Integritätsbedingungen für Geodaten

Von der Fakultät für Bauingenieurwesen und Geodäsie
der Gottfried Wilhelm Leibniz Universität Hannover
zur Erlangung des Grades

Doktor-Ingenieur (Dr.-Ing.)

genehmigte Dissertation von
Dipl.-Ing. Stefan Werder
geboren am 15.12.1979 in Wangen im Allgäu

Hannover 2014

Diese Arbeit ist auch veröffentlicht in:
Deutsche Geodätische Kommission bei der Bayerischen Akademie der Wissenschaften
Reihe C, Dissertationen, Heft Nr. 736, München 2014
ISBN 978-3-7696-5148-5, ISSN 0065-5325
www.dgk.badw.de

Vorsitzender der Prüfungskommission:	Prof. Dr.-Ing. Steffen Schön
Referentin:	Prof. Dr.-Ing. habil. Monika Sester
Korreferenten:	Prof. Dr.-Ing. habil. Christian Heipke
	Prof. Dr.-Ing. Wolfgang Reinhardt
Tag der Promotion:	25.03.2014

Abstract

Spatial data lay the foundation for decision making in research, economy and administration. Therefore, it is elementary that the used data is valid. Invalid data do not only lead to problems during processing and analysis, but can lead ultimately to erroneous decisions. If integrity constraints are defined, modelled and tested for all objects of a dataset, then invalid objects can be identified early and can be corrected, removed or marked as exception subsequently.

The processes that are presented within this dissertation make it possible to define, model and test integrity constraints in a simple, efficient and standard compliant way. This empowers producers and users of spatial data with the necessary basics and tools to use integrity constraints for their datasets.

This dissertation includes a requirement catalogue of 27 requirements covering all important aspects of integrity constraints, which serves with its detailed description and discussion not only as a reference but also as a basis for further research. Producers and users of spatial data can pick exactly those requirements from the catalogue that are of importance for their respective application.

The modelling of integrity constraints is based on the Model Driven Architecture (MDA) and the Object Constraint Language (OCL). By extending the OCL to the GeoOCL, spatial integrity constraints can be formalized platform independently. The GeoOCL is at the optimal abstraction level and is able to satisfy all requirements of the requirement catalogue. The suitability of the taken approach is demonstrated by many examples throughout the dissertation.

The definition, modelling and test of integrity constraints is demonstrated in this dissertation by two representative examples. In the first example, integrity constraints are analysed for land use information in both cadastral and topographic datasets provided by National Mapping Authorities. The approach is thereby representative for all datasets with coverages. Class descriptions for the different land use classes can be derived from first enriching the objects with geometric and topological measures and then analysing them with tools from descriptive statistics and exploratory data analysis. The class descriptions differ between the different land use classes and reveal interesting patterns and knowledge in the example datasets. Data mining is also used for extracting the relation between the classes of the cadastral and topographic dataset. The results of the classification show that the precision correlates with the complexity of the classification model and that not all land use classes can be predicted with equal precision.

Integrity constraints are also suitable for big datasets, which is demonstrated for buildings in open data with up to 31 million buildings per dataset. The approach is thereby representative for all datasets with point, line and polygonal geometries. Spatial partitioning is an important prerequisite for parallel processing. MapReduce workflows can then used to enrich the objects with geometric and topological measures, to determine the frequency of attribute values, and to filter the data. The data enrichment step scales linearly, which makes the approach suitable for processing any big dataset. The class descriptions of the buildings reveal once again interesting patterns and knowledge in the example datasets.

Keywords: Integrity constraints, Object Constraint Language, MapReduce

Zusammenfassung

Geodaten bilden die Grundlage für wichtige Entscheidungsprozesse in Forschung, Wirtschaft und Verwaltung. Eine elementare Anforderung an die verwendeten Daten ist deshalb, dass diese gültig sind. Fehlerhafte Daten sind nicht nur bei der Verarbeitung und Analyse der Daten problematisch, sondern können schließlich zu irrtümlich falschen Entscheidungen führen. Werden für alle Objekte eines Datensatzes Integritätsbedingungen definiert, modelliert und überprüft, so können ungültige Objekte frühzeitig identifiziert und anschließend entweder korrigiert, aus dem Datensatz entfernt oder als Ausnahme gekennzeichnet werden.

Mit den in dieser Arbeit vorgestellten Verfahren wird eine einfache, effiziente und standardkonforme Definition, Modellierung und Überprüfung von Integritätsbedingungen ermöglicht. Damit werden sowohl Produzenten als auch Nutzern von Geodaten die notwendigen Grundlagen und Werkzeuge an die Hand gegeben um Integritätsbedingungen für ihre Datensätze verwenden zu können.

Ein Anforderungskatalog mit 27 Anforderungen gibt nicht nur einen einzigartigen Überblick über alle wichtigen Aspekte von Integritätsbedingungen, sondern legt durch die detaillierte Beschreibung und Diskussion auch die Grundlage für weitere Forschungen. Aus den umfangreichen Anforderungen können von Produzenten und Nutzern dabei genau diejenigen ausgewählt werden, die für die jeweilige Anwendung von Relevanz sind.

Die Modellierung der Integritätsbedingungen basiert auf der Model Driven Architecture (MDA) und insbesondere auf der Object Constraint Language (OCL). Mit der Erweiterung der OCL zur GeoOCL können räumliche Bedingungen eindeutig plattformunabhängig formalisiert werden. Die GeoOCL befindet sich dabei auf der geeignetsten Abstraktionsebene der Formalisierung und kann zudem alle aufgestellten Anforderungen abdecken. Die vielen Beispiele von Bedingungen in der GeoOCL in der gesamten Arbeit belegen deutlich die Praxistauglichkeit der Sprache.

Die Definition, Modellierung und Überprüfung von Integritätsbedingungen wird in dieser Arbeit anhand zweier repräsentativer Beispiele veranschaulicht. Die Untersuchung der Flächennutzung in Geobasisdaten zeigt mehrere Anwendungen von Integritätsbedingungen auf. Das vorgestellte Vorgehen ist dabei repräsentativ für vollständig oder nahezu vollständig flächenüberdeckende Datensätze. Durch die Anreicherung der Objekte um geometrische und topologische Maße sowie deren Auswertung mittels deskriptiver Statistik und explorativer Datenanalyse können geeignete Klassenbeschreibungen für die Flächennutzungen erstellt werden. Diese unterscheiden sich signifikant zwischen den einzelnen Nutzungen und zeigen so interessante Muster und Wissen in den Daten auf. Das Data Mining kann jedoch auch für die Klassifikation eingesetzt werden, wobei die Zusammenhänge der Flächennutzung zwischen der Automatisierten Liegenschaftskarte (ALK) und dem Amtlichen Topographisch-Kartographischen Informationssystem (ATKIS) aufgedeckt werden. Dabei zeigt sich, dass die Klassifikationsgüte proportional zur Komplexität des verwendeten Modells ansteigt und nicht alle Flächennutzungen gleich gut prädiziert werden können.

Die Praktikabilität von Integritätsbedingungen selbst für umfangreiche Datenmengen zeigt die Untersuchung der Gebäude in Open Data mit bis zu 31 Millionen Objekten pro Datensatz. Das vorgestellte Vorgehen ist dabei repräsentativ für punkt-, linien- oder flächenhafte Datensätze. Um die Daten geeignet parallel verarbeiten zu können, müssen diese jedoch zuerst räumlich partitioniert werden. Die Anreicherung der Objekte um geometrische und topologische Maße, die Bestimmung der Häufigkeit von Attributwerten sowie die Filterung erfolgt anschließend mit MapReduce-Workflows. Die Anreicherung der Daten skaliert dabei annähernd linear, womit sich der gewählte Ansatz dazu eignet auf beliebig umfangreiche Daten angewendet zu werden. Die anschließende Klassenbeschreibung zeigt wiederum interessante Muster in den Daten auf.

Schlagworte: Integritätsbedingungen, Object Constraint Language, MapReduce

Inhaltsverzeichnis

1 Einleitung

1.1 Motivation

Geodaten bilden die Grundlage für wichtige Entscheidungsprozesse in Forschung, Wirtschaft und Verwaltung. Eine elementare Anforderung an die verwendeten Daten ist deshalb, dass diese gültig sind. Fehlerhafte Daten sind nicht nur bei der Verarbeitung und Analyse der Daten problematisch, sondern können schließlich zu irrtümlich falschen Entscheidungen führen. Werden für alle Objekte eines Datensatzes Integritätsbedingungen definiert, modelliert und überprüft, so können ungültige Objekte frühzeitig identifiziert und anschließend entweder korrigiert, aus dem Datensatz entfernt oder als Ausnahme gekennzeichnet werden. Der Begriff Integritätsbedingung wird dabei in dieser Arbeit als Synonym für die logische Konsistenz entsprechend des ISO 19113 (2002) Standards definiert, mit der die Einhaltung der logischen Regeln der Datenstrukturen, der Attributierung und der Beziehungen gefordert wird.

Integritätsbedingungen können für alle Objekte, sowohl für natürliche als auch künstliche, aufgestellt werden. Die Bedingungen können sich dabei auf die Thematik, Geometrie, Topologie und Dynamik der Objekte beziehen. So kann beispielsweise für Gebäude als Teil eines Liegenschaftskatasters gefordert werden, dass diese eine eindeutige Adresse aufweisen, eine Mindestgrundfläche nicht unterschreiten, sich nicht gegenseitig überlappen und korrekte Datumsangaben für die Genehmigung und Konstruktion aufweisen. Ebenso mannigfaltig wie die Integritätsbedingungen selbst kann auch deren Ursprung sein. Die Bedingungen können beispielsweise aus physikalischen Einschränkungen, aus Gesetzen und Vorschriften, aus Standards, aus ortsüblichen Gegebenheiten oder aus Beschränkungen der Repräsentation der Daten stammen.

Trotz ihrer großen Bedeutung finden Integritätsbedingungen häufig nur geringe Beachtung bei der Produktion und Nutzung von Geodaten. Defizite lassen sich dabei in den drei Bereichen der Definition, Modellierung und Überprüfung von Integritätsbedingungen identifizieren. Integritätsbedingungen sollten für jeden Datensatz definiert sein, da sie nicht nur festlegen wann Objekte gültig sind, sondern da sie auch aufzeigen können welche Beschränkungen ein Datensatz hinsichtlich der möglichen Nutzungen aufweist. Sind beispielsweise Gebäude in einem Datensatz nicht exakt überlappungsfrei, so kann der Datensatz zwar nicht für das Kataster verwendet werden, aber trotzdem für die Erstellung einer kleinmaßstäbigen Karte geeignet sein. Analog kann nach Überführung aller Anforderungen einer Anwendung in entsprechende Integritätsbedingungen eine größere Anzahl an Datensätzen auf ihre Eignung für die Anwendung untersucht werden. Oft unterbleibt die Definition der Integritätsbedingungen jedoch sowohl beim Produzenten als auch beim Nutzer von Geodaten, wobei letztere dann gezwungen sind Fehler zu beheben die bei der Verarbeitung der Daten auftreten. Werden Integritätsbedingungen definiert, so treten jedoch wiederum häufig Defizite in ihrer Modellierung auf. Dies ist darauf zurückzuführen, dass Bedingungen beispielsweise nur im Fließtext einer Datenspezifikation vermerkt sind oder integrativer Bestandteil einer einzelnen Software oder einer einzelnen Datenhaltungskomponente sind und damit nicht ohne weiteres auf andere Systeme übertragbar sind. Schließlich besteht bei der Überprüfung von Integritätsbedingungen ein mögliches Defizit darin, dass die vollständige Analyse aller Objekte umfangreicher Datensätze zeitlich so aufwändig sein kann, dass diese einfach unterbleibt.

1.2 Zielsetzung

Ziel der Arbeit ist es, ein neues Verfahren zu entwerfen um eine einfache, effiziente und standardkonforme Definition, Modellierung und Überprüfung von Integritätsbedingungen zu ermöglichen. Das Verfahren ist dabei allgemein gehalten, um dessen Anwendung für beliebige Geodaten zu gewährleisten. Die Ergebnisse der Arbeit sind damit für jeden von Interesse, der Geodaten produziert oder nutzt.

Die Arbeit bietet eine tiefgehende und umfangreiche Betrachtung von Integritätsbedingungen und deckt alle benötigten Aspekte ab. Dabei werden die meisten Aspekte vertieft behandelt, während wenige Aspekte lediglich kurz vorgestellt werden. Die Arbeit geht weit über andere Arbeiten hinaus, die lediglich Teilaspekte behandeln und damit oft nur isolierte Lösungsansätze bieten können.

Der erste Baustein der Arbeit basiert auf der detaillierten Analyse einer Vielzahl von Publikationen und zeigt durch eine Auflistung von möglichen Klassifikationen von Integritätsbedingungen sowie anhand einer detaillier-

ten Anforderungsanalyse die große Bandbreite des Themas auf. Dies soll Produzenten und Nutzer gleichermaßen sensibilisieren und dazu anregen über die Bedingungen nachzudenken, die in ihrem jeweiligen Anwendungsgebiet auftreten können.

Den zweiten Baustein der Arbeit bildet die Definition von Integritätsbedingungen. Die Bedingungen können erstens für einen Datensatz a priori festgelegt werden, sowohl vom Produzenten als Bestandteil der Datenmodellierung als auch vom Nutzer als Bestandteil der Anforderungsanalyse. Dies ist die übliche Herangehensweise für den Fall, dass fundiertes Wissen über die Beschaffenheit der Daten beziehungsweise deren geplante Anwendung vorhanden ist. Existiert dagegen kein fundiertes Wissen über die Beschaffenheit der Daten, dann können die Bedingungen auch aus einer Analyse der Daten gewonnen werden. Ein Beispiel für diesen zweiten Typ von Datensätzen ist das OpenStreetMap (Osm)-Projekt, das aufgrund seiner Vielzahl an freiwilligen Produzenten und seinem flexiblen Datenformat einen gewissen Grad an Varianz in der Beschaffenheit der Daten aufweist. Auf Basis der Analyse können für einen Datensatz dann erstmals Bedingungen aufgestellt werden, vorhandene Bedingungen um weitere Bedingungen ergänzt werden und Einblicke in die tatsächliche Verteilung der Daten gewonnen werden. Das fundierte Wissen über die Beschaffenheit der Daten ist aber auch oft dann nicht vorhanden, wenn zwei oder mehrere Datensätze miteinander integriert werden sollen. Für diesen dritten Datentyp werden durch die Analyse der Daten Einblicke in die Zusammenhänge der Daten ermöglicht, die dann als Bedingungen definiert werden können. Die Zusammenhänge können dabei so stark sein, dass aus einem Datensatz ein anderer Datensatz abgeleitet werden kann.

Für die Datenanalyse werden entsprechende Grundlagen der deskriptiven Statistik und explorativen Datenanalyse sowie Verfahren des maschinellen Lernens und des Data Mining vorgestellt und deren Anwendung an repräsentativen Beispielen veranschaulicht. Dies soll es Produzenten und Nutzer ermöglichen, die entsprechenden Integritätsbedingungen eines Datensatzes identifizieren und quantifizieren zu können.

Die Modellierung von Integritätsbedingungen bildet den dritten Baustein der Arbeit. Um eine wiederverwendbare, übertragbare und standardkonforme Repräsentation der Bedingungen zu erreichen, wird die plattformunabhängige Modellierungssprache Object Constraint Language (Ocl) (OMG, 2012) verwendet und um entsprechende Sprachkonstrukte für die Modellierung räumlicher Integritätsbedingungen erweitert. Dadurch wird es Produzenten und Nutzern ermöglicht Bedingungen eindeutig formal zu spezifizieren und in vielfältigen Implementierungen nutzen zu können.

Die Überprüfung von Integritätsbedingungen bildet schließlich den vierten und letzten Baustein der Arbeit. Dazu werden mögliche Ansätze und Implementierungen der Überprüfung kurz vorgestellt. Darüber hinaus wird mit Hadoop (White, 2012) ein Framework in Theorie und Praxis vorgestellt, mit dem umfangreiche Datensätze effizient parallel prozessiert werden können. So werden Produzenten und Nutzern die entsprechenden Werkzeuge an die Hand gegeben um Daten bezüglich der definierten und modellierten Integritätsbedingungen auch überprüfen zu können.

1.3 Gliederung

Die Arbeit gliedert sich in sieben Kapitel, die entsprechend den in der Zielsetzung definierten Bausteinen aufgebaut sind. Zuerst werden in Kapitel 2 Grundlagen vorgestellt und Begriffe definiert, die für das Verständnis der weiteren Kapitel benötigt werden. Diese umfassen neben allgemeinen Grundlagen auch Grundlagen der Erstellung formaler Modelle in der Softwareentwicklung, der raumbezogenen Datenmodellierung, der Statistik und des Data Mining sowie der Parallelisierung.

In Kapitel 3 werden auf Basis einer Analyse von publizierten Arbeiten einige Anwendungen vorgestellt, für die bereits Integritätsbedingungen verwendet werden. Um die Bandbreite von Integritätsbedingungen zu veranschaulichen werden zudem einige Klassifikationen aufgelistet. Schließlich werden existierende Ansätze zur Formalisierung vorgestellt.

Die vielseitigen Anforderungen an Integritätsbedingungen werden in Kapitel 4 in einem Anforderungskatalog mit insgesamt 27 Anforderungen zusammengefasst. Diese Aufstellung repräsentiert den status quo der Forschung und ermöglicht so erstmals einen vollständigen Überblick über alle wichtigen Aspekte von Integritätsbedingungen.

Die Modellierung und damit die formale Spezifikation von Integritätsbedingungen wird in Kapitel 5 diskutiert. Dazu wird die Ocl um entsprechende Sprachkonstrukte zur GeoOcl erweitert, um so unter anderem die in Kapitel 4 gestellten Anforderungen plattformunabhängig formalisieren zu können.

Anhand des Beispiels der Flächennutzung in Geobasisdaten werden in Kapitel 6 die Definition, Modellierung und Überprüfung von Integritätsbedingungen exemplarisch aufgezeigt. Dazu werden die Daten durch statisti-

sche Methoden analysiert um Bedingungen für bisher nicht untersuchte geometrische und topologische Maße aufzustellen und um einen Einblick in die tatsächliche Verteilung der Daten zu gewinnen. Zudem wird eine Klassifikation der Daten mit Verfahren des Data Mining durchgeführt um den Zusammenhang der beiden verwendeten Datensätze zu analysieren und so die Frage beantworten zu können, ob eine Ableitung eines Datensatzes aus einem anderen Datensatz möglich ist.

In Kapitel 7 wird anhand von Gebäuden in Open Data die parallele Verarbeitung umfangreicher Datensätze vorgestellt. Ebenso wie in Kapitel 6 werden auch entsprechende Integritätsbedingungen definiert, modelliert und überprüft. Die Daten werden durch statistische Methoden analysiert um erstmals Bedingungen für die Datensätze aufstellen zu können. Durch die Verwendung eines zweiten Datentyps wird unter anderem aufgezeigt, dass das entworfene Verfahren auf beliebige Geodaten übertragbar ist.

Das Kapitel 8 schließt die Arbeit mit einer Zusammenfassung und einem Ausblick ab.

2 Grundlagen

In diesem Kapitel werden Grundlagen vorgestellt und Begriffe definiert, die für das Verständnis der weiteren Kapitel benötigt werden. Dieses Kapitel ist dabei so gegliedert, dass einzelne Unterabschnitte auch direkt vor den thematisch zugeordneten weiteren Kapiteln der Arbeit gelesen werden können.

Zuerst werden in Abschnitt 2.1 allgemeine Grundlagen räumlicher Objekte vorgestellt und in Abschnitt 2.2 auf Aspekte der Datenqualität eingegangen. Für die Kapitel 3, 4 und 5 werden die benötigten Grundlagen der Object Constraint Language in Abschnitt 2.3 und der raumbezogenen Datenmodellierung in Abschnitt 2.4 vorgestellt. Anschließend werden für die Kapitel 6 und 7 allgemeine Grundlagen der Statistik in Abschnitt 2.5 und Grundlagen des Data Mining in Abschnitt 2.6 vorgestellt. Schließlich werden für das Kapitel 7 einige Grundlagen der Parallelisierung in Abschnitt 2.7 und der MapReduce-Ansatz in Abschnitt 2.8 vorgestellt.

2.1 Objekte mit Raumbezug

Objekte sind eine Repräsentation beschreibbarer Gegenstände der Realität. Die einzelnen Eigenschaften der Gegenstände werden durch ihre jeweils zugeordneten Objektattribute beschrieben. Diese können von thematischer, geometrischer, topologischer oder dynamischer Natur sein.

In den folgenden vier Unterabschnitten werden die vier Attributtypen Thematik, Geometrie, Topologie und Dynamik vorgestellt und anhand des Beispiels eines Gebäudes in Abbildung 2.1 veranschaulicht. Anschließend werden für den effizienten Zugriff auf räumliche Objekte in Unterabschnitt 2.1.5 räumliche Indexstrukturen und in Unterabschnitt 2.1.6 raumfüllende Kurven vorgestellt.

Abbildung 2.1: Beispiel für Attributtypen eines Objekts

2.1.1 Thematik

Thematische Attribute beziehen sich auf das Thema und damit die Bedeutung von Objekten. Das Thema des Objekts in Abbildung 2.1 enthält beispielsweise ein Gebäude als Teil des Katasters. Thematische Attribute werden häufig auch als Sach- oder Fachdaten bezeichnet.

Thematische Attribute lassen sich weiter unterteilen, wobei im Folgenden die Definitionen nach Han u. a. (2011) verwendet werden. *Nominale Attribute* beziehen sich auf Namen und nehmen als Werte Symbole oder Namen an. Jeder Wert repräsentiert eine Kategorie, einen Zustand oder eine Klasse. Die einzelnen Werte können nicht geordnet werden, da sie alle gleichwertig sind. Im Beispiel in Abbildung 2.1 ist die Objektart ein nominales Attribut. *Binäre Attribute* sind nominale Attribute mit lediglich zwei Zuständen. Sind diese gleichwertig, dann ist das binäre Attribut symmetrisch, ansonsten ist es asymmetrisch. Im Beispiel ist das Attribut Unterirdisch binär. *Ordinale Attribute* weisen eine Ordnung beziehungsweise Reihenfolge auf. Die einzelnen Werte können nach ihrer Größe geordnet werden, jedoch lässt sich der Abstand zwischen zwei Werten nicht bestimmen. Im Beispiel ist das Attribut Ausstattung ordinal. Nominale, binäre und ordinale Attribute sind qualitativ, da sie Eigenschaften nur beschreiben. *Numerische Attribute* sind hingegen quantitativ bestimmbar und werden durch ganze oder reelle Zahlen abgebildet. Im Beispiel in Abbildung 2.1 sind die beiden Attribute Stockwerke und Mietpreis numerische Attribute. Für numerische Attribute sollte soweit möglich die Einheit eines Wertes angegeben werden. Im Beispiel trifft dies auf das Attribut Mietpreis zu, das erst dann sinnvoll interpretierbar ist, wenn dessen Einheit [EUR pro m^2] angegeben wird.

Attribute werden zudem in diskret und kontinuierlich unterschieden. *Diskrete Attribute* besitzen eine abzählbare Menge an möglichen Werten und umfassen nominale, binäre und ordinale Attribute sowie numerische Attribute mit ganzen Zahlen. *Kontinuierliche Attribute* besitzen Werte die nicht abzählbar sind und umfassen numerische Attribute mit reellen Zahlen.

2.1.2 Geometrie

Die Geometrie beschreibt die Form, Größe und Lage von Objekten. Prinzipiell wird dabei in Vektor- und Rastergeometrien unterscheiden. Vektorielle Geometrien basieren auf den in Abbildung 2.2 dargestellten Primitiven, die auch als Basistypen bezeichnet werden. Der *Punkt* (engl. 'point') ist dabei das einfachste geometrische Primitiv und modelliert die zwei- oder dreidimensionale Geometrie eines punktuellen Objekts. Werden mehrere aufeinanderfolgende Punkte durch Geradenstücke miteinander verbunden, so entsteht eine *Linie* (engl. 'line', 'arc'). Sind der erste und letzte Stützpunkt einer Linie identisch, so ist diese geschlossen und kann als *Fläche* (engl. 'area', 'region') interpretiert werden. Rastergeometrien basieren dagegen auf einem gleichmäßigen Gitter, dessen rechteckige Zellen direkt aneinander anschließen. Weitere Details zu Vektor- und Rastergeometrien sind beispielsweise in Bartelme (2005) zu finden.

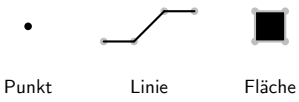


Abbildung 2.2: Zweidimensionale geometrische Primitive

Eine wichtige abgeleitete Geometrie ist das minimal umschließende Rechteck (engl. 'Minimum Enclosing Rectangle (MER)'). Das MER lässt sich für jedes zweidimensionale geometrische Primitiv bestimmen, wobei dieses für einen Punkt wiederum zu einem Punkt degradiert. In Abbildung 2.3 sind für die grau eingefärbte Fläche drei MER dargestellt, die auf jeweils unterschiedlichen Methoden basieren. Die MER sind jeweils so geformt und orientiert, dass ein Rechteck die Objektgeometrie vollständig enthält und dabei gleichzeitig die Breite des Rechtecks für die entsprechende Methode minimal ist, d.h. es gibt kein anderes Rechteck das ebenso die Geometrie enthält und eine geringere Breite aufweist.

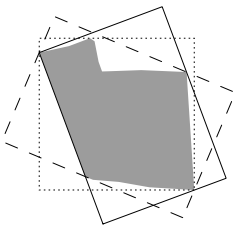


Abbildung 2.3: Unterschiedliche minimal umschließende Rechtecke für eine Geometrie

Für zwei der Methoden wird die *konvexe Hülle* einer Objektgeometrie verwendet. Die konvexe Hülle einer Geometrie enthält für jedes Paar von Stützpunkten deren Verbindung durch eine Gerade. Das Beispiel in Abbildung 2.4 zeigt die grundlegende Konstruktion der komplexen Hülle für eine einfache Objektgeometrie. Zuerst werden alle Stützpunkte der Geometrie paarweise durch Linien verbunden. Die außenliegenden Liniensegmente ergeben dann die konvexe Hülle der Geometrie.

