instance relation is part of a JEPD relation set. In such sets, a negation of a relation always corresponds to another relation or a disjunction of relations. For the example, this would be *disjoint* instead of *not intersect*:

"roads and lakes are disjoint".

This appears to be advantageous when considering aspects of human spatial cognition as people have more difficulties in reasoning with negated spatial relations (Schleipen et al., 2007).

4.4 Summary

The set of 17 abstract class relations enables the definition of class relations based on any binary instance relation. Typical applications of class relations are ontologies of classes (Donnelly and Bittner, 2005) and SICs (Tarquini and Clementini, 2008). For GI, class relations are of particular interest, because a semantic description of such data requires class relations, that base on spatial instance relations, such as topological or metric relations. The interoperable exchange of data of different domains and application areas requires semantic descriptions of the data. Class relations are useful for the formalisation of these descriptions. The logical properties of the class relations support an automatic processing, querying and comparing of such descriptions.

The set of abstract class relations is a qualitative representation of all (infinitely many) possible constraints. Only SICs, that base on the four cardinality properties (Equations 4.6 - 4.9) or $R_{LT.RT-all}(A, B)$ (Equation 4.21) can precisely be defined. Such notations as ERDs are more expressive with regard to the cardinality ratio constraints. However, the reasoning concepts, investigated in the next chapter, require a discrete set of abstract class relations. Also, it is assumed that the introduced set of abstract class relations can precisely represent a majority of the SICs used in practice.

Compared to other approaches of SIC formalisation (Section 2.3) the advantages of class relations are their logical soundness, their independence of a specific set of instance relations, the relatively high expressiveness and extensibility. The representation of SICs as class relations is clearer and more obvious for users than a representation in logical languages (Hadzilacos and Tryfona, 1992). A transformation of class relations into other formalisation formats is possible if the spatial relations and the logical expressions used for the definition of the cardinality properties and abstract class relations (negation $\neg$, universal and existential quantifiers $\forall$ and $\exists$, conjunction $\cap$, disjunction $\cup$ and membership of instances to a class) are supported.

Chapter 5

Reasoning Properties of Class Relations

This chapter investigates the logical properties of the set of 17 class relations. It is not an exhaustive analysis; only the required properties with regard to the purpose of this work are considered. These are the restrictions on the number of instances, the symmetry, the composition and the conceptual neighbourhood of class relations. The set of 17 abstract class relations is independent of concrete instance relations. Thus, it appears to be useful to separately analyse the reasoning properties of the abstract class relations and those of the instance relations, instead of researching a high number of class relations, that combines the two. The findings on the reasoning properties of the abstract class relations can be flexibly used in combination with any kind of instance relation.

5.1 Correlation Between the Number of Instances and the Feasibility of Class Relations

Class relations do not specify a concrete number of instances for the related classes. Nevertheless, some of the relations constrain the number of instances.

For most entity classes the number of existing individuals is unknown or variable. For these classes a dependency between class relations and the number of individuals of a class is irrelevant. However, 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* and a fixed time. Knowledge about these numbers should be considered when SICs are defined and for the reasoning on class relations.

Some class relations are invalid if one or both involved classes have less than three instances. If both classes have one instance, only class relations based on $R_{LT.RT-all}(A, B)$ (#17) are

Table 5.1: Restrictions of the number of instances of the abstract class relations

Abstract Class Relation	Minimal Required Instances		Comparison Number of A and B	Abstract Class Relation	Minimal Required Instances		Comparison Number of A and B
	A	B			A	B	
1.$R_{LD.RD}$	2	2	-	10.$R_{RD.RT}$	3	1	$A > B + 1$
2.R_{LD}	2	3	-	11.R_{LT}	2	3	-
3.R_{RD}	3	2	-	12.R_{RT}	3	2	-
4.R_{some}	3	3	-	13.$R_{LD.RD.LT.RT}$	2	2	$A = B$
5.$R_{LD.RD.LT}$	1	2	$A < B$	14.$R_{LD.LT.RT}$	2	3	$A < B$
6.$R_{LD.RD.RT}$	2	1	$A > B$	15.$R_{RD.LT.RT}$	3	2	$A > B$
7.$R_{LD.RT}$	2	2	-	16.$R_{LT.RT}$	2	2	-
8.$R_{RD.LT}$	2	2	-	17.$R_{LT.RT-all}$	1	1	-
9.$R_{LD.LT}$	1	3	$A + 1 < B$				

possible. If the first class A has one instance, the only possible abstract class relations are $R_{LD.RD.LT}(A, B)$ (#5), $R_{LD.LT}(A, B)$ (#9) and $R_{LT.RT-all}(A, B)$ (#17). If the second class B has one instance, only $R_{LD.RD.RT}(A, B)$ (#6), $R_{RD.RT}(A, B)$ (#10) and $R_{LT.RT-all}(A, B)$ (#17) are allowed. Table 5.1 shows for all 17 abstract class relations the minimal required instances.

The definition of class relations can also be restricted if the number of instances is more than three. If the number of instances of one of the classes is known, some abstract class relations allow for conclusions about the number of instances of the other class. There are seven abstract class relations that define such a proportion between the instances of the two classes (Table 5.1). This is also of interest if a SIC is defined for a single class (i.e., $A = B$). Since this implies that the number of instances is equal for both classes, only $R_{LD.RD.LT.RT}(A, A)$ (#13) is consistent, while the other six proportion restricting abstract class relations are impossible.

The number of instances can also restrict the possible combinations of class relations between two classes. For example, if exactly two instances of A exist, not more than two class relations based on $R_{LD.RT}(A, B)$ (#7) can be defined for one set of JEPD instance relations. A third $R_{LD.RT}(A, B)$ relation would require at least one more instance of A.

5.2 Reasoning on the Symmetry of Class Relations

Donnelly and Bittner (2005) have studied the transfer of logical properties of instance relations to class relations, such as their symmetry, transitivity and reflexiveness. This section deepens the analysis of the symmetry properties of the class relations.

Spatial relations between instances are usually either symmetric or have a well-defined converse relation. The symmetry properties of the class relations can be derived from the symmetry of the applied instance relations and the cardinality definitions of the abstract class relations. The converse of a class relation bases on the converse of the applied instance relation. If an abstract class relation is left-total / left-definite the converse relation is right-total / right-definite, and vice versa. The relations R_{some} (#4) and $R_{LT.RT-all}$ (#17) are symmetric. Table 5.2 summarises this correlation between symmetry properties of instance relations and those of the corresponding class relations. Therefore, the following symbols are defined:

r^i	Converse instance relation of r.
R^i	Converse class relation of R.

The following examples demonstrate the derivation of converse class relations. The class relations base on the symmetric instance relation *overlap* and the converse relations *contains* and *inside*:

$$(OVERLAP_{RD.LT}(Watermill, Stream))^i \quad := \quad OVERLAP_{LD.RT}(Stream, Watermill).$$

$$(CONTAINS_{LD.LT.RT}(Airport, A.Tower))^i \quad := \quad INSIDE_{RD.LT.RT}(A.Tower, Airport).$$

The examples show that not all class relations are symmetric, even if they are based on symmetric instance relations. However, based on the cardinality properties it can be proven that if an instance relation is symmetric or has a converse relation, there exists a converse relation for each of the corresponding class relations.

Table 5.2: Symmetry properties of the class relations

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

5.3 Composition of Class Relations

The composition of binary relations enables 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 at least some of the possible relations can be excluded. The composition as a mathematical operation and the application with spatial instance relations has been elucidated in Section 2.4. A transfer of this reasoning formalism to the class level is very useful for the work with SICs and other applications of class relations. In general, the composition of class relations depends on the composition of instance relations. It is essential that 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 abstract class relations together with the 8 topological relations between regions (Figure 2.2) would result in 136 topological class relations and almost 18500 compositions. Since such an amount of compositions is rather inconvenient, a two-level reasoning formalism has been proposed (Mäs, 2007b), which separates the compositions of the abstract class relations from those of the instance relations. Thus the composition of the abstract class relations can be defined independently of a concrete set of instance relations. Figure 5.1 illustrates how the overall composition of class relations can be derived from the two levels.

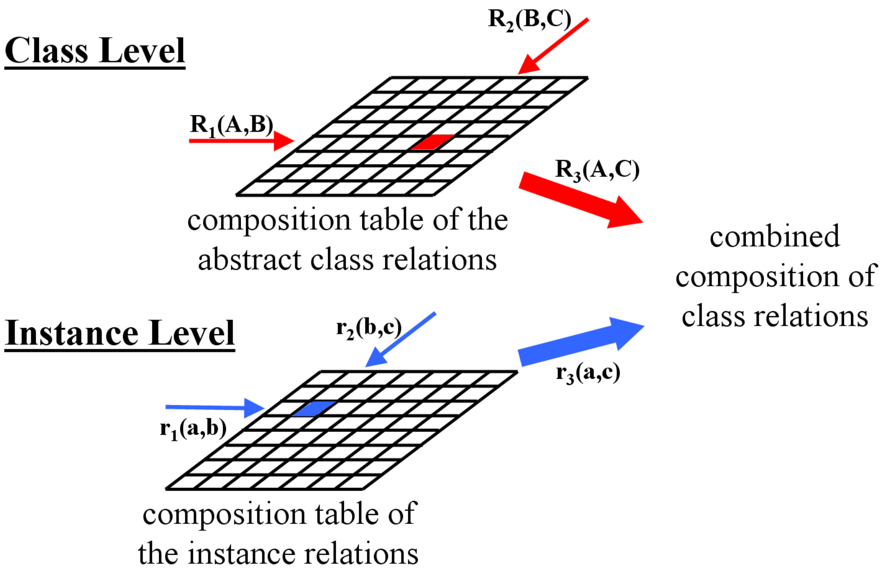

Class Level
$R_2(B,C)$
$R_1(A,B)$
$R_3(A,C)$
composition table of the abstract class relations
combined composition of class relations
Instance Level
$r_2(b,c)$
$r_1(a,b)$
$r_3(a,c)$
composition table of the instance relations

Figure 5.1: Two-level composition of class relations

The composition of instance relations is not further researched in this work. The following examples refer to the composition table of the topological relations between areal entities (Table 2.1).

The 17 abstract class relations have 289 compositions. Only some of them can be derived here, but the following three examples demonstrate the general inference approach and the application of the two-level composition.

The first example derives the composition for the introductory application scenario 1 (Chapter 1.1), which defines SICs between the entity classes airport, forest and airport tower. It was claimed that if the first two of the following constraints are defined, the third is inherently included and, therefore, redundant:

- Airports and forests are either disjoint or meet.
  (corresponds to $[MEET \cup DISJOINT]_{LT.RT-all}(Forest, Airport)$)
- Every airport contains at least one airport tower. Every airport tower is contained by an airport.
  (corresponds to $CONTAINS_{LD.LT.RT}(Airport, AirportTower)$)
- Forests and airport towers are disjoint.
  (corresponds to $DISJOINT_{LT.RT-all}(Forest, A.Tower)$)

This situation is shown in the ERD in Figure 5.2. With the two class relations between forest and airport and airport and airport tower given, the relation between the entity classes forest and airport tower can be derived. A possible arrangement of instance relations, which is consistent to the given SICs, is schematically represented in the left part of Figure 5.3. The arrows represent instance relations, while the cycles show instances of the different classes.

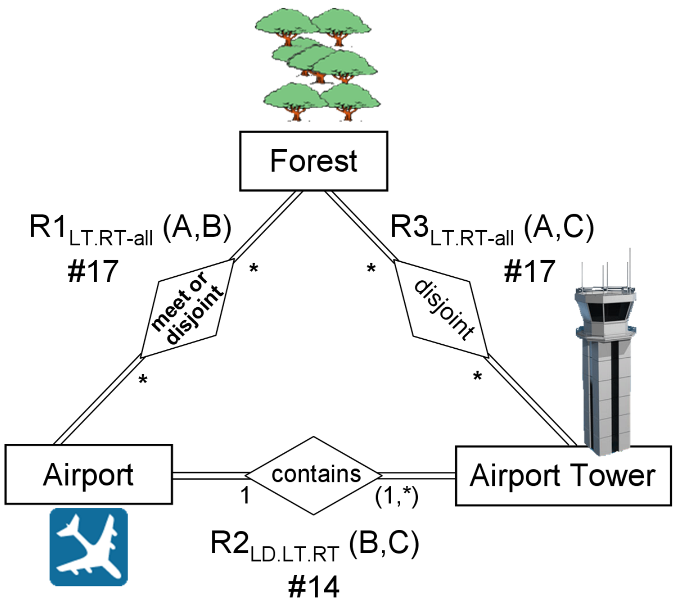


Figure 5.2: ERD of the SICs between the entity classes *forest*, *airport* and *airport tower*

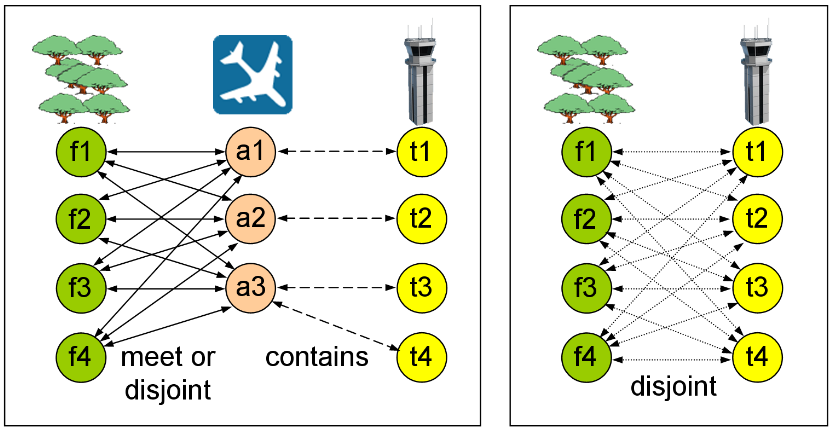


Figure 5.3: Possible arrangement of instance relations defined by the class relations between the classes *forest* / *airport* and *airport* / *airport tower* (left) and their composition (right)

In Figure 5.3 all forest instances have the same instance relation $r1$ ($meet \cup disjoint$) to all airport instances. All instance relations between the two classes are constraint and it is impossible to add a further arrow between these classes in the schema. Further, all airport instances have an $r2$ instance relation ($contains$) to at least one airport tower.

To infer the composition of the abstract class relations, every possible triple of forest, airport and airport tower instances has to be separately analysed. Whenever the relation $r1$ between the forest and the airport and the relation $r2$ between the airport and the airport tower is given, the relation $r3$ (or a disjunction of possible instance relations) between the forest and the airport tower instances can be inferred. For example, since forest $f1$ is related to airport $a1$ and airport $a1$ is related to airport tower $t1$, the relation between $f1$ and $t1$ can be deduced through composition (Table 2.1):

$$meet \cup disjoint(f1, a1) \ ; \ contains(a1, t1) \Rightarrow disjoint(f1, t1).$$

The combination of the inferences of all possible triples of instances leads to the abstract class relation between the classes forest and airport tower. For the example in Figure 5.3, each forest instance is related to every airport tower instance via at least one airport instance. Therefore, all forests must have the same instance relation $r3$ to all airport towers. Thus the composition of the abstract class relations must be (right box of Figure 5.3):

$$R1_{LT.RT-all}(Forest, Airport) \ ; \ R2_{LD.LT.RT}(Airport, A.Tower) \Rightarrow$$
$$R3_{LT.RT-all}(Forest, A.Tower).$$

The combination of the compositions of the two levels results in:

$$[MEET \cup DISJOINT]_{LT.RT-all}(Forest, Airport) \ ; \\ CONTAINS_{LD.LT.RT}(Airport, A.Tower) \Rightarrow \\ DISJOINT_{LT.RT-all}(Forest, A.Tower).$$

This result corresponds to the originally given SIC between forests and airport towers and the redundancy in the triple of SICs is proven for the arrangement shown in Figure 5.3. For these abstract class relations it is obvious, that the derived composition is independent of the number of instances and the relative arrangement of the instance relations. This means that the composition of the given abstract class relations will always lead to the same result.

The demonstrated two-level composition leads only to meaningful results if the composition of the instance relations does not result in a universal disjunction $\mathcal{U}$. Otherwise no inference is possible.

The 17 abstract class relations have 289 possible compositions. Many of them have differing results, depending on the number of instances of the three classes and the relative arrangement of the instance relations (Figure 5.4).

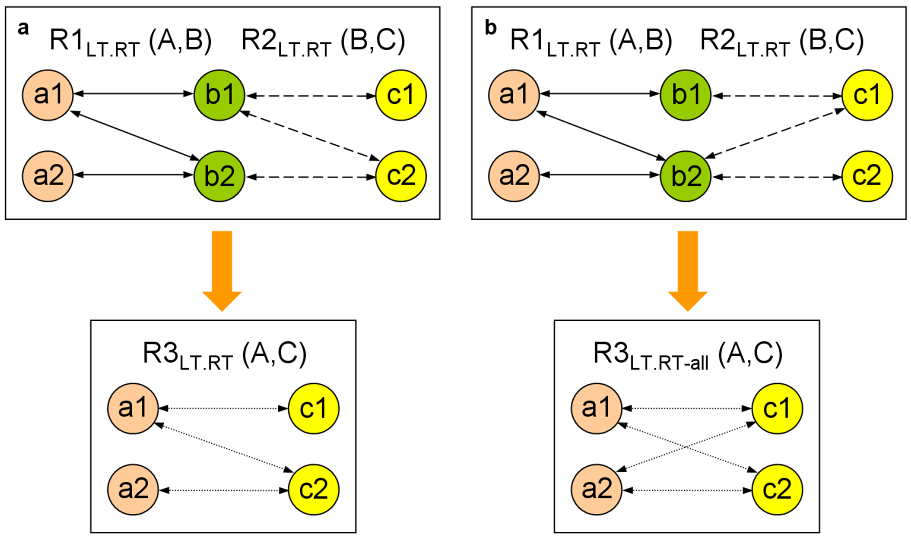


Figure 5.4: Influence of the relative arrangement of instance relations on the composition of the abstract class relations

The two boxes at the top of Figure 5.4 show possible constellations of the compositions of the $R1_{LT.RT}(A, B)$ and $R2_{LT.RT}(B, C)$ relations (#16). They only differ in an instance relation between the classes B and C: in the left upper box the instances $b1$ and $c2$ are related, whereas in the right upper box $b2$ and $c1$ are related. The number of instances of both classes, the abstract class relations and the total amount of instance relations, are equal. Nevertheless this difference leads to different compositions. For the left constellation the relation between the instances $a2$ and $c1$ cannot be inferred and the composition is $R3_{LT.RT}(A, C)$ (#16). The right constellation allows for a deduction of all four instance relations between A and C and thus the composition is $R3_{LT.RT-all}(A, C)$ (#17). Since the abstract class relations do not provide any information about the relative arrangement of the instance relations, none of the two results can be excluded. Thus the composition of the given abstract class relations is the disjunction of all possible results (logical XOR):

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

$R_{LT.RT}$ (#16) defines total participation constraints and excludes the unambiguousness cardinality constraints left-definite and right-definite for the involved classes (Equation 4.20). Thus, the composition of two such class relations will always lead to class relations that are again left-total and right-total and some of the instances of both classes will have more than one of the corresponding instance relations. Since only $R_{LT.RT}$ (#16) and $R_{LT.RT-all}$ (#17) fulfil this requirement, their disjunction is the exhaustive result of this composition.

Some compositions of abstract class relations have constellations in which no triple of instances with r1 and r2 relations exists and thus no inference for any r3 is possible. The third example (Figure 5.5) shows that the composition of $R1_{some}(A, B)$ (#4) and $R2_{LD}(B, C)$ (#2) is such a case.

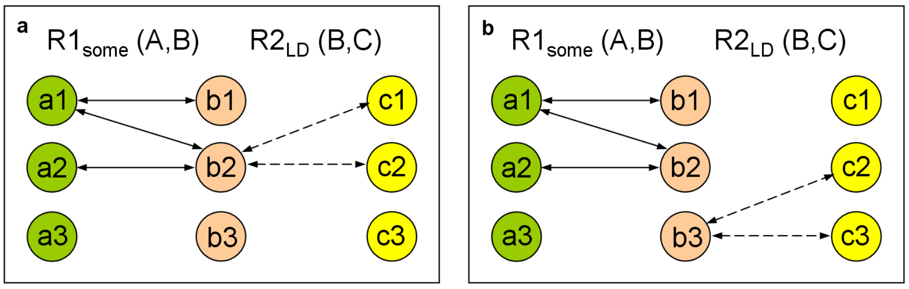


Figure 5.5: Composition of abstract class relations, which leads to a universal disjunction

The abstract class relations of the example define that some, but not all, instances of A and B are related by an instance relation $r1$ and some, but not all, instances of B and C are related by $r2$, while there is no C with more than one $r2$ relation. This does not imply that a triple of A, B and C instances exists that includes both instance relations $r1$ and $r2$ (Figure 5.5). While the situation in the left box allows for a deduction of the instance relations between $a1$, $a2$ and $c1$, $c2$, there is no inference possible for the arrangement of instance relations shown in the right box. Therefore, the composition of the class relations leads to a universal disjunction $\mathcal{U}_{\mathcal{U}}(A,C)$ of all possible abstract class relations and instance relations:

$$R1_{some}(A,B) \ ; \ R2_{LD}(B,C) \ \Rightarrow \ \mathcal{U}_{\mathcal{U}}(A,C).$$

This means that, based on the composition for the two given class relations, no implicit information can be deduced.

These examples show that the composition of the defined abstract class relations and the separation into two composition levels is possible. Nevertheless, a continuation of the manual analysis of all 289 possible compositions in such a way is hardly accomplishable. Even the automatic calculation of the overall composition table is complex and costly. Class relations neither specify the exact number of instances of the classes nor the exact arrangement of the instance relations. The defined abstract class relations require an analysis of all possible arrangements of instance relations for up to six instances for each of the three classes. If both classes of one relation have six instances than approximately 68,7 billion arrangements are possible. Each of these has to be separately analysed with all possible arrangements of the second relation. An analysis of classes with seven or more instances does not lead to additional results in the composition. With some heuristics, the calculation can be further optimised. The overall composition table is shown in Table 5.3 (Mäs, 2008).

Some of the compositions can be summarised by general rules, which deduce the composition results directly from the cardinality properties. This allows for a more convenient use of the composition. Some relatively obvious rules are:

- If the first abstract class relation is not right-total and the second relation is not left-total the composition is always a universal disjunction $\mathcal{U}_{\mathcal{U}}(A,C)$.
- If the first relation is $R1_{LD.RD.RT}(A,B)$ (#6) and the second is not left-total the composition is equal to the second abstract class relation.
- If the first relation is $R1_{LD.RD.RT}(A,B)$ (#6) and the second is left-total the composition has the same cardinality properties as the second relation, but it is not left-total. For $R2_{LT.RT-all}(B,C)$ (#17) this can be relation #6, #7, #10 or #12.
- If the first relation is not right-total and the second is $R2_{LD.RD.LT}(B,C)$ (#5) the composition is equal to the first abstract class relation.
- If the first relation is right-total and the second is $R2_{LD.RD.LT}(B,C)$ (#5) the composition has the same cardinality properties as the first relation, but it is not right-total. For $R1_{LT.RT-all}(A,B)$ (#17) this can be relation #5, #8, #9 or #11.

Table 5.3: Composition table of the 17 abstract class relations (Mäs, 2008)

2. Relation / 1. Relation		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
1	LD.RD	$\mathcal{U}$	$\mathcal{U}$	$\mathcal{U}$	$\mathcal{U}$	1	$\mathcal{U}$	$\mathcal{U}$	1,3	1,2	$\mathcal{U}$	1,2, 3,4	$\mathcal{U}$	1	1,2	1,3, 6,10	1,2,3, 4,6,7, 10,12	6, 7, 10,12
2	LD	$\mathcal{U}$	$\mathcal{U}$	$\mathcal{U}$	$\mathcal{U}$	2	$\mathcal{U}$	$\mathcal{U}$	1,2, 3,4	2	$\mathcal{U}$	1,2, 3,4	$\mathcal{U}$	2	2	1,2,3,4, 6,7,10, 12	1,2,3, 4,6,7, 10,12	6, 7, 10,12
3	RD	$\mathcal{U}$	$\mathcal{U}$	$\mathcal{U}$	$\mathcal{U}$	3	$\mathcal{U}$	$\mathcal{U}$	3	3,4	$\mathcal{U}$	3,4	$\mathcal{U}$	3	3,4	3,10	3,4, 10,12	10,12
4	some	$\mathcal{U}$	$\mathcal{U}$	$\mathcal{U}$	$\mathcal{U}$	4	$\mathcal{U}$	$\mathcal{U}$	3,4	4	$\mathcal{U}$	3,4	$\mathcal{U}$	4	4	3,4, 10,12	3,4, 10,12	10,12
5	LD.RD.LT	$\mathcal{U}$	$\mathcal{U}$	$\mathcal{U}$	$\mathcal{U}$	5	$\mathcal{U}$	$\mathcal{U}$	5,8	5,9	$\mathcal{U}$	5,8, 9,11	$\mathcal{U}$	5	5,9	5,8, 13,15	5,8,9, 11,13,14, 15,16,17	17
6	LD.RD.RT	1	2	3	4	1	6	7	3	2	10	4	12	6	7	10	12	6, 7, 10,12
7	LD.RT	1,2	2	1,2, 3,4	2,4	2	6,7	7	1,2, 3,4	2	6,7, 10,12	2,4	7,12	7	7	6,7, 10,12	7,12	6, 7, 10,12
8	RD.LT	$\mathcal{U}$	$\mathcal{U}$	$\mathcal{U}$	$\mathcal{U}$	8	$\mathcal{U}$	$\mathcal{U}$	8	8,11	$\mathcal{U}$	8,11	$\mathcal{U}$	8	8,11	8,15	8,11,15, 16,17	17
9	LD.LT	$\mathcal{U}$	$\mathcal{U}$	$\mathcal{U}$	$\mathcal{U}$	9	$\mathcal{U}$	$\mathcal{U}$	5,8, 9,11	9	$\mathcal{U}$	5,8, 9,11	$\mathcal{U}$	9	9	5,8,9,11, 13,14,15 16,17	5,8,9,11, 13,14,15, 16,17	17
10	RD.RT	1,3	2,4	3	4	3	6,10	7,12	3	4	10	4	12	10	12	10	12	10,12
11	LT	$\mathcal{U}$	$\mathcal{U}$	$\mathcal{U}$	$\mathcal{U}$	11	$\mathcal{U}$	$\mathcal{U}$	8,11	11	$\mathcal{U}$	8,11	$\mathcal{U}$	11	11	8,11, 15,16, 17	8,11, 15,16, 17	17
12	RT	1,2 3,4	2,4	1,2, 3,4	2,4	4	6,7, 10,12	7,12	3,4	4	6,7, 10,12	4	7,12	12	12	10,12	12	10,12
13	LD.RD.LT.RT	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
14	LD.LT.RT	1,2, 5,9	2,9	1,2,3, 4,5,8, 9,11	2,4, 9,11	9	6,7, 13,14	7,14	5,8, 9,11	9	6,7,10, 12,13,14, 15,16,17	9,11	7,12, 14,16, 17	14	14	13,14, 15,16, 17	14,16, 17	17
15	RD.LT.RT	1,3	2,4	3	4	8	6,10	7,12	8	11	10	11	12	15	16	15	16	17
16	LT.RT	1,2,3, 4,5,8, 9,11	2,4, 9,11	1,2,3, 4,5,8, 9,11	2,4, 9,11	11	6,7,10, 12,13,14, 15,16,17	7,12, 14,16, 17	8,11	11	6,7,10, 12,13,14, 15,16,17	11	7,12, 14,16, 17	16	16	15,16, 17	16,17	17
17	LT.RT-all	5,8, 9,11	9,11	5,8, 9,11	9,11	5,8, 9,11	17	17	5,8, 9,11	9,11	17	9,11	17	17	17	17	17	17

- If one of the relations is $R_{LD.RD.LT.RT}$ (#13) the composition is always equal to the corresponding other abstract class relation. Because of this property $R_{LD.RD.LT.RT}$ can represent the identity relation of classes if it is combined with an identity instance relation, for example $EQUAL_{LD.RD.LT.RT}(A, A)$.
- If the first relation is $R1_{LT.RT-all}(A, B)$ (#17) and the second is right-total the composition is always $R3_{LT.RT-all}(A, C)$.
- If the first relation is left-total and the second is $R2_{LT.RT-all}(B, C)$ (#17) the composition is always $R3_{LT.RT-all}(A, C)$.

The compositions that are defined by these rules, are highlighted in grey in Table 5.3. A set of rules that completely represents the composition table is the subject of further research and not investigated here.

For the abstract class relations and their compositions the properties of a relation algebra (Tarski, 1941) have been computationally checked. Some properties of the presented composition of abstract class relations are:

- The converse of a converse relation is equal to the original relation: $(R^i)^i = R$.
- All compositions with the identity relation (#13) are idempotent:

$$R \ ; \ R_{LD.RD.LT.RT} \ \Rightarrow \ R.$$
$$R_{LD.RD.LT.RT} \ ; \ R \ \Rightarrow \ R.$$

- The converse of a composition is equal to the composition of the converses of the two relations in reverse order:

$$(R1 \ ; \ R2)^i \ = \ R2^i \ ; \ R1^i.$$

- The associative property and the semiassociative property (Maddux, 1982) are not valid. The associative property holds if the compositions do not have different conclusions from two different reasoning paths. The semiassociative property proves if one of these reasoning paths produces a subset of the relations derived from the other one (Rodríguez and Egenhofer, 2000). Since the composition is neither associative nor semiassociative, the composition of the abstract class relations is nonassociative.

$$(R1 \ ; \ R2) \ ; \ R3 \ = \ R1 \ ; \ (R2 \ ; \ R3).$$
$$R \ ; \ (\mathcal{U} \ ; \ \mathcal{U}) \ = \ (R \ ; \ \mathcal{U}) \ ; \ \mathcal{U}.$$

Although the composition of class relations allows for some inferences about triples of class relations, a final conclusion requires a consideration of the conceptual neighbourhood of the class relations. The reasons for this are demonstrated in the following section.

5.4 Conceptual Neighbourhood of Class Relations

The conceptual neighbourhood represents continuous transformations between relations by linking relations that are connected by an atomic change. The conceptual neighbourhood of instance relations has been studied in Section 2.4. The conceptual neighbourhood of class relations has been introduced by Mäs (2008). In this approach, two class relations are considered as conceptually neighboured if they are linked to the same instance relation (or disjunction of instance relations) and they differ only in a single instance relation between two entities. The number of instances of the classes is considered fixed. In Figure 5.6 the conceptual neighbourhood of the class relations $R_{LD.RD}$ (#1) and R_{RD} (#3) is exemplarily illustrated. All arrows symbolise one instance relation of the same kind r. In the example, the addition of a further instance relation between the instances $a2$ and $b1$ in the right box leads to a transition of the class relation from $R_{LD.RD}$ to R_{RD}. An addition of an instance relation between other instances could also lead to other transitions.

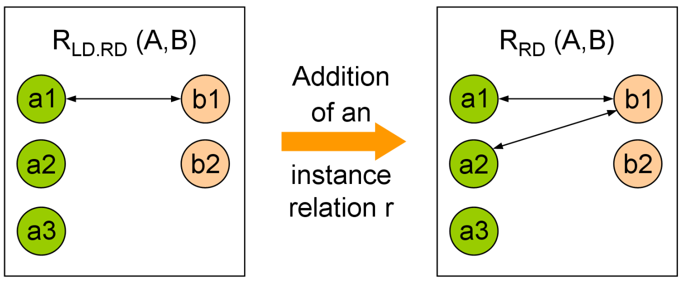


Figure 5.6: Conceptual neighbourhood between $R_{LD.RD}$ and R_{RD}

Figure 5.7 shows some of the possible transitions of the class relation if further instance relations are added to the class relation of the previous example. If an instance relation r is added to R_{RD} (#3), possible transitions are to $R_{RD.LT.RT}$ (#15) or to R_{RT} (#12) (Figure 5.7 - 2a and 2b). This shows that some class relations have several conceptual neighbours. Continuing the example, an r addition to $R_{RD.LT.RT}$ or to R_{RT} can lead in both cases to an $R_{LT.RT}$ (#16) class relation (Figure 5.7 - 3). The addition of another instance relation r to $R_{LT.RT}$ does not lead to a class relation transition. Thus the $R_{LT.RT}$ class relation is a conceptual neighbour of itself. If two r are added to the shown $R_{LT.RT}$ relation it finally becomes $R_{LT.RT-all}$ (#17) (Figure 5.7 - 4).

The 17 class relations have 45 conceptual neighbourhoods. Additionally nine class relations are conceptual neighbours of themselves (Table 5.4). The computation of all conceptual neighbourhoods between the defined class relations requires an analysis of all possible arrangements

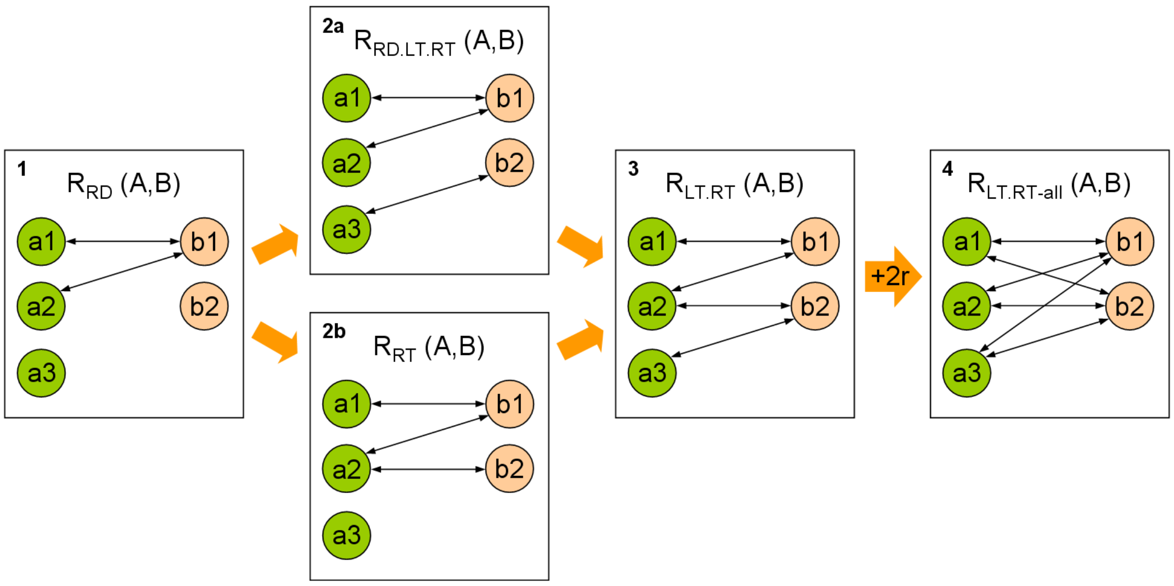


Figure 5.7: Conceptual neighbourhood between R_{RD}, $R_{RD.LT.RT}$, R_{RT}, $R_{LT.RT}$ and $R_{LT.RT-all}$

of instance relations for up to four instances for both classes. A higher number of instances does not lead to additional results.

Since the conceptual neighbourhood is defined through the addition or removal of a single instance relation, all neighbourhoods are directed. Table 5.4 represents the neighbourhoods which result from an addition ('+') and those which result from a removal ('−'). The symbol '±' marks class relations that are conceptual neighbours of themselves. If an addition or removal of an instance relation has changed a class relation it is impossible to get the same class relation again by further adding / removing of instance relations. The addition of instance relations ultimately leads to a $R_{LT.RT-all}$ (#17) class relation. A removal leads to $R_{LD.RD}$ (#1).[10]

The numbering of the abstract class relations has been chosen such that for all class relations the relations, that result from an addition, have a higher number and all converse relations are successive. Because of this order, all "removal neighbourhoods" appear in the lower triangle, while all "addition neighbourhoods" are in the upper triangle in Table 5.4.

The following example illustrates the practical use of the conceptual neighbourhood of class relations. Three class relations are defined for the classes A, B and C: $MEET_{some}(A, B)$ (#4), $CONTAINS_{LD.RD.LT.RT}(B, C)$ (#13) and $DISJOINT_{LT.RT}(A, C)$ (#16). These relations are analysed for conflicts through the comparison of the class relation composition

[10]Therefore both involved classes must have at least two entities (Table 5.1).

Table 5.4: Conceptual neighbourhood between the class relations: +/− corresponds to neighbourhood through addition / removal of an instance relation

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
1 LD.RD	±	+	+	+	+	+	+	+					+				
2 LD	-	±		+			+		+		+			+			
3 RD	-		±	+				+		+		+			+		
4 some	-	-	-	±							+	+				+	
5 LD.RD.LT	-								+		+			+			+
6 LD.RD.RT	-									+		+			+		+
7 LD.RT	-	-										+				+	
8 RD.LT	-		-								+					+	
9 LD.LT		-			-				±		+			+			+
10 RD.RT			-			-				±		+			+		+
11 LT		-		-	-			-	-		±					+	
12 RT			-	-		-	-			-		±				+	
13 LD.RD.LT.RT	-															+	
14 LD.LT.RT		-			-				-							+	
15 RD.LT.RT			-			-				-						+	
16 LT.RT				-			-	-			-	-	-	-	-	±	+
17 LT.RT-all					-	-			-	-						-	

$R(A,B)\ ;\ R(B,C)$ with the given third relation $R(A,C)$. The compositions of the corresponding instance and abstract class relations are:

$$meet(a,b)\ ;\ contains(b,c)\ \Rightarrow\ disjoint(a,c).$$

$$R1_{some}(A,B)\ ;\ R2_{LD.RD.LT.RT}(B,C)\ \Rightarrow\ R3_{some}(A,C).$$

Thus, the combination of the compositions of the two levels results in:

$$MEET_{some}(A,B)\ ;\ CONTAINS_{LD.RD.LT.RT}(B,C)\ \Rightarrow\ DISJOINT_{some}(A,C).$$

This result seems to be in conflict with the given third relation $DISJOINT_{LT.RT}(A,C)$. Figure 5.8 exemplarily illustrates this situation. The left box shows the given class relations and the middle box the inferred relation between A and C. In comparison with this, the right box shows that the given relation $DISJOINT_{LT.RT}(A,C)$ possibly differs from $DISJOINT_{some}(A,C)$ by only one disjoint instance relation (in this case $a3$ to $c3$). Thus $DISJOINT_{some}(A,C)$ and $DISJOINT_{LT.RT}(A,C)$ are conceptual neighbours. In Figure 5.8b the three disjoint instance relations of $DISJOINT_{some}(A,C)$ are implied by the class relations A to B and B to C. The composition does not allow for any conclusion about further relations between the instances of A and C. Also, it cannot be excluded that further pairs of A and C instances are *disjoint*. Hence the composition of $MEET_{some}(A, B)$ and $CONTAINS_{LD.RD.LT.RT}(B, C)$ does not contradict $DISJOINT_{LT.RT}(A, C)$ and the given triple of class relation is consistent. Beside $DISJOINT_{LT.RT}(A, C)$ (#16), also the direct conceptual neighbours of $DISJOINT_{some}(A, C)$ (#4) $DISJOINT_{LT}(A, C)$ (#11) and $DISJOINT_{RT}(A, C)$ (#12), and $DISJOINT_{LT.RT-all}(A, C)$ (#17) as a conceptual neighbour of $DISJOINT_{LT.RT}(A, C)$ have no conflict.