Abbildung 2.4: Konstruktion der komplexen Hülle einer Objektgeometrie

Die erste Methode erzeugt ein *achsenparalleles umschließendes Rechteck* (engl. 'bounding box'), das durch die in Richtung der Koordinatenachsen minimalen und maximalen Koordinaten der Stützpunkte begrenzt wird. Das so erzeugte MER eignet sich insbesondere für die Beschleunigung räumlicher Abfragen, jedoch nur bedingt zur Beschreibung der Form einer Geometrie. Für die Geometrie in Abbildung 2.3 ist das achsenparallele

umschließende Rechteck gepunktet dargestellt. Die zweite Methode erzeugt ein *minimal umschließendes Rechteck* (engl. 'minimum enclosing rectangle'), das auf der konvexen Hülle der Objektgeometrie basiert und damit an der Geometrie und nicht an den Koordinatenachsen ausgerichtet ist. Das minimal umschließende Rechteck enthält dabei mindestens zwei Stützpunkte der Objektgeometrie. In Abbildung 2.3 ist das minimal umschließende Rechteck gestrichelt dargestellt. Die dritte Methode erzeugt ein *minimal umschließendes angepasstes Rechteck* (engl. 'minimum enclosing adapted rectangle'). Dieses basiert wie das MER der zweiten Methode auf der konvexen Hülle, jedoch enthält das minimal umschließende angepasste Rechteck zwingend eine Linie der Objektgeometrie. In Abbildung 2.3 ist das minimal umschließende angepasste Rechteck als geschlossene Linie dargestellt. In dieser Arbeit wird bevorzugt das minimal umschließende angepasste Rechteck verwendet, dessen Breite zwar geringfügig größer sein kann als die des minimal umschließenden Rechtecks, dabei jedoch besser an die Form der Objektgeometrie angepasst ist.

Die *Orientierung* beschreibt die Lage von Geometrien in Bezug auf Koordinatensysteme oder relativ zueinander (siehe Abbildung 2.5). Die absolute Orientierung einer Geometrie beschreibt die Orientierung in Bezug auf ein Koordinatensystem, beispielsweise in Bezug zur Nordrichtung. Die relative Orientierung beschreibt die Lage zweier Geometrien zueinander. Diese kann sich wiederum auf ein Koordinatensystem beziehen, oder auf ausgezeichnete Charakteristika einer Geometrie. So ist in Abbildung 2.5 beispielsweise der Eingang eines Gebäudes angegeben, womit in Bezug auf diesen die Bezeichnungen links, rechts, vor und hinter verwendet werden können um die Lage einer zweiten Geometrie zu beschreiben.

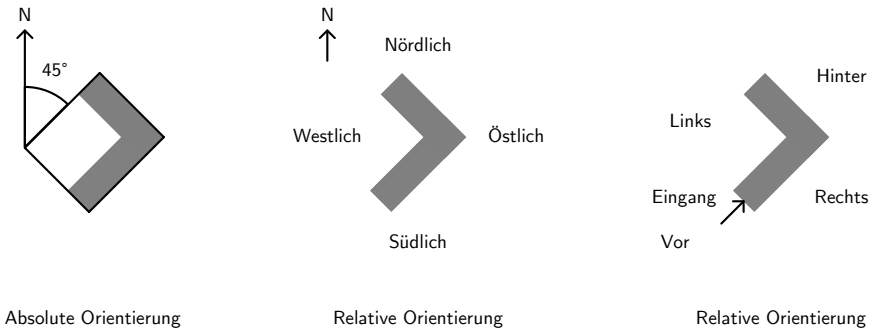


Abbildung 2.5: Absolute und relative Orientierung

Die *Distanz* quantifiziert die Entfernung zwischen zwei Geometrien (siehe Abbildung 2.6). Diese kann sich direkt auf die Geometrien selbst oder auf abgeleitete Geometrien beziehen. So wird beispielsweise in Abbildung 2.6 die abgeleitete Distanz als Distanz zwischen den minimal umschließenden Rechtecken zweier Geometrien bestimmt.

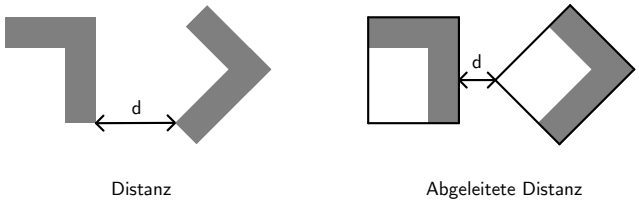


Abbildung 2.6: Distanz und abgeleitete Distanz

2.1.3 Topologie

Die Topologie beschreibt die Beziehung von Objekten zueinander. Analog zu den geometrischen Primitiven sind die topologischen Primitive in Abbildung 2.7 dargestellt. Ein *Knoten* (engl. 'node') ist das einfachste topologische Primitiv. Die Verbindung zweier Knoten erfolgt durch eine *Kante* (engl. 'edge'), die in ihrer Navigierbarkeit entweder ungerichtet sein kann oder gerichtet. Ist die Kante gerichtet, dann kann nur von einem der beiden Knoten zum anderen navigiert werden, nicht jedoch zurück. Ist der Start- und Endpunkt einer Reihe aufeinanderfolgender Kanten identisch, so bilden diese eine *Masche* (engl. 'face'). Aus den vorgestellten topologischen Primitiven lassen sich komplexe Konstrukte ableiten, wie beispielsweise ein Straßennetzwerk, dessen Knoten den Kreuzungen und dessen Kanten den Straßengeometrien entsprechen. Dabei können den Kanten zusätzlich Gewichte zugewiesen werden, wie beispielsweise die Länge der Geometrie, um dann komplexe Routingalgorithmen auf dem Netzwerk ausführen zu können.

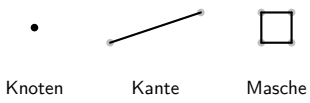


Abbildung 2.7: Zweidimensionale topologische Primitive

Die Beziehung zwischen zwei Geometrien kann durch sogenannte *topologische Relationen* ausgedrückt werden. In Egenhofer (1989) und Egenhofer und Herring (1991) wird das als 9 Intersection Method (9IM) bekannte Modell zur Beschreibung der Schnittmengen von zweidimensionalen Geometrien im zweidimensionalen Raum definiert. Die 9IM basiert auf dem Schnitt zweier Geometrien, wobei für jede Geometrie A das Innere A°, der Rand ∂A und das Äußere A^- berücksichtigt wird (siehe Abbildung 2.8). Durch den Schnitt zweier Geometrien A und B ergeben sich dann acht benannte topologische Relationen in der 9IM, die in Abbildung 2.9 dargestellt sind. In Gleichung 2.1 ist die Schnittmatrix für den Fall aufgelistet, dass die topologische Relation der Geometrien A und B 'covers' ist. Die Schnittmatrizen für die weiteren topologischen Relationen sind in Egenhofer und Herring (1991) angegeben. Dort werden auch die topologischen Relationen zwischen zwei Linien sowie zwischen einer Linie und einem Polygon detailliert erläutert.

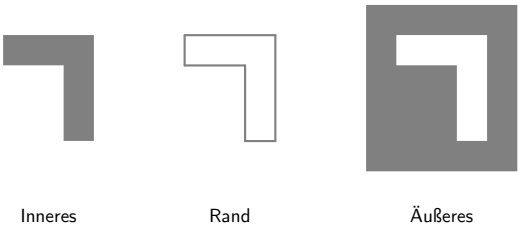


Abbildung 2.8: Inneres, Rand und Äußeres einer Geometrie

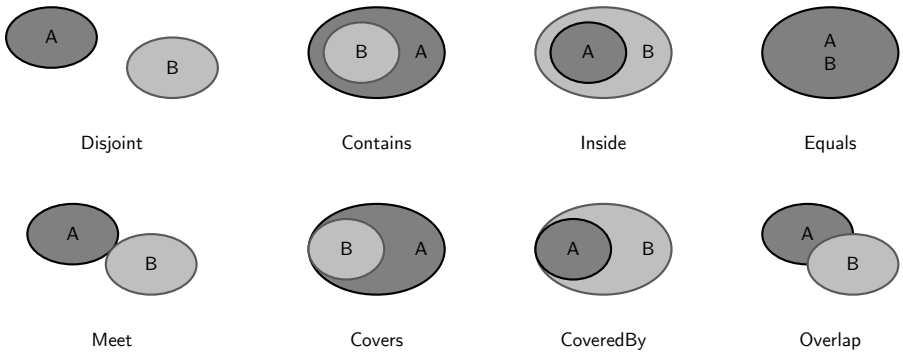


Abbildung 2.9: Topologische Relationen zwischen zwei Flächen nach Egenhofer und Herring (1991)

$$\text{covers}(A,B) = \begin{array}{c|ccc} & B^\circ & \partial B & B^- \\ \hline A^\circ & \neg\varnothing & \neg\varnothing & \neg\varnothing \\ \partial A & \varnothing & \neg\varnothing & \neg\varnothing \\ A^- & \varnothing & \varnothing & \neg\varnothing \end{array} \tag{2.1}$$

Neben der 9IM existieren weitere Varianten um die topologische Relation von Geometrien ausdrücken zu können. In Clementini u. a. (1993) wird für den Schnitt zweier Geometrien nicht nur zwischen leerer und nicht leerer Menge unterschieden, sondern stattdessen die Dimension der Schnittgeometrie bzw. -geometrien S nach Gleichung 2.2 berücksichtigt. So kann die 9IM zur Dimensionally Extended 9 Intersection Method (DE-9IM) erweitert werden, indem die Elemente in den Schnittmatrizen die Dimension des jeweiligen Schnitts bezeichnen. In Clementini u. a. (1993) werden zudem lediglich fünf benannte topologischen Relationen für das sogenannte Calculus Based Model (CBM) aufgestellt ('touch', 'in', 'cross', 'overlap', 'disjoint'), die sich nur auf den Schnitt des Inneren und des Rands von Geometrien beziehen, d.h. das Äußere wird bei den Relationen nicht berücksichtigt. Dagegen stellen Randell u. a. (1992) zehn sogenannte dyadische topologische Relationen auf, die zusammen mit ihren Inversen den Region Connected Calculus (RCC) bilden. Sowohl das CBM als auch der RCC werden im Vergleich zur 9IM seltener verwendet, da letztere beispielsweise in räumlichen Standards referenziert wird (siehe Abschnitt 2.4).

$$\dim(S) = \begin{cases} -, & \text{falls } S = \varnothing \\ 0, & \text{falls } S \text{ mindestens einen Punkt und keine Linien oder Flächen enthält} \\ 1, & \text{falls } S \text{ mindestens eine Linie und keine Flächen enthält} \\ 2, & \text{falls } S \text{ mindestens eine Fläche enthält} \end{cases} \tag{2.2}$$

2.1.4 Dynamik

Die Dynamik von Objekten kann durch Zeitpunkte, Intervalle und Zyklen beschrieben werden. Ein *Zeitpunkt* (engl. 'instant') bezeichnet exakt einen zeitlichen Zustand. Ein *Intervall* (engl. 'interval', 'period') bezeichnet den Zeitraum zwischen zwei Zeitpunkten. Ein *Zyklus* (engl. 'cycle') bezeichnet eine feste Aneinanderreihung sich zeitlich wiederholender Intervalle, beispielsweise der Jahreszeiten. Zeitpunkte und Intervalle können auf einer eindimensionalen Achse als Punkte und Linien abgetragen werden, weshalb die Zeit häufig in Ergänzung der drei Dimensionen von Geometrien als vierte Dimension bezeichnet wird. Sowohl Zeitpunkte als auch Intervalle sind im ISO 19108 (2002) Standard definiert und deren Beziehungen zu den räumlichen Dimensionen beschrieben.

Für Zeitpunkte und Intervalle lassen sich auf der eindimensionalen Achse geometrische und topologische Beziehungen aufstellen. Geometrische Beziehungen bezeichnen dabei die Distanz zwischen zwei Zeitpunkten oder Intervallen. Topologische Beziehungen bezeichnen die relative Lage von Zeitpunkten und Intervallen. Dabei können zwischen zwei zeitlichen Intervallen insgesamt 13 Beziehungen aufgestellt werden, die auch als *Allen-Relationen* bezeichnet werden (Allen, 1983). Diese Beziehungen sind in Abbildung 2.10 abgebildet, wobei die sechs symmetrischen Beziehungen von Y zu X nicht enthalten sind. Die Allen-Relationen werden auch im ISO 19108 (2002) Standard verwendet, jedoch mit abweichenden Bezeichnungen.

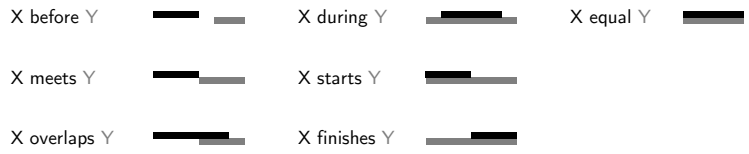


Abbildung 2.10: Beziehungen zeitlicher Intervalle nach Allen (1983)

2.1.5 Räumliche Indexstrukturen

Um den Zugriff auf räumliche Daten zu beschleunigen werden häufig *räumliche Indexstrukturen* verwendet. Diese unterteilen den Raum in Partitionen und weisen dadurch jeder Geometrie eindeutig eine oder mehrere Partitionen zu. Dadurch können beispielsweise effizient die Nachbarn einer Geometrie bestimmt werden, ohne über alle Geometrien eines Datensatzes iterieren zu müssen. In dieser Arbeit wird Bezug auf die zwei räumlichen Indexstrukturen Quadtree und R-Baum genommen, die im Folgenden kurz vorgestellt werden. Weitere Details können den jeweils referenzierten Publikationen entnommen werden.

Der in Finkel und Bentley (1974) vorgestellte *Quadtree* unterteilt den Raum rekursiv in quadratische überlappungsfreie Flächen, die jeweils eine maximale Kapazität aufweisen. Überschreitet die Anzahl der Elemente einer Fläche die maximale Kapazität, dann wird die Fläche in vier gleich große Quadranten unterteilt. Im Beispiel in Abbildung 2.11 ist die räumliche Repräsentation sowie die entsprechende Repräsentation als Baum eines Quadtrees dargestellt, wobei die maximale Kapazität zwei Elemente beträgt. Die Flächen 0 und 1 enthalten jeweils maximal zwei Elemente und sind deshalb nicht weiter unterteilt. Die Fläche 2 enthält insgesamt vier Elemente und ist deshalb rekursiv in vier Quadranten unterteilt. Die Fläche 3 enthält insgesamt neun Elemente und ist deshalb in vier Quadranten unterteilt, wobei die Fläche 3.1 wiederum unterteilt ist. So halbieren sich beim Quadtree die Seitenlängen der Flächen durch jede Unterteilung.

Der *R-Baum* nach Guttman (1984) basiert auf achsenparallelen umschließenden Rechtecken, bei denen die Blattknoten jeweils das MER einer Geometrie repräsentieren. Mehrere benachbarte MER werden in einer mehrstufigen Hierarchie zu Clustern zusammengefasst. Im Beispiel in Abbildung 2.12 ist die räumliche Repräsentation sowie die entsprechende Repräsentation als Baum eines R-Baums dargestellt.

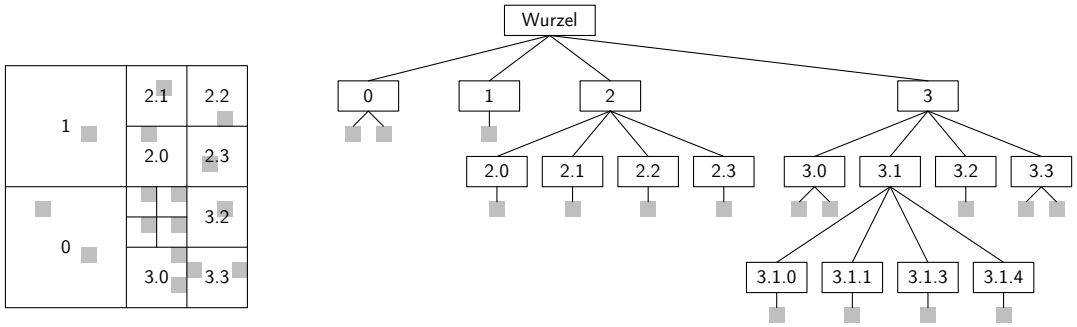


Abbildung 2.11: Beispiel für einen Quadtree

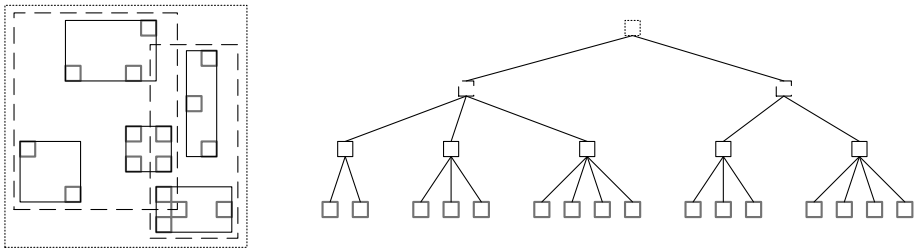


Abbildung 2.12: Beispiel für einen R-Baum

2.1.6 Raumfüllende Kurven

Mit *raumfüllenden Kurven* können zwei- oder mehrdimensionale Daten auf eine Dimension abgebildet werden. Ausgehend von einem Einheitsquadrat das in vier Quadrate unterteilt ist, durchläuft eine raumfüllende Kurve alle Quadrate exakt einmal, wobei die Kurve lückenlos und ohne Selbstüberschneidungen ist. Die Quadrate können wiederum rekursiv unterteilt werden und somit füllt die Kurve schließlich die gesamte Fläche aus. Somit kann die Fläche als eindimensionale Linie repräsentiert werden. In dieser Arbeit wird Bezug auf die Hilbert- und die Z-Kurve genommen, die im Folgenden kurz vorgestellt werden. Weitere Details können den jeweils referenzierten Publikationen entnommen werden.

Die *Hilbert-Kurve* ist nach Hilbert (1891) benannt und in der ersten bis dritten Ordnung in Abbildung 2.13 dargestellt[1]. Die Ordnung wird dabei auch als Iteration oder Auflösung bezeichnet. Das Grundmuster der Hilbert-Kurve ist durch die erste Ordnung definiert. Wird die Ordnung um eins erhöht, dann wird das Grundmuster durch Spiegelungen und Rotationen geeignet vervielfacht, wobei die allgemeinen Eigenschaften von raumfüllenden Kurven erhalten bleiben. Dabei wird die Kurve umso dichter, je höher die Ordnung gewählt wird.

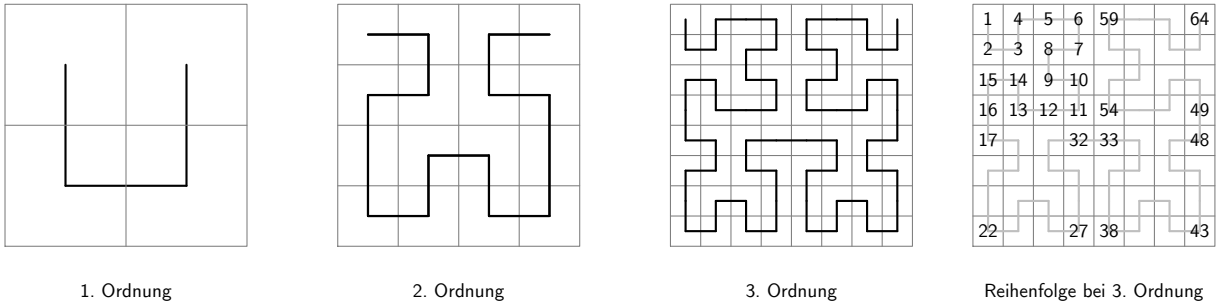


Abbildung 2.13: Hilbert-Kurve

[1] Im Gegensatz zu Hilbert (1891) ist die Hilbert-Kurve in Abbildung 2.13 um 180° gedreht dargestellt, um eine bessere Vergleichbarkeit mit der Z-Kurve zu gewährleisten.

Eine weitere wichtige raumfüllende Kurve ist die *Z-Kurve*, die in Morton (1966) vorgestellt wird[2]. Das Grundmuster der Z-Kurve ist durch die erste Ordnung definiert und entspricht der Form des Buchstabens 'Z'. Die weiteren Ordnungen ergeben sich wiederum durch Spiegelungen und Rotationen. Der z-Wert eines Punkts in der Fläche kann dabei einfach durch eine Verschränkung der Bits der Koordinaten des Punkts ermittelt werden. Dagegen ist die Bestimmung des Werts eines Punktes in einer Hilbert-Kurve rechnerisch aufwändiger.

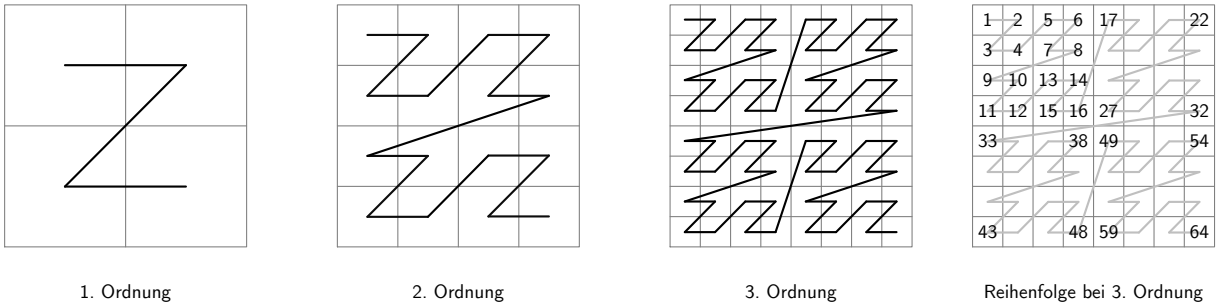


Abbildung 2.14: Z-Kurve

2.2 Datenqualität

Daten sind lediglich eine Repräsentation der Realität, die aufgrund ihrer Natur komplex, mannigfaltig und dynamisch ist. Ein Datenmodell wird immer für einen begrenzten Zweck entworfen. Es vereinfacht und aggregiert die abzubildenden Elemente der Realität und lässt die anderen Elemente aus. Ein Datensatz ist deshalb immer, zumindest zu einem gewissen Grad, unpräzise, ungenau, unvollständig und veraltet.

Datenqualität wird im ISO 19101 (2002) Standard definiert als Gesamtheit der Eigenschaften eines Produktes die dazu beitragen festgelegte und implizite Anforderungen zu erfüllen (engl. 'totality of characteristics of a product that bear on its ability to satisfy stated and implied needs'). Die Brauchbarkeit (engl. 'fitness for use') eines Datensatzes bestimmt also dessen Qualität.

Beschreibung der Datenqualität

Für die Beschreibung der Datenqualität spezifiziert der ISO 19113 (2002) Standard nicht-quantitative und quantitative Elemente. Die nicht-quantitativen *Übersichtselemente* der Datenqualität stellen allgemeine Informationen über den Zweck, die Nutzung und die Herkunft eines Datensatzes bereit. Der Zweck entspricht der Sicht des Produzenten und charakterisiert dessen Absicht hinter der Erzeugung des Datensatzes und die beabsichtigen Verwendungszwecke. Die Nutzung entspricht der Sicht des Benutzers und stellt Informationen über die Anwendungstypen bereit, für die die Daten genutzt werden. Die Herkunft beschreibt die geschichtliche Entwicklung des Datensatzes von seiner Erfassung bis zu seinem aktuellen Zustand.

Im Gegensatz zu den Übersichtselementen spezifizieren die *Elemente der Datenqualität* quantitative, und damit messbare, Informationen inwieweit die Daten die Anforderungen der Produktspezifikation erfüllen. Der ISO 19113 (2002) Standard spezifiziert fünf Elemente der Datenqualität:

- *Vollständigkeit*: Vorhandensein und Fehlen von Objekten, ihren Attributen und Beziehungen
- *Logische Konsistenz*: Grad der Einhaltung der logischen Regeln der Datenstrukturen, der Attributierung und der Beziehungen (Datenstrukturen können konzeptuell, logisch oder physikalisch sein)
- *Räumliche Genauigkeit*: Genauigkeit der Lage von Objekten
- *Zeitliche Genauigkeit*: Genauigkeit der zeitlichen Attribute und zeitlichen Beziehungen von Objekten
- *Thematische Genauigkeit*: Genauigkeit der quantitativen Attribute und Korrektheit der nicht-quantitativen Attribute und der Klassifizierung der Objekte und ihrer Beziehungen

[2] Die Z-Kurve wird auch als Lebesgue-Kurve, Morton-Code oder Morton-Ordnung bezeichnet.

Logische Konsistenz

Von den fünf Elementen der Datenqualität ist nur die logische Konsistenz von Bedeutung für diese Arbeit. Diese Auswahl basiert auf den Klassen von Evaluationsmethoden für die Datenqualität, wie sie im ISO 19114 (2003) Standard definiert sind. *Indirekte Evaluationsmethoden* nutzen externes Wissen, wie Übersichtselemente der Datenqualität oder andere Qualitätsberichte, für die Bestimmung der Qualität eines Datensatzes. *Direkte Evaluationsmethoden* bestimmen die Datenqualität durch Vergleich des Datensatzes mit internen oder externen Referenzinformationen. Interne Referenzinformationen werden nur aus dem Datensatz selbst bestimmt. Externe Referenzinformationen erfordern einen externen, d.h. zweiten, Datensatz höherer Qualität, um diesen mit dem ersten Datensatz vergleichen zu können. Diese externen Daten werden auch als Referenzdaten bzw. Kontrolldaten bezeichnet. Um beispielsweise die Vollständigkeit eines Datensatzes zu bestimmen kann die Gesamtanzahl der Objekte mit der entsprechenden Anzahl des Referenzdatensatzes verglichen werden. Der Quotient der beiden Zahlen kann dann als Maß für die Qualität verwendet werden. In dieser Arbeit wird jedoch vorausgesetzt, dass ein Referenzdatensatz entweder nicht verfügbar ist oder es keinen Datensatz höherer Qualität gibt. Außer der logischen Konsistenz verlangen alle anderen Elemente der Datenqualität das Vorhandensein eines Referenzdatensatzes, weshalb diese im Folgenden nicht weiter berücksichtigt werden.