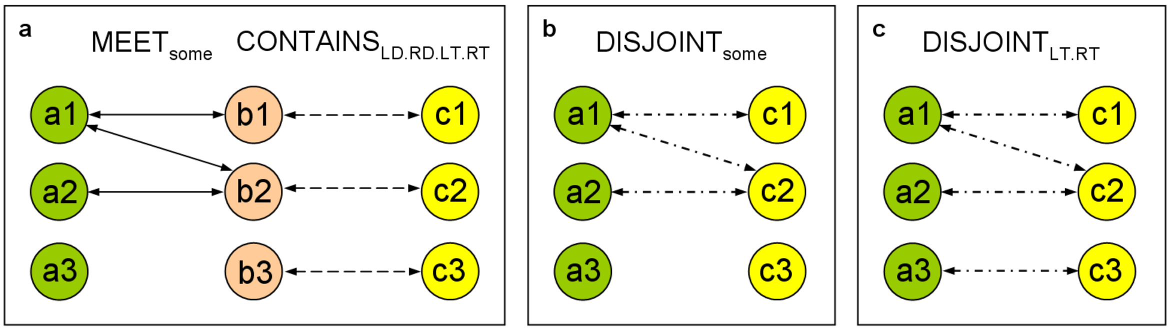


Figure 5.8: Use of the conceptual neighbourhood for the composition of class relations

In general, a class relation $R3$ is not in conflict with a composition $R1\ ;\ R2 \Rightarrow R3^*$ if $R3^*$ and $R3$ base on the same instance relation $r3$ and the addition of further $r3$ instance relations to $R3^*$ can lead to a transition to class relation $R3$. For this, the result of the composition $R3^*$ and $R3$ do not need to be direct conceptual neighbours. There can also be further class relation transitions between the two class relations. Nevertheless the conceptual neighbourhood points out which $R3$ class relations are valid, since it shows which transitions are possible for a certain class relation $R3^*$.

Thus the check of conflicts in a triple of class relations consists of two steps: at first the comparison of the composition of two relations with the given third. If they are equal, the triple of relations is conform to the introduced composition of class relations and there is no obvious conflict. If these two relations base on the same instance relation but have different abstract class relations, the second step checks their conceptual neighbourhood as described above. If the given third relation is not a corresponding conceptual neighbour of the composition the triple of class relations is in conflict. If the composition and the third relation base on different instance and abstract class relations they are not necessarily in conflict. In this case some more complex rules have to be checked for the detection of conflicts. These rules are discussed in the next chapter.

5.5 Summary

In this chapter the logical properties - the restrictions on the number of instances, the symmetry, the composition and the conceptual neighbourhood of the defined set of 17 class relations have been investigated. The definitions and reasoning rules of the class relations are described independently of a specific set of instance relations. The introduced two-level composition of class relations allows for a separate analysis of instance relations and abstract class relations. The overall reasoning formalism can be used with any spatial or non-spatial set of instance relations. The only requirements imposed on the instance relations are that they are part of a JEPD set of relations and have defined converse relations and compositions.

The calculation of the conceptual neighbourhood, and in particular the composition of the class relations, is costly in terms of calculation time. For a standard PC in 2008 (3GHz CPU and 2GB RAM), it takes approximately two months to calculate the compositions of the 17 class relations. If the set of class relations is extended, for example, by further cardinality ratio constraints (e.g., 0..2), this will increase these calculation costs exponentially. Because of this, further work should deal with the direct derivation of the reasoning properties of the class relations from their cardinality properties. This will deepen the understanding of the logics and support possible extensions by additional cardinality properties.

The described concepts separately analyse the reasoning properties of the abstract class relations and those of the instance relations. However, some combinations of instance and abstract class relations lead to conflicts that cannot be found in this way. For example, the combination of $EQUAL_{LD.RT}(A, B)$ (#7) and $R_{LD.RD.LT.RT}(B, C)$ (#13) is impossible (Figure 5.9). This is due to the specific identity properties of the *equal* instance relation and the cardinality properties of the two abstract class relations.

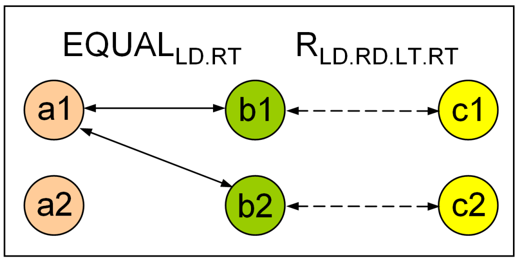


Figure 5.9: Inconsistent combination of class relations

$EQUAL_{LD.RT}(A, B)$ requires at least one instance of A that is *equal* to at least two instances of B, because it is defined as not right-definite (Equation 4.17). Since *equal* is symmetric and transitive this implies that the corresponding B instances are also *equal* ($b1$ and $b2$ in

Figure 5.9). Thus if one of these B instances has an instance relation to a C instance, the other B must have the same relation to this C. This means that both Bs should have the same instance relation to both Cs. This is in conflict with the $R_{LD.RD.LT.RT}(B, C)$ relation, because this class relation is defined as left-definite and right-definite (Equation 4.26). A general description of such conflicts is the subject of further research.

The following chapter shows how the described logics can be used to solve a class relation CSP and to find conflicts in sets of class relations.

Chapter 6

Basics for Checking the Consistency of Semantic Integrity Constraints

The previous two chapters discussed the formal definition of SICs through class relations and their reasoning properties. The application of these reasoning techniques for checking consistency in networks of binary relations is a constraint satisfaction problem (CSP). For networks of instance relations, such a proof of consistency has been elucidated in Section 2.4. The proof of consistency of class relation networks (i.e., sets of SICs) is similar, but mainly because of the following two reasons more complex:

1. It is possible to define more than one class relation between two classes, even if the applied instance relations are part of the same JEPD set of relations. Between instances only one relation of a JEPD set can be defined.
2. The class relations themselves are more complex than instance relations. While it is relatively simple to recognise if two instance relations are in conflict, such a comparison of class relations is extensive.

In analogy to a CSP at the instance level, a CSP solution of class relation networks is combining the proof of node, arc and path consistency, which is investigated in the following section. The subsequent sections provide rules and conditions for the detection of redundancies, conflicts and possible restrictions of class relations. These rules have to be checked during the CSP solution. Altogether, this enables to cope with the higher complexity of class relations.

6.1 Constraint Satisfaction Problems in Class Relation Networks

In a consistent network of JEPD relations, the following three conditions are fulfilled: node consistency, arc consistency and path consistency. This section investigates how the reasoning properties of class relations, researched in the previous chapter, provide the basis for checking these three consistency requirements in networks of class relations.

6.1.1 Node Consistency in Class Relation Networks

For instance relation networks, node consistency is ensured, if every node has an identity relation. For class relation networks this means that every class must have a relation to itself. This identity class relation bases on the corresponding identity instance relation, for example *equal* when using the topological relations of Figure 2.2. The selection of the abstract class relation depends on whether or not the instance identity relation fulfils the **identity criterion** (Guarino, 1999) for this class:

$$r_A^{id} \quad := \quad \forall x, y (Inst(x, A) \cap Inst(y, A) \cap r(x, y)) \;\Rightarrow\; x = y. \qquad (6.1)$$

This means that r_A^{id} holds for a class A if and only if there is no identity relation r between two different instances of A. Parcels are a typical example of a class for which the topological identity relation *equal* holds the identity criterion. If two parcel instances are *equal* then they also have the same geometry and the same identity. If the identity criterion holds for a class A and an instance identity relation r, the identity class relation is $R_{LD.RD.LT.RT}(A, A)$ (#13), for example for parcels $EQUAL_{LD.RD.LT.RT}(Parcel, Parcel)$ (Figure 6.1a).

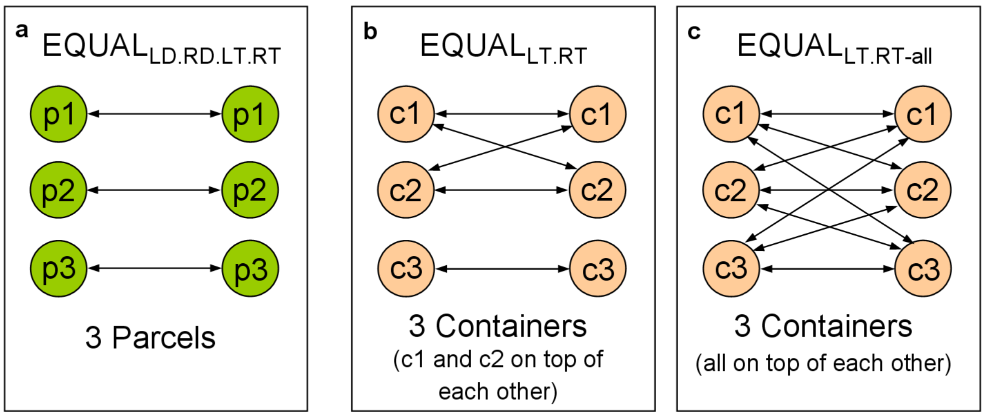


Figure 6.1: Identity class relations for the classes *parcel* and *shipping container*

Shipping containers are a counterexample of a class for which *equal* does not hold the identity criterion. Since two such containers can be on top of each other, they can be topologically *equal* in a two dimensional representation so that the identity criterion is not fulfilled for this relation. The identity class relation is, therefore, either based on the abstract class relation:

- $R_{LD.RD.LT.RT}(A, A)$ (#13) if no container instances are on top of each other (corresponds to Figure 6.1a),

- $R_{LT.RT}(A, A)$ (#16) if some but not all container instances are on top of each other (Figure 6.1b), or
- $R_{LT.RT-all}(A, A)$ (#17) if all container instances are on top of each other (Figure 6.1c).

At the level of the entity classes it is impossible to make an assumption on how many instances are on top of each other. Thus, the topological identity class relation for shipping containers is the disjunction of the three class relations:

$$EQUAL_{LD.RD.LT.RT}(C, C) \cup EQUAL_{LT.RT}(C, C) \cup EQUAL_{LT.RT-all}(C, C).$$

Whether an instance identity relation fulfils the identity criterion for a class, cannot be determined automatically. It should be part of the class definition in the conceptual data schema.

6.1.2 Arc Consistency in Class Relation Networks

A network of relations is arc consistent if every edge of the network has an edge in the reverse direction, that is, every relation has a converse relation. It has been shown in Section 5.2 that, if an instance relation is symmetric or has a converse relation, there is also a converse relation for each of the corresponding class relations.

6.1.3 Path Consistency in Class Relation Networks

For the proof of path consistency, the compositions of all possible node triples must be checked. Therefore, the two-level composition of the class relations (Section 5.3) can be used. The algorithm that checks the path consistency in class relation networks is similar to the one described in Section 2.4.3 for instance relation networks. It also consists of three procedures. The differences to the procedures at the instance level mainly result from the higher complexity at the class level. In particular, the detection of conflicts is more complex. While at the instance level an empty intersection of two relations detects an inconsistency, such a simple operation is not feasible at the class level. Because of this, the detection of conflicts and redundancies is now only executed in the first of the three procedures. This `ADD` procedure inserts a new relation r_{ij} of the entities i and j to the set of relations N, which contains all known relations. The procedure is also called, when class relations, derived through symmetry or composition, are added to the network.

```
To ADD $r_{ij}$
  begin
    OK $\leftarrow$ $true$;
    For each relation $[r_{ij}]_a$ of N do
    . begin
    .   If ($r_{ij}$ = $[r_{ij}]_a$)
    .     then begin
    .       check Redundancy;
    .       OK $\leftarrow$ $false$;
    .     end;
    .   If ($r_{ij}$ $\neq$ $[r_{ij}]_a$)
    .     then begin
    .       If ($r_{ij}$ is in conflict with $[r_{ij}]_a$)
    .       . then begin
    .       .   If (restricted $r_{ij}$ is not in conflict with $[r_{ij}]_a$)
    .       .     then begin
    .       .       ADD(restricted $r_{ij}$);
    .       .       OK $\leftarrow$ $false$;
    .       .     end;
    .       .   If ($r_{ij}$ is not in conflict with restricted $[r_{ij}]_a$)
    .       .     then begin
    .       .       remove $[r_{ij}]_a$ from N;
    .       .       ADD(restricted $[r_{ij}]_a$);
    .       .     end;
    .       .   If (restricted $r_{ij}$ is not in conflict with restricted
    .       .       $[r_{ij}]_a$)
    .       .     then begin
    .       .       remove $[r_{ij}]_a$ from N;
    .       .       ADD(restricted $[r_{ij}]_a$);
    .       .       ADD(restricted $r_{ij}$);
    .       .       OK $\leftarrow$ $false$;
    .       .     end;
    .       .   If (no consistent restriction is possible)
    .       .     then begin
    .       .       Inconsistency found;
    .       .       OK $\leftarrow$ $false$;
    .       .     end;
    .       . end;
```

```
38        .         If ($r_{ij}$ is not in conflict with $[r_{ij}]_a$)
39        .         . then begin
40        .         .    If ($[r_{ij}]_a$ is disjunction which contains $r_{ij}$)
41        .         .       then remove $[r_{ij}]_a$ from N;
42        .         .    If ($r_{ij}$ is disjunction which contains $[r_{ij}]_a$)
43        .         .       then OK $\leftarrow$ $false$;
44        .         . end;
45        .       end;
46        . end;
47        If (OK)
48          then begin
49            add $r_{ij}$ to N;
50            put pair <i,j> on Queue;
51          end
52      end;
```

Before the first class relation is inserted, all classes are related by $\mathcal{U}_{LT.RT-all}(A, B)$. This relation corresponds to the universal disjunction of the instance relations, which is defined when the path consistency is checked at the instance level.

When the ADD procedure is called, the r_{ij} relation is separately compared with all known relations between the entities i and j of the set of relations N. The variable a represents a counter for these relations. Relations that are part of a class relation disjunction, are also separately compared. The OK variable is initially set to *true*. If the comparison of relations detects a conflict or a redundancy this variable is set to $false$ to signalise that r_{ij} shall not be added to N.

The procedure first checks whether the two relations r_{ij} and $[r_{ij}]_a$ are equal (lines 6−10), which potentially indicates a redundancy. The detection of redundancies is discussed in Section 6.2.

If the two compared relations are not equal, the procedure continues with checking whether the relations contradict. The conditions that are analysed during this check (line 13) and a comparison to the instance level, are elucidated in Section 6.3.

If a conflict is detected it is sometimes possible to further restrict one or both of the compared relations to maintain a consistent set of class relations. This is tested in lines 15-31 of the pseudo code. Some possible restrictions are listed in Section 6.4. If $[r_{ij}]_a$ is restricted it is removed from the constraint set N. If a restricted relation shall be added to the constraint set, the ADD procedure is called recursively, because in order to assure consistency, the restricted relation has to be compared again with all known relations. If none of the possible restrictions leads to a consistent set of relations, an inconsistency is detected (lines 32-36).

Finally, if the two relations r_{ij} and $[r_{ij}]_a$ are not equal and do not conflict, but at least one of them is a disjunction, it has to be assured that this relation does not contain the other. If the new class relation r_{ij} is fully contained by a defined disjunction of class relations, the disjunction is deleted and replaced by the new class relation (lines 40-41). Correspondingly, if a new class relation disjunction is added, which fully contains a defined class relation $[r_{ij}]_a$, the disjunction is not added (lines 42-43). For example, $MEET_{some}(A, B)$ is fully contained in the disjunction $MEET_{some}(A, B) \cup INSIDE_{LT.RT}(A, B)$. Thus only $MEET_{some}(A, B)$ will remain in the set of class relations N.

If the OK variable still holds the value *true* after the comparison, the relation r_{ij} is added to the set of class relations N (line 49). In this case the constrained pair of entities i, j is placed in the queue for propagation (line 50).

```
1    To CHECKCONSISTENCY
2      While Queue is not empty do
3        begin
4          Get next <i,j> from Queue;
5          PROPAGATE(i,j);
6        end;
```

In analogy to the instance level CSP algorithm, the consistency check of the class relation set is started by the CHECKCONSISTENCY procedure, which calls the PROPAGATE procedure as long as there are entries in the queue.

The PROPAGATE procedure calculates the compositions of all $[r_{ij}]_a$ relations of the classes i and j with all known relations. Firstly, it calculates the compositions of $[r_{ij}]_a$ and $[r_{jk}]_b$ (lines 7-11) and then the compositions of $[r_{ki}]_c$ and $[r_{ij}]_a$ (lines 12-16) for all classes k. In contrast to the corresponding procedure used at the instance level, the composition results are not directly compared and added to the set of known relations N. For each composition result, the ADD procedure is called, which then checks for conflicts, redundancies and restriction possibilities, adds the consistent new relations to N and correspondingly puts the pair of nodes i, k or k, j in the queue for further propagation.

```
To PROPAGATE i,j
  begin
    For each node k do
      begin
      . For each relation $[r_{ij}]_a$ of N do
      .   begin
      .     For each relation $[r_{jk}]_b$ of N do
      .       begin
      .         $new_{ik} \leftarrow [r_{ij}]_a;[r_{jk}]_b$ ;
      .         ADD($new_{ik}$);
      .       end;
      .     For each relation $[r_{ki}]_c$ of N do
      .       begin
      .         $new_{kj} \leftarrow [r_{ki}]_c;[r_{ij}]_a$ ;
      .         ADD($new_{kj}$);
      .       end;
      .   end;
      end;
  end;
```

6.2 Detection of Redundancies

A set of class relations contains a redundancy if an explicitly defined constraint can be deduced from the other explicit relations of the set. With the described algorithms, it is possible to detect redundancies if an explicitly defined class relation can be *directly* derived through symmetry or composition of other explicit class relations or the composition with their converse relations. To accomplish this it is necessary to record whether a relation is explicitly defined or the result of an identity, symmetry or composition inference. Compositions that base on composition results, cannot directly point out redundancies, because the possibly redundant relation might have been one of the originally composed relations. A solution of this problem is to store which explicit relations have been involved in the deduction of an implicit relation. Otherwise, it is impossible to detect all redundancies with the described algorithms.

The aim of a redundancy-free set of relations can be extended towards the minimal subsets of a class relation set, that is, a set with a minimal number of relations that completely contains the knowledge of the overall set. Different approaches to calculate such minimal subsets for temporal interval relations have been researched (Rodríguez et al., 2004). For SICs as an application of class relations such subsets are of particular interest, since they would enable the minimisation of the number of SICs that have to be checked during quality assurance. The calculation of such minimal subsets for instance relation sets is complex and a transfer to the class level ambitious. Because of this, it is beyond the scope of this thesis and a subject of future research.