Für die logische Konsistenz definiert der ISO 19113 (2002) Standard für Unterelemente der Datenqualität:

- *Konzeptuelle Konsistenz*: Einhaltung der Regeln des konzeptuellen Schemas
- *Konsistenz des Wertebereiches*: Einhaltung der Werte des Wertebereiches
- *Konsistenz des Formats*: Grad zu welchem Daten in Übereinstimmung mit der physischen Struktur des Datensatzes gespeichert sind
- *Topologische Konsistenz*: Korrektheit der explizit gespeicherten topologischen Eigenschaften eines Datensatzes

Im Kontext dieser Arbeit wird der Begriff Integritätsbedingung als Synonym für die logische Konsistenz entsprechend des ISO 19113 (2002) Standards verwendet. Die in dieser Arbeit definierten Integritätsbedingungen fordern demnach alle die Einhaltung der logischen Konsistenz und damit der logischen Regeln der Datenstrukturen, der Attributierung und der Beziehungen.

2.3 Model Driven Architecture

Die Model Driven Architecture (MDA) ist ein von der Object Management Group (OMG) definiertes Framework, bei dem formale Modelle die Grundlage für die Softwareentwicklung bilden (Hitz u. a., 2005). Die Spezifikation eines Systems erfolgt in der MDA mit plattformunabhängigen Modellen, die dann mit Hilfe von Transformationen zuerst in plattformspezifische Modelle und schließlich in plattformspezifische Implementierungen überführt werden können.

Im Folgenden werden in Unterabschnitt 2.3.1 zuerst einige grundlegende Konzepte der MDA vorgestellt. Anschließend wird in Unterabschnitt 2.3.2 die Modellierungssprache Unified Modeling Language (UML) und in Unterabschnitt 2.3.3 die dazugehörige Sprache zur Definition von Integritätsbedingungen mit dem Namen Object Constraint Language (OCL) vorgestellt. Die OCL wird in Kapitel 5 erweitert um auch räumliche Integritätsbedingungen definieren zu können.

2.3.1 Grundlegende Konzepte

Die Transformation von der Spezifikation bis zur Implementierung basiert auf drei aufeinanderfolgenden Schritten und deren Modellen, die in OMG (2003) und Hitz u. a. (2005) beschrieben werden:

- Zuerst wird ein Modell mit hohem Abstraktionslevel erzeugt, das als *Platform Independent Model* (PIM) bezeichnet wird. Dieses Modell ist unabhängig von der Implementierungstechnik und erlaubt so einen Austausch der Implementierungsplattform ohne Änderungen des Modells. Dabei beeinflusst der Grad der Abstraktion die Anzahl der unterstützten Plattformen. Bei einem hohen Abstraktionsgrad sind viele Plattformen für die Umsetzung geeignet, wohingegen bei einem niedrigeren Abstraktionsgrad entsprechend weniger Plattformen geeignet sind.

– Das PIM wird anschließend in ein Modell transformiert, das die Details einer bestimmten Implementierungsplattform berücksichtigt. Dieses Modell wird als *Platform Specific Model* (PSM) bezeichnet. Aus einem PIM können dabei mehrere plattformspezifische Modelle erzeugt werden.

– Schließlich wird das PSM in *Code für eine spezifische Plattform* transformiert. Je näher das PSM am eigentlichen Code ist, desto geringer sind die in diesem Schritt anfallenden Änderungen.

Die drei Schritte sind nochmals in Abbildung 2.15 dargestellt und um ein Beispiel ergänzt. Im Beispiel wird das PIM mit der Modellierungssprache UML erstellt, die in Unterabschnitt 2.3.2 detaillierter beschrieben wird. Anschließend wird das PIM in zwei plattformspezifische Modelle transformiert. Das erste PSM definiert die Speicherung von Objekten in einer Datenbank und basiert deshalb auf der Structured Query Language (SQL). Das zweite PSM definiert die Speicherung von Objekten in der Programmiersprache Java, sogenannten Java Beans. Die Transformation der plattformspezifischen Modelle in Code kann wie bei den Java Beans eindeutig sein, kann jedoch wie beim SQL-Modell auch von der eingesetzten Plattform abhängen, da beispielsweise herstellerspezifische Definitionen berücksichtigt werden müssen.

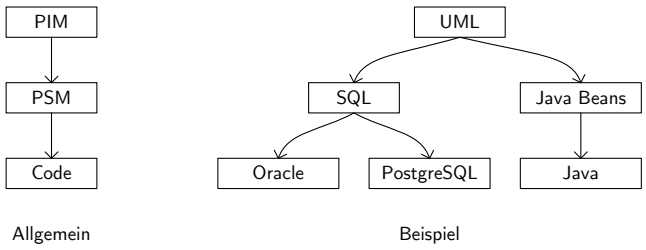


Abbildung 2.15: Transformationen in der MDA

Die MDA verfolgt dabei drei prinzipielle Ziele, die in OMG (2003) beschrieben werden. Das erste Ziel ist *Portabilität*, da auf Basis eines einmalig definierten PIM Implementierungen für verschiedene Plattformen abgeleitet werden können. Dadurch kann beispielsweise die aktuell in einem System eingesetzte Softwaretechnologie einfacher durch eine andere Softwaretechnologie ersetzt werden. Das zweite Ziel der MDA ist *Interoperabilität*, da zwei Systeme die dasselbe PIM verwenden einfach miteinander kommunizieren können. Das dritte Ziel ist *Wiederverwendbarkeit*, da Elemente des Modells eines Systems einfach für die Modelle anderer Systeme übernommen werden können. Zudem sind für die Transformationen zwischen den einzelnen Modellen teilweise automatische oder halbautomatische Werkzeuge verfügbar, die damit dem Einsatz der MDA für die Softwareentwicklung zusätzliche Attraktivität verleihen.

2.3.2 Unified Modeling Language

Die UML ist eine grafische Modellierungssprache für die Spezifikation, Konstruktion und Dokumentation von Softwaresystemen (OMG, 2011b). Die UML basiert auf objektorientierten Konzepten und kann universell für unterschiedlichste Anwendungsgebiete und Implementierungsplattformen eingesetzt werden. Die UML ist von der OMG und als International Organization for Standardization (ISO) 19505 standardisiert.

Mit der UML können Modelle mit Hilfe von insgesamt 14 Diagrammen sowie ergänzenden textuellen Definitionen spezifiziert werden. In dieser Arbeit wird die aktuelle Version 2.4.2 der UML verwendet, die in OMG (2011b) und OMG (2011c) definiert ist. Die Diagramme lassen sich in die zwei Gruppen Struktur- und Verhaltensdiagramme unterscheiden. Für diese Arbeit ist lediglich das Klassendiagramm von Interesse, das zu den Strukturdiagrammen zählt, und im Folgenden entsprechend seiner Definition in OMG (2011b), OMG (2011c) und Hitz u. a. (2005) vorgestellt wird.

Klassendiagramm

Ein UML-Klassendiagramm enhält Klassen mit deren Attributen und Operationen sowie die Beziehungen zwischen den Klassen. In Abbildung 2.16 ist ein entsprechend der UML-Stilrichtlinien entworfenes Klassendiagramm dargestellt, das grundlegende räumliche Geometrien modelliert. Dessen Elemente werden im Folgenden zuerst allgemein und dann am Beispiel vorgestellt.

Zuerst werden die Elemente einer Klasse vorgestellt. Eine Klasse ist im UML-Diagramm durch ein Rechteck gekennzeichnet, das in bis zu drei Abschnitte unterteilt sein kann. Der *Name* der Klasse wird im obersten rechteckigen Abschnitt zentriert und mit fettgedruckter Schrift dargestellt. Ist die Klasse abstrakt, dann wird deren Namen zusätzlich kursiv hervorgehoben. Im mittleren rechteckigen Abschnitt folgen die *Attribute* der Klasse.

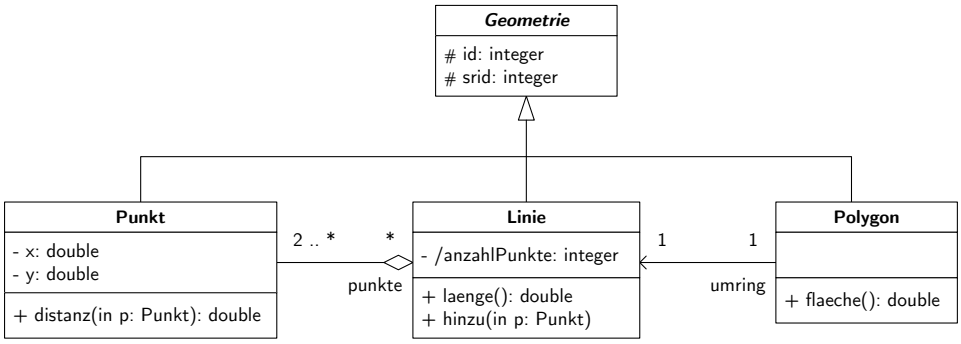


Abbildung 2.16: UML-Klassendiagramm

Dem Attributnamen vorangestellt ist die Sichtbarkeit des Attributs, dessen mögliche Angaben in Tabelle 2.1 aufgelistet sind. Ein Attribut kann zusätzlich mit einem vorangestellten Schrägstrich als abgeleitetes Attribut gekennzeichnet werden, d.h. der Wert des Attributs kann aus den Werten der anderen Attribute der Klasse bestimmt werden. Der Datentyp des Attributs ist vom Namen durch einen Doppelpunkt getrennt. Für jedes Attribut kann zudem dessen Multiplizität in eckigen Klammern angegeben werden, mit der die minimale und maximale Anzahl der Werte des Attributs festgelegt wird (siehe Tabelle 2.2). Ist die Multiplizität eines Attributs nicht explizit angegeben, dann nimmt das Attribut exakt einen Wert an. Im unteren rechteckigen Abschnitt folgen schließlich die *Operationen* der Klasse. Im Anschluss an die Sichtbarkeit und den Namen der Operation sind deren Parameter in Klammern angegeben. Die Parameter werden vergleichbar zu den Attributen definiert, jedoch kann zusätzlich deren Typ angegeben werden. Parameter die den Typ 'in' aufweisen werden von der aufrufenden Instanz an die Operation übergeben, wohingegen Parameter vom Typ 'out' von der Operation an die aufrufende Instanz übergeben werden. Die beiden Verhaltensweisen können außerdem durch den Typ 'inout' kombiniert werden. Nach dem Doppelpunkt folgt schließlich der Datentyp des Rückgabewerts der Operation.

Sichtbarkeit	Kurzform	Bedeutung
public	+	Öffentlich sichtbar
package	~	Nur innerhalb des Pakets sichtbar
private	-	Nur innerhalb der Klasse sichtbar
protected	#	Nur innerhalb der Klasse und deren Unterklassen sichtbar

Tabelle 2.1: Sichtbarkeit von Attributen in UML-Klassendiagrammen

Multiplizität	Minimum	Maximum	Angabe
Exakt 1 (Standardwert)	1	1	1 .. 1
Von 0 bis 3	0	3	0 .. 3
Beliebig viele		*	*
2 oder mehr	2	*	2 .. *

Tabelle 2.2: Multiplizität in UML-Klassendiagrammen

Die Beziehungen zwischen den Klassen werden durch Linien unterschiedlichen Typs gekennzeichnet. In Abbildung 2.17 sind die wichtigsten Beziehungen in UML-Klassendiagrammen dargestellt. Die *Generalisierung* entspricht der Vererbung, bei der eine Unterklasse alle Eigenschaften einer Oberklasse erbt und diese durch weitere Eigenschaften erweitern kann. Die *Aggregation* stellt eine Teil-Ganzes-Beziehung zwischen zwei Klassen dar, bei der ein Ganzes aus mehreren Teilen besteht. Dabei wird die Multiplizität der beteiligten Klassen am jeweiligen Linienende angegeben (siehe Tabelle 2.2). Zusätzlich kann ein Attributname an den Linienenden angegeben werden, der einem entsprechenden Attribut der Klasse entspricht. Die *Komposition* ist eine strenge Form der Aggregation, bei der die Teile maximal einem Ganzen zugeordnet sein können und zudem ohne das Ganze nicht existieren können. Die allgemeinste Beziehung zwischen zwei Klassen ist die *Assoziation*. Die Beziehung kann dabei gerichtet sein, so dass die Assoziation nur von einer Quell-Klasse zu einer Ziel-Klasse bestimmt werden kann, jedoch nicht umgekehrt.

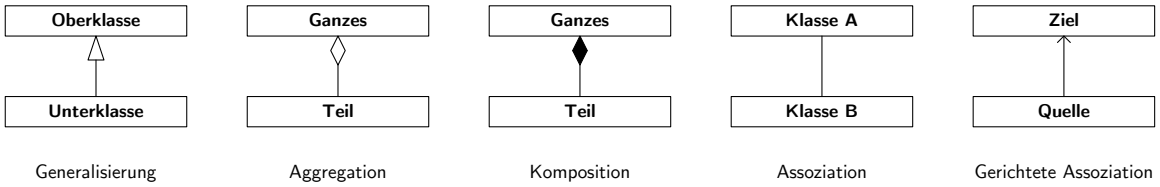


Abbildung 2.17: Beziehungen in UML-Klassendiagrammen

Beispiel

Im Beispiel erben die Klassen Punkt, Linie und Polygon die Eigenschaften der abstrakten Klasse Geometrie. Die Klasse *Geometrie* definiert einen ganzzahligen Identifikator und ein ganzzahliges Koordinatensystem[3] als Attribute. Diese sind nur innerhalb der Klasse und deren Unterklassen sichtbar. Die Klasse *Punkt* verfügt über zwei private Attribute x und y für die Speicherung der Koordinaten als Gleitkommazahlen. Zusätzlich bietet die Klasse die Operation distanz an, die als Eingabe einen weiteren Punkt erwartet. Der Rückgabewert dieser Operation ist eine Gleitkommazahl. Die Klasse *Linie* ist mit der Klasse Punkt über eine Aggregation verbunden, d.h. das Ganze (die Linie) besteht aus mehreren Teilen (die zugeordneten Stützpunkte). Eine Linie besteht dabei aus zwei oder mehr Punkten und ein einzelner Punkt kann beliebig vielen Linien zugeordnet sein. Die Punkte sind als Attribut punkte der Linie modelliert, das damit einer mehrelementigen Liste entspricht. Die Klasse Linie hat zudem ein zweites Attribut anzahlPunkte, das ein abgeleitetes Attribut ist, da dessen Wert aus der Anzahl der Element der Liste punkte bestimmt werden kann. Zusätzlich verfügt die Klasse Linie über eine öffentlich sichtbare Operation mit dem Namen laenge, die einen Gleitkommawert zurück gibt, sowie über eine Operation zum hinzufügen eines weiteren Punkts zur Linie. Die Klasse *Polygon* ist schließlich über eine gerichtete Assoziation mit der Klasse Linie verbunden, wobei die Linie als Attribut umring modelliert ist. Jedes Polygon ist mit genau einer Linie assoziiert und eine Linie kann nur einem einzigen Polygon zugeordnet sein. Die Klasse Polygon bietet schließlich eine Operation flaeche an, die einen Gleitkommawert zurückgibt.

2.3.3 Object Constraint Language

Die OCL ist eine formale Sprache um Ausdrücke in UML-Modellen beschreiben zu können (OMG, 2012). Meistens entsprechen diese Ausdrücke invarianten Bedingungen, die für das modellierte System eingehalten werden müssen. Mit den Ausdrücken können jedoch auch Abfragen für die Objekte eines Modells aufgestellt werden. Die OCL ist von der OMG und als ISO 19507 (2012) standardisiert.

In UML-Diagrammen können häufig nicht alle Aspekte einer Spezifikation so modelliert werden, dass diese eindeutig, vollständig und konsistent wiedergegeben werden. Deshalb werden UML-Diagramme oft durch natürlichsprachige Definitionen ergänzt, die jedoch nicht selten uneindeutig sind. Ein besserer Ansatz ist dagegen die Kombination von UML-Diagrammen mit OCL-Ausdrücken. In dieser Arbeit wird die aktuelle Version 2.3.1 der OCL verwendet, die in OMG (2012) definiert ist. Im Folgenden werden die zentralen Konzepte der OCL entsprechend den Ausführungen in OMG (2012) und Warmer und Kleppe (2004) vorgestellt.

Die OCL ist eine formale Sprache, die auf mathematischer Mengenlehre und Prädikatenlogik basiert. Im Gegensatz zu vielen anderen formalen Sprachen verwendet die Notation der OCL jedoch keine mathematischen Symbole, sondern stattdessen eindeutig definierte Schlüsselwörter. Dadurch ist sie einfach zu verstehen und anzuwenden. Zudem ist die OCL eine reine Spezifikationssprache. Erstens wird dadurch garantiert, dass die Evaluierung der Ausdrücke keinen Seiteneffekt haben und somit keine Veränderungen am Modell vorgenommen werden. Stattdessen ist das Ergebnis einer Evaluierung stets ein einfacher Wert, wie beispielsweise ein Wahrheitswert. Zweitens beschreiben die Ausdrücke in der OCL lediglich was getan werden muss, aber nicht wie. Deshalb sind alle Belange der Implementierung nicht durch die OCL abgedeckt, was die OCL zu einer plattformunabhängigen Sprache macht. Schließlich ist die OCL eine stark typisierte Sprache, d.h. dass jeder Ausdruck einen Typ aufweisen muss. Dadurch kann ein Ausdruck bereits bei der Modellierung überprüft werden, beispielsweise um durch einen Vergleich die inkompatiblen Typen Integer und String als Fehler identifizieren zu können.

[3] Das Koordinatensystem kann beispielsweise über einen eindeutigen numerischen Code, den sogenannten Spatial Reference System Identifier (kurz 'srid'), definiert werden. Die Zuordnung der Identifikatoren zu den Koordinatensystemen wird durch die European Petroleum Survey Group (EPSG) festgelegt. So entspricht beispielsweise der Code EPSG:4326 dem World Geodetic System (WGS) '84.

Spezifizierung von OCL-Ausdrücken

OCL-Ausdrücke können entweder als Teile eines UML-Diagramms oder in textueller Form angegeben werden. In UML-Diagrammen werden OCL-Ausdrücke als Kommentare angegeben, die den jeweiligen Elementen über eine gestrichelte Linie zugeordnet werden (siehe Abbildung 2.18). Alternativ können OCL-Ausdrücke jedoch auch als Text ohne grafische Repräsentation angegeben werden (siehe Programm 2.1). In dieser Arbeit wird nur die textuelle Repräsentation verwendet, da diese erstens mächtiger ist und zweitens entsprechend erweiterte UML-Diagramme bei vielen Ausdrücken schnell zu unübersichtlich werden.

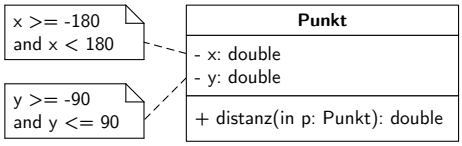


Abbildung 2.18: OCL-Bedingungen in einem UML-Klassendiagramm

```
context Punkt
  inv laenge: x >= -180 and x < 180
  inv breite: y >= -90 and y <= 90
```

Programm 2.1: OCL-Bedingungen als Text

Basistypen und Operationen

Die OCL verfügt über einige vordefinierte Basistypen und Operationen, die unabhängig vom verwendeten UML-Modell zur Verfügung gestellt werden. Die Basistypen der OCL sind im Klassendiagramm in Abbildung 2.19 dargestellt. Die auf diesen Basistypen definierten Operationen sind in Tabelle 2.3 aufgelistet.

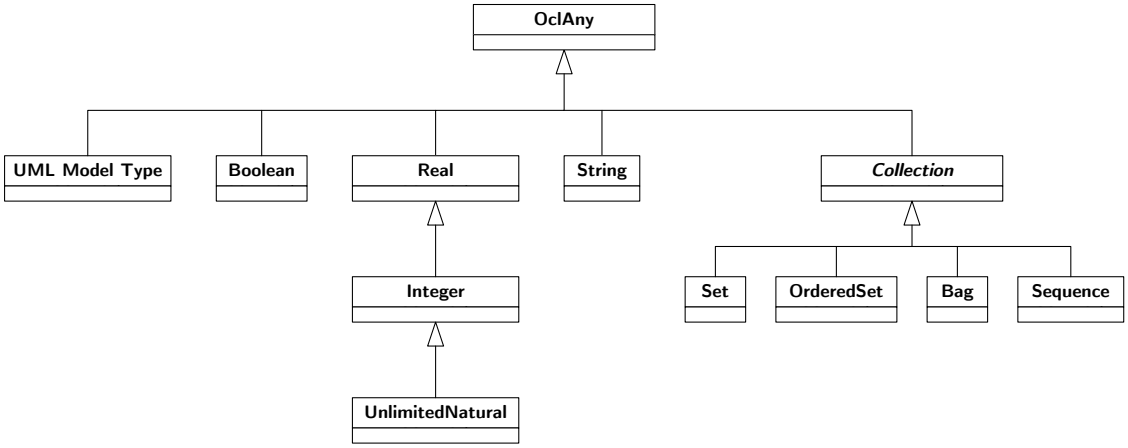


Abbildung 2.19: UML-Klassendiagramm der Basistypen der OCL

Der Typ beziehungsweise die Klasse *OclAny* ist die Oberklasse aller Klassen der OCL sowie aller Klassen in UML-Modellen. Demnach kann ein OCL-Ausdruck alle Operationen der Klasse OclAny auf jede Klasse eines UML-Modells anwenden, unabhängig davon welche konkreten Klassen in einem beliebigen UML-Modell verwendet werden. Die Operationen des Typs OclAny umfassen Tests auf Gleichheit von Objekten sowie Tests bezüglich der Vererbungshierarchie von Klassen. So überprüft die Operation oclIsTypeOf ob die Klasse einer bestimmten anderen Klasse entspricht. Dagegen prüft die Operation oclIsKindOf ob die Klasse einer bestimmten anderen Klasse entspricht oder deren Unterklasse ist.

Die primitiven Typen der OCL umfassen die Klassen *Boolean* für Wahrheitswerte, *Real* für Gleitkommazahlen, *Integer* für Ganzzahlen, *UnlimitedNatural* für Ganzzahlen inklusive dem Wert unendlich[4] sowie die Klasse *String* für Zeichenketten. Die Operationen der Klasse Boolean ermöglichen die logische Verknüpfung von Wahrheitswerten. Die Operationen der Klasse Real ermöglichen die üblicherweise benötigten Berechnungen wie Addition

[4] Der Wert unendlich der Klasse *UnlimitedNatural* wird benötigt um die Multiplizität '*' in UML-Diagrammen abbilden zu können. In OCL wird der Wert unendlich ebenso als '*' ausgedrückt.

und Multiplikation, Ableitungen auf Basis eines Werts sowie Vergleiche zwischen zwei Werten. Diese Operationen stehen aufgrund der Vererbung auch den Unterklassen Integer und UnlimitedNatural zur Verfügung. Die beiden Klassen für Ganzzahlen überschreiben einige der Operationen der Oberklasse Real mit spezifischen Operationen für ihren jeweiligen Datentyp. Die Operationen der Klasse String ermöglichen schließlich den Vergleich, die Konvertierung sowie den Zugriff auf Teile von Zeichenketten. Zudem können alle primitiven Datentypen in den Typ String umgewandelt werden sowie aus einem String erzeugt werden.

Neben den primitiven Typen definiert die OCL auch Auflistungen. Diese sind aus einer Menge von Elementen des gleichen Typs beziehungsweise derselben Klasse T (für engl. 'template') zusammengesetzt. Die abstrakte Klasse *Collection* bildet die Oberklasse aller Auflistungsklassen. Sie definiert Operationen zum Vergleich von Auflistungen, zum Zählen von Elementen, zur Überprüfung ob gewisse Elemente in einer Auflistung enthalten sind sowie zur Konvertierung von Auflistungen. Sofern der Typ der Elemente dies unterstützt, können auch die Elemente mit dem minimalen oder maximalen Wert bestimmt sowie die Elementwerte aufsummiert werden. Die OCL definiert vier konkrete Klassen für Auflistungen, die sich darin unterscheiden ob Elemente doppelt in einer Auflistung vorkommen können und ob die Elemente sortiert sind. Ein *Set* erlaubt keine Duplikate und weist keine Sortierung der Elemente auf. In einem *OrderedSet* sind die einzelnen Elemente dagegen sortiert, d.h. sie haben eine fest definierte Reihenfolge. Die Klasse *Bag* erlaubt Duplikate, jedoch sind die Elemente der Auflistung unsortiert. Die Klasse *Sequence* erlaubt schließlich Duplikate und weist eine Sortierung der Elemente auf. In Tabelle 2.3 sind lediglich die Operationen der Klasse Sequence aufgelistet, da diese Auflistungsklasse für diese Arbeit am wichtigsten ist. Die Operationen der Klasse Sequence ermöglichen den Zugriff auf Elemente basierend auf ihrem Index sowie das hinzufügen von Elementen am Anfang, am Ende und an einem beliebigem Index der Elementliste.

Für Auflistungen sieht die OCL weitere Operationen vor, die auf Iteratoren basieren. Dabei wird für jedes Element einer Auflistung überprüft, ob es eine bestimmte Bedingung erfüllt. In Tabelle 2.4 sind die wichtigsten Operationen mit Iteratoren i und Bedingungen b für die Klassen Collection und Sequence aufgelistet.

Zudem unterstützt die OCL die Erstellung von Auflistungen aus einem Attribut oder einer Operation einer Klasse mit dem collect Operator. Dabei werden die Werte aller Instanzen zu einer Auflistung hinzugefügt. Zum Beispiel werden im Programm 2.2 für alle Punkte einer Linie die x-Werte zu einer Auflistung hinzugefügt und von dieser Auflistung das Minimum und Maximum bestimmt. So kann gefordert werden, dass die Linie mindestens eine Länge von 10 m in Richtung der x-Achse aufweist.

```
context Linie
  inv minDifferenzX: punkte->collect(x)->max() - punkte->collect(x)->min() > 10 -- [Meter]
```

Programm 2.2: Erstellung einer Auflistung mit dem collect Operator der OCL

Weitere Grundlagen und Beispiel

Die für diese Arbeit benötigten weiteren Grundlagen der OCL werden im Folgenden anhand des in Unterabschnitt 2.3.2 eingeführten Beispiels der Modellierung von Geometrien vorgestellt. Die OCL-Ausdrücke im Programm 2.3 ergänzen dabei das UML-Klassendiagramm in Abbildung 2.16 und verbessern damit dessen Spezifikationsqualität.

Der Kontext (engl. 'context') eines OCL-Ausdrucks gibt die Entität im Modell an, für die der Ausdruck gilt. Dieser kann sich auf eine Klasse, eine Schnittstelle, einen Datentyp oder auf eine Komponente beziehen. Eine Invariante wird mit dem Schlüsselwort inv gekennzeichnet und ist ein logischer Ausdruck, der für alle Instanzen des Kontextes zu jeder Zeit wahr sein muss. Invarianten eigenen sich damit hervorragend zur Definition von Integritätsbedingungen. Invarianten können optional benannt werden, um diese einfacher referenzieren zu können. Ein OCL-Ausdruck kann dabei alle Elemente des UML-Diagramms verwenden, wie beispielsweise alle Attribute, Operationen und Beziehungen.

Die erste Invariante des Kontextes *Geometrie* fordert, dass für jede Instanz der Klasse Geometrie der Wert des Attributs id größer gleich Null ist. Der Ausdruck ist dabei äquivalent zum im Kommentar angegebenen Ausdruck, der das Schlüsselwort self verwendet, das den Kontext definiert. Ist der Kontext eindeutig, dann kann das Schlüsselwort self weggelassen werden. Die zweite Invariante bezieht sich statt auf eine einzelne Instanz des Kontextes auf alle Instanzen des Kontextes und fordert, dass das Attribut id eindeutig für alle Instanzen ist, d.h. dass kein Wert doppelt vorkommt. Die OCL-Operation allInstances liefert dabei alle Instanzen als Auflistung, auf der die auflistungsbasierte OCL-Operation isUnique ausgeführt wird. Dabei ist zu beachten, dass Operationen auf primitiven Datentypen durch einen vorangehenden Punkt (.) spezifiziert werden, wohingegen Operationen

Typ	Operationen auf dem Typ
OclAny	`= (o: OclAny): Boolean, <> (o: OclAny): Boolean, oclIsTypeOf(type: Classifier): Boolean, oclIsKindOf(type: Classifier): Boolean, oclType(): Classifier ...`
Boolean	`or(b: Boolean): Boolean, xor(b: Boolean): Boolean, and(b: Boolean): Boolean, not: Boolean, implies(b: Boolean): Boolean, toString(): String`
Real	`+ (r: Real): Real, - (r: Real): Real, * (r: Real): Real, / (r: Real): Real, -: Real, abs(): Real, floor(): Integer, round(): Integer, max(r: Real): Real, min(r: Real): Real, < (r: Real): Boolean, > (r: Real): Boolean, <= (r: Real): Boolean, >= (r: Real): Boolean, toString(): String`
Integer	`+ (i: Integer): Integer, - (i: Integer): Integer, * (i: Integer): Integer, / (i: Integer): Real, -: Integer, abs(): Integer, div(i: Integer): Integer, mod(i: Integer): Integer, max(i: Integer): Integer, min(i: Integer): Integer, toString(): String`
UnlimitedNatural	`+ (u: UnlimitedNatural): UnlimitedNatural, * (u: UnlimitedNatural): UnlimitedNatural, / (u: UnlimitedNatural): Real, div(u: UnlimitedNatural): UnlimitedNatural, mod(u: UnlimitedNatural): UnlimitedNatural, max(u: UnlimitedNatural): UnlimitedNatural, min(u: UnlimitedNatural): UnlimitedNatural, < (u: UnlimitedNatural): Boolean, > (u: UnlimitedNatural): Boolean, <= (u: UnlimitedNatural): Boolean, >= (u: UnlimitedNatural): Boolean, toInteger(): Integer, toString(): String`
String	`+ (s: String): String, size(): Integer, concat(s: String): String, substring(lower: Integer, upper: Integer): String, toBoolean(): Boolean, toReal(): Real, toInteger(): Integer, toUpperCase(): String, toLowerCase(): String, indexOf(s: String): Integer, equalsIgnoreCase(s: String): Boolean, at(i: Integer): String, characters(): Sequence(String), < (s: String): Boolean, > (s: String): Boolean, <= (s: String): Boolean, >= (s: String): Boolean`
Collection	`= (c: Collection(T)): Boolean, <> (c: Collection(T)): Boolean, size(): Integer, includes(o: T): Boolean, excludes(o: T): Boolean, count(o: T): Integer, includesAll(c2: Collection(T)): Boolean, excludesAll(c2: Collection(T)): Boolean, isEmpty(): Boolean, notEmpty(): Boolean, max(): T, min(): T, sum(): T, product(c2: Collection(T2)): Set(Tuple(first: T, second: T2)), asSet(): Set(T), asOrderedSet(): OrderedSet(T), asBag(): Bag(T), asSequence(): Sequence(T), flatten(): Collection(T2)`
Sequence	`count(o: T): Integer, = (s: Sequence(T)): Boolean, union (s: Sequence(T)): Sequence(T), flatten(): Sequence(T2), append (o: T): Sequence(T), prepend(o: T): Sequence(T), insertAt(index: Integer, o: T): Sequence(T), subSequence(lower: Integer, upper: Integer): Sequence(T), at(i: Integer): T, indexOf(o: T): Integer, first(): T, last(): T, including(o: T): Sequence(T), excluding(o: T): Sequence(T), reverse(): Sequence(T), sum(): T ...`

Tabelle 2.3: Basistypen und Operationen der OCL nach OMG (2012)

Typ	Operation	
Collection	`exists(i | b): Boolean`	Wahr, falls b für mindestens ein Element wahr ist
	`forAll(i | b): Boolean`	Wahr, falls b für alle Elemente wahr ist
	`isUnique(i | b): Boolean`	Wahr, falls b für jedes Element einen unterschiedlichen Wert aufweist
	`any(i | b): Collection(T)`	Gibt alle Elemente zurück, für die b wahr ist
	`one(i | b): Boolean`	Wahr, falls b für exakt ein Element wahr ist
Sequence	`select(i | b): Sequence(T)`	Gibt alle Elemente zurück, für die b wahr ist
	`reject(i | b): Sequence(T)`	Gibt alle Elemente zurück, für die b falsch ist

Tabelle 2.4: Iteratoren der OCL nach OMG (2012)

```
context Geometrie
  inv idPositiv: id >= 0 -- äquivalent zu self.id >= 0
  inv idEindeutig: Geometrie.allInstances()->isUnique(id) -- alternativ als Geometrie.allInstances
      ()->forAll(g1, g2 | g1 <> g2 implies g1.id <> g2.id)
  inv sridDE: srid = 25831 or srid = 25832 or srid = 25833 or srid = 4326

context Punkt
  inv laenge: srid = 4326 implies x >= -180 and x < 180
  inv breite: srid = 4326 implies y >= -90 and y <= 90

context Linie
  derive: anzahlPunkte = punkte->size()
  inv anzahlPunkte: anzahlPunkte >= 2
  inv sridEq: punkte->forAll(p | p.srid = self.srid)

context Polygon
  inv minDreieck: umring.punkte->size() >= 4
  inv umring: let startPunkt: Punkt = umring.punkte->first(), endPunkt: Punkt = umring.punkte->last
      () in startPunkt.x = endPunkt.x and startPunkt.y = endPunkt.y
  inv sridEq: umring.srid = self.srid
```

Programm 2.3: Integritätsbedingungen in OCL für das UML-Klassendiagramm

auf Auflistungen durch einen vorangehenden Pfeil (→) spezifiziert werden. Teilweise gibt es in der OCL auch mehrere Möglichkeiten um eine Invariante auszudrücken, was an der kommentierten alternativen Invariante ersichtlich ist. Wie die dritte Invariante der Klasse Geometrie aufzeigt, können mehrere OCL-Ausdrücke miteinander durch logische Operatoren verknüpft werden, da jeder Ausdruck selbst einen Wahrheitswert zurückliefert. Die Invariante gibt alle Werte an, die das Attribut srid für das Koordinatensystem annehmen darf[5].

Die beiden Invarianten des Kontextes *Punkt* fordern, dass die Werte für die beiden Koordinaten den Anforderungen an geographische Koordinaten genügen, jedoch nur falls das Koordinatensystem dem WGS '84 entspricht. Das Schlüsselwort beziehungsweise die Operation implies definiert dabei einen Filter, der vergleichbar zu dem auch in der OCL definierten if-then-else-endif Konstrukt ist.

Der erste OCL-Ausdruck des Kontextes *Linie* zeigt wie eine Ableitungsregel für ein abgeleitetes Attribut definiert wird. Die Anzahl der Punkte entspricht dabei der Größe der entsprechenden Liste, die durch die Aggregation gegeben ist. Die Operation size ist dabei eine von der OCL definierte Operation für Auflistungen. Zudem können die Multiplizitäten eines UML-Diagramms auch als Invarianten in der OCL definiert werden, wie die zweite Invariante zeigt.

Die erste Invariante des Kontextes *Polygon* fordert, dass ein Polygon aus mindestens drei Punkten bestehen muss. Die gerichtete Assoziation kann dabei über das Attribut der Klasse angesprochen werden. Die zweite Invariante fordert dass der Umring geschlossen ist, indem die Invariante aufgestellt wird dass der Start- und Endpunkt des Umrings übereinstimmen. Die Invariante verwendet dabei lokale Variablen, die über das Schlüsselwort let-in definiert werden. Die beiden Operationen first und last sind dabei wiederum von der OCL bereitgestellte Operationen für Auflistungen.

Analog lassen sich globale Variablen und Operationen über das Schlüsselwort def für einen Kontext definieren. So wird beispielsweise im Programm 2.4 eine globale Variable definiert, die dem ersten Punkt des Umrings eines Polygons entspricht. Dort ist auch eine globale Operation definiert, die überprüft, ob das Polygon ein n-Eck ist. Die Operation verwendet dazu einen eigenen Parameter, der unabhängig von den Attributen der Klasse ist.

```
context Polygon
  def: startPunktUmring: Punkt = umring.punkte->first()
  def: istNEck(n: Integer): Boolean = umring.anzahlPunkte - 1 = n
```

Programm 2.4: Definition globaler Variablen und Operationen in der OCL

Nachdem im Programm 2.3 die Invarianten für die Attribute definiert sind, werden im Programm 2.5 sogenannte Vor- und Nachbedingungen (engl. 'pre-, post-conditions') für die Operationen des UML-Klassendiagramms

[5] Die EPSG-Codes (siehe Unterabschnitt 2.3.2) entsprechen den drei in Deutschland vorkommenden UTM-Zonen sowie dem WGS '84.

aus Abbildung 2.16 aufgestellt. Eine Vorbedingung muss erfüllt sein, wenn die Operation mit der Ausführung beginnt. Entsprechend muss eine Nachbedingung erfüllt sein, wenn die Operation die Ausführung beendet.

```
context Punkt::distanz(p: Punkt)
  pre: p.x <> self.x and p.y <> self.y
  post: result > 0

context Linie::laenge()
  pre: --
  post: result > 0

context Linie::hinzu(p : Punkt)
  pre: p.srid = self.srid
  post: punkte = punkte@pre->including(p)

context Polygon::flaeche()
  pre: --
  post: result > 0
```

Programm 2.5: Integritätsbedingungen in OCL für die Operationen des UML-Klassendiagramms

Die Vorbedingung des Kontextes *Punkt::distanz(p: Punkt)* fordert, dass eine Distanz nur dann berechnet werden kann, wenn der an die Operation übergebene Punkt unterschiedliche Koordinaten hat als die Instanz der Klasse Punkt, für die diese Operation aufgerufen wird. Für die Vorbedingung wird im Programm 2.5 explizit das Schlüsselwort self verwendet, um den Bezug zur Klasse deutlich zu machen. Die Nachbedingung fordert, dass das Ergebnis der Operation größer als Null ist. Die gleiche Nachbedingung wird auch für den Kontext *Linie::laenge()* und den Kontext *Polygon::flaeche()* gefordert.

Für den Kontext *Linie::hinzu(p : Punkt)* wird als Vorbedingung gefordert, dass das Koordinatensystem des Punkts gleich dem Koordinatensystem der Linie ist. Die Nachbedingung nimmt Bezug auf den Zustand des Attributs punkte vor Ausführung der Operation. Dieser Zustand wird in OCL mit dem Schlüsselwort @pre gekennzeichnet. Mit der OCL-Operation including wird eine Auflistung um ein weiteres Element ergänzt. Die Nachbedingung überprüft also, ob die Auflistung beim Beenden der Operation um das neue Element erweitert wurde.

2.4 Raumbezogene Datenmodellierung

Raumbezogene Daten können auf Basis mehrerer Standards modelliert werden. Ein wichtiger Standard ist das räumliche Schema der ISO 19107 (2003), das die Grundlage für weitere Standards wie beispielsweise die Simple Features des Open Geospatial Consortium (OGC) und die Geography Markup Language (GML) bildet. Diese bilden wiederum die Grundlage für die Speicherung räumlicher Objekte in relationalen räumlichen Datenbanken sowie für den interoperablen Austausch räumlicher Daten.

Im Folgenden wird in Unterabschnitt 2.4.1 zuerst der Simple Feature Standard der OGC zur Modellierung von Geometrien vorgestellt. Anschließend werden in Unterabschnitt 2.4.2 kurz das geometrische und topologische Modell des ISO 19107 (2003) Standards vorgestellt. Vorrangig wird in dieser Arbeit der Simple Feature Standard verwendet, deshalb wird dieser im Folgenden entsprechend ausführlicher ausgeführt.

2.4.1 Simple Feature Standard des OGC

Der Simple Feature Standard des OGC implementiert ein Profil des räumlichen Schemas des ISO 19107 (2003) Standards. In Abbildung 2.20 sind die Klassen des OGC Simple Feature (2011) Standards vereinfacht dargestellt. Die dazugehörigen Operationen sind in Tabelle 2.5 aufgelistet. Die Klassen sowie die Operationen auf diesen Klassen werden im Folgenden entsprechend ihrer Definition in OGC Simple Feature (2011) vorgestellt.

Das Geometriemodell unterstützt 0-, 1- und 2-dimensionale Geometrien, die in 2-, 3- und 4-dimensionalen Koordinatenräumen existieren können. Neben den zweidimensionalen Koordinaten x und y können Objekte auch z-Werte aufweisen, die üblicherweise für die Angabe einer Höhe verwendet werden, sowie m-Werte, die für die Speicherung von Messwerten verwendet werden können.

Die Oberklasse aller Geometrieklassen ist die abstrakte Klasse *Geometry*. Diese verfügt über drei Gruppen von Operationen, die in Tabelle 2.5 aufgelistet sind. Die erste Gruppe umfasst grundlegende Operationen zur

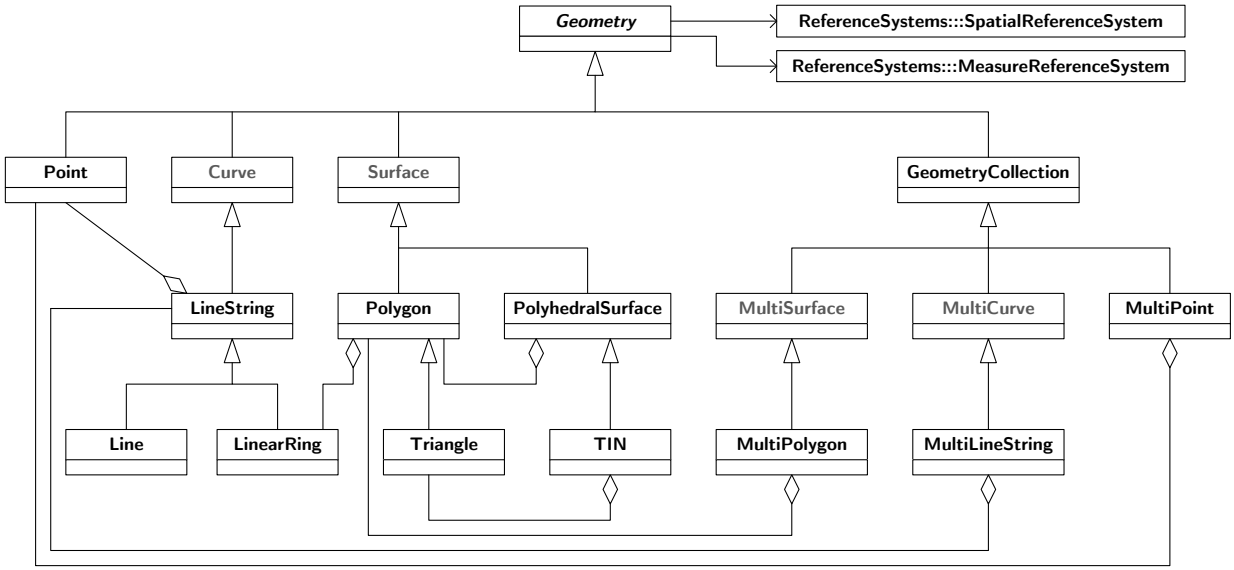


Abbildung 2.20: Geometriemodell nach OGC Simple Feature (2011)

Typ	Operationen auf dem Typ
Geometry	`dimension(): Integer, coordinateDimension(): Integer, spatialDimension(): Integer, geometryType(): String, SRID(): Integer, envelope(): Geometry, asText(): String, asBinary(): Binary, isEmpty(): Boolean, isSimple(): Boolean, is3D(): Boolean, isMeasured(): Boolean, boundary(): Geometry`
	`equals(o: Geometry): Boolean, disjoint(o: Geometry): Boolean, intersects(o: Geometry): Boolean, touches(o: Geometry): Boolean, crosses(o: Geometry): Boolean, within(o: Geometry): Boolean, contains(o: Geometry): Boolean, overlaps(o: Geometry): Boolean, relate(o: Geometry, matrix: String): Boolean, locateAlong(mValue: Double): Geometry, locateBetween(mStart: Double, mEnd: Double): Geometry`
	`distance(o: Geometry): Double, buffer(distance: Double): Geometry, convexHull(): Geometry, intersection(o: Geometry): Geometry, union(o: Geometry): Geometry, difference(o: Geometry): Geometry, symDifference(o: Geometry): Geometry`
Point	`X(): Double, Y(): Double, Z(): Double, M(): Double`
Curve	`length(): Double, startPoint(): Point, endPoint(): Point, isClosed(): Boolean, isRing(): Boolean`
LineString	`numPoints(): Integer, pointN(n: Integer): Point`
Surface	`area(): Double, centroid(): Point, pointOnSurface(): Point, boundary(): MultiCurve`
Polygon	`exteriorRing(): LineString, numInteriorRing(): Integer, interiorRingN(n: Integer): LineString`
PolyhedralSurface	`numPatches(): Integer, patchN(n: Integer): Polygon, boundingPolygons(p: Polygon): MultiPolygon, isClosed(): Boolean`
GeometryCollection	`numGeometries(): Integer, geometryN(n: Integer): Geometry`
MultiCurve	`isClosed(): Boolean, length(): Double`
MultiSurface	`area(): Double, centroid(): Point, pointOnSurface(): Point`

Tabelle 2.5: Geometrietypen und Operationen nach OGC Simple Feature (2011)

Abfrage des Geometrietyps, der Dimension, des räumlichen Referenzsystems und zur Konvertierung in eine textuelle oder binäre Repräsentation. Zudem können für jede Geometrie das minimale umschließende Rechteck über die Operation envelope sowie die umschließende Geometrie über die Operation boundary abgefragt werden. Die zweite Gruppe von Operationen umfasst die topologischen Operationen des DE-9IM, mit denen beispielsweise überprüft werden kann ob sich zwei Geometrien schneiden. Die dritte und letzte Gruppe umfasst Operationen zur räumlichen Analyse, wie die Bildung eines Puffers um eine Geometrie und die Bestimmung der konvexen Hülle einer Geometrie. Die Gruppe umfasst zudem auch Operationen, die sich auf jeweils zwei Geometrien beziehen. Zu diesen gehört die Berechnung der Distanz zweier Geometrien, deren Schnittmenge, deren Vereinigung sowie deren Differenz.

Die Klasse *Point* modelliert einen mindestens zweidimensionalen Punkt und verfügt über Operationen zum direkten Zugriff auf die Koordinatenwerte der x-, y-, z- und m-Achse. Die Klasse *LineString* repräsentiert eine zweidimensionale Kurve, die aus mehreren konsekutiven Stützpunkten besteht. Die Stützpunkte sind jeweils paarweise durch Liniensegmente miteinander verbunden, so dass zwischen den Stützpunkten linear interpoliert werden kann. Mit den Operationen der Klasse LineString kann die Gesamtlänge der Kurve sowie der Start- und Endpunkt bestimmt werden. Zudem kann mit der Operation isClosed überprüft werden ob die Kurve geschlossen ist, also der Start- und Endpunkt identisch sind. Die Operation isRing prüft zusätzlich zur Geschlossenheit, ob die einzelnen Liniensegmente sich nicht selbst überschneiden. Eine solche Geometrie wird auch als einfach bezeichnet. Mit dem Wissen über die Gesamtanzahl der Stützpunkte kann schließlich selektiv auf die einzelnen Stützpunkte des LineString zugegriffen werden. Die Klasse *Line* entspricht einem einzelnen Liniensegment, d.h. einem LineString mit exakt zwei Stützpunkten. Die Klasse *LinearRing* entspricht einem LineString, der sowohl geschlossen als auch einfach ist. Die Klasse *Surface* repräsentiert eine zweidimensionale ebene Fläche, die aus einem äußeren LinearRing und optional mehreren inneren Ringen besteht, d.h. die Fläche kann Löcher aufweisen. Die Fläche muss dabei nicht unbedingt in der x-y-Ebene liegen, solange sie durch eine Affintransformation in eine solche Fläche überführt werden kann. Die Klasse Surface bietet Operationen zur Bestimmung der Fläche, der Umringe, des Schwerpunkts sowie eines Punktes der garantiert innerhalb der Fläche liegt[6]. Die Klasse *Polygon* repräsentiert eine ebene Fläche, deren Ringe über entsprechende Operationen gezielt abgefragt werden können. Die Klasse *Triangle* als Unterklasse der Klasse Polygon definiert ein Dreieck bestehend aus exakt drei nicht kollinearen Punkten ohne innere Ringe. Die Klasse *PolyhedralSurface* besteht aus mehreren Polygonen, die als Patches bezeichnet werden, die zum Teil an ihren Rändern miteinander verbunden sind. Dadurch können dreidimensionale Körper modelliert werden. Auf die einzelnen Patches kann über entsprechende Operationen der Klasse PolyhedralSurface zugegriffen werden. Zudem kann mit der Operation isClosed überprüft werden, ob ein Objekt dieser Klasse einen dreidimensionalen Körper modelliert. Die Klasse *TIN* repräsentiert schließlich eine Dreiecksvermaschung, d.h. alle Polygone sind Objekte der Klasse Triangle.

Die Klasse *GeometryCollection* modelliert eine Auflistung von Geometrien, die alle das gleiche Referenzsystem aufweisen müssen. Ansonsten werden an die Elemente der Auflistung keine weiteren Anforderungen gestellt, so können beispielsweise Objekte der Klassen Point und LineString in einer gemeinsamen GeometryCollection enthalten sein. Die Operationen der Klasse GeometryCollection erlauben den wahlfreien Zugriff auf deren Elemente durch Angabe des gewünschten Index in der Auflistung. Die erste Unterklasse von GeometryCollection ist die Klasse *MultiPoint*, die nur Objekte der Klasse Point enthalten kann. Die Elemente der Klasse *MultiCurve* müssen Objekte der Klasse Curve oder deren Unterklassen sein. Die Operationen der Klasse MultiCurve ermöglichen die Prüfung, ob alle Geometrien der Auflistung geschlossen sind, sowie die Berechnung der aufsummierten Länge der enthaltenen Geometrien. Die Klasse MultiCurve selbst ist nicht instantiierbar, jedoch deren Unterklasse *MultiLineString*. Die Klasse *MultiSurface* bildet schließlich eine Auflistung von Elementen der Klasse Surface. Als Operationen werden die Berechnung der aufsummierten Fläche der enthaltenen Geometrien sowie die Bestimmung des Schwerpunkts und eines Punktes auf den Geometrien unterstützt.

2.4.2 Räumliches Schema der ISO 19107

Das geometrische Modell des ISO 19107 (2003) Standards ist in Abbildung 2.21 abgebildet[7]. Da der in Unterabschnitt 2.4.1 vorgestellte Simple Feature Standard der OGC auf diesem Modell basiert, werden im Folgenden lediglich einige Unterschiede zwischen diesen beiden Modellen vorgestellt. Die Klasse *GM_Complex* modelliert eine Auflistung von primitiven Geometrien, die sich jedoch nicht überlappen dürfen. Die Klasse *GM_Aggregate* modelliert auch eine Auflistung von primitiven Geometrien. Sie weist jedoch keinerlei Anforderungen an die in

[6] Der Schwerpunkt einer ebenen Fläche liegt nicht zwingend innerhalb der Fläche, beispielsweise liegt der Schwerpunkt bei einem Rechteck mit zentriertem rechteckigem Loch in eben diesem.
[7] Der Präfix GM_, der allen Klassennamen voransteht, ist in der Abbildung 2.21 aus Gründen der Leserlichkeit weggelassen.

der Auflistung enthaltenen Geometrien auf und ist damit mit der Klasse *GeometryCollection* des Simple Feature Standards gleichzusetzen. Zudem definiert der ISO 19107 (2003) Standard für die spezifizierten geometrischen Klassen entsprechend auch Operationen auf diesen Klassen.