6.3 Detection of Conflicts

A set of class relations is in conflict if the set is not realisable by a data set. This means that there is no data set that fulfils all constraints, defined by the class relations. Such class relation sets are logically inconsistent.

A conflict is found if a class relation is added to a defined set of relations and the relation is in contradiction to an implicitly or explicitly defined class relation between the same classes. Such a comparison of two class relations is more complex than the corresponding comparison of (spatial) relations at the instance level. In an implementation of an instance relation network, each relation can be represented by a vector of binary elements (Hernández, 1994). The length of the vector corresponds to the number of possible relations. This number is usually relatively small, as the relations are qualitative representations. Each element of the vector is associated to one of the relations. If a vector element is set to true, this relation is feasible for the two entities, otherwise the relation is excluded. The comparison of the relations bases on the comparison operations of the vectors. Two relations are in conflict if they relate the same entities without at least one vector element in common that is set to true.

Such an approach is infeasible for class relations due to two reasons. First is the huge number of different class relations. For example, the combination of the eight topological relations of areal entities with the 17 abstract class relations yields 136 class relations. If all possible 12.869 disjunctions of the eight instance relations are considered more than 200.000 different class relations can be defined. When the binary element vectors have such a size, their handling and comparison is inefficient. The second reason is: the fact that more than one valid class relation can be defined between two classes cannot be considered this way. Thus, some explicit rules are required to allow for the comparison of class relations between the same classes.

In the following some relatively obvious conditions on class relations that constrain the same classes are listed. The prove of such rules has a great impact on the usefulness of the overall algorithm. If the applied rule set is too weak, conflicts might not be recognised. On the other hand, since the rules have to be checked whenever a class relation is added or changed in the overall set, they are also affecting the calculation costs of the consistency check. The list of conditions provided below is not necessarily exhaustive and can be extended if further rules are derived. Tests with a prototypical implementation (Chapter 7) have shown that this set of rules achieves considerable results.

The first group of conditions must be checked if class relations are defined for a single class ($R(A, A)$). Every class can have, apart from the identity class relation, further class relations to itself. These relations must fulfil the following conditions:

- A class relation based on the abstract class relations #5, #6, #9, #10, #14 or #15 is infeasible as $R(A, A)$. This results from the fact that, if a class relation is defined for a single class, the number of instances is equal on both sides of the relation. This is not conform to the restrictions on the proportion of the number of instances of the six referred abstract class relations (Section 5.1).
- A class relation based on the abstract class relation $R_{LD.RD.LT.RT}(A, A)$ (#13) and an instance identity relation cannot include further instance relations as a disjunction. All instances of the class have an identity relation to themselves. At the class level this is expressed, for instance, by $EQUAL_{LD.RD.LT.RT}(A, A)$. A disjunction with another instance relation also changes the abstract class relation. Thus a class relation, such as $[EQUAL \cup MEET]_{LD.RD.LT.RT}(A, A)$ is impossible.
- Class relations that include an identity instance relation must be based on the abstract class relations $R_{LD.RD.LT.RT}(A, A)$ (#13), $R_{LT.RT}(A, A)$ (#16) or on $R_{LT.RT-all}(A, A)$ (#17). As defined by the previous condition, the combination of class relation #13 with an identity instance relation corresponds to the identity relations of all instances. According to the conceptual neighbourhood of this abstract class relation (Section 5.4) the inclusion of further instance relations leads to a transition to the abstract class relations #16 or #17.
- Class relations based on the abstract class relation $R_{LT.RT-all}(A, A)$ (#17) must include the identity instance relations, since it restricts all relations of the instances of the class and, therefore, also the identity relations of the instances.
- If the instance relation (or disjunction of relations) of a $R(A, A)$ class relation is symmetric, the applied abstract class relation must also be symmetric. If such a class relation is based on a non-symmetric abstract class relation the corresponding converse defines different cardinality restrictions on the same instance relation. This is in conflict to the original relation.

The second group of conditions must be fulfilled by all class relations that are added. The conditions specify rules for the comparison of class relations that relate the same classes.

- It is impossible to define two class relations for the same classes based on the same instance relation (or the same disjunction of instance relations).
- If the instance relation (or disjunction of instance relations) of one class relation is fully contained in the disjunction of instance relations of a second class relation between the same classes, the transition of the former to the latter abstract class relation through the addition of instance relations must be possible. This can be proven with the conceptual neighbourhood of the abstract class relations.
  For example, $[MEET \cup COVERS]_{LD.RD}(A, B)$ (#1) is not in conflict with $[MEET \cup COVERS \cup OVERLAP]_{LD.LT.RT}(A, B)$ (#14), since an addition of *overlap* instance relation(s) to $R_{LD.RD}(A, B)$ can lead to $R_{LD.LT.RT}(A, B)$. $R_{LD.RT}(A, B)$ (#7) would not fulfil this requirement, for instance.
- There can only be one class relation between two classes based on $R_{LT.RT-all}$ (#17).
- An $R_{LT.RT-all}(A, B)$ (#17) class relation defines the possible instance relations for the other class relations between these classes. For example, all class relations between A and B must be based on the instance relations *meet* or *disjoint*, if $[MEET \cup DISJOINT]_{LT.RT-all}(A, B)$ is defined.
- If the number of instances of both constrained classes are known some proportion restricting abstract class relations and some combinations of abstract class relations are impossible (Chapter 5.1).

6.4 Restriction of Class Relations

A detected conflict needs not necessarily point out an inconsistency in the constraint set. If two class relations have a conflict, it is sometimes possible to restrict one or both of them further, to arrive at a consistent set of class relations. Two restriction possibilities must be considered:

- If a class relation that is part of a new class relation disjunction is in conflict with a defined class relation, the conflicting class relation is deleted from the disjunction. The same happens, if the single class relation is added and the disjunction has been previously defined. For example, the first relation of the disjunction $INSIDE_{LT.RT-all}(A, B)$ $\cup$ $CONTAINS_{LT.RT}(A, B)$ is in conflict with $[MEET \cup CONTAINS]_{LT.RT-all}(A, B)$. Thus the relation is deleted from the disjunction and only $CONTAINS_{LT.RT}(A, B)$ and $[MEET \cup CONTAINS]_{LT.RT-all}(A, B)$ remain in the set of class relations.
- Some inconsistencies can be corrected by restricting the applied instance relations of one of the conflicting class relations. For example $[MEET \cup COVERS]_{LT.RT-all}(A, B)$ and $[MEET \cup INSIDE]_{LT.RT}(A, B)$ are in conflict with respect to the fourth condition of

the second group: the *inside* instance relation creates a conflict, because it is not contained in the disjunction of instance relations of the $R_{LT.RT-all}(A, B)$ class relation. If the second class relation is restricted to $[MEET]_{LT.RT}(A, B)$ it is still in accordance with the original relation, and it is also consistent with the first relation. Similarly, the class relation $[MEET \cup COVERS]_{LT.RT}(A, B)$ is also in conflict with the first relation, because it has the same instance relations (first condition of the second group). For a consistent result it can be restricted to the $[MEET]_{LT.RT}(A, B) \cup [COVERS]_{LT.RT}(A, B)$ class relation disjunction.

During the consistency check, a set of class relations is assumed to be consistent until a conflict is found that cannot be corrected by one of these methods.

6.5 Summary

The introduced CSP algorithm enables the detection of conflicts and redundancies in networks of class relations. In general, the algorithm is similar to the one discussed for the instance relations. However, due to the higher complexity, the detection whether two relations between the same classes are in conflict is more extensive.

The chapter provides a set of conditions for the detection of conflicts and redundancies. These rules have a great impact on the usefulness of the overall algorithm. Due to unconsidered conditions, conflicting class relations might pass without recognition. On the other hand, they also influence the calculation time of a consistency check, since all rules have to be checked for each entry in the set of known class relations.

The algorithm does not allow for a resolution of the detected conflicts. The constraint relaxation algorithms (Hernández, 1994) required for this are outside the scope of this work. In general, it is useful to keep the origin of a class relation as metainformation, that is, if it is explicitly defined, an identity relation or deduced through symmetry or composition. This information can also help users to find the class relations which cause conflicts.

Chapter 7

Application

The proposed reasoning methodology has many applications. The following two sections present the results and experiences of a prototypical implementation of the algorithm for checking the consistency of SIC sets. In the third section, the resolution of the second application scenario demonstrates the practical use of this implementation. Finally, some further application areas are discussed.

7.1 Prototypical Implementation

The proposed algorithm for checking the consistency of sets of SICs has been implemented as a plug-in extension of the ontology modelling and knowledge acquisition platform Protégé. Protégé is a free, open source ontology editor and knowledge base framework. It implements a rich set of knowledge modelling structures and actions that support the creation, visualisation, and manipulation of ontologies in various representation formats. Protégé can be extended by way of a plug-in architecture and a Java-based Application Programming Interface (API) for building knowledge-based tools and applications.[11] The SIC Checker Plug-in is implemented as a so-called tab widget plug-in. This plug-in type allows for functionality and application extensions, which appear on a new tab in the Protégé user interface. The user interface of the SIC Checker Plug-in is discussed in the next section.

Figure 7.1 schematically illustrates the information exchange between the Protégé ontology editor and the SIC Checker Plug-in. Protégé provides a Knowledge Base (KB), which contains the specified concepts of the ontology. This KB is accessible for plug-ins via a Java API. The SIC Checker Plug-in utilises the tree of the class concepts from the Protégé KB. A further information exchange is currently not implemented. One of the reasons for choosing the Protégé ontology editor as a basis platform for the implementation was the possibility to combine the developed reasoning methodology with existing inference capabilities of Protégé and to transfer the defined SICs back to the KB of the ontology editor for integration and

[11]http://protege.stanford.edu/ (last visited at 3th August 2009)

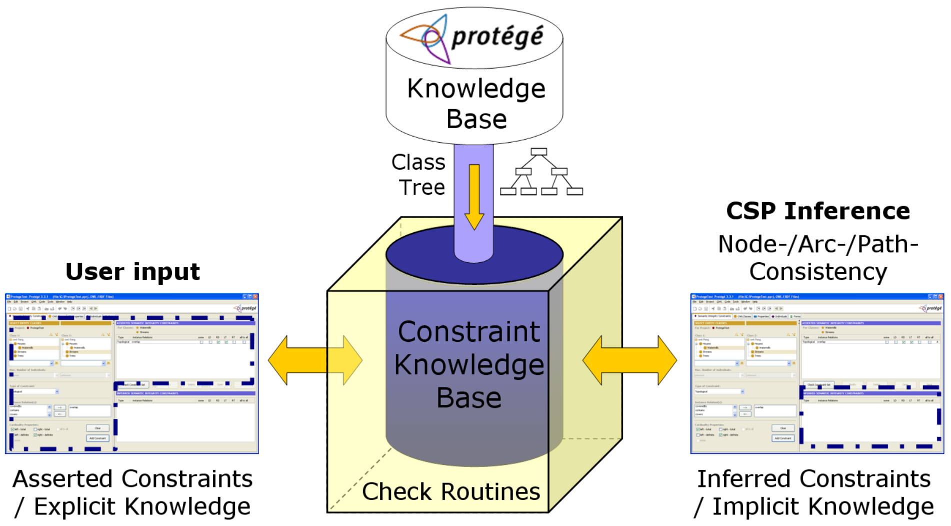

protégé
Knowledge Base
Class Tree
User input
Asserted Constraints / Explicit Knowledge
Constraint Knowledge Base
Check Routines
CSP Inference
Node-/Arc-/Path-Consistency
Inferred Constraints / Implicit Knowledge

Figure 7.1: Information exchange between Protégé ontology editor and SIC Checker Plug-in

possibly storing in standard ontology languages like for example the Semantic Web Rule Language (SWRL). This remains open for future work.

The SIC Checker Plug-in allows to specify SICs for the classes defined in the Protégé KB. The SICs are stored in the Constraint KB, which is managed by the plug-in and separated from the Protégé KB. The Constraint KB also contains the lists of relations for each set of instance relations and the abstract class relations, references to the corresponding identity relations and the symmetry and composition rules. Such an architecture with separated KBs and reasoners can be used to overcome deficiencies in expressiveness and reasoning capabilities in one of the system components (Grütter and Bauer-Messmer, 2007).

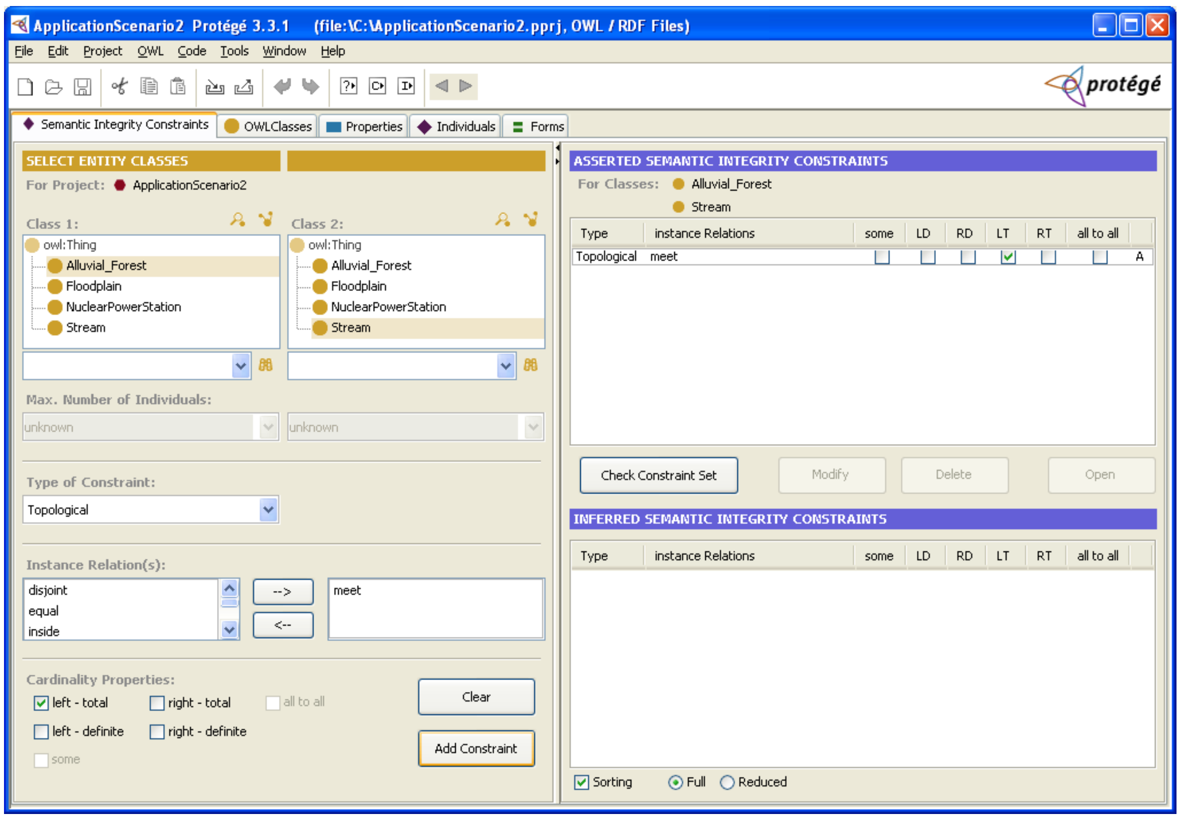
The SICs, which are entered into the Constraint KB originate either from user input or from inference. Before a SIC of one of these sources is added to the Constraint KB the constraint has to pass checking routines, which detect redundancies and conflicts with the SICs already stored in the Constraint KB. These checking routines are described in Sections 6.2 and 6.3. Possible restrictions of class relations, which are elucidated in Section 6.4, are only applied on the inferred SICs. SICs, which are asserted by the user, are considered as fully consistent and therefore not restricted.

The inference of SICs is based on the CSP solution of class relation networks, which has been researched in Section 6.1. This allows inferring identity[12] and converse relations, as well as compositions of SICs. Reasoning on knowledge about the maximal number of existent instances of the classes (Section 5.1) is not implemented in the prototype.

[12]For the prototype it is assumed that the instance identity relation fulfils the identity criterion (Equation 6.1) for all classes.

7.2 User Interface

Figure 7.2 shows a user interface for the definition and consistency check of SICs between two classes. Therewith the definition of a SIC, based on the defined class relations, consists of four main steps (left part of Figure 7.2):


ApplicationScenario2 Protégé 3.3.1 (file:\C:\ApplicationScenario2.pprj, OWL / RDF Files)
File Edit Project OWL Code Tools Window Help
protégé
Semantic Integrity Constraints
OWLClasses
Properties
Individuals
Forms
SELECT ENTITY CLASSES
owl:Thing
Alluvial_Forest
Floodplain
NuclearPowerStation
Stream
disjoint
equal
inside
-->
<--
Clear
Add Constraint
ASSERTED SEMANTIC INTEGRITY CONSTRAINTS
Check Constraint Set
Modify
Delete
Open
INFERRED SEMANTIC INTEGRITY CONSTRAINTS

For Project: ApplicationScenario2

Class 1: Alluvial_Forest

Class 2: Stream

Max. Number of Individuals: unknown | unknown

Type of Constraint: Topological

Instance Relation(s): meet

Cardinality Properties: [✓ left - total] right - total all to all left - definite right - definite some

For Classes: Alluvial_Forest, Stream

Type	instance Relations	some	LD	RD	LT	RT	all to all	
Topological	meet	☐	☐	☐	✓	☐	☐	A

Type	instance Relations	some	LD	RD	LT	RT	all to all

[✓ Sorting] [✓ Full] Reduced

Figure 7.2: Screenshot of the user interface of the Protégé SIC Checker Plug-in

Firstly, the user selects the entity classes, which shall be restricted by the SIC. After the selection, explicitly and implicitly defined SICs between these two classes are displayed in the tables on the right side of the window. For the definition of spatial SICs 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 encoded in XML Metadata Interchange (XMI) or Geography Markup Language (GML) application schema 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. For the prototypical implementation, only areal geometry types are considered. For non-spatial SICs, like for example temporal constraints, the restricted attributes and the attribute type (e.g., temporal interval) must be known or entered by the user. If the user knows of the maximal number of existent instances of a class, he or she can enter them after the entity class selection. These input boxes are not enabled in the user interface, since the reasoning algorithms, which check their consistency (Section 5.1), have not been included in the prototypical implementation.

Secondly, the user selects the type of SIC he or she wants to define, for instance a topological or directional spatial SIC or a non-spatial constraint such as a temporal SIC. These types are classified according to the semantic domains of the instance relations (Chapter 3).

In the third step, the user selects one or more instance relations, which the class relation of the desired SIC is based on. The assortment of instance relations, made available by the interface, is adjusted to the geometry types (or attribute types of non-spatial SICs) of the entity classes and the type of SIC, 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, 1990) for the region, line and point geometries is necessary.

The fourth step is the selection of the cardinality properties of the class relation, which have been introduced in Section 4.2. The class relation properties can be separately activated, which is 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 “some” check box is not enabled, if one of the others is checked.

The final class relation results from the combination of the selected instance relation(s) and the cardinality properties. The new SIC is added to the Constraint KB when the “Add Constraint” button is pressed. The interface shown in Figure 7.2 contains the settings of the example SIC, which defines an $MEET_{LT}$ relation between the classes $alluvialforest$ and $stream$. The same SIC is shown in the table of asserted constraints on the right top of the figure.

The inference of implicit SICs is started when the user presses the “Check Constraint Set” button. This enables to control the reasoning process, which is the main objective of the prototype. For a realistic application it would be more reasonable to call the reasoning algorithms directly when the SICs are defined by the user. This would enable to discover inconsistencies as early as possible. The SICs, which have been derived for the selected classes, are visualised in the table of inferred constraints on the right bottom of the figure.

Further buttons allow to clear the SIC input mask, and to modify and delete SICs from the table of asserted constraints.

7.3 Harmonisation and Integration of Semantic Integrity Constraints

The objective of this section is to illustrate the application of the developed reasoning algorithms and the prototypical implementation in practice and to show, how results of the inference can be interpreted. Therefore the issues of harmonisation and integration of SICs, which have been addressed in the second application scenario (Section 1.2), are lead to a solution. To recall the application scenario Figure 7.3 repeats its illustration.

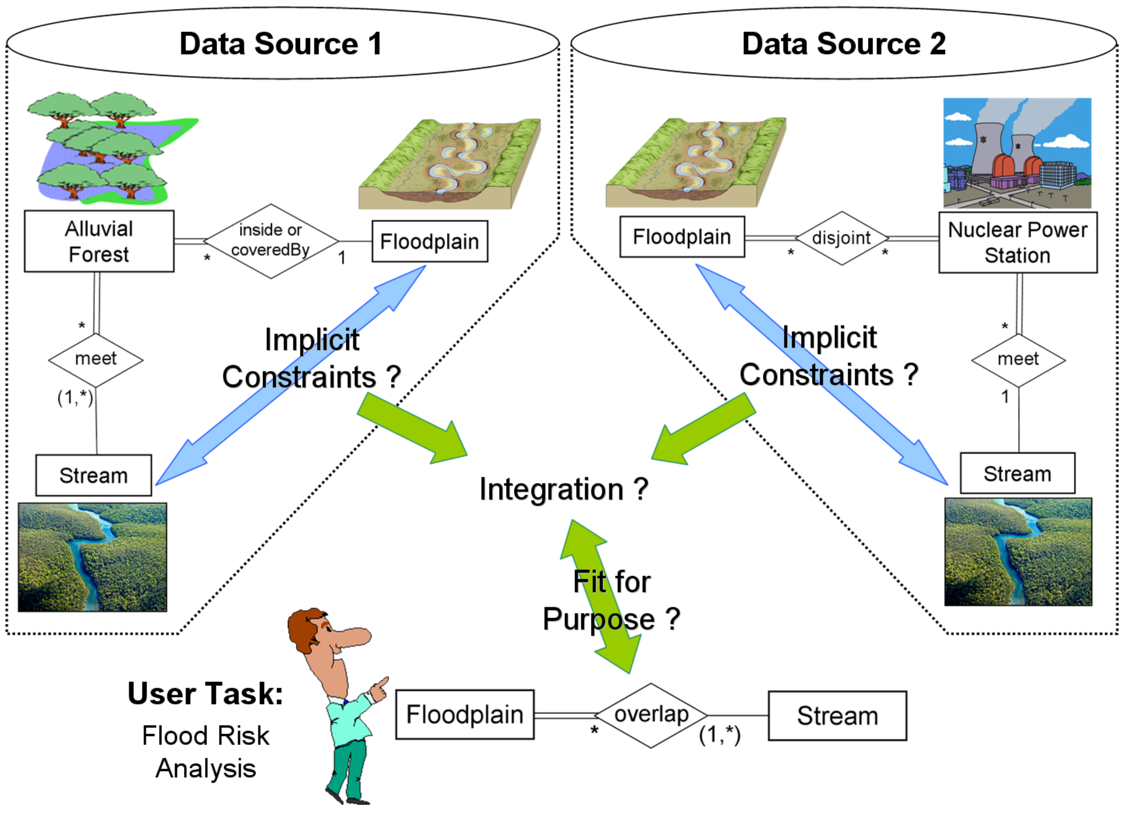


Figure 7.3: Scenario for the harmonisation of spatial SICs (repetition of Figure 1.2)

In the scenario, a user wants to analyse the flooding risk of a particular area. To enable this analysis the data has to conform to the following SIC:

- All floodplains must overlap with at least one stream.
  (corresponds to $OVERLAP_{LT}(Floodplain, Stream)$)

Two data providers offer the corresponding information about floodplains and streams for the required region. The user downloads metadata of the two data sources. These include the data schemas and the SICs, which have been checked during the quality assurance of the data sets. The first data source offers data for the three entity classes *floodplain*, *stream* and *alluvial forest* with the following SICs:

- All alluvial forests must be inside or covered by a floodplain.
  (corresponds to $[INSIDE \cup COVEREDBY]_{RD.LT}(\mathit{AlluvialForest}, \mathit{Floodplain})$)
- All alluvial forests must meet with at least one stream.
  (corresponds to $MEET_{LT}(\mathit{AlluvialForest}, \mathit{Stream})$)

Similar the second data source offers three entity classes with the constraints:

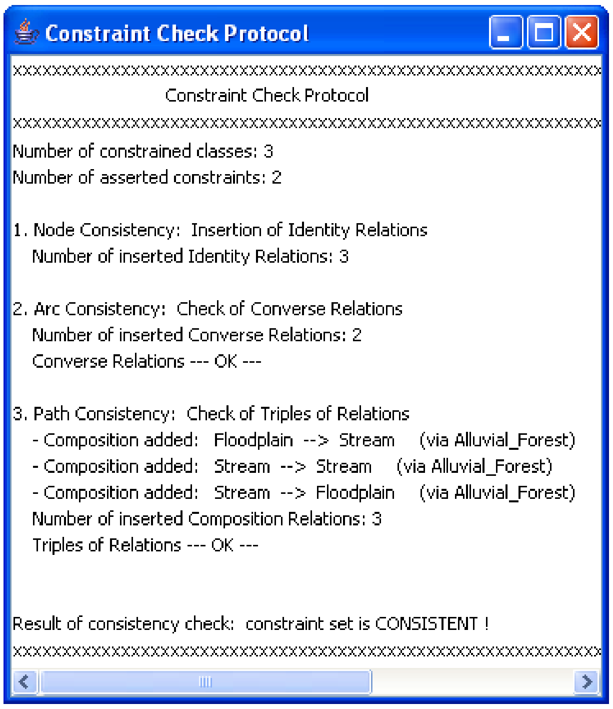
- All nuclear power stations must be disjoint from all floodplains.
  (corresponds to $DISJOINT_{LT.RT-all}(\mathit{NuclearPowerStation}, \mathit{Floodplain})$)
- All nuclear power stations must meet with exactly one stream.
  (corresponds to $MEET_{RD.LT}(\mathit{NuclearPowerStation}, \mathit{Stream})$)

Based on this situation three questions have been raised in the introduction of this thesis, which have to be answered to decide whether the data fit for the purposes of the user:

1. Are there implicit SICs defined within the two constraint sets?
The providers of both available data sets have not proven a SIC between the classes *floodplain* and *stream*. Therefore it has to be checked, if the constraint sets of the data sources have implicitly defined constraints between these classes. To deduce the constraints for the first data source the two given SICs have to be entered into the Protégé SIC Checker Plug-in. The process and results of the consistency check of these constraints are summarised in the checking protocol shown in Figure 7.4. The proof of node consistency led to an insertion of the three identity relations of the involved classes into the Constraint KB. During the proof of arc consistency, the converse relations of the two given SICs have been inserted. The composition inference during the proof of path consistency derived three implicitly defined relations. One of them is between the required classes floodplain and stream. The checking algorithm calculates all possible compositions of the asserted and inferred SICs. Only those, which derived new knowledge, are listed in the checking protocol. The overall result of the consistency check confirms, that the constraint set of data source one is (not surprisingly) consistent.

The derived relation between the classes *floodplain* and *stream* is visualised in the table of inferred SICs in Figure 7.5. This SIC is a disjunction of nine relations, which base on the same disjunction of instance relations and differing abstract class relations. The disjunction of instance relations is derived as follows (the first relation in the equation results from the converse relation of the given SIC between *alluvial forest* and *floodplain*; Table 2.1):

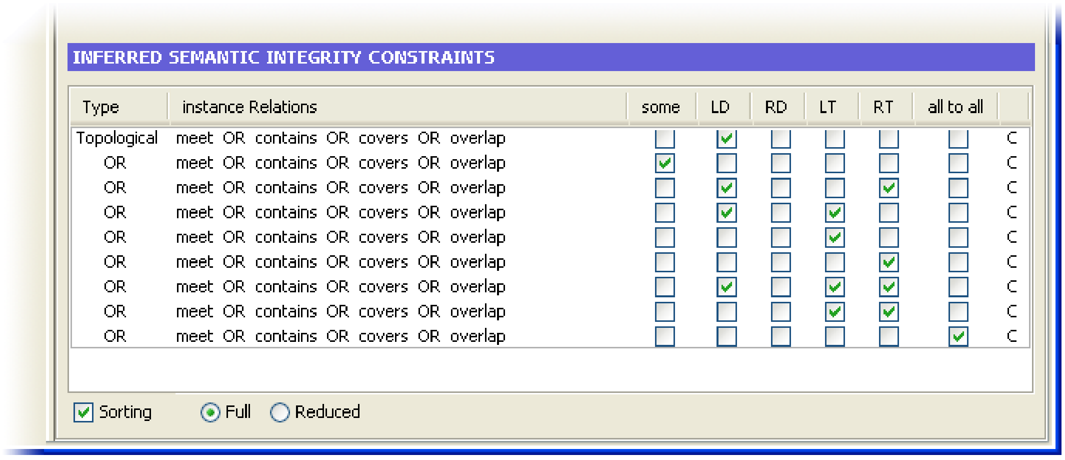
$$\mathit{contains} \cup \mathit{covers}(f1, a1)\ ;\ \mathit{meet}(a1, s1) \Rightarrow \mathit{meet} \cup \mathit{contains} \cup \mathit{covers} \cup \mathit{overlap}(f1, s1).$$



Constraint Check Protocol

```
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
                    Constraint Check Protocol
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
Number of constrained classes: 3
Number of asserted constraints: 2

1. Node Consistency:  Insertion of Identity Relations
   Number of inserted Identity Relations: 3

2. Arc Consistency:  Check of Converse Relations
   Number of inserted Converse Relations: 2
   Converse Relations --- OK ---

3. Path Consistency:  Check of Triples of Relations
   - Composition added:   Floodplain  -->  Stream    (via Alluvial_Forest)
   - Composition added:   Stream  -->  Stream    (via Alluvial_Forest)
   - Composition added:   Stream  -->  Floodplain    (via Alluvial_Forest)
   Number of inserted Composition Relations: 3
   Triples of Relations --- OK ---


Result of consistency check:  constraint set is CONSISTENT !
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
```

Figure 7.4: Protocol of the SIC consistency check of the first data source



INFERRED SEMANTIC INTEGRITY CONSTRAINTS

Type	instance Relations	some	LD	RD	LT	RT	all to all	
Topological	meet OR contains OR covers OR overlap	☐	✓	☐	☐	☐	☐	C
OR	meet OR contains OR covers OR overlap	✓	☐	☐	☐	☐	☐	C
OR	meet OR contains OR covers OR overlap	☐	✓	☐	☐	✓	☐	C
OR	meet OR contains OR covers OR overlap	☐	✓	☐	✓	☐	☐	C
OR	meet OR contains OR covers OR overlap	☐	☐	☐	✓	☐	☐	C
OR	meet OR contains OR covers OR overlap	☐	☐	☐	☐	✓	☐	C
OR	meet OR contains OR covers OR overlap	☐	✓	☐	✓	✓	☐	C
OR	meet OR contains OR covers OR overlap	☐	☐	☐	✓	✓	☐	C
OR	meet OR contains OR covers OR overlap	☐	☐	☐	☐	☐	✓	C

[✓] Sorting [✓ Full] Reduced

Figure 7.5: Derived SIC for the classes *floodplain* and *stream* of data source one

The composition of the corresponding abstract class relations is (Table 5.3):

$$R1_{LD.RT}(Floodplain, AlluvialForest)\ ;\ R2_{LT}(AlluvialForest, Stream) \Rightarrow$$
$$R3_{LD}(Floodplain, Stream) \cup R3_{some}(Floodplain, Stream).$$

Figure 7.5 represents this result in the upper two rows of the disjunction. The other seven relations of the disjunction result from the conceptual neighbourhood of the two directly derived class relations. The reason, why these relations have to be included, is explained in Section 5.4.

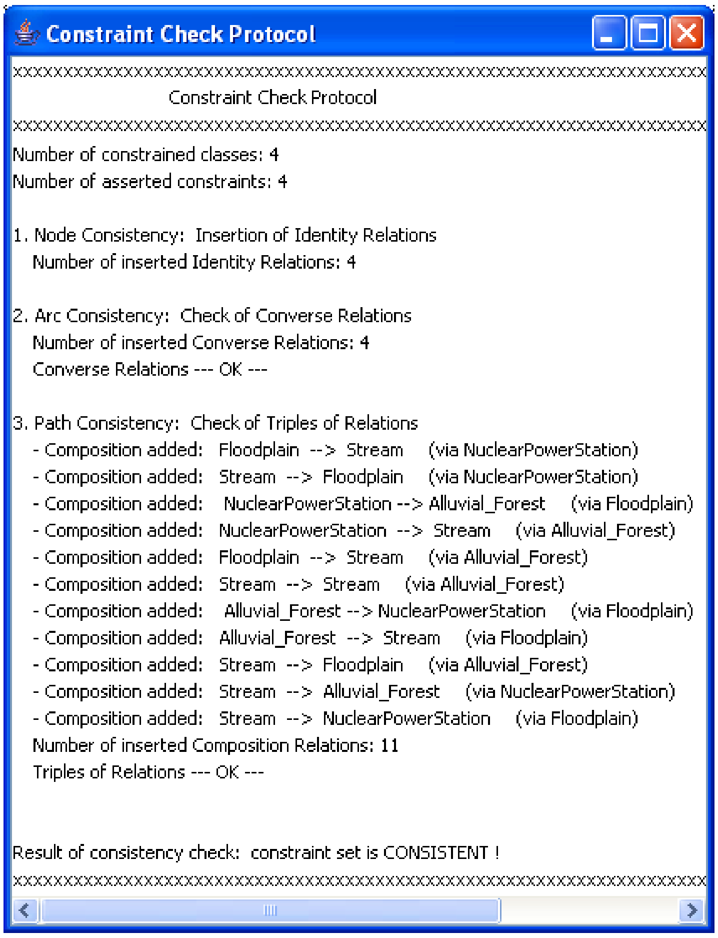
The disjunction of nine relations (Figure 7.5) shows that the two given SICs define a very weak restriction of the relations between *floodplain* and *stream* instances. Because of that, it is hardly possible for a human user to derive this constraint manually. Nevertheless, it is possible that other SICs are in conflict with the derived disjunction. This issue is analysed to answer the second question.

2. Is it possible to integrate the two data sets or are there explicit or implicit SICs that contradict each other?
The inference of implicitly defined SICs of the second data source leads to comparable results, such that there is no need of further discussion here and it is directly continued with the integration of the constraint sets. An integration by means of the SIC Checker Plug-in will discover contradictions and inequalities between the constraint sets. Differing SICs between the common classes indicate different quality requirements of the two data sets. The differing quality of the data sets might have to be aligned before the integration of the data. For the integration, all four given constraints of the two data sources have to be entered into the Protégé SIC Checker Plug-in. Figure 7.6 shows the protocol of the consistency check.

The protocol confirms that the SICs of the two data sources are not contradicting. Eleven implicit relations, which have been derived during the path consistency check, have been logged. Two times a constraint has been derived between the classes *floodplain* and *stream* (and two times in reverse direction), one of them by a composition via the *nuclear power station* class and the second via *alluvial forest*.

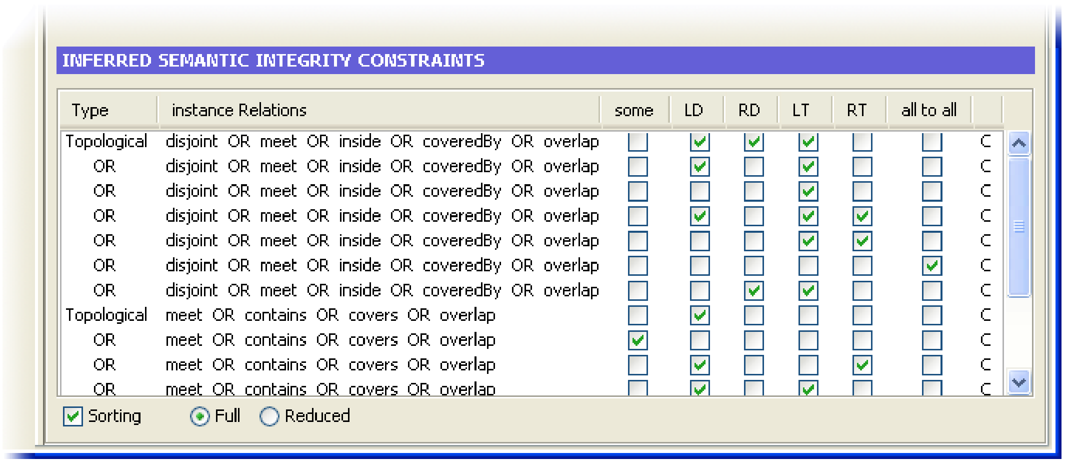
The derived SICs are shown in the table of inferred constraints in Figure 7.7. The table of inferred SICs for the classes *floodplain* and *stream* contains two disjunctions. The upper one is a disjunction of seven relations, all based on the same disjunction of five instance relations and differing abstract class relations. This SIC is the composition result of the two given constraints of the second data source:

$$DISJOINT_{LT.RT-all}(Floodplain, NPS)\ ;\ MEET_{RD.LT}(NPS, Stream)$$



```
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
                    Constraint Check Protocol
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
Number of constrained classes: 4
Number of asserted constraints: 4

1. Node Consistency:  Insertion of Identity Relations
   Number of inserted Identity Relations: 4

2. Arc Consistency:  Check of Converse Relations
   Number of inserted Converse Relations: 4
   Converse Relations --- OK ---

3. Path Consistency:  Check of Triples of Relations
   - Composition added:   Floodplain  -->  Stream    (via NuclearPowerStation)
   - Composition added:   Stream  -->  Floodplain    (via NuclearPowerStation)
   - Composition added:   NuclearPowerStation --> Alluvial_Forest    (via Floodplain)
   - Composition added:   NuclearPowerStation  -->  Stream    (via Alluvial_Forest)
   - Composition added:   Floodplain  -->  Stream    (via Alluvial_Forest)
   - Composition added:   Stream  -->  Stream    (via Alluvial_Forest)
   - Composition added:   Alluvial_Forest --> NuclearPowerStation    (via Floodplain)
   - Composition added:   Alluvial_Forest  -->  Stream    (via Floodplain)
   - Composition added:   Stream  -->  Floodplain    (via Alluvial_Forest)
   - Composition added:   Stream  -->  Alluvial_Forest    (via NuclearPowerStation)
   - Composition added:   Stream  -->  NuclearPowerStation    (via Floodplain)
   Number of inserted Composition Relations: 11
   Triples of Relations --- OK ---


Result of consistency check:  constraint set is CONSISTENT !
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
```