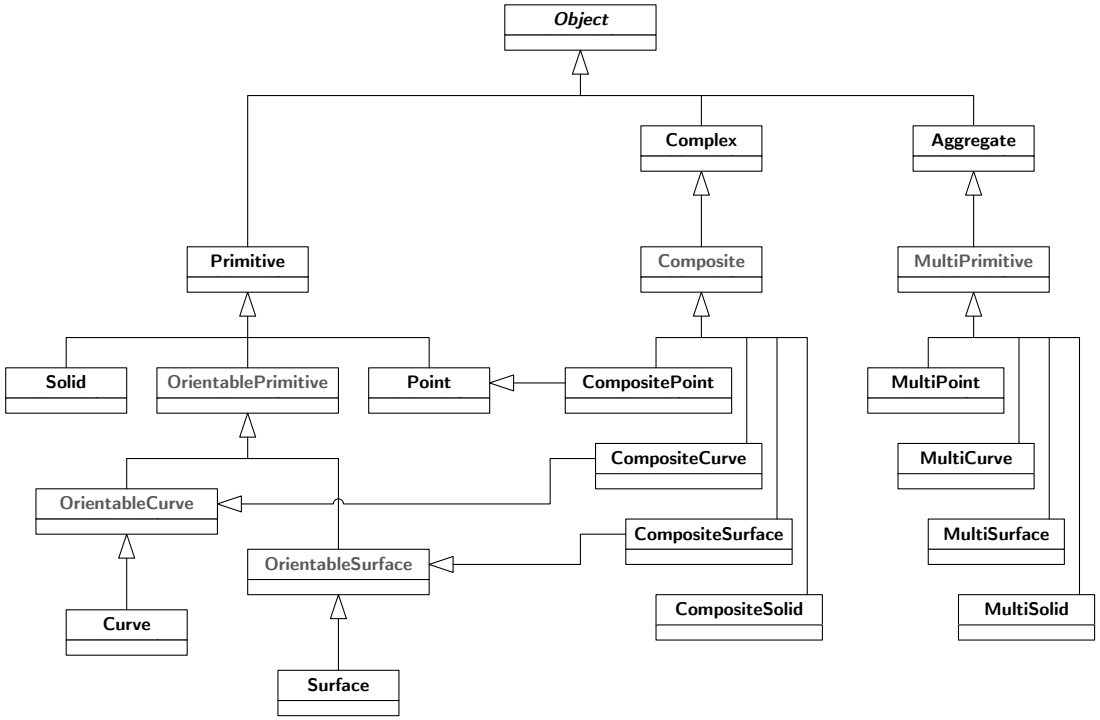


Abbildung 2.21: Geometriemodell nach ISO 19107 (2003)

Der ISO 19107 (2003) Standard definiert neben dem geometrischen Modell auch ein topologisches Modell. Dieses Modell besteht aus Knoten (engl. 'node'), Kanten (engl. 'edge'), Maschen (engl. 'face') und Raumelementen (engl. 'solid'). Diese Primitive können zudem zu komplexeren Elementen zusammengesetzt werden. Außerdem unterstützt das Topologiemodell gerichtete Knoten, Kanten, Maschen und Raumelemente.

2.5 Statistik

Die *Statistik* befasst sich mit der Erhebung, Aufbereitung und Analyse von Daten sowie der Interpretation der daraus resultierenden Ergebnisse (Bourier, 2011). Die Statistik lässt sich in vier Teilgebiete untergliedern, deren Definitionen nach Rinne (2003) und Bourier (2011) in der folgenden Auflistung zusammengefasst werden:

- *Deskriptive Statistik*: Die deskriptive Statistik (siehe Unterabschnitt 2.5.1) beschreibt Daten durch numerische Maße und grafische Abbildungen. Die Daten werden dabei vollständig erhoben, d.h. alle relevanten Objekte werden erfasst.
- *Explorative Datenanalyse*: Die explorative Datenanalyse (siehe Unterabschnitt 2.5.3) untersucht Daten um deren Beschreibung zu vereinfachen und um bisher verborgene Aspekte der Daten offenzulegen. Die explorative Datenanalyse kann als Bindeglied zwischen deskriptiver und inferentieller Statistik verstanden werden. Sie verwendet beschreibende Methoden um Daten zu analysieren und liefert Indizien die mit Hilfe der inferentiellen Statistik überprüft werden können.
- *Wahrscheinlichkeitsrechnung*: Die Wahrscheinlichkeitsrechnung (siehe Unterabschnitt 2.5.2) bildet die Grundlage für stochastische Modelle, die im Gegensatz zu deterministischen Modellen zufällige Einflüsse enthalten. Die Methoden der Wahrscheinlichkeitsrechnung sind eine Grundvoraussetzung für die inferentielle Statistik.
- *Inferentielle Statistik*: Mit der inferentiellen Statistik[8] wird anhand von vorliegenden Daten auf diejenigen Ursachenkomplexe geschlossen, die diese Daten erstellt haben könnten. Die Daten werden dabei

[8] Die inferentielle Statistik wird auch als induktive, schließende oder Inferenz-Statistik bezeichnet.

nicht vollständig erhoben, sondern es wird aus einer Grundgesamtheit von N Objekten mit Hilfe eines Auswahlverfahrens eine Stichprobe von $n < N$ Objekten gezogen (siehe Abbildung 2.22). Anhand der Daten der Stichprobe wird dann auf die Grundgesamtheit geschlossen. Die inferentielle Statistik umfasst das Testen von konkurrierenden Erklärungshypothesen für die Daten (siehe Unterabschnitt 2.5.4) und das bestmögliche Schätzen von Modellparametern aus den Daten (siehe Abschnitt 2.6).

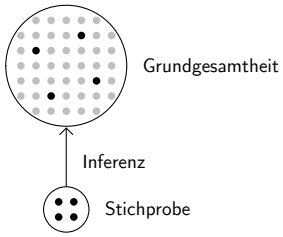


Abbildung 2.22: Stichprobe, Inferenz und Grundgesamtheit

Beispiel

Die in diesem Abschnitt vorgestellten statistischen Kennzahlen sowie Abbildungen werden, soweit sinnvoll, anhand des in Tabelle 2.6 aufgelisteten Beispiels am Ende jedes Unterabschnitts veranschaulicht. In der Tabelle sind die (gerundeten) Flächen und Bevölkerung der 16 Bundesländer Deutschlands mit Stand Ende 2011 angegeben. Die Daten stammen vom statistischen Bundesamt der Bundesrepublik Deutschland (DESTATIS, 2012).

Bundesland	Fläche $[\mathrm{km}^2]$	Bevölkerung
Baden-Württemberg	35 751	10 786 227
Bayern	70 550	12 595 891
Berlin	892	3 501 872
Brandenburg	29 484	2 495 635
Bremen	419	661 301
Hamburg	755	1 798 836
Hessen	21 115	6 092 126
Mecklenburg-Vorpommern	23 194	1 634 734
Niedersachsen	47 614	7 913 502
Nordrhein-Westfalen	34 098	17 841 956
Rheinland-Pfalz	19 854	3 999 117
Saarland	2569	1 013 352
Sachsen	18 420	4 137 051
Sachsen-Anhalt	20 450	2 313 280
Schleswig-Holstein	15 800	2 837 641
Thüringen	16 173	2 221 222

Tabelle 2.6: Bundesländer mit Fläche und Bevölkerung im Jahr 2011 nach DESTATIS (2012)

2.5.1 Deskriptive Statistik

Die zur Beschreibung der Daten eingesetzten numerischen Maße lassen sich in Lage-, Streuungs- und Zusammenhangsmaße untergliedern. Zur grafischen Abbildung können unter anderem Stabdiagramme und Histogramme verwendet werden. Die für diese Arbeit wichtigen Maße und Abbildungen der deskriptiven Statistik werden im Folgenden entsprechend ihrer Definition in Hartung u. a. (2002) vorgestellt.

Stabdiagramm und Histogramm

Die Häufigkeit von Attributwerten bildet die Grundlage für das Stabdiagramm und das Histogramm. Sei x_i der Wert des Attributs X des i-ten Objekts aus einer Menge von n Objekten. Sei ferner l die Anzahl der voneinander verschiedenen Werte beziehungsweise Ausprägungen $a_1, \ldots, a_l$ aller Werte von X. Dann bezeichnet die *absolute Häufigkeit* H_j die Anzahl der Objekte deren Wert x_i gleich a_j ist. Die *relative Häufigkeit* h_j normiert die absolute Häufigkeit auf das Intervall $[0, 1]$. Um die große Anzahl von Werten numerischer Attribute sinnvoll zu gruppieren, werden diese p überlappungsfreien Klassen zugeordnet, die jeweils ein Teilintervall des gesamten Wertebereichs abdecken. Die Gesamtanzahl H_i der Objekte der Klasse K_i wird dann als *absolute Klassenhäufigkeit* bezeichnet. Die *relative Klassenhäufigkeit* normiert wiederum die absolute Klassenhäufigkeit auf das Intervall $[0, 1]$.

Die *Klasseneinteilung* (engl. 'binning') des Wertebereichs in einzelne *Bereiche* (engl. 'bin', 'bucket') kann nach unterschiedlichen Kriterien erfolgen. Der Wertebereich kann in gleichgroße Intervalle (engl. 'equal width') unterteilt werden, wobei festgelegt werden muss ob die Intervalle linksoffen $(a, b]$ oder rechtsoffen $[a, b)$ sein sollen. Die Bereiche können auch so gebildet werden, dass diese jeweils eine feste (maximale) Anzahl von Werten enthalten (engl. 'equal frequency'). Alternativ kann die Klasseneinteilung manuell oder anhand beliebiger anderer Kriterien erfolgen.

Beim Stabdiagramm und Histogramm werden auf der Ordinate alle Werte a_j beziehungsweise alle Klassen K_i abgetragen (siehe Abbildung 2.23). Die absolute oder relative Häufigkeit wird entsprechend auf der Abszisse abgetragen, wobei diese beim Stabdiagramm durch die Länge einer Linie und beim Histogramm durch die Fläche eines Rechtecks abgebildet wird. Das Stabdiagramm eignet sich vor allem für die Visualisierung diskreter Werte, während das Histogramm vor allem für die Visualisierung numerischer Werte geeignet ist.

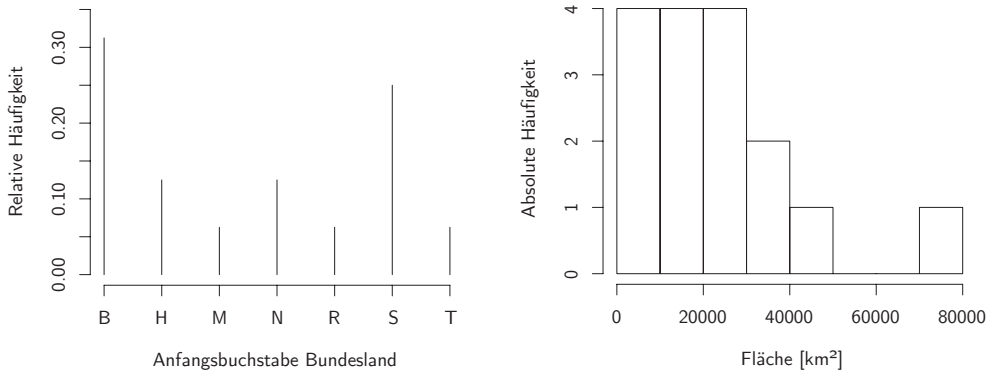


Abbildung 2.23: Stabdiagramm der Anfangsbuchstaben der Bundesländer und Histogramm der Fläche des Beispiels

Lagemaße

Lagemaße geben das 'Zentrum' der Werte eines Attributs an. Das *arithmetische Mittel* $\overline{x}$, das häufig als Mittelwert bezeichnet wird, lässt sich nach Gleichung 2.3 aus den einzelnen Attributwerten x_i der Objekte berechnen. Der *Median* $\tilde{x}_{0,5}$, der auch als Zentralwert bezeichnet wird, trennt die nach Größe sortierten Werte x_i derart, dass die Hälfte der Werte kleiner oder gleich dem Median ist und die andere Hälfte entsprechend größer oder gleich dem Median ist. Der Median kann nach Gleichung 2.4 bestimmt werden.

$$\overline{x} = \frac{1}{n} \cdot \sum_{i=1}^{n} x_i = \frac{1}{n} \cdot \sum_{j=1}^{l} (a_j \cdot H_j) \tag{2.3}$$

$$\tilde{x}_{0,5} = \begin{cases} x_{(n+1)/2}, & \text{falls } n \text{ ungerade} \\ \frac{1}{2} \cdot \left(x_{n/2} + x_{(n+2)/2}\right), & \text{falls } n \text{ gerade} \end{cases} \tag{2.4}$$

Eine Verallgemeinerung des Medians stellt das *α-Quantil* $\tilde{x}_\alpha$ dar, das die nach Größe sortierten Werte anhand eines Prozentsatzes α in zwei Mengen trennt. Die Berechnung des α-Quantils ist in Gleichung 2.5 angegeben. Einige spezielle Werte von α sind benannt und in Tabelle 2.7 aufgeführt.

$$\tilde{x}_\alpha = \begin{cases} x_k, & \text{falls } n \cdot \alpha \text{ keine ganze Zahl ist, wobei } k \text{ die auf } n \cdot \alpha \text{ folgende ganze Zahl ist} \\ \frac{1}{2} \cdot (x_k + x_{k+1}), & \text{falls } n \cdot \alpha \text{ eine ganze Zahl ist, wobei } k = n \cdot \alpha \text{ ist} \end{cases} \tag{2.5}$$

α	Name
0,5	Median, Q_2
0,25	Unteres Quartil, Q_1
0,75	Oberes Quartil, Q_3
$d \cdot 0{,}1$	d-tes Dezil
$p \cdot 0{,}01$	p-tes Perzentil

Tabelle 2.7: Spezielle α-Quantile

Der *Modus*, der auch als Modalwert bezeichnet wird, ist der häufigste Wert eines Attributs. Der Modus lässt sich demnach nicht nur für numerische Attribute, sondern auch für nominale Attribute bestimmen.

Streuungsmaße

Streuungsmaße beschreiben die Abweichung der Attributwerte vom Zentrum einer Häufigkeitsverteilung. Die *Spannweite* (engl. 'range') lässt sich aus der Differenz des größten zum kleinsten Attributwert, den sogenannten Extremwerten, berechnen. Liegen an den Rändern des Wertebereichs Ausreißer, so beeinflussen diese die Spannweite extrem. Ein robusteres Maß für die Streuung ist deshalb der *Quartilsabstand*, der auch als Interquartilsabstand (engl. 'interquartile range') bezeichnet wird. Dieser berechnet sich aus der Differenz des oberen zum unteren Quartil (siehe Tabelle 2.7). Die *Varianz* s^2 der Attributwerte lässt sich nach Gleichung 2.6 berechnen und bezieht sich auf den mittleren quadratischen Abstand der Attributwerte zum arithmetischen Mittel. Die Wurzel aus der Varianz ergibt die *Standardabweichung* s, welche die gleiche Einheit beziehungsweise Dimension besitzt wie die Attributwerte. Zum Vergleich der Streuung verschiedener Attribute oder Datensätze ist der *Variationskoeffizient* geeignet. Dieser drückt das Verhältnis der Streuung zum arithmetischen Mittel mit der Gleichung $v = s/\overline{x}$ aus. Der Variationskoeffizient ist jedoch nur dann ein sinnvolles Streuungsmaß, wenn alle Attributwerte positiv sind.

$$s^2 = \frac{1}{n-1} \cdot \sum_{i=1}^{n} (x_i - \overline{x})^2 = \frac{1}{n-1} \cdot \left(\sum_{i=1}^{n} x_i^2 - n \cdot \overline{x}^2 \right) \tag{2.6}$$

Zusammenhangsmaße

Zusammenhangsmaße beschreiben den gegenseitigen Zusammenhang der Werte von zwei oder mehr Attributen. Die Stärke der Abhängigkeit lässt sich numerisch durch einen Korrelationswert ausdrücken. Der funktionale Zusammenhang lässt sich durch eine Regressionsanalyse bestimmen (siehe Unterabschnitt 2.6.5). Im Folgenden werden die Zusammenhänge zwischen zwei Variablen untersucht. Die Verallgemeinerung auf mehrere Variablen ist beispielsweise in Hartung u. a. (2002) beschrieben.

Sei y_i der Wert des Attributs Y des i-ten Objekts aus einer Menge von n Objekten. Sei ferner m die Anzahl der voneinander verschiedenen Werte beziehungsweise Ausprägungen $b_1, \ldots, b_m$ aller Werte von Y. Dann bezeichnet die absolute Häufigkeit H_{jk} genau die Anzahl der Objekte deren Wert x_i gleich a_j und deren Wert y_i gleich b_k ist. Werden die absoluten Häufigkeiten für ein bestimmtes j beziehungsweise k aufsummiert, dann ergibt sich daraus die *absolute Randhäufigkeit* $H_{j.}$ beziehungsweise $H_{.k}$ nach Gleichung 2.7.

$$H_{j.} = \sum_{k=1}^{m} H_{jk}, \quad H_{.k} = \sum_{j=1}^{l} H_{jk} \tag{2.7}$$

Mit diesen Annahmen kann für zwei numerische Attribute X und Y der *Korrelationskoeffizient (nach Bravais-Pearson)* r_{XY} aus Gleichung 2.8 bestimmt werden. Die Werte des Korrelationskoeffizienten r_{XY} liegen im Intervall $[-1, 1]$, wobei der Wert 0 keiner Korrelation zwischen den Attributen X und Y entspricht. Die Werte 1 beziehungsweise -1 weisen auf eine positive beziehungsweise negative Korrelation, und damit auf einen starken *linearen Zusammenhang* der beiden Attribute, hin.

$$r_{XY} = \frac{\sum_{j=1}^{l}\sum_{k=1}^{m}\left((a_j - \overline{a}) \cdot (b_k - \overline{b}) \cdot H_{jk}\right)}{\sqrt{\left(\sum_{j=1}^{l}(a_j - \overline{a})^2 \cdot H_{j\cdot}\right) \cdot \left(\sum_{k=1}^{m}(b_j - \overline{b})^2 \cdot H_{\cdot k}\right)}} = \frac{\sum_{i=1}^{n}(x_i \cdot y_i) - n \cdot \overline{x} \cdot \overline{y}}{\sqrt{\left(\sum_{i=1}^{n} x_i^2 - n \cdot \overline{x}^2\right) \cdot \left(\sum_{i=1}^{n} y_i^2 - n \cdot \overline{y}^2\right)}} \tag{2.8}$$

Für zwei ordinale Attribute X und Y kann der *Spearmansche Rangkorrelationskoeffizient* r_S nach Gleichung 2.9 bestimmt werden. Die Gleichung entspricht der Gleichung des Korrelationskoeffizienten nach Bravais-Pearson mit dem Unterschied, dass anstatt der Attributwerte sogenannte Rangzahlen verwendet werden. Dazu werden die Werte x_i eines Attributs X anhand ihrer Größe aufsteigend angeordnet. Dadurch entsteht eine geordnete Reihe mit den Werten $x_{(1)}, \ldots, x_{(n)}$. Die *Rangzahl* $R(x_{(i)})$ beschreibt dann die Position, den sogenannten Rang, des Werts $x_{(i)}$ in der geordneten Reihe. Sind alle Werte voneinander verschieden, dann entspricht die Rangzahl $R(x_{(i)})$ dem Wert i. Die Rangzahl von Werten mit der gleichen Ausprägung a_j entspricht dem arithmetischen Mittel aller Ränge dieser Ausprägung.

$$r_S = \frac{\sum_{i=1}^{n}\left(R(x_i) \cdot R(y_i)\right) - n \cdot \overline{R(x)} \cdot \overline{R(y)}}{\sqrt{\left(\sum_{i=1}^{n} R(x_i)^2 - n \cdot \overline{R(x)}^2\right) \cdot \left(\sum_{i=1}^{n} R(y_i)^2 - n \cdot \overline{R(y)}^2\right)}} \tag{2.9}$$

Beispiel

Für das Beispiel der Flächen und Bevölkerung der 16 Bundesländer Deutschlands sind die Werte der Lage- und Streuungsmaße in Tabelle 2.8 angegeben. Diese spiegeln die ungleiche Verteilung der Attributwerte des Beispiels (siehe Tabelle 2.6) wider. Die Fläche ist minimal für die Stadtstaaten und das Saarland und mit deutlichem Abstand maximal für Bayern. Die Bevölkerung ist ebenso ungleich verteilt und mit deutlichem Abstand maximal für Baden-Württemberg, Bayern und Nordrhein-Westfalen.

Die ungleiche Verteilung führt zu einer großen Spannweite und zu einer großen Standardabweichung, die auch durch einen hohen Variationskoeffizient ausdrückt wird. Das Beispiel belegt zudem deutlich die Vorteile der beiden robusten Maße Median und Quartilsabstand. Dadurch, dass nur die acht Attributwerte berücksichtigt werden, die in der der geordneten Reihe in der Mitte liegen, haben die vier minimalen und vier maximalen Werte keinen Einfluss auf diese beiden Maße. Dies führt entsprechend zu einem signifikanten Abstand des Medians zum arithmetischen Mittel. Der Korrelationskoeffizient zwischen Fläche und Bevölkerung beträgt 0,71, d.h. die Attribute haben einen deutlichen positiven linearen Zusammenhang. Obwohl die genauen Größen der Werte in Tabelle 2.8 wichtig sind, ermöglichen sie dennoch nur eine bedingt übersichtliche Beschreibung der Daten. Die in Unterabschnitt 2.5.3 vorgestellte explorative Datenanalyse bietet Ansätze zur einfacheren Darstellung der Daten.

	Fläche $\left[\text{km}^2\right]$	Bevölkerung
Arithmetisches Mittel $\overline{x}$	22 321	5 115 234
Median $\tilde{x}_{0,5}$	20 152	3 169 756
Spannweite	70 131	17 180 655
Unteres Quartil Q_1	9185	2 010 029
Oberes Quartil Q_3	31 791	7 002 814
Quartilsabstand	22 607	4 992 785
Standardabweichung s	18 686	4 842 391
Variationskoeffizient $[\cdot 100]$	84	95

Tabelle 2.8: Maße der deskriptiven Statistik für die Attribute des Beispiels

2.5.2 Wahrscheinlichkeitsrechnung

Die Methoden der Wahrscheinlichkeitsrechnung sind eine Grundvoraussetzung für die inferentielle Statistik (siehe Unterabschnitt 2.5.4) und das Data Mining (siehe Abschnitt 2.6). Im Folgenden werden die für diese Arbeit wichtigen Begriffe entsprechend ihrer Definition in Hartung u. a. (2002) und Rinne (2003) vorgestellt.

Grundlagen der Wahrscheinlichkeitsrechnung

Ein *Zufallsexperiment* ist ein nach einer bestimmten Vorschrift beliebig oft wiederholbarer Vorgang, dessen jeweilige Ergebnisse vom Zufall abhängen. Der Ausgang eines Zufallsexperiments lässt sich also nicht mit Sicherheit im voraus bestimmen. Ein *Elementarereignis* ω bezeichnet ein mögliches Ergebnis eines Zufallsexperiments. Die Menge aller möglichen Ergebnisse bildet entsprechend den *Ereignisraum* Ω. Zufällige *Ereignisse* sind Teilmengen von Ω und werden mit einem großen lateinischen Buchstaben $(A, B, \dots)$ gekennzeichnet. Für diese sind Mengenoperationen wie Vereinigung, Durchschnitt, logische Differenz und Komplement definiert.

Obwohl sich bei einem Zufallsexperiment nicht voraussagen lässt welches Ergebnis eintritt, sind doch manche Ergebnisse wahrscheinlicher als andere. Die *Wahrscheinlichkeit* eines Ereignisses A lässt sich auf mehrere Arten quantifizieren und wird als $P(A)$ angegeben[9]. Die *subjektive Wahrscheinlichkeit* beruht auf der Einschätzung von Experten. Die *objektive Prior-Wahrscheinlichkeit*, die auch als Laplace-Wahrscheinlichkeit bezeichnet wird, lässt sich nach Gleichung 2.10 berechnen. Die *objektive Posterior-Wahrscheinlichkeit*, die auch als statistische Wahrscheinlichkeit bezeichnet wird, kann nach Gleichung 2.11 ermittelt und als relative Häufigkeit interpretiert werden.

$$P(A) = \frac{N(A)}{N} = \frac{\text{Anzahl der für } A \text{ günstigen Fälle}}{\text{Anzahl aller möglichen (und gleichmöglichen) Fälle}} \tag{2.10}$$

$$P(A) = \lim_{n\to\infty} \frac{n(A)}{n} = \frac{\text{Anzahl der beobachteten Versuchsausgänge mit dem Ergebnis } A}{\text{Anzahl der Versuchswiederholungen für großes } n} \tag{2.11}$$

Die *bedingte Wahrscheinlichkeit* $P(B|A)$ definiert die Wahrscheinlichkeit für das Ereignis B unter der Bedingung, dass das Ereignis A bereits eingetreten ist. Die bedingte Wahrscheinlichkeit lässt sich nach Gleichung 2.12 bestimmen. In diesem Zusammenhang werden zwei Ereignisse A und B als *stochastisch unabhängig* bezeichnet, wenn die Gleichung $P(A \cap B) = P(A) \cdot P(B)$ erfüllt ist[10].

$$P(B \mid A) = \frac{P(B \cap A)}{P(A)} \tag{2.12}$$

Zufallsvariablen und Verteilungen

Eine Funktion $X(\omega \in \Omega) = x$ heißt eindimensionale *Zufallsvariable*, wenn diese jedem Elementarereignis ω eindeutig einen Wert $x \in \mathbb{R}$ zuordnet und die Wahrscheinlichkeit $P(X(\omega) \leq x)$ definiert ist. Im Allgemeinen wird der Bezug zu ω weggelassen und damit die Zufallsvariable als $X = x$, beziehungsweise im endlichen oder abzählbaren Fall als $X = x_i$ angegeben.

Die Verteilung einer Zufallsvariable lässt sich durch ihre *Verteilungsfunktion* nach Gleichung 2.13 ausdrücken. Für diskrete Zufallsvariablen lässt sich die Verteilung zusätzlich durch die *Wahrscheinlichkeitsfunktion* ausdrücken. Für stetige Zufallsvariablen lässt sich die Verteilung zusätzlich durch die Dichtefunktion, kurz *Dichte*, ausdrücken. Wichtige Eigenschaften der Wahrscheinlichkeitsfunktion und der Dichte sind in Tabelle 2.9 zusammengefasst.

$$F(x) := P(X \leq x), \quad x \in \mathbb{R} \tag{2.13}$$

Lage-, Streuungs- und Formparameter

Die Verteilungsfunktion einer *parametrischen Verteilung* enthält Parameter, welche die Lage, Streuung und Form bestimmen. Die Parameter ähneln dabei den Lage- und Streuungsmaßen der deskriptiven Statistik.

[9] Die Definition der Wahrscheinlichkeit erfüllt die drei Kolmogoroffschen Axiome: $P(A) \geq 0$ für alle $A \subset \Omega$, $P(\Omega) = 1$, die Wahrscheinlichkeit paarweiser disjunkter Ereignisse ist gleich der Summe ihrer Einzelwahrscheinlichkeiten.

[10] Die Gleichung $P(A \cap B) = P(A) \cdot P(B)$ ist äquivalent zu den beiden Gleichungen $P(B|A) = P(B)$ und $P(A|B) = P(A)$.

	Wahrscheinlichkeitsfkt.	Dichte
Bezeichnung	$P(X = x)$	$f(x)$
Definitionsbereich	$-\infty < x < +\infty$	$-\infty < x < +\infty$
Wertebereich	$0 \leq P_i \leq 1$	$f(x) \geq 0$
Normierung	$\sum_i P_i = 1$	$\int_{-\infty}^{+\infty} f(x)\,dx = 1$
Beziehung zu $F(x)$	$= \sum_{x_i \leq x} P_i$	$= \int_{-\infty}^{x} f(u)\,du$

Tabelle 2.9: Wichtige Eigenschaften der Wahrscheinlichkeitsfunktion und Dichte

Der wichtigste Lageparameter ist der *Erwartungswert* $E(X)$, der auch Mittelwert genannt wird. Der Erwartungswert lässt sich sich nach Gleichung 2.14 bestimmen. Die *Varianz* σ^2, beziehungsweise die Standardabweichung σ, ist der wichtigste Streuungsparameter. Die Varianz einer Verteilung kann nach Gleichung 2.15 berechnet werden.

$$\mu := E(X) = \begin{cases} \sum_i x_i \cdot P_i, & \text{falls X diskret} \\ \int_{-\infty}^{+\infty} x\, f(x)\,dx, & \text{falls X stetig} \end{cases} \tag{2.14}$$

$$\sigma^2 := V(X) := E\left[(X-\mu)^2\right] = \begin{cases} \sum_i (x_i - \mu)^2 \cdot P_i, & \text{falls X diskret} \\ \int_{-\infty}^{+\infty} (x-\mu)^2\, f(x)\,dx, & \text{falls X stetig} \end{cases} \tag{2.15}$$

Weitere Parameter ergeben sich aus den Momenten beziehungsweise den zentralen Momenten der Verteilung. *Momente* sind Erwartungswerte von speziellen Funktionen $g(X)$, beispielsweise Potenzen in X. Dabei entspricht das erste Moment $\mu_1' = E(X^1)$ dem Erwartungswert. Die zentralen Momente μ_r lassen sich allgemein nach Gleichung 2.16 berechnen. Das zweite zentrale Moment μ_2 entspricht demnach der Varianz. Momentenverhältnisse der zentralen Momente beschreiben zusätzlich zu den Lage- und Streuungsparametern die Form einer Verteilung. Die *Schiefe* S einer Verteilung kann nach Gleichung 2.17 bestimmt werden. Die Wölbung K, die auch als Kurtosis bezeichnet wird, folgt nach Gleichung 2.18.

$$\mu_r := E\left[(X-\mu)^r\right] \tag{2.16}$$

$$S = \frac{\mu_3}{\mu_2^{3/2}} = \begin{cases} < 0 & \text{rechtssteile Verteilung} \\ = 0 & \text{symmetrische Verteilung} \\ > 0 & \text{linkssteile Verteilung} \end{cases} \tag{2.17}$$

$$K = \frac{\mu_4}{\mu_2^2} = \begin{cases} < 3 & \text{platykurtische/flachgipflige Verteilung} \\ = 3 & \text{mesokurtische/normalgipflige Verteilung} \\ > 3 & \text{leptokurtische/steilgipflige Verteilung} \end{cases} \tag{2.18}$$

Normalverteilung

Die *Normalverteilung* ist die wichtigste Verteilung in der Statistik. Sie ist eine symmetrische Verteilung und durch die beiden Parameter Erwartungswert und Standardabweichung eindeutig festgelegt. Ihre Dichte kann nach Gleichung 2.19 berechnet werden. Die Dichte weist eine glockenförmige Gestalt auf und hat Wendepunkte an den Stellen $\mu \pm \sigma$. Die Schiefe einer Normalverteilung ist $S = 0$ und die Wölbung $K = 3$.

$$f(x) = \frac{1}{\sqrt{2 \cdot \pi} \cdot \sigma} \cdot e^{-\frac{(x-\mu)^2}{2 \cdot \sigma^2}}, \quad x \in \mathbb{R}, \mu \in \mathbb{R}, \sigma \in \mathbb{R}^+ \tag{2.19}$$

Die Normalverteilung einer Zufallsvariable X lässt sich kurz als $X \sim N(\mu, \sigma)$ schreiben. Durch die Transformation in Gleichung 2.20 lässt sich jede Normalverteilung in eine *Standardnormalverteilung* $Y \sim N(0, 1)$ überführen. Die Zufallsvariable Y wird dann als standardnormalverteilt bezeichnet.

$$Y := \frac{X - \mu}{\sigma}, \text{ mit } E(Y) = 0 \text{ und } V(Y) = 1 \tag{2.20}$$

2.5.3 Explorative Datenanalyse

Der Begriff der explorativen Datenanalyse (EDA) (engl. 'exploratory data analysis') wurde durch Tukey (1977) geprägt. Das zentrale Motiv der EDA ist die *Gewinnung neuer Einblicke in die Daten.* Durch einfache Arithmetik sowie einfach zu erzeugende Abbildungen soll die EDA es ermöglichen, die Beschreibung der Daten zu vereinfachen und bisher verborgene Aspekte der Daten offenzulegen. Tukey (1977) vergleicht die EDA mit der Arbeit eines Detektivs, der gleichermaßen die entsprechenden Werkzeuge und das notwendige logische Denkvermögen für seine Analyse benötigt. Die EDA soll Indizien liefern, die dann mit Hilfe der inferentiellen Statistik überprüft werden können.

Die für diese Arbeit wichtigen Werkzeuge werden im Folgenden entsprechend ihrer Definition in Hartung u. a. (2002) vorgestellt. Diese werden ergänzt um deutsche Bezeichnungen nach Polasek (1994). Detaillierte Hintergründe zur EDA sind beispielsweise in Tukey (1977) und in Hoaglin u. a. (2000) erläutert.

Lage- und Streuungsmaße

Die Definition der Lage- und Streuungsmaße der EDA unterscheiden sich teilweise von den entsprechenden Definitionen der deskriptiven Statistik (siehe Unterabschnitt 2.5.1). Die Grundlage der Maße bildet die nach Größe aufsteigend geordnete Reihe $x_{(1)}, \ldots, x_{(n)}$ der Werte x_i eines Attributs X. Dann beschreibt der *Tiefe-Index* T die Entfernung, die sogenannte Tiefe, des Werts x_i von den Extremwerten des Wertebereichs des Attributs. Das Minimum $x_{(1)}$ und das Maximum $x_{(n)}$ der Attributwerte haben jeweils die Tiefe 1. Um Werte mit der gleichen Tiefe dennoch unterscheiden zu können, wird die Bezeichnung des kleineren Werts um ein L (für engl. 'lower') und die des größeren Werts um ein U (für engl. 'upper') erweitert. Die geordnete Reihe der Attributwerte beginnt demnach mit den Werten $x_{1.L}, x_{2.L}, x_{3.L}$ und endet mit den Werten $x_{3.U}, x_{2.U}, x_{1.U}$.

Der *Median* m wird in der EDA über die Tiefe $T(m) = (n + 1)/2$ bestimmt und kann nach Gleichung 2.21 ermittelt werden. Die Definition ist damit äquivalent zur entsprechenden Definition der deskriptiven Statistik.

$$m = \begin{cases} x_{(n+1)/2}, & \text{falls } n \text{ ungerade} \\ \frac{1}{2} \cdot \left(x_{n/2.L} + x_{(n+2)/2.U} \right), & \text{falls } n \text{ gerade} \end{cases} \tag{2.21}$$

Weitere Lagemaße der EDA entsprechen in etwa den α-Quantilen der deskriptiven Statistik. Die *hinges* h (dt. 'Angelpunkte'[11]) entsprechen in etwa den Quartilen und sind durch ihre Tiefe $T(h) = ([T(m)] + 1)/2$ bestimmt. Dabei wird durch die Gaußklammer $[z]$ die Zahl z zur nächstkleineren ganzen Zahl abgerundet. Der obere beziehungsweise untere hinge kann nach Gleichung 2.22 ermittelt werden[12]. Weitere Halbierungen der Tiefe ergeben die *eights* e mit $T(e) = ([T(h)] + 1)/2$, *sixteenths* d mit $T(d) = ([T(e)] + 1)/2$ usw. analog zu den hinges.

$$h_{L/U} = \begin{cases} x_{T(h).L/U}, & \text{falls } T(h) \text{ eine ganze Zahl ist} \\ \frac{1}{2} \cdot \left(x_{T(h)-0,5.L/U} + x_{T(h)+0,5.L/U} \right), & \text{sonst} \end{cases} \tag{2.22}$$

Eine wichtige Zusammenstellung der Lagemaße bildet das sogenannte *5-Zahlenmaß* (engl. '5-number summary'), welches sich aus den beiden Extremwerten, den beiden hinges sowie dem Median ergibt. Dieses lässt sich kompakt in einem Pentagramm (engl. 'letter-value display') darstellen (siehe Abbildung 2.24). Die Angabe eines h hinter einem Wert bedeutet dabei, dass dieser um 0,5 größer ist.

Als Maße für die Streuung werden in der EDA der *hinge-Abstand* (engl. *h-spread*), der *eights-Abstand* (*eights-spread*) usw. verwendet.

[11] Der Begriff Angelpunkte bezieht sich bildlich auf eine Tür- oder Fensterangel, an deren Gelenk eine Tür oder ein Fenster hängt. Auf die EDA übertragen entsprechen die Gelenke den hinges und die Türen oder Fenster jeweils einem Viertel der geordneten Attributwerte.

[12] Nach Hartung u. a. (2002) sind die hinges identisch mit dem unteren beziehungsweise oberen Quartil, falls $T(h)$ eine ganze Zahl ist und zusätzlich $(n - 1)$ oder $(n - 2)$ durch 4 teilbar ist. Ansonsten liegen die hinges etwas näher am Median als die Quartile.

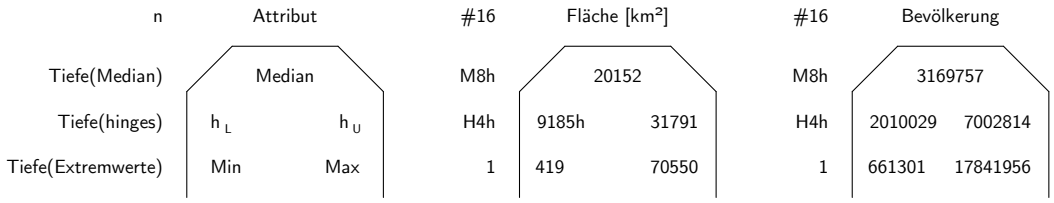


Abbildung 2.24: Pentagramm der Fläche und der Bevölkerung des Beispiels

Box-Plot

Ein *Box-Plot* (engl. 'box-and-whisker plot') stellt Lage- und Streuungsmaße von Attributwerten anschaulich und übersichtlich dar (siehe Abbildung 2.25). Das zentrale Element der Abbildung ist ein Rechteck vom unteren zum oberen hinge, die sogenannte Box. In der Box wird die Lage des Medians mit einem Querstrich gekennzeichnet. In der einfachen Variante des Box-Plots, die dem 5-Zahlenmaß entspricht, werden zusätzlich zwei Linien, sogenannte 'whisker', ausgehend von der Box bis zu den Extremwerten abgetragen. Die Extremwerte werden wiederum mit Querstrichen gekennzeichnet.

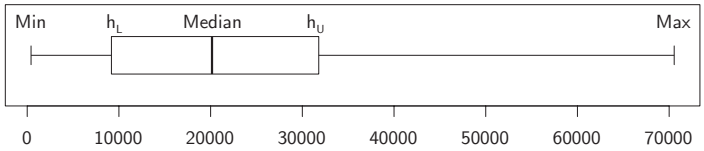


Abbildung 2.25: Einfacher Box-Plot der Fläche des Beispiels

In der erweiterten Variante des Box-Plots, die in Tukey (1977) auch als schematischer Plot (engl. 'schematic plot') bezeichnet wird, werden zusätzliche Maße zur einfacheren Identifikation von Ausreißern dargestellt (siehe Abbildung 2.26). Der dabei entscheidende Abstand ist der sogenannte *Schritt* (engl. 'step'), der dem anderthalbfachen Abstand der hinges entspricht. Durch diesen werden die inneren und äußeren Zäune (engl. 'inner, outer fence') definiert. Die *inneren Zäune* (abgekürzt als 'Inn. Zaun' in Abbildung 2.26) befinden sich innerhalb eines Schritts von den hinges und können nach Tabelle 2.10 ermittelt werden. Anstatt der durchgezogenen Linien zu den Extremwerten werden in der erweiterten Variante des Box-Plots gestrichelte Linien zu denjenigen Werten abgetragen, die sich gerade noch innerhalb des inneren Zauns befinden. Diese Werte werden wiederum mit Querstrichen gekennzeichnet. Die *äußeren Zäune* befinden sich innerhalb von zwei Schritten von den hinges und können nach Tabelle 2.10 bestimmt werden. Werte innerhalb des inneren Zauns werden als *Ausreißer* (auch Außenpunkte, engl. 'outside') bezeichnet. Werte außerhalb des äußeren Zauns werden entsprechend als *krasse Ausreißer* (auch Fernpunkte, engl. 'far out') bezeichnet.

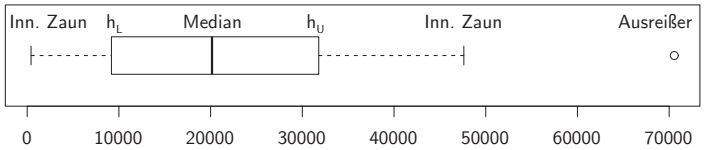


Abbildung 2.26: Erweiterter Box-Plot der Fläche des Beispiels

Datentransformationen

In der EDA werden *Datentransformationen* (engl. 're-expression') eingesetzt um eine bessere Übersicht über die Daten zu erhalten. Dabei wird eine monotone Funktion f auf alle Werte x_i eines Attributs X angewendet. Durch die Monotonie von f wird sichergestellt, dass die transformierten Werte einer geordneten Reihe wiederum geordnet sind. In der EDA werden Transformationen der Form x^p verwendet, welche die sogenannte Exponentenleiter (engl. 'ladder of power') bilden. Die nach Polasek (1994) wichtigsten Werte von p sind in Tabelle 2.11 zusammengefasst. Dabei ist zu beachten, dass alle Werte x_i positiv sein müssen, da ansonsten die Reihenfolge der Werte durch die Transformation nicht erhalten bleibt. Gegebenfalls ist dies durch die Addition einer Konstante zu gewährleisten.

	Untergrenze	Obergrenze
Innerer Zaun	$h_L - 1{,}5 \cdot$ *(h-spread)*	$h_U + 1{,}5 \cdot$ *(h-spread)*
Äußerer Zaun	$h_L - \ 3 \cdot$ *(h-spread)*	$h_U + \ 3 \cdot$ *(h-spread)*

Tabelle 2.10: Grenzen der inneren und äußeren Zäune eines Box-Plots

p	3	2	1	$\frac{1}{2}$	0	$-\frac{1}{2}$	-1	-2
Transformation	x^3	x^2	x	$\sqrt{x}$	$ln\, x$	$-\frac{1}{\sqrt{x}}$	$-\frac{1}{x}$	$-\frac{1}{x^2}$

Tabelle 2.11: Exponentenleiter für Datentransformationen nach Hartung u. a. (2002)

Quantile-Quantile-Plot

Mit einem *Quantile-Quantile-Plot* (Q-Q-Plot) kann überprüft werden, ob die Werte x_i eines Attributs X einer bestimmten Verteilung unterliegen. Dazu werden die nach Größe geordneten Werte $x_{(i)}$ als $i/(n+1)$-Quantile interpretiert. In einem zweidimensionalen Koordinatensystem werden dann Punkte der Form $(\xi_{i/(n+1)}, x_{(i)})$ abgetragen, wobei $\xi_{i/(n+1)}$ die entsprechenden theoretische Quantile der zu vergleichenden Verteilung repräsentieren (siehe Unterabschnitt 2.5.2). Liegen die Punkte annähernd auf einer Geraden, dann unterliegen die Attributwerte von X der vergleichenden Verteilung. Weichen nur die ersten oder letzten Punkte von der Gerade ab, dann können diese als Ausreißer interpretiert werden. In Abbildung 2.27 sind zwei unterschiedliche Attribute, dargestellt durch Punkte, gegen die Normalverteilung $N(\mu, \sigma)$ des jeweiligen Attributs, dargestellt als Linie, abgetragen.

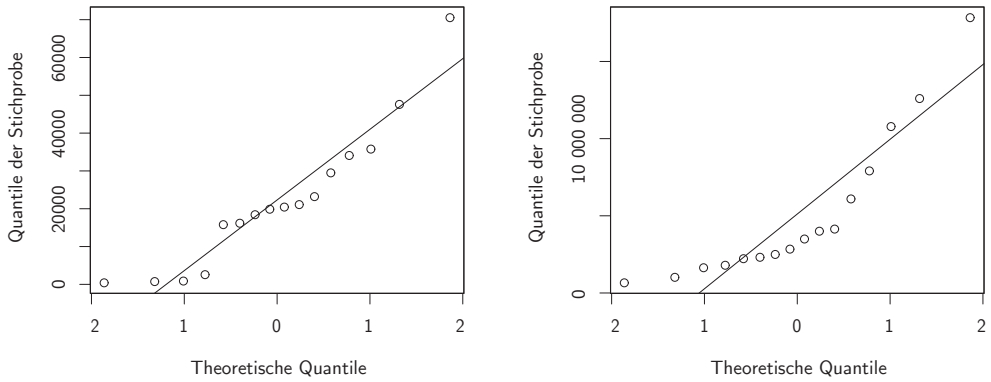


Abbildung 2.27: Quantile-Quantile-Plots der Fläche und der Bevölkerung des Beispiels

Scatter-Plot

Die Zusammenhänge zwischen mehreren quantitativen Attributen $X_1, \ldots, X_j$ von n Objekten können in einem *Scatter-Plot* übersichtlich dargestellt werden. Dabei werden für alle Objekte die Werte jedes der j Attribute gegen die Werte jedes anderen Attributs $k \neq j$ in einem zweidimensionalen Koordinatensystem abgetragen (siehe Abbildung 2.28). Die so entstehenden $j \cdot (j-1)$ Koordinatensysteme werden in einer quadratischen $j \times j$ Matrix angeordnet, deren Diagonale unbesetzt ist.

Beispiel

Die Kombination der Lage- und Streuungsmaße im Pentagramm (siehe Abbildung 2.24) ermöglicht einen einfacheren und strukturierteren Überblick über die Maße als eine tabellarische Darstellung (siehe Tabelle 2.8). Die grafische Abbildung der Lage- und Streuungsmaße in einem Box-Plot vereinfacht deren Interpretation nochmals. Aus dem Box-Plot der Fläche in Abbildung 2.26 lassen sich mehrere Schlüsse ziehen. Erstens ist der einzige Ausreißer, nämlich die Fläche des Bundeslands Bayern, deutlich gekennzeichnet und dessen Abweichung von den restlichen Werten direkt ersichtlich. Zweitens liegt der Median der Fläche der Bundesländer mittig innerhalb der hinges, was auf eine gleichmäßige Verteilung der Werte hindeutet. Der Box-Plot der Bevölkerung in Abbildung 2.29 zeigt ebenso deutlich den Ausreißer, nämlich die Bevölkerung von Nordrhein-Westfalen. Zudem liegt der Median deutlich näher am unteren hinge, was auf eine Konzentration der Daten im unteren Wertebereich hindeutet.

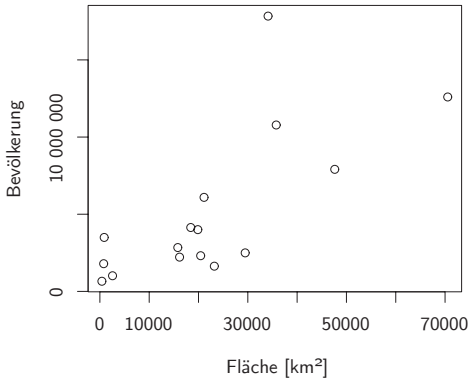


Abbildung 2.28: Scatter-Plot der Fläche und der Bevölkerung des Beispiels

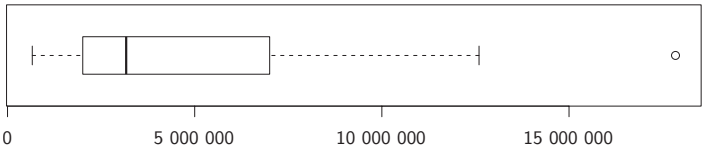


Abbildung 2.29: Erweiterter Box-Plot der Bevölkerung des Beispiels

Die beiden Quantile-Quantile-Plots in Abbildung 2.27 zeigen, dass abgesehen von wenigen Ausreißern an den Rändern des jeweiligen Wertebereichs, die Werte nahe den dargestellten Normalverteilungen liegen. Dadurch lässt sich die Hypothese aufstellen, dass sowohl die Fläche als auch die Bevölkerung der Bundesländer annähernd normalverteilt sind. Diese Hypothese kann durch die inferentielle Statistik im Folgenden Unterabschnitt statistisch fundiert überprüft werden.

Der Scatter-Plot in Abbildung 2.28 zeigt schließlich anschaulich den Zusammenhang der beiden Attribute Fläche und Bevölkerung. Als Teil der deskriptiven Statistik liegt bereits der Korrelationskoeffizient mit einem Wert von 0,71 vor, was als deutlicher positiver linearer Zusammenhang interpretiert werden kann. Im Scatter-Plot liegen entsprechend viele Punkte nahe einer Gerade, wobei wiederum deutlich die beiden Ausreißer in Fläche (Bayern) und Bevölkerung (Nordrhein-Westfalen) erkennbar sind.

2.5.4 Inferentielle Statistik

Mit der inferentiellen Statistik wird anhand einer Stichprobe auf diejenigen Ursachenkomplexe geschlossen, die diese Daten erstellt haben könnten. Die inferentielle Statistik lässt sich in zwei Bereiche untergliedern. Der erste Teil umfasst das Testen von konkurrierenden Erklärungshypothesen für die Daten und wird in diesem Abschnitt vorgestellt. Der zweite Teil behandelt das bestmögliche Schätzen von Modellparametern aus den Daten und wird aufgrund seines Umfangs in Abschnitt 2.6 vorgestellt.

Im Folgenden werden zuerst allgemein die Begriffe und Ziele von statistischen Tests anhand ihrer Definition in Rinne (2003) eingeführt. Diese können sich auf die Parameter, die Verteilung oder die Zusammenhänge von Zufallsvariablen beziehen. Anschließend wird der Test auf Normalverteilung vorgestellt, der zu den sogenannten Anpassungstests (engl. 'goodness of fit tests') zählt. Mit Anpassungstests kann überprüft werden, wie gut sich die Daten einer Stichprobe an eine theoretische Verteilung anpassen.

Statistische Tests

Statistische Tests basieren auf der Überprüfung von Hypothesen. Mit einer *statistischen Hypothese* wird eine Behauptung über die Eigenschaften einer oder mehrerer Zufallsvariablen aufgestellt. Die Behauptung kann sich dabei auf die Parameter, die Verteilung oder die Zusammenhänge von Zufallsvariablen beziehen. Um einen statistischen Test durchführen zu können müssen immer zwei Hypothesen aufgestellt werden. Die *Nullhypothese* H_0 ist die zu prüfende Hypothese und H_1 die dazu komplementäre *Alternativhypothese*, die behauptet dass die Nullhypothese nicht zutrifft. Mit einem *statistischen Test* kann dann für eine Zufallsstichprobe überprüft werden ob die Nullhypothese zutrifft oder verworfen werden muss.

Die Entscheidung eines statistischen Tests kann jedoch nie mit absoluter Sicherheit erfolgen, d.h. die Wahrscheinlichkeit für einen Fehler kann lediglich minimiert werden. Ein *Fehler erster Art* tritt auf, wenn die Null-

hypothese irrtümlich verworfen wird, obwohl diese zutrifft. Dagegen tritt ein *Fehler zweiter Art* auf, wenn die Nullhypothese irrtümlich angenommen wird, obwohl diese nicht zutrifft.