Figure 7.6: Protocol of the consistency check of the SICs of both data sources



INFERRED SEMANTIC INTEGRITY CONSTRAINTS

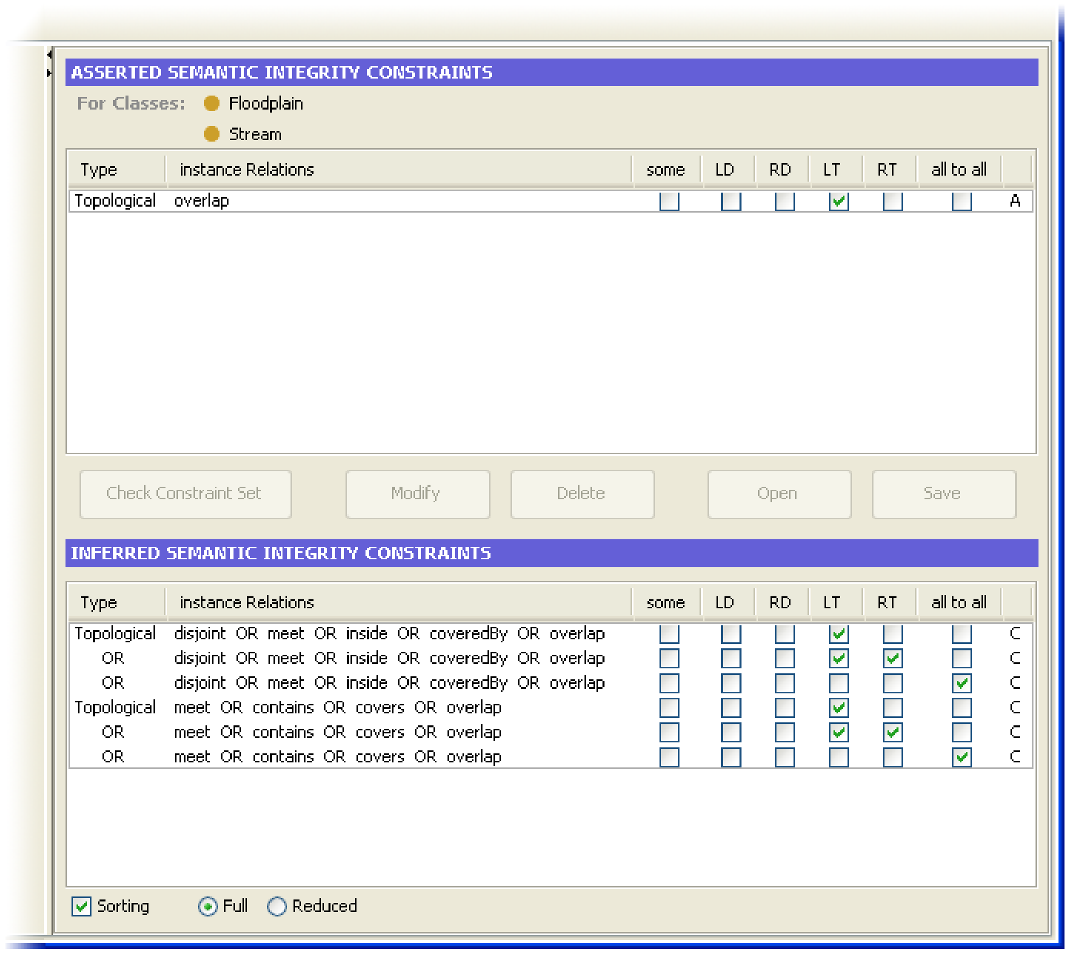
Type	instance Relations	some	LD	RD	LT	RT	all to all	
Topological	disjoint OR meet OR inside OR coveredBy OR overlap		✓	✓	✓			C
OR	disjoint OR meet OR inside OR coveredBy OR overlap		✓		✓			C
OR	disjoint OR meet OR inside OR coveredBy OR overlap				✓			C
OR	disjoint OR meet OR inside OR coveredBy OR overlap		✓		✓	✓		C
OR	disjoint OR meet OR inside OR coveredBy OR overlap				✓	✓		C
OR	disjoint OR meet OR inside OR coveredBy OR overlap						✓	C
OR	disjoint OR meet OR inside OR coveredBy OR overlap			✓	✓			C
Topological	meet OR contains OR covers OR overlap		✓					C
OR	meet OR contains OR covers OR overlap	✓						C
OR	meet OR contains OR covers OR overlap		✓			✓		C
OR	meet OR contains OR covers OR overlap		✓		✓			C

[✓ Sorting] [✓ Full] Reduced

Figure 7.7: Derived SICs for the classes *floodplain* and *stream* of both data sources integrated

The second disjunction (which is not fully visible in Figure 7.7) equals to the one, which has been directly derived from the SICs of the first data source (Figure 7.5). Each one of the disjunctions can be derived from the constraints of one data source, independently of those of the other. They are also not restricting each other (Section 6.4). This leads to a conclusion regarding the quality requirements: the constraint sets of both data sources define weak constraints between the classes *floodplain* and *stream*. Concerning the SICs an integration of the two data sets is possible without any restrictions or required alignments of quality.

3. How does the integrated data fit to the user's SIC and the planned analysis?
Finally, it has to be proven, if the integrated data meets the quality requirements of the user. Therefore the given constraints of the two data sources and the SIC, required by the user, have to be entered into the Protégé SIC Checker Plug-in and a consistency check has to be conducted.


ASSERTED SEMANTIC INTEGRITY CONSTRAINTS

For Classes: ● Floodplain
● Stream

Type	instance Relations	some	LD	RD	LT	RT	all to all	
Topological	overlap	☐	☐	☐	✓	☐	☐	A

INFERRED SEMANTIC INTEGRITY CONSTRAINTS

Type	instance Relations	some	LD	RD	LT	RT	all to all	
Topological	disjoint OR meet OR inside OR coveredBy OR overlap	☐	☐	☐	✓	☐	☐	C
OR	disjoint OR meet OR inside OR coveredBy OR overlap	☐	☐	☐	✓	✓	☐	C
OR	disjoint OR meet OR inside OR coveredBy OR overlap	☐	☐	☐	☐	☐	✓	C
Topological	meet OR contains OR covers OR overlap	☐	☐	☐	✓	☐	☐	C
OR	meet OR contains OR covers OR overlap	☐	☐	☐	✓	✓	☐	C
OR	meet OR contains OR covers OR overlap	☐	☐	☐	☐	☐	✓	C

✓ Sorting [✓ Full] Reduced

Figure 7.8: SICs of the integrated data sources and the user for the classes *floodplain* and *stream*

Again, this check does not discover any contradictions between the five SICs. However, the overall inference has an influence on the inferred relations between the classes *floodplain* and *stream* (Figure 7.8). The table of asserted constraints at the top of the figure contains the SIC of the user. Two constraint disjunctions are inferred from the given five constraints, shown in the lower table. The two disjunctions originate from the disjunctions inferred from the integrated SICs (Figure 7.7), but are restricted to the relations, which are not contradicting the user's SIC. The reason of this restriction is: all relations of the original disjunctions contain the *overlap* instance relation of the user's SIC in their instance relation disjunctions. Therefore, only relations based on abstract class relations, which are conceptual neighbours of R_{LT} through addition of instance relations are consistent (Sections 5.4 and 6.3 for a derivation). The remaining constraint disjunctions could also be fully restricted to the user's SIC, since both contain a relation, which bases on the instance relation *overlap* and the R_{LT} abstract class relation. However, since this conclusion is not unique, the constraints are not automatically restricted by the plug-in. The user can remove such restrictable constraints from the table view by selecting the "Reduced" choice box at the bottom of the window. All together, this proves that the user's SIC is more restricting than the inferred constraints.

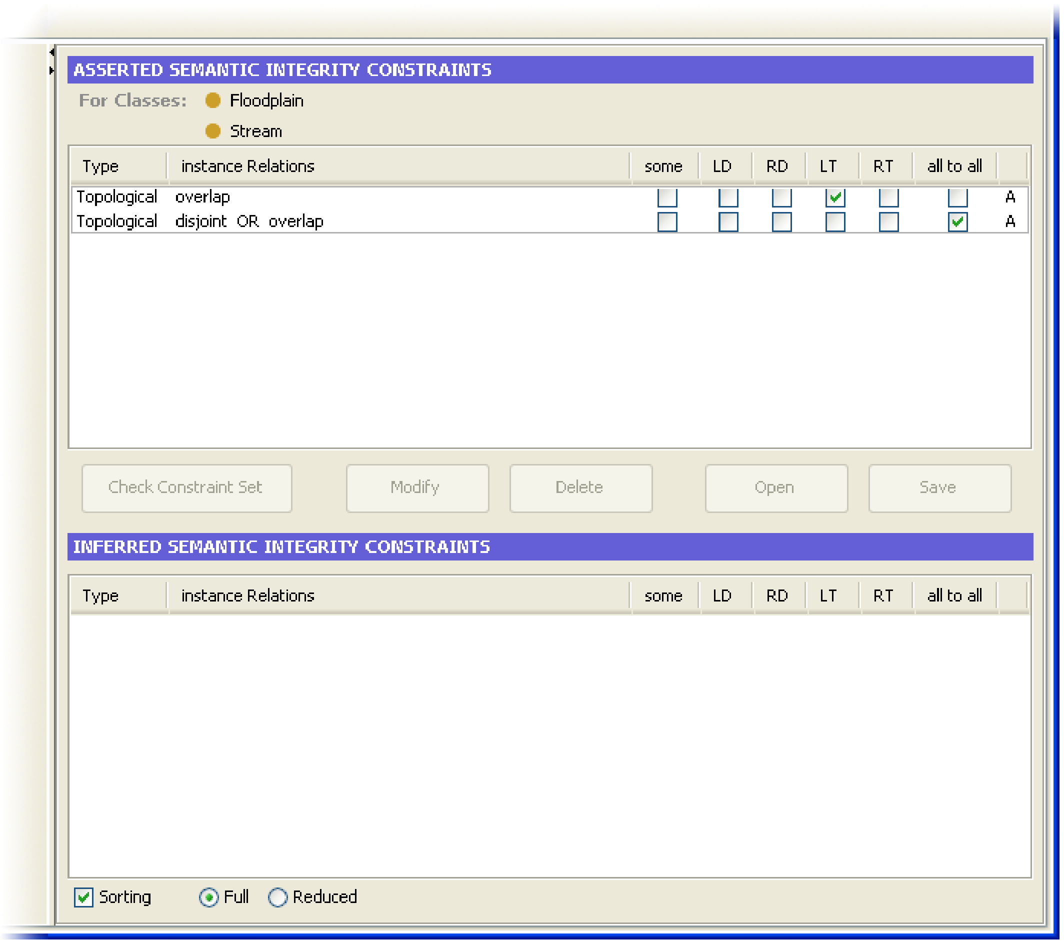
The restriction of inferred constraints can be continued by defining an additional SIC, which restricts all instance relations between *floodplains* and *streams*:

- All floodplains must overlap with or be disjoint from all streams.
  (corresponds to $[OVERLAP \cup DISJOINT]_{LT.RT-all}(Floodplain, Stream)$)

The result of the corresponding consistency check is shown in Figure 7.9: the inferred constraint disjunctions are completely restricted such that only the two explicitly defined SICs remain between the classes *floodplain* and *stream*. This means that there is no additional implicit knowledge available. The constraint set as a whole is nevertheless consistent.

To summarise the results of the second application scenario: it has been confirmed that it is possible to integrate the constraint sets of both data sources without conflicts. Furthermore, the integrated SICs of the two data sets and the SIC of the user can be combined to a single, logically consistent set of constraints. The SIC of the user, which has to be hold by the data to enable the flooding analysis, is more restricting than the inferred constraints of the integrated sets. Thus it is advisable for the user to prove the SIC on the data to create a (with respect to the SICs) harmonised data set.

This application scenario demonstrates the relevance of the developed reasoning methodology for data integration and harmonisation in general. It enables to discover conflicts and quality discrepancies through an analysis of the SICs independently of the actual data. Therewith this part of the harmonisation can be conducted at the metadata level, without checking single instances against the SICs. The only requirement is that the semantics of the common entity classes of the data sets can be considered as similar. A survey of approaches, which enable such similarity measure, can be found in (Schwering, 2008).


ASSERTED SEMANTIC INTEGRITY CONSTRAINTS

For Classes: ● Floodplain
● Stream

Type	instance Relations	some	LD	RD	LT	RT	all to all	
Topological	overlap	☐	☐	☐	✓	☐	☐	A
Topological	disjoint OR overlap	☐	☐	☐	☐	☐	✓	A

INFERRED SEMANTIC INTEGRITY CONSTRAINTS

Type	instance Relations	some	LD	RD	LT	RT	all to all

[✓ Sorting] [✓ Full] Reduced

Figure 7.9: SIC tables of the classes *floodplain* and *stream* after checking the consistency of all six given constraints

7.4 Further Application Areas

The two application scenarios (Sections 1.1 and 1.2) demonstrate the usefulness of the class relation based reasoning methodology for quality assurance and integration and harmonisation of spatial data. Some further application areas are:

Management of Spatial Semantic Integrity Constraints
In GIS applications, like for example in mapping agencies or utility companies, it is not unusual to have sets of more than one hundred SICs to assure logical consistency. Accordingly, the calculation effort of such a quality assurance is very high. Personal discussions with such users affirmed, that there is a great demand for the management of SICs and therewith also for the overall process of assuring logical consistency. The developed reasoning methodology can support the management of SICs with consistency and redundancy checks if new SICs

are defined. Furthermore, it might be necessary to enable a **versioning of constraint sets**, in particular if multiple parties are involved in the acquisition and management of the data or parts of the data. The comparison of different SIC set versions facilitates the control of its evolution and minimises the efforts of the quality assurance calculation, when parts of the data are checked against an outdated version of the SIC set.

Geospatial Ontologies and Conceptual Data Modelling

The similarities and differences between ontologies and conceptual data models are well researched (Spyns et al., 2002; Fonseca et al., 2003; Jarrar et al., 2003). SICs are part of both, since they describe the semantics of the modelled concepts and specify integrity rules for the data. An integration of SICs into geospatial ontologies supports information retrieval in the so-called **Geospatial Semantic Web** (Egenhofer, 2002; Fonseca, 2008a) and **ontology based geospatial data integration** (Fonseca, 2008b). The developed reasoning methodology can facilitate these processes as well as the proof of the internal consistency of the ontological axioms.

Advanced database modelling tools like the one introduced in (Stoyanov and Harper, 2009) support the development at all data modelling levels (Section 2.1) with a single software. This allows for a tight synchronisation between the models of the different levels and an improved communication between modellers, users and database architects. If such tools facilitate the specification of SICs at the level of conceptual models they should also incorporate the checking of internal consistency of the constraints and the proof of the mapping of the constraints to the models at the implementing levels.

Semantic Similarity Measures

If the proposed algorithm is applied for ontology or data integration, it is a key requirement that the concepts of common classes are known to be similar. If class names and attribute definitions differ in the data models, this similarity might become difficult to determine. In these cases, a semantic similarity analysis should be conducted. Such similarity measures are usually based on a semantic distance or difference calculation, which takes the parts, functions (i.e., affordances) and attributes of the class definitions as well as semantic interrelations between classes (is−a and part/whole relations) into account as distinguishing features (Rodríguez et al., 1999; Rodríguez and Egenhofer, 2004). Schwering (2008) has given a review of existing approaches to semantic similarity measurement.

The analysis of conflicts in sets of class relations could also serve as an extension to such similarity measures. If, for example, an analysis of the SICs of two data sets shows contradictions, the similarity measure of the concerning classes could be decreased. An approach which considers such reasoning on class relations in a similarity analysis is unknown, but their inclusion in future similarity measures has for example been claimed in (Schwering, 2006).

Usability Evaluation

The ISO defines usability (also called fitness for use) as the "extent, to which a product can be used by specified users to achieve specified goals with effectiveness, efficiency and satisfaction in a specified context of use" (ISO, 1998). The requirement to evaluate the usability of Web accessible geodata with regard to a particular application led to a lively discussion in the GI community about the existing quality and metadata describing standards. It has been argued that the established quality elements (Section 2.2) would be producer centric and not meaningful to assess the fitness of the data for a given purpose (Goodchild, 2007; Devillers et al., 2007).

The resolution of the second application scenario in the previous section demonstrates a usability evaluation based on a SIC, which specifies user requirements, and the SICs of the available data sets. In general, if a specific application has quality requirements on the logical consistency which can be specified as SIC, the proposed algorithm can be used to check the usability of the data sets and to discover corresponding quality discrepancies. This serves to evaluate the model quality with regard to a particular application.

Chapter 8

Discussion of Results

This dissertation presents a new approach to reasoning on relations among classes. It bases on a qualitative description of cardinality restrictions, which has, to the best knowledge of the author, nobody researched before. The work was motivated by the inability to compare pairs of Semantic Integrity Constraints (SICs) and to find conflicts and redundancies in SIC sets. Because of that, current GISs are not capable of recognising and handling inconsistencies between different data sources and inconsistencies of the data with regard to user requirements.

8.1 Summary of Contributions

Three major contributions are elaborated: (i) the categorisation of SICs, (ii) the framework for the formal definition of SICs and (iii) the reasoning methodology for the detection of conflicting and redundant SICs.

The proposed categorisation (Chapter 3) provides a basis for further work on the definition, formalisation, management and validation of internal consistency of SIC sets. The classification distinguishes the constraints according to the involved types of conditions and profoundly differentiates the properties and aspects restricted by SICs. The new concepts extend the well-established classification of database ICs of Elmasri and Navathe (1994). A particular focus has been laid on ICs, which restrict spatial properties and spatial relations. ICs on temporal, geometric and topological primitives are considered as domain constraints. Further on the categorisation distinguishes SICs according to the semantic domains of the involved properties and relations. The categorisation should be considered as a starting point for further discussion and refinement.

The framework for formal definition of SICs (Chapter 4) is based on a set of 17 abstract class relations. This qualitative description of cardinality restrictions is novel. The definitions and reasoning rules of the class relations are described independently of a concrete set of instance relations, which makes them applicable for many spatial and non-spatial relations. Main advantages over the formalisation approaches listed in Section 2.3 are the consideration of the cardinality definitions of the restricted (spatial) relations, the clear representation of the cardinality restriction (e.g., in comparison to Object Constraint Language (OCL) definitions) and their logical soundness. The interoperable exchange of data of different domains and application areas requires semantic descriptions of the data. The proposed class relations are useful for the formalisation of these descriptions. The logical properties of the class relations support an automatic processing, querying and comparing of such descriptions. Nevertheless, the scientific investigation of class relations and their applications is currently still in the early stages.

With the introduced reasoning algorithm, it is possible to detect conflicts and redundancies in sets of SICs, which has hardly been a research topic before. The overall reasoning algorithm is based on the symmetry, composition and conceptual neighbourhood of class relations (Chapter 5). The definitions and reasoning rules of the class relations are described independently of a specific set of instance relations. The introduced two-level composition of class relations allows for a separate analysis of instance relations and abstract class relations. Therewith the overall reasoning formalism can be used with spatial or non-spatial sets of instance relations. The conditions of this use have been elucidated: it is essential that the applied instance relations belong to the same set of JEPD relations and that this set allows for reasoning on symmetry and compositions. The application of these reasoning techniques for checking consistency in networks of binary relations is a CSP. The procedures for the solution of the CSP for class relations are similar but more complex than the corresponding procedures at the instance level (Chapter 6). Main reasons for the higher complexity are the required rules and conditions for the detection of redundancies, conflicts and possible restrictions of class relations, and the fact, that multiple SICs can be defined between each pair of classes.