In einem *Signifikanztest* kann die Wahrscheinlichkeit für einen Fehler erster Art durch das *Signifikanzniveau* α vorab festgelegt werden. Beispielsweise bezeichnet ein Signifikanzniveau von $\alpha = 0{,}05$ eine fünfprozentige Wahrscheinlichkeit für einen Fehler erster Art. Die Durchführung eines solchen Signifikanztests besteht aus mehreren Schritten. Als Grundlage für den Test werden zuerst die Null- und die Alternativhypothese aufgestellt sowie das Signifikanzniveau festgelegt. Um eine Entscheidung über die Annahme der Nullhypothese treffen zu können wird anschließend eine *Prüfgröße* T, die auch als Testfunktion bezeichnet wird, bestimmt. Die Verteilung dieser Prüfgröße hängt von den zu testenden Hypothesen H_0 und H_1 ab und muss unter der Annahme der Gültigkeit der Nullhypothese bekannt sein. Anhand des Signifikanzniveaus und der Verteilung der Prüfgröße wird dann für diese der *Ablehnungsbereich* ermittelt. Dies ist derjenige Wertebereich der Prüfgröße, in dem die Nullhypothese abgelehnt wird. Der Ablehnungsbereich ist durch einen oder zwei sogenannte kritische Werte begrenzt, die diesen vom Annahmebereich abgrenzen. Schließlich wird für eine Zufallsstichprobe der tatsächliche Wert der Prüfgröße bestimmt. Ist dieser im Ablehnungsbereich enthalten, so wird die Nullhypothese H_0 auf dem Signifikanzniveau α verworfen und somit die Alternativhypothese H_1 angenommen. Ansonsten wird die Nullhypothese angenommen.

Test auf Normalverteilung

Für den Test von Stichproben auf eine Normalverteilung können nach Yazici und Yolacan (2007) etwa 40 verschiedene Anpassungstests verwendet werden. Bei diesen Tests sagt die Nullhypothese H_0 aus, dass die vorliegenden Werte $N(\mu, \sigma)$-verteilt sind. Dagegen umfasst die Alternativhypothese H_1 alle übrigen Verteilungen. In Yazici und Yolacan (2007) werden die verschiedenen Anpassungstests für unterschiedliche Stichprobengrößen und für unterschiedliche tatsächlich vorliegende Verteilungen[13] untersucht. Dabei konnte kein gleichmäßig bester Test ermittelt werden, d.h. alle Tests unterscheiden sich in ihrer Leistungsfähigkeit in Abhängigkeit vom Stichprobenumfang und der tatsächlich vorliegenden Verteilung. Falls keine Kenntnisse über die tatsächlich vorliegende Verteilung verfügbar sind und der Stichprobenumfang groß ist, wird in Yazici und Yolacan (2007) die Verwendung des Jarque-Bera-Tests empfohlen.

Der *Jarque-Bera-Test*, der in Jarque und Bera (1987) definiert wird, vergleicht die Schiefe S und Wölbung K der tatsächlich vorliegenden Verteilung der Stichprobe mit den entsprechenden theoretischen Standardmomenten einer Normalverteilung ($S = 0$ und $K = 3$, siehe Unterabschnitt 2.5.2). Die Berechnung der Schiefe und Wölbung sind in Gleichung 2.23 angegeben. Die Prüfgröße T lässt sich nach Gleichung 2.24 bestimmen und ist unter H_0 asymptotisch χ^2-verteilt mit zwei Freiheitsgraden. Die Nullhypothese H_0, nämlich das Vorliegen einer Normalverteilung, muss beim Jarque-Bera-Test demnach genau dann auf dem Signifikanzniveau α verworfen werden, wenn die Prüfgröße $T > \chi^2_{2;1-\alpha}$ ist.

$$S = \frac{\frac{1}{n} \cdot \sum_{i=1}^{n} (x_i - \overline{x})^3}{\left(\frac{1}{n} \cdot \sum_{i=1}^{n} (x_i - \overline{x})^2 \right)^{\frac{3}{2}}}, \quad K = \frac{\frac{1}{n} \cdot \sum_{i=1}^{n} (x_i - \overline{x})^4}{\left(\frac{1}{n} \cdot \sum_{i=1}^{n} (x_i - \overline{x})^2 \right)^{2}} \tag{2.23}$$

$$T = n \cdot \left(\frac{S^2}{6} + \frac{(K-3)^2}{24} \right) \sim \chi^2_2 \tag{2.24}$$

Beispiel

Für das Beispiel wurde anhand der Quantile-Quantile-Plots der EDA in Unterabschnitt 2.5.3 die Hypothese aufgestellt, dass die Werte beider Attribute annähernd normalverteilt sind. Diese Hypothese lässt sich nun mit einem statistischen Anpassungstest an die Normalverteilung überprüfen. Dazu werden für den Jarque-Bera-Test jeweils die Schiefe und Wölbung der Verteilung der Werte berechnet und mit diesen Werten die Prüfgrößen T bestimmt. Schließlich wird ermittelt, ob der Wert der Prüfgröße im Ablehnungsbereich liegt. Die Größen der beiden Jarque-Bera-Tests mit einem Signifikanzniveau von $\alpha = 0,05$ sind in Tabelle 2.12 aufgeführt. Die Nullhypothese H_0, mit der behauptet wird dass die Attributwerte einer Normalverteilung unterliegen, wird entsprechend für die Fläche auf dem Signifikanzniveau 0,05 angenommen, während sie für die Bevölkerung auf dem Signifikanzniveau 0,05 verworfen werden muss.

[13] Als alternative Verteilungen werden in Yazici und Yolacan (2007) die Beta-, Gamma-, Log-Normal-, Weibull- und t-Verteilung verwendet.

Attribut	Schiefe	Wölbung	Prüfgröße T	Kritischer Wert $\chi^2_{2;0,95}$	Testentscheidung
Fläche	0,99	3,90	3,17	5,99	H_0 annehmen
Bevölkerung	1,44	4,09	6,28	5,99	H_0 verwerfen

Tabelle 2.12: Größen der Jarque-Bera-Tests für die Attribute des Beispiels

2.6 Data Mining

Data Mining bezeichnet den Prozess der Entdeckung von interessanten Mustern und Wissen in umfangreichen Daten (Han u. a., 2011). Der Begriff wird synonym zum Prozess der Entdeckung von Wissen in Daten, dem sogenannten *Knowledge Discovery from Data* (KDD)-Prozess, verwendet. Streng genommen ist Data Mining nur ein Schritt im KDD-Prozess, wenn auch der entscheidende.

Der KDD-Prozess wird unterschiedlich in einzelne Schritte aufgeteilt, sowohl betreffend deren Anzahl als auch deren Definition. In dieser Arbeit wird die Definition nach Han u. a. (2011) verwendet, die im Folgenden vorgestellt wird. Anschließend werden kurz die Unterschiede zu zwei weiteren gebräuchlichen Definitionen aufgezeigt.

Der KDD-Prozess nach Han u. a. (2011) ist in sieben Schritte gegliedert, von denen die ersten vier Schritte unter dem Begriff Datenvorverarbeitung zusammengefasst werden können:

1. *Datenbereinigung*: Entfernen von Rauschen und inkonsistenten Daten
2. *Datenintegration*: Integration von Daten unterschiedlichen Ursprungs
3. *Datenreduktion*: Selektion relevanter Daten für die Analyse
4. *Datentransformation*: Transformation und Vereinigung von Daten
5. *Data Mining*: Extraktion von interessanten Mustern
6. *Auswertung der Muster*: Identifikation tatsächlich interessanter Muster anhand entsprechender Maße
7. *Wissensrepräsentation*: Visualisierung und Wissensrepräsentation für Benutzer

Eine in der Wissenschaft häufig verwendete Unterteilung des KDD-Prozesses[14] in neun Schritte wird in Fayyad u. a. (1996) vorgestellt. Diese unterscheidet sich nur gering von der Definition nach Han u. a. (2011). Dem Prozess wird als erster Schritt die Einarbeitung in die Aufgabenstellung und Ziele der Anwendung vorangestellt. Die Auswertung der Muster wird in Fayyad u. a. (1996) als Interpretation bezeichnet, die als ein Ergebnis die erneute Ausführung der bis dahin erfolgten Schritte erforderlich machen kann. Der letzte Schritt umfasst zusätzlich Entscheidungen, die auf Basis des gewonnenen Wissens erfolgen, sowie die Überprüfung auf Konflikte zu bereits bekanntem Wissen.

In der Industrie wird häufig das in Chapman u. a. (2000) definierte Prozessmodell CRISP-DM (CRoss-Industry Standard Process for Data Mining) verwendet[15]. CRISP-DM unterteilt den KDD-Prozess in sechs Schritte und deren zugeordnete Aktivitäten, Berichte und Daten. Die sieben Schritte in Han u. a. (2011) finden sich ähnlich im CRISP-DM wieder. Das Prozessmodell CRISP-DM ist jedoch auf den Einsatz in Unternehmen ausgerichtet, indem beispielsweise betriebswirtschaftliche Ziele berücksichtigt werden.

Im Folgenden werden die einzelnen Schritte des KDD-Prozesses detaillierter vorgestellt. Zuerst werden in Unterabschnitt 2.6.1 die vier Schritte der Datenvorverarbeitung einzeln diskutiert. Anschließend wird in Unterabschnitt 2.6.2 eine Gruppierung der Verfahren des Data Mining vorgestellt. Von den insgesamt fünf Gruppen werden zwei detaillierter vorgestellt, nämlich das Verfahren der Klassenbeschreibung (Unterabschnitt 2.6.3) und das Verfahren für die Klassifikation und Regression (Unterabschnitt 2.6.4 und 2.6.5). Durch die Verwendung räumlicher Daten ergeben sich einige Änderungen für das Data Mining, die in Unterabschnitt 2.6.6 diskutiert werden. Die Auswertung der Muster als weiterer Schritt im KDD-Prozess wird in Unterabschnitt 2.6.7 beschrieben. Der letzte Schritt der Wissensrepräsentation wird schließlich in Unterabschnitt 2.6.8 vorgestellt.

[14] In Fayyad u. a. (1996) wird KDD als Knowledge Discovery in Databases bezeichnet. Der Prozess lässt sich jedoch ohne Einschränkungen auf andere Datenrepräsentationen übertragen.

[15] CRISP-DM ist das Ergebnis eines von der Europäischen Union geförderten Projekts der vier Industriepartner NCR Systems Engineering Copenhagen, DaimlerChrysler AG, SPSS Inc. und OHRA Verzekeringen en Bank Groep B.V..

2.6.1 Datenvorverarbeitung

Die Datenvorverarbeitung fasst die vier Schritte Datenbereinigung, Datenintegration, Datenreduktion und Datentransformation zusammen. Zwischen den einzelnen Schritten treten Überlappungen auf, weshalb diese im Folgenden gemeinsam vorgestellt werden. Die aufbereiteten Daten können anschließend durch Verfahren des Data Mining analysiert werden.

Datenbereinigung

Die Datenbereinigung (engl. 'data cleaning', teilweise auch 'data cleansing') bildet den ersten Schritt der Datenvorverarbeitung. Unvollständige, verrauschte oder inkonsistente Daten werden korrigiert, da diese sonst entweder von den Algorithmen des Data Mining nicht verarbeitet werden können oder fehlerhafte Ergebnisse erzeugen.

Objekte eines Datensatzes sind dann *unvollständig*, wenn einem nicht optionalen Attribut kein Wert zugewiesen ist. In Han u. a. (2011) werden insgesamt sechs Ansätze vorgestellt um unvollständige Daten zu bereinigen. Der einfachste Ansatz besteht darin die entsprechenden Objekte zu ignorieren, indem die Daten im weiteren nicht berücksichtigt werden. Ein Benutzer kann die fehlenden Werte auch manuell eintragen, jedoch ist dieser Ansatz bei umfangreichen Datenmengen nur bedingt sinnvoll. Ein weiterer Ansatz besteht darin allen fehlenden Werten eine globale Konstante zuzuweisen, beispielsweise die Konstante 'Unbekannt'. Den fehlenden Werten kann auch der Mittelwert oder Median der Werte aller Objekte oder der Werte aller Objekte derselben Klasse zugeordnet werden. Schließlich kann der wahrscheinlichste Wert auch durch Data Mining bestimmt werden, beispielsweise durch Regression oder einen Entscheidungsbaum (siehe Unterabschnitt 2.6.4 und 2.6.5).

Rauschen ist ein zufälliger Fehler oder eine zufällige Varianz eines gemessenen Attributs (Han u. a., 2011). Um das Rauschen zu verringern können die Attributwerte geglättet werden (engl. 'smoothing'). Die Glättung kann lokal unter Berücksichtigung der Nachbarschaften oder global erfolgen.

Inkonsistenzen treten unter anderem auf, wenn die logische Konsistenz (siehe Abschnitt 2.2) verletzt ist. Die Werte eines Attributs müssen beispielsweise den korrekten Datentyp aufweisen um die konzeptuelle Konsistenz zu gewährleisten. Außerdem müssen Attributwerte ihren zugewiesenen Wertebereich konsistent berücksichtigen. Komplexere Inkonsistenzen lassen sich nach Han u. a. (2011) durch Data Mining aufdecken, indem Regeln für die Daten aufgestellt und überprüft werden. Die Korrektur von Inkonsistenzen kann anschließend iterativ durch entsprechende Transformationen der Daten erfolgen.

Datenintegration

Durch die Datenintegration werden Objekte mehrerer Datensätze zu einem gemeinsamen Datensatz zusammengeführt. Die Integration von räumlichen Datensätzen ist detailliert in Werder (2010) beschrieben.

Bei der Datenintegration müssen nach Han u. a. (2011) mehrere Aspekte berücksichtigt werden. Es muss sichergestellt werden dass sowohl die Attribute der Schemata der Datensätze als auch deren Objekte korrekt einander zugeordnet werden. Dabei müssen Konflikte zwischen Attributwerten berücksichtigt werden, die beispielsweise durch unterschiedliche Repräsentation, Skalierung, Kodierung oder Abstraktionsebenen entstehen können. Ebenso müssen Duplikate von Objekte vermieden werden. Schließlich können redundante Attribute teilweise durch Korrelationsanalysen aufgedeckt werden.

Datenreduktion

Der Datenumfang lässt sich durch mehrere Methoden reduzieren. Erstens kann die Anzahl der Objekte reduziert werden, indem Objekte aggregiert oder nicht berücksichtigt werden. Zweitens kann die Anzahl der Attribute reduziert werden, indem Attribute zusammengefasst oder nicht berücksichtigt werden. Drittens kann die Anzahl der möglichen Werte eines Attributs reduziert werden.

Die Reduktion der Daten für das Data Mining verfolgt mehrere Ziele. Durch die Reduktion der zu verarbeitenden Daten wird in der Regel die Berechnung beschleunigt. Manche Algorithmen des Data Mining sind bezüglich der verarbeitbaren Daten derart eingeschränkt oder so komplex, dass ihre Anwendung erst nach einer Datenreduktion möglich oder sinnvoll ist. Zudem sinkt häufig die Leistungsfähigkeit der verwendeten Algorithmen wenn irrelevante Attribute in die Berechnungen einbezogen werden. Nicht zuletzt ergeben sich durch die Berücksichtigung einer geringeren Anzahl an Attributen einfachere Modelle, die damit auch für den Benutzer verständlicher sind.

Trotz der genannten Vorteile hat die Datenreduktion den Nachteil, dass nicht alle Daten in das Data Mining einfließen und damit die Ergebnisse von einer Berechnung unter Berücksichtigung aller Daten abweichen können.

Die Vor- und Nachteile einer Reduktion der Daten sollten deshalb im Einzelfall sorgfältig abgewägt werden. Im Folgenden werden einige Methoden zur Datenreduktion entsprechend ihrer Definition in Han u. a. (2011) vorgestellt, wobei diese dort teilweise der Datentransformation zugeordnet werden.

Die *Anzahl der Objekte* kann durch die Aggregation mehrerer Objekte zu einem einzelnen Objekt reduziert werden. Dabei werden die Attributwerte der einzelnen Objekte summiert oder aggregiert. Beispielsweise können alle Städte eines Landkreises zu einem Objekt zusammengefasst werden, das dann Auskunft über die Gesamteinwohnerzahlen des Landkreises geben kann.

Durch die Ziehung von Stichproben aus dem Gesamtdatensatz kann ebenso die Anzahl der Objekte reduziert werden. Eine Zufallsstichprobe kann durch zufälliges Ziehen mit oder ohne Zurücklegen bereits gezogener Objekte erstellt werden. Sind die Daten bereits in p disjunkte Partitionen[16] aufgeteilt, beispielsweise in Partitionen mit jeweils einer festen Anzahl an Objekten, dann kann aus diesen eine Zufallsstichprobe von $n < p$ Partitionen gezogen werden. Ist dagegen eine Gruppierung der Objekte aufgrund ihrer Attributwerte in disjunkte Klassen, sogenannte Strata, möglich, dann kann durch eine stratifizierte Stichprobe die Verteilung der Objekte auf die einzelnen Klassen berücksichtigt werden. Dabei werden getrennt aus jeder Klasse Zufallsstichproben gezogen, wobei die Anzahl der zu ziehenden Objekte entsprechend der Klassenverteilung gewählt wird. Dadurch wird sichergestellt, dass auch Objekte weniger häufiger Klassen angemessen in der Stichprobe repräsentiert sind.

Die *Anzahl der Attribute* kann durch die Zusammenfassung von Attributen reduziert werden. Bei der Zusammenfassung wird aus den Werten mehrerer Attribute ein neues Attribut erzeugt, das die Informationen der Attribute effektiver abbildet. Attribute können durch mathematische Operationen, wie Addition oder Division, oder durch komplexere Formeln beziehungsweise Prozesse zusammengefasst werden. Beispielsweise können für Rechtecke die zwei Attribute Länge und Breite zu einem einzelnen Attribut, der Fläche des Rechtecks, zusammengefasst werden. Ein Beispiel für komplexere Prozesse zur Zusammenfassung von Attributen sind projektive Methoden, zu denen die Hauptkomponentenanalyse zählt. Diese bilden eine Anzahl von Attributen, die sogenannte Dimension, auf einen Raum mit weniger Dimensionen ab, der entsprechend durch weniger Attribute beschrieben werden kann.

Durch die Selektion relevanter Attribute kann ebenso die Anzahl der Attribute reduziert werden. Dabei werden die für eine bestimmte Aufgabe irrelevanten oder redundanten Attribute aus den Eingabedaten entfernt beziehungsweise deselektiert. Prinzipiell ergeben sich für n Attribute eines Objekts 2^n mögliche Kombinationen für die Bildung von Teilmengen dieser Attribute. Um nicht alle Teilmengen erschöpfend testen zu müssen, werden meist heuristische Methoden verwendet um die Selektion zu beschleunigen. Diese Methoden sind üblicherweise gierig (engl. 'greedy'), da sie in jedem Schritt jeweils das beste Attribut aus den verfügbaren Attributen auswählen. Durch die lokale Optimierung der Attributauswahl wird meist auch die global optimale Teilmenge ausgewählt. Als Beispiel für die Selektion der optimalen Teilmenge wird im Folgenden die Klassifizierung von Objekten anhand ihrer Attribute verwendet. Bei der schrittweisen Vorwärtsselektion (engl. 'stepwise forward selection') wird ausgehend von einer leeren Menge in jedem Schritt dasjenige Attribut hinzugefügt, welches die Klassifizierung optimal verbessert. Beispiele für Maße zur Attributselektion werden in Unterabschnitt 2.6.4 vorgestellt. Bei der schrittweisen Rückwärtseliminierung (engl. 'stepwise backward elimination') wird ausgehend von der Menge aller Attribute in jedem Schritt dasjenige Attribut eliminiert, welches die Klassifizierung am wenigsten verbessert. Die schrittweise Vorwärtsselektion und Rückwärtselimination können auch kombiniert eingesetzt werden. Alternativ können auch diejenigen Attribute selektiert werden, die in einem Entscheidungsbaum (siehe Unterabschnitt 2.6.4) zur Klassifizierung verwendet werden.

Die *Anzahl der möglichen Werte eines Attributs* kann für numerische Attribute durch die Diskretisierung der Werte reduziert werden. Die Diskretisierung kann durch eine Klasseneinteilung (siehe Unterabschnitt 2.5.1) erfolgen, wie sie beispielsweise bei der Erstellung eines Histogramms verwendet wird. Zusätzlich können iterativ die einzelnen Bereiche eines Histogramms zu größeren Bereichen zusammengefasst werden um die Anzahl der möglichen Werte hierarchisch in mehreren Stufen beziehungsweise Auflösungen zu verringern. Die Diskretisierung kann jedoch auch auf Verfahren des Data Mining basieren, beispielsweise auf den Teilungspunkten eines Entscheidungsbaums (siehe Unterabschnitt 2.6.4).

Für nominale Attribute kann die Anzahl der möglichen Werte eines Attributs durch die Erstellung von Konzepthierarchien reduziert werden. Durch diese werden weniger abstrakte Konzepte zu abstrakteren Konzepten zusammengefasst. Beispielsweise kann der administrative Bezug eines Gebäudes auf der Ebene von Gebäudeblöcken, Stadtteilen, Städten, Landkreisen oder Bundesländern definiert werden. Umso abstrakter das Konzept dabei ist, umso geringer ist die Anzahl der möglichen Attributwerte.

[16] In Han u. a. (2011) werden die Partitionen als Cluster bezeichnet.

Datentransformation

Durch die Datentransformation werden Daten für das Data Mining aufbereitet. Die Methoden umfassen Methoden zur Verringerung von Rauschen sowie zur Aggregation, Zusammenfassung von Attributen, Diskretisierung und Bildung von Konzepthierarchien (siehe Datenreduktion). Eine weitere Methoden zur Transformation der Daten ist die Normalisierung, die im Folgenden entsprechend der Definition in Han u. a. (2011) vorgestellt wird.

Der Einfluss von Maßeinheiten auf die Datenanalyse kann durch eine *Normalisierung* verringert werden. Durch diese wird der Wertebereich eines Attributs auf ein kleineres Intervall, meist $[-1, 1]$ oder $[0, 1]$, abgebildet. Die Normalisierung weist zudem jedem Attribut das gleiche Gewicht zu, was beispielsweise bei distanzbasierten Verfahren des Data Mining von Bedeutung ist. So hat ein Attribut mit einem großen Wertebereich, wie beispielsweise der Flächeninhalt von Gebäuden, den gleichen Einfluss wie ein Attribut mit einem kleinem Wertebereich, beispielsweise die Anzahl der Stockwerke.

2.6.2 Verfahren des Data Mining

Durch das Data Mining werden interessante Muster aus umfangreichen Daten extrahiert. Für die Extraktion der Muster stehen vielfältige Verfahren zur Auswahl, die nach Han u. a. (2011) in fünf Gruppen unterteilt werden können:

- *Klassen- / Konzeptbeschreibung*: Die einzelnen Objekte eines Datensatzes sind häufig einer geringen Anzahl an Klassen beziehungsweise Konzepten zugeordnet. Das Ziel des Data Mining ist dann eine zusammengefasste und zugleich präzise Beschreibung der Klassen. Diese Beschreibungen lassen sich prinzipiell durch die Charakterisierung der Klassen, durch die Unterscheidung der Klassen oder durch eine Kombination der beiden Methoden erstellen.
- *Klassifikation und Regression*: Das Ziel der Klassifikation ist die Aufstellung eines Modells, durch das die Klassen eines Datensatzes beschrieben und voneinander getrennt werden können. Das Modell wird auf Basis der Analyse eines Trainingsdatensatzes aufgestellt, für den die Klassen der Objekte bekannt sind. Mit dem so erstellten Modell können für Objekte, deren Klassen apriori unbekannt sind, die zugehörigen Klassen prädiziert werden. Die Klassifikation entsprechend dieser Definition wird auch als überwachtes Lernen bezeichnet, da die Klassen der Objekte a priori bekannt sind.

  Im Gegensatz zur Klassifikation, die auf die Prädiktion diskreter Größen beschränkt ist, können durch die Regressionsanalyse numerische Größen prädiziert werden, um beispielsweise Werte für unvollständige Attribute eines Objekts zu bestimmen.
- *Häufige Muster, Beziehungen und Korrelationen*: Häufige Muster (engl. 'frequent patterns') können sich auf unterschiedliche Elemente beziehen. Eine häufige Menge von Objekten (engl. 'frequent itemset') bezeichnet Objekte, die in einer Transaktion häufig zusammen auftreten. Ein Beispiel dafür sind Lebensmittel, die häufig zusammen eingekauft werden. Ein (häufiges) sequentielles Muster (engl. '(frequent) sequential pattern') bezeichnet Elemente, die häufig zeitlich aufeinanderfolgen. Ein (häufiges) strukturelles Muster (engl. '(frequent) structural pattern') bezeichnet häufig auftretende Strukturen, wie beispielsweise häufige Anordnungen von Elementen in einem Graphen. Die Analyse von häufigen Mustern hat das Ziel, interessante Beziehungen (engl. 'association') und Korrelationen in Datensätzen zu entdecken.
- *Analyse von Clustern*: Ein Cluster bezeichnet Objekte, die aufgrund ihrer Eigenschaften eine Gruppe bilden. Durch das Clustering werden Objekte so gruppiert, dass die Ähnlichkeit der Objekte innerhalb einer Gruppe maximiert und die Ähnlichkeit zwischen Objekten unterschiedlicher Gruppen minimiert wird. Die Analyse von Clustern entsprechend dieser Definition wird auch als unüberwachtes Lernen bezeichnet, da die Cluster beziehungsweise Klassen der Objekte a priori unbekannt sind.
- *Ausreißeranalyse*: Üblicherweise werden Ausreißer durch die Datenvorverarbeitung vor dem Data Mining entfernt. In manchen Anwendungen sind jedoch die genau die Ausreißer von Interesse, beispielsweise um ungewöhnliche Transaktionen bei Bankgeschäften aufzudecken. Ausreißer können unter anderem durch statistische Tests oder durch die Analyse von Clustern entdeckt werden.