The prototypical implementation of the approach (Chapter 7) verifies the feasibility and shows, how it can be integrated into the SIC definition workflow. As possible application areas have been discussed: the quality assurance of geodata, geodata integration and harmonisation, data modelling and ontology engineering, semantic similarity measurements and usability evaluation. The illustrated use in the application scenario showed that working with the tool is still complex and merely for experts, who have the required domain knowledge and understanding of the SIC formalisation. Anyway, using the tool will not be an everyday task and the provided support for the work with SICs is unprecedented.

8.2 Restrictions and Future Research Topics

A major progress for the quality assurance with SICs would be if a minimal set of SICs could be determined. Such a set specifies all restrictions with a minimal number of SICs. This could minimise the calculation costs of the quality assurance. With the proposed algorithm only directly derived redundancies can be detected, which will not necessarily lead to a minimal set. Rodríguez et al. (2004) researched different approaches to calculate such minimal sets for temporal interval relations. A transfer of these methods to the class level would be of high interest. A further contribution in this direction could be the consideration of the actual calculation costs of checking a data set against a single SIC. These costs depend on the checking operation, which corresponds to the spatial relation of the SIC, but also on the number of instances in the data set. This would enable to detect cases, for which it is more reasonable to check two instead of one SIC to minimise calculation costs.

Extensions of the reasoning methodology and an improvement of its practical applicability require research of the supported instance relations, abstract class relations and the inheritance of class relations:

So far, the developed algorithm supports only binary instance relations. This could be extended towards ternary relations (e.g., a is between b and c) and relations between even more instances. For a broader use, also combinations with unary relations are required. The use of the introduced concepts is currently restricted by the unavailability of composition tables for many of the spatial or non-spatial relations. Furthermore, at least composition tables for topological relations between entities with simple and possibly different geometries, like points or linestrings, must be established. The application of other spatial relations is mostly hampered by the lack of a common understanding of their concepts. Moreover, the different aspects of space are not independent of each other. To consider this when reasoning on class relations, corresponding composition tables which contain the compositions of relations of multiple spatial aspects and resolution, are required (for example, a is within b and b is south of c).

This approach is restricted to binary relations between entire entity classes. Relations between three or more classes or between subsets of classes are not considered. Further more, only total participation and a cardinality ratio of 0..1 are included as cardinality properties of the class relations. Nevertheless, this framework provides a basis, which can be extended for other possibly more complex types of class relations. For an extension by further cardinality ratio constraints (e.g., 0..2) it has to be considered, that this will increase the calculation costs of the compositions exponentially. For an optimal usability and user guidance in implementing software, research regarding the ability of users to work with cardinality restrictions while defining SICs should be conducted. This includes research on the cognitive plausibility of the developed logics.

The class concepts described in data models or ontologies are hierarchically structured, usually. The subclasses inherit properties of their superclasses in the hierarchy. The inheritance of SICs in class hierarchies has not been researched so far. Depending on the distinguishing properties of a categorisation, not all SICs are directly inherited. Nevertheless, there are obvious dependencies between super- and subclasses, which have to be formally described and integrated into the developed algorithm. Similarly, complex SICs, which define spatial relations between classes with restricted thematic attributes, can be considered. Since such SICs are defining subsets of a class, the logics for checking consistency are comparable to a subclass / superclass dependency.

Certainly, it cannot be expected that these extensions will lead to real practical applications in the near future, but it will definitely be interesting and challenging to work in this relatively untouched field of GI science.

Abbreviations and Acronyms

API	Application Programming Interface
CSP	Constraint Satisfaction Problem
DBMS	Database Management System
DDL	Data Definition Language
DI-9IM	Dimensionally Extended 9 Intersection Model
DQWG	Data Quality Working Group
ERD	Entity Relationship Diagram
GI	Geographic Information
GIS	Geographic Information System
GML	Geography Markup Language
IC	Integrity Constraint
INSPIRE	Infrastructure for Spatial Information in Europe
ISO	International Organization for Standardization
ISO-TC211	ISO - Technical Committee 211
JEPD	jointly exhaustive and pairwise disjoint
KB	Knowledge Base
OCL	Object Constraint Language
OGC	Open Geospatial Consortium
RDF	Resource Description Framework
RuleML	Rule Markup Language
SDI	Spatial Data Infrastructure
SIC	Semantic Integrity Constraint
SQUIRL	Spatial QUality and Integration Rules Language
SWRL	Semantic Web Rule Language
UML	Unified Modeling Language
XMI	XML Metadata Interchange
XML	Extensible Markup Language

List of Symbols

x, y, z	Denote variables for individuals / instances. Every instance must belong 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 instance x has the relation r to instance y; x and y are said to participate on the relationship instance r. The meta-variable r can stand for any binary relation between instances (e.g., a topological relation) or for a disjunction of such relations. The validity of the binary relation depends on the properties of the instances, for example for spatial relations the geometry of the instances. Every relationship instance r can be associated with a class relation R (same letter as upper case).
$R_{<cp>}(A, B)$	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 instance relation r (same letter as lower case) or a disjunction of instance relations. If a class relation $R_{<cp>}(A, B)$ is defined at least one r must be existent between the instances of A and B. The placeholder $< cp >$ stands for the cardinality properties of the class relation.
r^i	Converse instance relation of r
R^i	Converse class relation of R
r_A^{id}	Denotes that the instance relation r fulfils the identity criterion (Equation 6.1) for the class A.

$r1 \cup r2$	Union / disjunction of the two instance relations
$r1 \cap r2$	Intersection of the two instance relations
$r1\ ;\ r2$	Composition / relative product of the two instance relations
$R1 \cup R2$	Union / disjunction of the two class relations
$R1 \cap R2$	Intersection of the two class relations
$R1\ ;\ R2$	Composition / relative product of the two class relations
$\mathcal{U}$	Universal disjunction / universal relation of the corresponding set of relations
$:=$	Definition, for example $R1 := ...$ can be read as "$R1$ is defined as ..."
$\Rightarrow$	Implication; for example $R1 \Rightarrow R2$ can be read as "if $R1$ then $R2$"
$\forall$	Universal quantifier
$\exists$	Existential quantifier
$\neg$	Negation

Bibliography

Allen, J. F. (1983). Maintaining knowledge about temporal intervals. *Communications of the ACM*, 26(11):832–843. 24, 27, 30, 32, 42

Bartelme, N. (2005). *Geoinformatik: Modelle, Strukturen, Funktionen (4th edition)*. Springer. 19

Becker, L., Ditt, H., Hinrichs, K., and Voigtmann, A. (1999). Constraints and triggers: a method for ensuring data quality in an object-oriented geo database kernel. In *GIS '99: Proceedings of the 7th ACM international symposium on Advances in geographic information systems*, pages 160–161. ACM Press. 12

Belussi, A., Negri, M., and Pelagatti, G. (2006). An iso tc 211 conformant approach to model spatial integrity constraints in the conceptual design of geographical databases. In Roddick, J. F., Benjamins, V. R., Cherfi, S. S.-S., Chiang, R. H. L., Claramunt, C., Elmasri, R., Grandi, F., Han, H., Hepp, M., Lytras, M. D., Misic, V. B., Poels, G., Song, I.-Y., Trujillo, J., and Vangenot, C., editors, *Advances in Conceptual Modeling - Theory and Practice, ER 2006 Workshop CoMoGIS, Proceedings*, volume 4231 of *Lecture Notes in Computer Science*, pages 100–109. Springer. 23

Borges, K. A. V., Laender, A. H. F., and Clodoveu A. Davis, J. (1999). Spatial data integrity constraints in object oriented geographic data modeling. In *GIS '99: Proceedings of the 7th ACM international symposium on Advances in geographic information systems*, pages 1–6. ACM Press. 20, 23, 37

Bravo, L. and Rodríguez, M. A. (2009). Semantic integrity constraints for spatial databases. In Arenas, M. and Bertossi, L., editors, *Proceedings of the 3rd Alberto Mendelzon International Workshop on Foundations of Data Management (AMW09)*, volume 450 of *CEUR workshop proceedings*. RWTH Aachen. 12, 23

Brodeur, J. and Badard, T. (2008). Modeling with iso 191xx standards. In Shekhar, S. and Xiong, H., editors, *Encyclopedia of GIS*, pages 705–716. Springer. 20

Casanova, M., Wallet, T., and D'Hondt, M. (2000). Ensuring quality of geographic data with uml and ocl. In Evans, A., Kent, S., and Selic, B., editors, *UML 2000 - The Unified Modeling Language, Advancing the Standard, Third International Conference, Proceedings*, volume 1939 of *Lecture Notes in Computer Science*, pages 225–239. Springer. 23

Casanova, M., Wallet, T., and D'Hondt, M. (2001). Explicit domain knowledge model in geographic information systems. In *Proceedings of the Thirteenth International Conference on Software Engineering & Knowledge Engineering (SEKE'2001)*, pages 331–340. 23

Cicerone, S. and Felice, P. D. (2004). Cardinal directions between spatial objects: the pairwise-consistency problem. *Information Sciences*, 164(1-4):165–188. 45

Claramunt, C. and Jiang, B. (2001). An integrated representation of spatial and temporal relationships between evolving regions. *Journal of Geographical Systems*, 3(4):411–428. 47

Clementini, E. and Felice, P. D. (1997). A global framework for qualitative shape description. *Geoinformatica*, 1(1):11–27. 25, 46

Clementini, E. and Felice, P. D. (2000). Spatial operators. *SIGMOD Record*, 29(3):31–38. 50

Cockcroft, S. (1997). A taxonomy of spatial data integrity constraints. *Geoinformatica*, 1(4):327–343. 37

Cockcroft, S. (2004). The design and implementation of a repository for the management of spatial data integrity constraints. *Geoinformatica*, 8(1):49–69. 37, 39, 41, 47

Codd, E. F. (1990). *The relational model for database management: version 2*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA. 11

Cohn, A. G. (1995). A hierarchical representation of qualitative shape based on connection and convexity. In Frank, A. U. and Kuhn, W., editors, *Spatial Information Theory: A Theoretical Basis for GIS, International Conference COSIT 1995, Proceedings*, volume 988 of *Lecture Notes in Computer Science*, pages 311–326. Springer. 25, 46

Cohn, A. G. (2008). Conceptual neighborhood. In Shekhar, S. and Xiong, H., editors, *Encyclopedia of GIS*, page 123. Springer. 32

Cohn, A. G., Gotts, N. M., Cui, Z., Randell, D. A., Bennett, B., and Gooday, J. M. (1998). Exploiting temporal continuity in qualitative spatial calculi. In Golledge, R. G. and Egenhofer, M. J., editors, *Spatial and Temporal Reasoning in Geographical Information Systems*, pages 5–24. Oxford University Press. 32

Cohn, A. G. and Hazarika, S. M. (2001). Qualitative spatial representation and reasoning: An overview. *Fundamenta Informaticae*, 46(1-2):1–29. 17, 25, 43, 46

Devillers, R., Bédard, Y., Jeansoulin, R., and Moulin, B. (2007). Towards spatial data quality information analysis tools for experts assessing the fitness for use of spatial data. *International Journal of Geographical Information Science*, 21(3):261–282. 108

Ditt, H., Becker, L., Voigtmann, A., and Hinrichs, K. (1997). Constraints and triggers in an object-oriented geo databasekernel. In *DEXA Workshop*, pages 508–513. 20, 37

Donnelly, M. and Bittner, T. (2005). Spatial relations between classes of individuals. In Cohn, A. G. and Mark, D. M., editors, *Spatial Information Theory, International Conference, COSIT 2005, Proceedings*, volume 3693 of *Lecture Notes in Computer Science*, pages 182–199. Springer. 23, 52, 53, 55, 56, 59, 60, 63, 67, XIII

Doorn, J. H. and Rivero, L. C. (2002). *Database Integrity: Challenges and Solutions*. IGI Publishing, Hershey, PA, USA. 12

Duboisset, M., Pinet, F., Kang, M.-A., and Schneider, M. (2005). Precise modeling and verification of topological integrity constraints in spatial databases: From an expressive power study to code generation principles. In Delcambre, L. M. L., Kop, C., Mayr, H. C., Mylopoulos, J., and Pastor, O., editors, *Conceptual Modeling - ER 2005, 24th International Conference on Conceptual Modeling, Proceedings*, volume 3716 of *Lecture Notes in Computer Science*, pages 465–482. Springer. 23

Ebbinghaus, H. D. (1977). *Einführung in die Mengenlehre*. Wissenschaftliche Buchgesellschaft Darmstadt. 26

Egenhofer, M. J. (1994). Deriving the composition of binary topological relations. *Journal of Visual Languages and Computing*, 5(2):133–149. 27, 28, XV

Egenhofer, M. J. (1997). Consistency revisited. *Editorial in GeoInformatica*, 1(4):323–325. 12, 21, 22

Egenhofer, M. J. (2002). Toward the semantic geospatial web. In Voisard, A. and Chen, S.-C., editors, *ACM-GIS 2002, Proceedings of the Tenth ACM International Symposium on Advances in Geographic Information Systems*, pages 1–4. ACM. 107

Egenhofer, M. J. and Al-Taha, K. K. (1992). Reasoning about gradual changes of topological relationships. In *Theories and Methods of Spatio-Temporal Reasoning in Geographic Space, International Conference GIS - From Space to Territory: Theories and Methods of Spatio-Temporal Reasoning, September 21-23, 1992, Proceedings*, volume 639 of *Lecture Notes in Computer Science*, pages 196–219. Springer. 32, 33, XIII

Egenhofer, M. J., Clementini, E., and Felice, P. D. (1994). Evaluating inconsistencies among multiple representations. In *Sixth International Symposium on Spatial Data Handling*, pages 901–920. 49

Egenhofer, M. J. and Frank, A. U. (1992). Object-oriented modeling for gis. *Journal of the Urban and Regional Information Systems Association*, 4(2):3–19. 48

Egenhofer, M. J. and Franzosa, R. D. (1991). Point set topological relations. *International Journal of Geographical Information Systems*, 5:161–174. 44

Egenhofer, M. J. and Herring, J. R. (1990). Categorizing binary topological relations between regions, lines, and points in geographic databases. Technical report, Department of Surveying Engineering, University of Maine. 24, 25, 44, 50, 98, XIII

Egenhofer, M. J. and Mark, D. M. (1995). Modelling conceptual neighbourhoods of toplogical line-region relations. *International Journal of Geographical Information Systems*, 9(5):555–565. 32

Egenhofer, M. J. and Sharma, J. (1993). Assessing the consistency of complete and incomplete topological information. *Geographical Systems*, 1:47–68. 32

Elmasri, R. and Navathe, S. B. (1994). *Fundamentals of database systems (2nd edition)*. Benjamin - Cummings Publishing Company, Inc. 12, 16, 19, 35, 36, 38, 39, 40, 41, 48, 54, 55, 109

Fonseca, F. T. (2008a). Geospatial semantic web. In Shekhar, S. and Xiong, H., editors, *Encyclopedia of GIS*, pages 388–391. Springer. 107

Fonseca, F. T. (2008b). Ontology-based geospatial data integration. In Shekhar, S. and Xiong, H., editors, *Encyclopedia of GIS*, pages 812–815. Springer. 107

Fonseca, F. T., Davis, C. A., and Câmara, G. (2003). Bridging ontologies and conceptual schemas in geographic information integration. *GeoInformatica*, 7(4):355–378. 107

Frank, A. U. (1992). Qualitative spatial reasoning about distances and directions in geographic space. *Journal of Visual Languages and Computing*, 3:343–371. 24, 27, 45, 47

Frank, A. U. (2001). Tiers of ontology and consistency constraints in geographical information systems. *International Journal of Geographical Information Science*, 15(7):667–678. 37, 39

Freksa, C. (1992a). Temporal reasoning based on semi-intervals. *Artificial Intelligence*, 54(1):199–227. 24, 27, 32

Freksa, C. (1992b). Using orientation information for qualitative spatial reasoning. In Frank, A. U., Campari, I., and Formentini, U., editors, *Theories and Methods of Spatio-Temporal Reasoning in Geographic Space, International Conference GIS - From Space to Territory: Theories and Methods of Spatio-Temporal Reasoning, Proceedings*, volume 639 of *Lecture Notes in Computer Science*, pages 162–178. Springer. 24, 27, 45, 46

Friis-Christensen, A., Tryfona, N., and Jensen, C. S. (2001). Requirements and research issues in geographic data modeling. In Aref, W. G., editor, *ACM-GIS 2001, Proceedings of the Ninth ACM International Symposium on Advances in Geographic Information Systems,*, pages 2–8. ACM. 20, 37

Goodchild, M. F. (2007). Beyond metadata: Towards user-centric description of data quality. In *Proceedings of the 5th International Symposium on Spatial Data Quality, ISSDQ 2007*. 108

Gottfried, B. (2007). Characterising straightness qualitatively. In Fabrikant, S. I. and Wachowicz, M., editors, *The European Information Society Leading the Way with Geo-information, Proceedings of the AGILE 2007*, Lecture Notes in Geoinformation and Cartography, pages 419–433. Springer. 46

Goyal, R. K. and Egenhofer, M. J. (2001). Similarity of cardinal directions. In Jensen, C. S., Schneider, M., Seeger, B., and Tsotras, V. J., editors, *Advances in Spatial and Temporal Databases, 7th International Symposium, SSTD 2001, Proceedings*, volume 2121 of *Lecture Notes in Computer Science*, pages 36–58. Springer. 24, 45

Grigni, M., Papadias, D., and Papadimitriou, C. H. (1995). Topological inference. In *Proceedings of the International Joint Conference of Artificial Intelligence IJCAI (1)*, pages 901–907. 25, 27, 32, 44

Gröger, G. and Plümer, L. (1997). Provably correct and complete transaction rules for gis. In *GIS '97: Proceedings of the 5th ACM International Workshop on Advances in Geographic Information Systems*, pages 40–43. ACM. 35

Grütter, R. and Bauer-Messmer, B. (2007). Towards spatial reasoning in the semantic web: A hybrid knowledge representation system architecture. In Fabrikant, S. I. and Wachowicz, M., editors, *The European Information Society Leading the Way with Geo-information, Proceedings of the AGILE 2007*, Lecture Notes in Geoinformation and Cartography, pages 349–364. Springer. 96

Guarino, N. (1999). The role of identity conditions in ontology design. In Freksa, C. and Mark, D. M., editors, *Spatial Information Theory: Cognitive and Computational Foundations of Geographic Information Science, International Conference COSIT '99, Proceedings*, volume 1661 of *Lecture Notes in Computer Science*, pages 221–234. Springer. 84

Hadzilacos, T. and Tryfona, N. (1992). A model for expressing topological integrity constraints in geographic databases. In *Proceedings of the International Conference GIS - From Space to Territory: Theories and Methods of Spatio-Temporal Reasoning on Theories and Methods of Spatio-Temporal Reasoning in Geographic Space*, pages 252–268. Springer. 22, 63

Hernández, D. (1994). *Qualitative Representation of Spatial Knowledge*, volume 804 of *Lecture Notes in Artificial Intelligence*. Springer. 24, 27, 30, 31, 32, 47, 90, 93

Hernández, D., Clementini, E., and Felice, P. D. (1995). Qualitative distances. In Frank, A. U. and Kuhn, W., editors, *Spatial Information Theory: A Theoretical Basis for GIS, International Conference COSIT 1995, Proceedings*, volume 988 of *Lecture Notes in Computer Science*, pages 45–57. Springer. 27, 47

Hornsby, K. and Egenhofer, M. J. (1997). Qualitative representation of change. In Hirtle, S. C. and Frank, A. U., editors, *Spatial Information Theory: A Theoretical Basis for GIS, International Conference COSIT '97, Proceedings*, volume 1329 of *Lecture Notes in Computer Science*, pages 15–33. Springer. 48

Hornsby, K., Egenhofer, M. J., and Hayes, P. J. (1999). Modeling cyclic change. In Chen, P. P., Embley, D. W., Kouloumdjian, J., Liddle, S. W., and Roddick, J. F., editors, *Advances in Conceptual Modeling: ER '99 Workshops on Evolution and Change in Data Management, Reverse Engineering in Information Systems, and the World Wide Web and Conceptual Modeling, Proceedings*, volume 1727 of *Lecture Notes in Computer Science*, pages 98–109. Springer. 24, 32, 42

ISO (1998). 9241 part 11: Guidance on usability. 108

ISO/TC211 (2000). 19109 - geographic information - rules for application schema. 19, 22

ISO/TC211 (2002a). 19107 - geographic information - spatial schema. 19, 38, 40, 47

ISO/TC211 (2002b). 19108 - geographic information - temporal schema. 19, 41, 42

ISO/TC211 (2002c). 19113 - geographic information - quality principles. 21

ISO/TC211 (2002d). 19115 - geographic information - metadata. 21

ISO/TC211 (2003). 19114 - geographic information - quality evaluation. 21

ISO/TC211 (2004). 19125 - geographic information - simple feature access. 41, 44

ISO/TC211 (2005). 19138 - geographic information - data quality measures. 21

Jarrar, M., Demey, J., and Meersman, R. (2003). On using conceptual data modeling for ontology engineering. In Spaccapietra, S., March, S. T., and Aberer, K., editors, *Journal on Data Semantics 1*, volume 2800 of *Lecture Notes in Computer Science*, pages 185–207. Springer. 107

Joos, G. (1999a). Assessing the quality of geodata by testing consistency with respect to the conceptual data schema. In Craglia, M. and Onsrud, H., editors, *Geographic Information Research: Trans-Atlantic Perspectives*, pages 509 – 519. Taylor & Francis. 22, 23

Joos, G. (1999b). *Zur Qualität von objektstrukturierten Geodaten.* PhD thesis, University of the Bundeswehr, Munich. 19, 23

Kainz, W. (1995). Logical consistency. In Guptill, S. C. and Morrison, J. L., editors, *Elements of Spatial Data Quality*, pages 109–137. Elsevier Science, Oxford. 21

Knauff, M., Rauh, R., and Renz, J. (1997). A cognitive assessment of topological spatial relations: Results from an empirical investigation. In Hirtle, S. C. and Frank, A. U., editors, *Spatial Information Theory: A Theoretical Basis for GIS, International Conference COSIT '97, Proceedings*, volume 1329 of *Lecture Notes in Computer Science*, pages 193–206. Springer. 43, 44

Kurata, Y. (2008). The 9^+-intersection: A universal framework for modeling topological relations. In Cova, T. J., Miller, H. J., Beard, K., Frank, A. U., and Goodchild, M. F., editors, *Geographic Information Science, 5th International Conference, GIScience 2008*, volume 5266 of *Lecture Notes in Computer Science*, pages 181–198. Springer. 24

Kurata, Y. and Egenhofer, M. J. (2006). The head-body-tail intersection for spatial relations between directed line segments. In Raubal, M., Miller, H. J., Frank, A. U., and Goodchild, M. F., editors, *Geographic Information Science, 4th International Conference, GIScience 2006, Proceedings*, volume 4197 of *Lecture Notes in Computer Science*, pages 269–286. Springer. 27, 32

Leyton, M. (1988). A process-grammar for shape. *Artificial Intelligence*, 34(2):213–247. 46

Louwsma, J., Zlatanova, S., Lammeren, R., and Oosterom, P. (2006). Specifying and implementing constraints in gis–with examples from a geo-virtual reality system. *Geoinformatica*, 10(4):531–550. 12, 22, 23, 37

Mackworth, A. K. and Freuder, E. C. (1985). The complexity of some polynomial network consistency algorithms for constraint satisfaction problems. *Artifical Intelligence*, 25(1):65–74. 30

Maddux, R. D. (1982). Some varieties containing relation algebras. *Transactions of the American Mathematical Society*, 272(2):501–526. 75

Maddux, R. D. (2006). *Relation Algebras*, volume 150 of *Studies in logic and the foundations of mathematics*. Elsevier Science & Technology. 25, 26

Mäs, S. (2007a). Checking the integrity of spatial integrity constraints. In Bank, B., Egenhofer, M. J., and Kuijpers, B., editors, *Constraint Databases, Geometric Elimination and Geographic Information Systems*, volume 07212 of *Dagstuhl Seminar Proceedings*. Internationales Begegnungs- und Forschungszentrum für Informatik (IBFI), Schloss Dagstuhl, Germany. 51, 56

Mäs, S. (2007b). Reasoning on spatial semantic integrity constraints. In Winter, S., Duckham, M., Kulik, L., and Kuipers, B., editors, *Spatial Information Theory, 8th International Conference, COSIT 2007, Proceedings*, volume 4736 of *Lecture Notes in Computer Science*, pages 285–302. Springer. 17, 51, 56, 68

Mäs, S. (2008). Reasoning on spatial relations between entity classes. In Cova, T. J., Miller, H. J., Beard, K., Frank, A. U., and Goodchild, M. F., editors, *Geographic Information Science, 5th International Conference, GIScience 2008, Proceedings*, volume 5266 of *Lecture Notes in Computer Science*, pages 234–248. Springer. 73, 74, 76, XV

Mäs, S. and Reinhardt, W. (2009). Categories of geospatial and temporal integrity constraints. In *International Conference on Advanced Geographic Information Systems and Web Services, GEOWS 2009*, pages 146–151. IEEE Computer Society. 39, 40, 41, XIII

Mäs, S., Reinhardt, W., Kandawasvika, A., and Wang, F. (2005a). Concepts for quality assurance during mobile online data acquisition. In Abdul-Rahman, A., Zlatanova, S., and Coors, V., editors, *8th AGILE Conference on Geographic Information Science - Proceedings*, pages 3–12. 12

Mäs, S., Wang, F., and Reinhardt, W. (2005b). Using ontologies for integrity constraint definition. In Wu, L., Shi, W., Fang, Y., and Tong, Q., editors, *Proceedings of the 4th International Symposium On Spatial Data Quality, ISSDQ 2005*, pages 304–313. 23, 51

Mostafavi, M.-A., Edwards, G., and Jeansoulin, R. (2004). An ontology-based method for quality assessment of spatial data bases. In Frank, A. U. and Grum, E., editors, *Third International Symposium on Spatial Data Quality, ISSDQ, Proceedings*, volume 28a of *GeoInfo Series Vienna*, pages 49–66. TU Vienna. 12, 23

OGC (2005). Filter encoding implementation specification, version 1.1.0. 42

Oosterom, P. v. (2006). Constraints in spatial data models, in a dynamic context. In Drummond, J., Billen, R., João, E., and Forrest, D., editors, *Dynamic and Mobile GIS: Investigating Changes in Space and Time*, pages 104–137. Taylor & Francis. 20, 22, 23

Papadias, D. and Egenhofer, M. J. (1997). Algorithms for hierarchical spatial reasoning. *GeoInformatica*, 1(3):251–273. 27, 30, 32

Pizano, A., Klinger, A., and Cardenas, A. (1989). Specification of spatial integrity constraints in pictorial databases. *Computer*, 22(12):59–71. 22

Plümer, L. and Gröger, G. (1997). Achieving integrity in geographic information systems maps and nested maps. *Geoinformatica*, 1(4):345–367. 38, 48

Pundt, H. (2002). Field data collection with mobile gis: Dependencies between semantics and data quality. *Geoinformatica*, 6(4):363–380. 12, 23

Renz, J. (2002). *Qualitative Spatial Reasoning with Topological Information*, volume 2293 of *Lecture Notes in Computer Science*. Springer. 24, 25

Rodríguez, A., de Weghe, N. V., and Maeyer, P. D. (2004). Simplifying sets of events by selecting temporal relations. In Egenhofer, M. J., Freksa, C., and Miller, H. J., editors, *Geographic Information Science, Third International Conference, GIScience 2004, Proceedings*, volume 3234 of *Lecture Notes in Computer Science*, pages 269–284. Springer. 29, 32, 43, 90, 111, XV

Rodríguez, M. A. and Egenhofer, M. J. (2000). A comparison of inferences about containers and surfaces in small-scale and large-scale spaces. *Journal of Visual Languages and Computing*, 11(6):639–662. 75

Rodríguez, M. A. and Egenhofer, M. J. (2004). Comparing geospatial entity classes: an asymmetric and context-dependent similarity measure. *International Journal of Geographical Information Science*, 18(3):229–256. 107

Rodríguez, M. A., Egenhofer, M. J., and Rugg, R. D. (1999). Assessing semantic similarities among geospatial feature class definitions. In Vckovski, A., Brassel, K. E., and Schek, H.-J., editors, *Interoperating Geographic Information Systems, Second International Conference, INTEROP '99, Proceedings*, volume 1580 of *Lecture Notes in Computer Science*, pages 189–202. Springer. 107

Salehi, M., Bédard, Y., Mostafavi, M. A., and Brodeur, J. (2007). On languages for the specification of integrity constraints in spatial conceptual models. In Hainaut, J.-L., Rundensteiner, E. A., Kirchberg, M., Bertolotto, M., Brochhausen, M., Chen, Y.-P. P., Cherfi, S. S.-S., Doerr, M., Han, H., Hartmann, S., Parsons, J., Poels, G., Rolland, C., Trujillo, J., Yu, E. S. K., and Zimányi, E., editors, *Advances in Conceptual Modeling - Foundations and Applications, ER 2007 Workshops: CMLSA, FP-UML, ONISW, QoIS, RIGiM, SeCoGIS*, volume 4802 of *Lecture Notes in Computer Science*, pages 388–397. Springer. 23

Sanderson, M., Ramage, S., and van Linden, L. (2009). Sdi communities: Data quality and knowledge sharing. In *GSDI 11 world conference: Spatial Data Infrastructure Convergence: Building SDI Bridges to Address Global Challenges, Proceedings*. 11, 23

Schleipen, S., Ragni, M., and Fangmeier, T. (2007). Negation in spatial reasoning. In Hertzberg, J., Beetz, M., and Englert, R., editors, *KI 2007: Advances in Artificial Intelligence, 30th Annual German Conference on AI, KI 2007, Proceedings*, volume 4667 of *Lecture Notes in Computer Science*, pages 175–189. Springer. 63

Schlieder, C. (1996). Qualitative shape representation. In Burrough, P. A. and Frank, A. U., editors, *Geographic Objects with Indeterminate Boundaries*, pages 123–140. Taylor & Francis. 46

Schwering, A. (2006). *Semantic Similarity Measurement including Spatial Relations for Semantic Information Retrieval of Geo-Spatial Data*. PhD thesis, Westfälische Wilhelms Universität Münster. 107

Schwering, A. (2007). Evaluation of a semantic similarity measure for natural language spatial relations. In Winter, S., Duckham, M., Kulik, L., and Kuipers, B., editors, *Spatial Information Theory, 8th International Conference, COSIT 2007, Proceedings*, volume 4736 of *Lecture Notes in Computer Science*, pages 116–132. Springer. 32

Schwering, A. (2008). Approaches to semantic similarity measurement for geo-spatial data: A survey. *Transactions in GIS*, 12(1):5–29. 105, 107

Servigne, S., Ubeda, T., Puricelli, A., and Laurini, R. (2000). A methodology for spatial consistency improvement of geographic databases. *Geoinformatica*, 4(1):7–34. 12, 22, 37, 38, 40, 41

Sharma, J. and Flewelling, D. M. (1995). Inferences from combined knowledge about topology and directions. In Egenhofer, M. J. and Herring, J. R., editors, *Advances in Spatial Databases, 4th International Symposium, SSD'95, Proceedings*, volume 951 of *Lecture Notes in Computer Science*, pages 279–291. Springer. 47

Skiadopoulos, S. and Koubarakis, M. (2001). Composing cardinal direction relations. In Jensen, C. S., Schneider, M., Seeger, B., and Tsotras, V. J., editors, *Advances in Spatial and Temporal Databases, 7th International Symposium, SSTD 2001, Proceedings*, volume 2121 of *Lecture Notes in Computer Science*, pages 299–320. Springer. 24, 45

Skiadopoulos, S. and Koubarakis, M. (2005). On the consistency of cardinal direction constraints. *Artificial Intelligence*, 163(1):91–135. 45

Spyns, P., Meersman, R., and Jarrar, M. (2002). Data modelling versus ontology engineering. *SIGMOD Rec.*, 31(4):12–17. 107

Stoyanov, P. and Harper, S. (2009). *An Oracle White Paper - An Introduction to Oracle SQL Developer Data Modeler.* available at: http://www.oracle.com/technology/products/database/datamodeler/pdf/sqldeveloperdat amodeleroverview.pdf ; last visited on August 10th 2009. 107

Tarquini, F. and Clementini, E. (2008). Spatial relations between classes as integrity constraints. *Transactions in GIS*, 12(s1):45–57. 17, 23, 52, 63

Tarski, A. (1941). On the calculus of relations. *The Journal of Symbolic Logic*, 6(3):73–89. 26, 75

Tryfona, N. and Egenhofer, M. J. (1997). Consistency among parts and aggregates: A computational model. *Transactions in GIS*, 1(3):189–206. 49

Ubeda, T. and Egenhofer, M. J. (1997). Topological error correcting in gis. In *SSD '97: Proceedings of the 5th International Symposium on Advances in Spatial Databases*, pages 283–297. Springer. 22, 38, 51, 62

Vallieres, S., Brodeur, J., and Pilon, D. (2006). Spatial integrity constraints: A tool for improving the internal quality of spatial data. In DEVILLERS, R. and JEANSOULIN, R., editors, *Fundamentals of Spatial Data Quality*, Geographical Information System Series, chapter 9, pages 161–178. ISTE Ltd., London. 12

Varzi, A. C. (1994). On the boundary between mereology and topology. In Casati, R., Smith, B., and White, G., editors, *Philosophy and the Cognitive Sciences: Proceedings of the 16th International Wittgenstein Symposium 1993*, volume 21 of *Schriftreihe der Wittgenstein-Gesellschaft*, pages 423–442. Vienna: Hölder-Pichler-Tempsky. 48

Wang, F. (2008). *Handling Data Consistency through Spatial Data Integrity Rules in Constraint Decision Tables.* PhD thesis, University of the Bundeswehr, Munich. 20, 22, 23, 35

Wang, F. and Reinhardt, W. (2007). Extending geographic data modeling by adopting constraint decision table to specify spatial integrity constraints. In Fabrikant, S. I. and Wachowicz, M., editors, *The European Information Society Leading the Way with Geo-information, Proceedings of the AGILE 2007*, Lecture Notes in Geoinformation and Cartography, pages 435–454. Springer. 23

Watson, P. (2007). Formal languages for expressing data consistency rules and implications for reporting of quality metadata. In *Proceedings of the 5th International Symposium on Spatial Data Quality, ISSDQ 2007.* 23

Werder, S. (2009). Formalization of spatial constraints. In *Proceedings of the 12th AGILE International Conference on Geographic Information Science.* 23, 51

Yan, H., Chu, Y., Li, Z., and Guo, R. (2006). A quantitative description model for direction relations based on direction groups. *Geoinformatica*, 10(2):177–196. 24, 45

List of Figures

List of Tables

Danksagung

Die Idee zu diesem Promotionsthema entstand während des vom Bundesministerium für Bildung und Forschung (BMBF) geförderten Projektes “Weiterentwicklung von Geodiensten”. Ich möchte mich für die Unterstützung, welche ich beim recherchieren, forschen und schreiben dieser Arbeit erhalten habe, bedanken.

Mein ganz besonderer Dank gilt Herrn Prof. Dr.-Ing. Wolfgang Reinhardt, Lehrstuhl für Geoinformatik der Universität der Bundeswehr München, für die Unterstützung während der gesamten Promotionszeit. Seine Anleitung, Kritik, Aufgeschlossenheit und Geduld gaben mir den Rückhalt aber auch die notwenigen Freiheiten für die Durchführung dieser Arbeit. Die von ihm geleitete Arbeitsgemeinschaft Geoinformationssysteme (AGIS) ist eine optimale Umgebung für das wissenschaftliche Arbeiten und ich habe meine Zeit dort sehr genossen.

Meinem Zweitgutachter, Herrn Prof. Dr.-Ing. Max J. Egenhofer, Universität von Maine, danke ich besonders für das Interesse, welches er bereits in einer sehr frühen Phase der Arbeit gezeigt hat, und die Unterstützung in der Endphase. Seine Vorarbeiten weckten bei mir das Interesse für das logische Schließen mit räumlichen Relationen und bilden eine wichtige Grundlage dieser Arbeit.

Herrn Prof. Dr.-Ing. habil. Wilhelm Caspary (i.R.) möchte ich für die Übernahme des Vorsitzes des Promotionsverfahrens und den Review meiner Arbeit danken. Die Diskussionen mit ihm lieferten wertvolle kleinere und größere Denkanstöße.

Für das angenehme und motivierende Arbeitsumfeld und die vielen aufschlussreichen fachlichen Diskussionen möchte ich mich herzlichst bei den Mitarbeitern der AGIS insbesondere bei Admire, Fei, Thorsten und Simone bedanken.

Heike und Ronja danke ich für die emotionale Unterstützung und den familiären Rückhalt, das Verständnis, die Liebe und die Geduld, welche besonders während der arbeitsreicheren Phasen der Promotionszeit beansprucht wurde. Meinen Eltern und Großeltern möchte ich für die Unterstützung während meines Studiums an der TU Dresden danken.

Curriculum Vitae

Name	Stephan Mäs
Date of Birth	12th of September 1977
Place of Birth	Weimar, Germany
Contact	StephanMaes@web.de

Education

1997-2003	Diplom Ingenieur in Geodesy (Dipl.-Ing. Univ.), Technical University of Dresden, Germany
12/2001 - 5/2002	International Affiliate, Department of Geomatics, University of Cape Town, South Africa
09/2002 - 07/2003	Internship and diploma at the Fraunhofer IGD, Darmstadt, Diploma Thesis: “Conception and Implementation of a Location Based Service for Mobile Devices”

Professional Experience

8/2003 - 9/2003	Research Assistent (wissenschaftliche Hilfskraft) at Fraunhofer IGD, Darmstadt
4/2004 - 4/2010	Research Associate (wissenschaftlicher Mitarbeiter) in AGIS Research Group, University of Bundeswehr Munich, Germany
Projects:	
2004 - 2005	Advancement of Geoservices, BMBF/DFG: mobile data acquisition via standardised Web services, quality assurance
2005 - 2010	INDOOR, BMBF/DLR: 3D building modelling, CityGML and indoor navigation
2006 - 2010	EduGI, e-Learning Program of the European Union: annually exchange of e-Learning courses between European universities
2009 - 2010	EUROSUR, European Commission DG JLS: conception of a pre-frontier intelligence picture within the framework of a European border surveillance system