Prinzipiell eignen sich zur Prüfung der Datenintegrität alle vorgestellten Verfahren des Data Mining. In dieser Arbeit werden jedoch nicht alle Verfahren betrachtet. Die Regression wird nur kurz vorgestellt, da für die verwendeten Datensätze in den Kapiteln 6 und 7 keine Notwendigkeit für die Prädiktion numerischer Werte besteht. Häufige Muster, Beziehungen und Korrelationen werden ebenso nicht berücksichtigt. Die Analyse von Clustern wird in Unterabschnitt 2.6.6 lediglich kurz vorgestellt, da diese nur bedingt zur Aufstellung von Regeln geeignet ist. Da die Regeln für die Prüfung der Datenintegrität in dieser Arbeit so aufgestellt werden, dass sie die

gültigen Objekte und dementsprechend nicht die Ausreißer beschreiben, wird das Verfahren der Ausreißeranalyse nicht eingesetzt.

Im Folgenden werden in Unterabschnitt 2.6.3 die Klassenbeschreibung, in Unterabschnitt 2.6.4 die Klassifikation und in Unterabschnitt 2.6.5 die Regression vorgestellt.

2.6.3 Charakterisierende und diskriminierende Klassenbeschreibung

Klassenbeschreibungen lassen sich durch die Methoden der Charakterisierung und Diskriminierung sowie deren Kombination erstellen. Die zwei Methoden werden im Folgenden anhand ihrer Definition in Han u. a. (2011) kurz vorgestellt.

Die *Charakterisierung* bezeichnet die Zusammenfassung der charakteristischen Eigenschaften der Objekte einer Klasse. Ein einfacher aber effizienter Ansatz zur Charakterisierung ist die statistische Untersuchung der Daten, die auch die deren graphische Darstellung umfasst (siehe Abschnitt 2.5). Weitere Ansätze zur Charakterisierung benötigen eine spezielle Datenhaltung, wie beispielsweise in sogenannten Data Warehouses. Das Ergebnis der Charakterisierung sind die Beschreibungen der Klassen, die in Form von charakteristischen Regeln dargestellt werden können.

Bei der *Diskriminierung* werden die Eigenschaften einer Klasse mit den Eigenschaften einer oder mehrerer anderer Klassen verglichen, um so eine Abgrenzung der Klasse zu ermöglichen. Das Ergebnis der Diskriminierung sind die Unterschiede zwischen Klassen, die in Form von diskriminierenden Regeln dargestellt werden können.

In dieser Arbeit wird das Verfahren der Klassenbeschreibung extensiv genutzt um charakteristische Regeln für Objektklassen zu erstellen. Die erstellten Regeln basieren auf einer statistischen Untersuchung der Daten und werden sowohl für das Data Mining von Geobasisdaten in Abschnitt 6.3 als auch für das Data Mining von Open Data in Abschnitt 7.6 eingesetzt. Dagegen werden in dieser Arbeit keine diskriminierenden Regeln aufgestellt, da die Regeln für eine Objektklasse in sich geschlossen und unabhängig von den Regeln anderer Klassen sein sollen.

2.6.4 Klassifikation

Bei der Klassifikation wird ein Modell aufgestellt, durch das die Klassen eines Datensatzes beschrieben und voneinander getrennt werden können. Im Folgenden werden basierend auf den Ausführungen in Han u. a. (2011) zuerst einige grundlegende Begriffe definiert und dann die Klassifikation mit Entscheidungsbäumen vorgestellt. Schließlich werden zwei Vereinfachungen von Entscheidungsbäumen vorgestellt.

Zur Klassifikation von Datensätzen existieren neben der Erstellung von Entscheidungsbäumen sowie dessen Vereinfachungen viele weitere Klassifikatoren, wie neuronale Netzwerke, Support Vector Machines oder Bayessche Verfahren (siehe beispielsweise Han u. a. (2011)). Die genannten Klassifikatoren sind Entscheidungsbäumen teilweise bezüglich der Aspekte Genauigkeit, Geschwindigkeit, Robustheit und Skalierbarkeit überlegen. Der wichtigste Aspekt für diese Arbeit ist jedoch die Interpretierbarkeit eines Klassifikators, da das erstellte Modell in einfach nachvollziehbare formale Regeln überführt werden soll. Diesbezüglich ist die Klassifikation mit Entscheidungsbäumen sowie dessen Vereinfachungen den anderen genannten Klassifikatoren deutlich überlegen.

Grundlagen der Klassifikation

Die Klassifikation von Daten besteht aus zwei Schritten. Im ersten Schritt, der als *Training* bezeichnet wird, wird das Modell aus bereits klassifizierten Objekten gelernt. Die Objekte des Trainingsdatensatzes weisen dabei neben ihren sonstigen Attributen ein nominales Attribut auf, das die Klasse der Objekte bezeichnet. Das Ergebnis des Trainings ist eine Zuordnungsfunktion, die basierend auf den sonstigen Attributen eines Objektes dessen Klasse prädiziert. Im zweiten Schritt, der als *Klassifikation* bezeichnet wird, werden anhand der Zuordnungsfunktion die Klassen von a priori unklassifizierten Objekten bestimmt.

Die Genauigkeit der Klassifikation, die als *Klassifikationsgüte* bezeichnet wird, gibt an, wie gut das Modell beziehungsweise die Zuordnungsfunktion die Klassen prädiziert. Für einen Testdatensatz, der bereits klassifizierte Objekte enthält, werden die bekannten Klassen der Objekte mit den prädizierten Klassen der Objekte verglichen. Die Klassifikationsgüte gibt dann den Anteil der korrekt klassifizierten Objekte an. Dabei ist es wichtig, dass die Objekte des Testdatensatzes von den Objekten des Trainingsdatensatzes unabhängig sind, da sonst die Klassifikationsgüte als zu optimistisch bestimmt wird. Dies ist darauf zurückzuführen, dass die

Zuordnungsfunktion zu einer Überanpassung an die Daten neigt, d.h. sie berücksichtigt besondere Anomalien des Trainingsdatensatzes, die jedoch nicht repräsentativ für die Grundgesamtheit sind.

Eine Klassifikation kann auch erstellt werden, wenn nur ein einzelner Datensatz vorhanden ist. Dazu werden die Objekte des Datensatzes in zwei zufällige Teilmengen unterteilt, wobei die erste Teilmenge für das Training und die zweite Teilmenge für den Test verwendet wird. Eine einfache Methode ist die Unterteilung der Objekte anhand eines festen Prozentsatzes (engl. 'holdout'), wobei typischerweise zwei Drittel der Objekte für das Training und das verbleibende Drittel für den Test verwendet werden.

Modellselektion

Um mehrere Modelle und damit Klassifikatoren vergleichen zu können, kann die *k-fache Kreuzvalidierung* (engl. 'k-fold cross validation') verwendet werden. Bei der k-fachen Kreuzvalidierung werden die Objekte zufällig in k Teilmengen annähernd gleichen Umfangs unterteilt. Das Training und der Test werden dann k mal durchgeführt, wobei im i-ten Schritt die i-te Teilmenge für den Test und die übrigen $k - 1$ Teilmengen für das Training verwendet werden. Die Klassifikationsgüte der k-fachen Kreuzvalidierung wird schließlich klassisch über die Anzahl der korrekt klassifizierten Objekte ermittelt. Da jedes Objekt nur in einem der k Testdatensätze enthalten ist, wird es somit auch nur ein einziges Mal klassifiziert. Das Ziel einer k-fachen Kreuzvalidierung ist es dabei jedoch nicht, eines der k Modellinstanzen als beste Modellinstanz zu selektieren. Stattdessen wird die k-fache Kreuzvalidierung verwendet, um die Genauigkeit verschiedener Modelle vergleichen zu können, um schließlich das geeignetste Modell auswählen zu können. Bei einer stratifizierten k-fachen Kreuzvalidierung (engl. 'stratified k-fold cross validation') werden die Teilmengen so gewählt, dass die Verteilung der Objekte auf die Klassen in allen Teilmengen ungefähr der entsprechenden Verteilung im Gesamtdatensatz entspricht (siehe stratifizierte Stichprobe in Unterabschnitt 2.6.1).

Entscheidungsbäume

Ein Entscheidungsbaum kodiert aufeinanderfolgende Entscheidungen in Form eines gerichteten Baums. Der Baum besteht aus einem Wurzelknoten, beliebig vielen inneren Knoten sowie mindestens zwei Blattknoten. Anhand des Abbildung 2.30 dargestellten Beispiels wird im Folgenden der Aufbau eines Entscheidungsbaums detaillierter vorgestellt. Im Beispiel werden Vermessungspunkte anhand ihrer Attributwerte den drei Klassen Grenzpunkt, Trigonometrischer Punkt und Höhenfestpunkt zugeordnet. Am *Wurzelknoten* und an jedem *inneren Knoten* wird der Wert eines Objektattributs getestet. Am Wurzelknoten Material wird beispielsweise der Wert des Attributs Material getestet. Die möglichen Entscheidungen des Tests werden als Äste beziehungsweise Kanten des Baums angegeben, wobei das Attribut Material im Beispiel die beiden Werte 'Stein' oder 'Metall' annehmen kann. An den *inneren Knoten* werden weitere Tests anhand der Objektattribute ausgeführt. In den *Blattknoten* stehen schließlich die den Objekten zugeordneten Klassen. Beispielsweise wird für einen Vermessungspunkt mit den Attributwerten 'Material=Metall' und 'Ausrichtung=vertikal' die Klasse Grenzpunkt prädiziert.

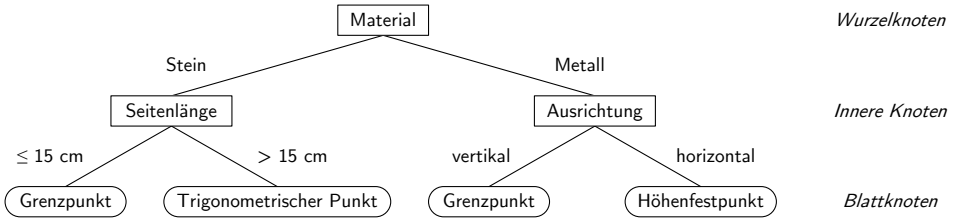


Abbildung 2.30: Beispiel für einen einfachen Entscheidungsbaum zur Klassifikation von Vermessungspunkten

Anhand des Beispiels werden im Folgenden zwei weitere Eigenschaften von Entscheidungsbäumen vorgestellt. Der Entscheidungsbaum im Beispiel ist binär, da der Wurzelknoten und jeder innere Knoten jeweils nur zwei unterschiedliche Ergebnisse für einen Test aufweisen. Allgemein unterstützen Entscheidungsbäume jedoch beliebig viele Entscheidungen für den Test eines Attributwerts. Diese lassen sich aber durch Umformungen immer auf einen binären Entscheidungsbaum abbilden. Die *Tiefe* eines Entscheidungsbaums bezeichnet die maximale Anzahl der zu testenden Attribute vom Wurzelknoten bis zu den Blattknoten. Beispielsweise hat der Baum in Abbildung 2.30 die Tiefe zwei.

Algorithmen zur Aufstellung von Entscheidungsbäumen wurden bereits in den späten 1970er und frühen 1980er Jahren entwickelt. Die beiden ersten und unabhängig voneinander entwickelten Algorithmen sind die binären Entscheidungsbäume im Buch 'Classification and Regression Trees' (CART) von Breiman u. a. (1984) und der

ID3-Algorithmus (engl. 'iterative dichotomiser') von Quinlan (1986). Eine Verbesserung des ID3-Algorithmus stellt der C4.5-Algorithmus dar, der im Buch von Quinlan (1993) vorgestellt wird, und von dem eine Implementierung in dieser Arbeit verwendet wird.

Im Folgenden wird die Strategie zur Aufstellung eines einfachen Entscheidungsbaums entsprechend der Beschreibung in Han u. a. (2011) vorgestellt, die zudem als Pseudo-Code im Programm 2.6 angegeben ist:

- Der Algorithmus benötigt drei Eingaben. Die Datenpartition D enthält anfänglich alle klassifizierten Trainingsobjekte. Der Parameter *attributliste* ist eine Auflistung aller Attribute der Objekte. Die *AttributSelektionsMethode* gibt eine Heuristik an, mit der dasjenige Attribut ausgewählt werden kann, das die gegebenen Objekte am besten in disjunkte Partitionen unterteilt. Die Partitionen sollen dabei möglichst rein sein, d.h. nur Objekte derselben Klasse enthalten. Die Methode verwendet dafür ein Maß für die Attributselektion, wie beispielsweise den Informationsgewinn, der im weiteren noch erklärt wird.
- Der Entscheidungsbaum beginnt mit einem einzelnen Knoten N, der die Trainingsobjekte der Datenpartition D enthält.
- Falls alle Objekte in D derselben Klasse C angehören, dann wird der Knoten N in einen Blattknoten umgewandelt und mit der Klasse C gekennzeichnet.
- Ansonsten wird die *AttributSelektionsMethode* angewendet um das *teilungskriterium* zu ermitteln. Das Teilungskriterium enthält das *teilungsattribut*, anhand dessen die Teilung durchgeführt wird, sowie den *teilungspunkt* oder die *teilungsmenge*.

  Im Beispiel des Entscheidungsbaums zur Klassifikation von Vermessungspunkten in Abbildung 2.30 ist das Attribut Seitenlänge ein Teilungsattribut. Der Wert '15 cm' ist der zugehörige Teilungspunkt, der die Objekte in D in zwei disjunkte Mengen unterteilt.

  Im gleichen Beispiel ist das Attribut Ausrichtung auch ein Teilungsattribut, und die Menge ('vertikal', 'horizontal') die zugehörige Teilungsmenge, welche die Objekte in D in zwei disjunkte Mengen unterteilt.
- Falls das Teilungsattribut diskret ist, dann wird dieses aus der Attributliste entfernt, da alle Möglichkeiten der Testentscheidung in der Teilungsmenge bereits enthalten sind.
- Anschließend wird der Knoten N mit dem Teilungskriterium gekennzeichnet. Für jede der j Testentscheidungen wird ein Ast in den Entscheidungsbaum eingeführt und die Objekte in D entsprechend den j Partitionen zugeteilt.
- Der Algorithmus benutzt dieselbe Strategie rekursiv um für die Objekte der j Partitionen wiederum Entscheidungsbäume aufzubauen.
- Die rekursive Ausführung wird nur abgebrochen, wenn eine der folgenden terminalen Bedingungen erfüllt ist:

  1. Alle Objekte der Partition D im Knoten N gehören derselben Klasse C an.
  2. Die Attributliste ist leer, d.h. alle Attribute wurden bereits aus ihr entfernt. Der Knoten N wird dann in einen Blattknoten umgewandelt. Dieser wird durch einen Mehrheitsentscheid mit der häufigsten Klasse der Objekte der Partition D gekennzeichnet.
  3. Für einen Ast sind keine Objekte in der Partition Dj enthalten. Es wird dann ein Blattknoten erzeugt, der die häufigste Klasse der Objekte in D enthält.
- Schließlich wird der erzeugte Entscheidungsbaum als Ergebnis des Algorithmus zurückgegeben.

Die entscheidende Komponente bei der Erstellung eines Entscheidungsbaums ist die Attributselektion, mit der das Teilungskriterium bestimmt wird. Die Attributselektion kann auf unterschiedlichen Maßen basieren, von denen im Folgenden zwei vorgestellt werden. Weitere Maße werden beispielsweise in Han u. a. (2011) vorgestellt. Dabei bezeichne D die Datenpartition bestehend aus klassifizierten Trainingsobjekten, die insgesamt m Klassen $C_1, \ldots, C_m$ zugeordnet seien. Sei weiter $C_{i,D}$ die Menge der Objekte der Klasse C_i in D. Die Anzahl der Objekte in $C_{i,D}$ sei $|C_{i,D}|$ und analog $|D|$ die Anzahl der Objekte in D.

Das erste Maß für die Attributselektion ist der *Informationsgewinn* (engl. 'information gain'), der im ID3-Algorithmus verwendet wird und auf der Informationstheorie von Shannon (1948) basiert. Das Attribut mit dem höchsten Informationsgewinn partitioniert die Objekte der Datenpartition D in einem Knoten N so, dass die Partitionen möglichst rein sind. In den erzeugten Partitionen wird dadurch die zur Klassifizierung der

```
Algorithmus: ErstelleEntscheidungsbaum. Erstellt einen Entscheidungsbaum basierend auf Objekten
    einer Datenpartition D.

Eingabe:
  - D, Datenpartition bestehend aus klassifizierten Trainingsobjekten
  - attributliste, Attributliste mit Attributkandidaten
  - AttributSelektionsMethode, Methode zur Bestimmung des Teilungskriteriums, das die Objekte 'am
      besten' in einzelne Partitionen unterteilt. Das Kriterium besteht aus einem teilungsattribut
       und entweder einem teilungspunkt oder einer teilungsmenge.

Ausgabe: Ein Entscheidungsbaum.

Methode:
  Erstelle einen Knoten N;
  if Objekte in D gehören alle der gleichen Klasse C an then
    return N als Blattknoten mit der Klasse C;
  if attributliste ist leer then
    return N als Blattknoten mit der Mehrheitsklasse in D; // Mehrheitsentscheid
  apply AttributSelektionsMethode(D, attributliste) um 'bestes' teilungskriterium zu finden;
  if teilungsattribut ist diskret then
    attributliste ← attributliste − teilungsattribut; // Entferne teilungsattribut
  Kennzeichne Knoten N mit dem teilungskriterium;
  for each ausgabe j des teilungskriterium // Partitioniert die Objekte und erstellt Unterbäume für
       jede Partition
    let Dj die Menge der Objekte in D welche der Testentscheidung j angehören // Eine Partition
    if Dj ist leer then
      füge einen Blattknoten mit der Mehrheitsklasse in D zum Knoten N hinzu;
    else füge den Knoten, der von ErstelleEntscheidungsbaum(Dj, attributliste) zurückgegeben wird,
         zu N hinzu;
  endfor
  return N;
```

Programm 2.6: Pseudo-Code zur Erstellung eines Entscheidungsbaums nach Han u. a. (2011)

Objekte benötigte Information minimiert. Dieser Ansatz minimiert die Anzahl der zu erwartenden Tests für die Klassifikation eines Objekts und stellt zugleich sicher, dass ein einfacher Entscheidungsbaum aufgestellt wird.

Die erwartete Information zur Klassifikation eines Objekts in D wird durch die Gleichung 2.25 angegeben. Dabei bezeichnet p_i die Wahrscheinlichkeit, dass ein beliebiges Objekt aus D der Klasse C_i zugeordnet ist. Der Wert $\mathit{Info}\,(D)$ gibt dann den Mittelwert für die Information an, die zur Klassifikation eines Objekts in D benötigt wird. Dieser Wert wird auch als Entropie von D bezeichnet.

$$\mathit{Info}\,(D) = -\sum_{i=1}^{m} \left(p_i \cdot \log_2(p_i)\right), \text{ mit } p_i = \frac{|C_{i,D}|}{|D|}, \; p_i > 0 \tag{2.25}$$

Wird ein diskretes Attribut A mit den Werten $a_1, \ldots, a_v$ zur Partitionierung von D im Knoten N verwendet, dann folgen daraus v disjunkte Partitionen $D_1, \ldots, D_v$. Dabei enthält die Partition D_j genau die Objekte, für die das Attribut A den Wert a_j aufweist. Durch die Partitionierung verringert sich die benötigte Information für die Klassifikation auf die in Gleichung 2.26 angegebene Größe $\mathit{Info}_A(D)$, in der die Anzahl der Objekte in jeder Partition als Gewicht einfließt. Je geringer die benötigte Information für die Klassifikation nach der Partitionierung ist, umso reiner sind die erstellten Partitionen. Die Differenz zwischen den beiden Maßen bildet den Informationsgewinn $\mathit{Gain}\,(A)$, der in Gleichung 2.27 angegeben ist. Bei der Attributselektion auf Basis des Informationsgewinns wird dann genau dasjenige Attribut A gewählt, für das der Informationsgewinn $\mathit{Gain}\,(A)$ maximal beziehungsweise $\mathit{Info}_A(D)$ minimal ist.

$$\mathit{Info}_A(D) = -\sum_{j=1}^{v} \left(\frac{|D_j|}{|D|} \cdot \mathit{Info}\,(D_j)\right) \tag{2.26}$$

$$\mathit{Gain}\,(A) = \mathit{Info}\,(D) - \mathit{Info}_A(D) \tag{2.27}$$

Wird stattdessen ein kontinuierliches Attribut A zur Partitionierung von D im Knoten N verwendet, dann muss der optimale Teilungspunkt ermittelt werden, der die Objekte in D in exakt zwei disjunkte Partitionen D_1 und D_2 teilt. Dazu werden die insgesamt j Werte von A in aufsteigender Reihenfolge sortiert und für jeden Mittelpunkt von zwei Werten a_i und a_{i+1} mit $i = 1, \ldots, j-1$ wird die Information $\mathit{Info}_A(D)$ ermittelt.

Der Teilungspunkt mit dem kleinsten Wert für $Info_A(D)$ wird schließlich ausgewählt und die Objekte in D entsprechend geteilt.

Ein kontinuierliches Attribut A kann jedoch auch zur Erstellung von mehr als zwei Partitionen verwendet werden. Dazu müssen lediglich mehrere Teilungspunkte ermittelt werden. Dies wird am Beispiel in Tabelle 2.13 deutlich, in dem eine Menge von Werten mit zugeordneten Klassen jeweils durch eine unterschiedliche Anzahl an Teilungspunkten partitioniert wird. Wird die Anzahl und Lage der Teilungspunkte so gewählt, dass jede Partition jeweils nur Objekte derselben Klasse enthält, dann ist die Partitionierung optimal und damit der Informationsgewinn maximal. Im Beispiel ist dies bei sechs Teilungspunkten der Fall. Um eine zu große Anzahl an Teilungspunkten zu vermeiden, wird häufig eine minimale Anzahl an Objekten pro Partition gefordert. Im Beispiel folgen aus einer minimalen Partitionsgröße von drei Objekten insgesamt zwei Teilungspunkte. Von den so erzeugten Partitionen ist nur eine rein, wohingegen zwei Partitionen jeweils ein Objekt einer abweichenden Klasse enthalten. Der Informationsgewinn fällt im Beispiel noch geringer aus, wenn nur ein einziger Teilungspunkt verwendet wird.

	a_1	a_2	a_3	a_4	a_5	a_6	a_7	a_8	a_9	a_{10}	a_{11}	a_{12}	$Info_A(D)$
Wert	0	2	4	5	8	9	10	14	16	17	20	21	—
Klasse	A	A	B	A	B	B	B	C	C	C	B	C	—
6 Teilungspunkte	A	A ∘	B ∘	A ∘	B	B	B ∘	C	C	C ∘	B ∘	C	0
2 Teilungspunkte	A	A	A	A ∘	B	B	B ∘	C	C	C	C	C	0,57
1 Teilungspunkt	B	B	B	B	B	B	B ∘	C	C	C	C	C	0,88

Tabelle 2.13: Beispiel für die Partitionierung eines kontinuierlichen Attributs

Das zweite Maß zur Attributselektion ist das *Informationsgewinn-Verhältnis* (engl. 'gain ratio'), das im C4.5-Algorithmus verwendet wird. Dieses Maß kompensiert eine Problematik bei der Verwendung des Informationsgewinns, nämlich die Tendenz Attribute mit vielen Werten bevorzugt zu selektieren. Je weniger Objekte eine Partition umfasst, umso höher ist tendenziell die Wahrscheinlichkeit, dass die Partition rein ist. Ein Extrembeispiel ist die Partitionierung der Objekte nach einem eindeutigen Identifikator, die $|D|$ Partitionen mit jeweils einem einzelnen Objekt erzeugt. Das Informationsgewinn-Verhältnis $GainRatio(A)$, das in Gleichung 2.29 angegeben ist, verwendet deshalb eine Normalisierung mit dem Wert $SplitInfo_A(D)$ aus Gleichung 2.28. Der Wert $SplitInfo_A(D)$ gibt die potentielle Information an, die durch die Aufteilung von D in v Partitionen erzeugt wird. Wird das Informationsgewinn-Verhältnis zur Attributselektion verwendet, so wird genau dasjenige Attribut gewählt, für welches das Informationsgewinn-Verhältnis $GainRatio(A)$ maximal ist.

$$SplitInfo_A(D) = -\sum_{j=1}^{v} \left(\frac{|D_j|}{|D|} \cdot \log_2 \frac{|D_j|}{|D|} \right) \tag{2.28}$$

$$GainRatio(A) = \frac{Gain(A)}{SplitInfo_A(D)} = \frac{Info(D) - Info_A(D)}{SplitInfo_A(D)} \tag{2.29}$$

Wird ein Entscheidungsbaum als Zuordnungsfunktion für die Klassifikation verwendet, dann neigt dieser wie auch andere Zuordnungsfunktionen zu einer Überanpassung an die Daten. Besondere Anomalien des Trainingsdatensatzes, die jedoch nicht repräsentativ für die Grundgesamtheit sind, finden sich somit im Entscheidungsbaum wieder. Um der Überanpassung zu begegnen, können Entscheidungsbäume so zurückgeschnitten werden, dass die statistisch am wenigsten stabilen Äste entfernt werden (engl. 'pruning'). Das Ziel des *Zurückschneidens* ist das aufwiegen der Komplexität eines Entscheidungsbaum gegen dessen Klassifikationsgüte, d.h. komplexe Folgen von Entscheidungen werden verworfen, wenn diese nur zu einen marginalen Zugewinn der Klassifikationsgüte führen. Im Folgenden werden die beiden grundlegenden Ansätze zum Zurückschneiden am Beispiel des Entscheidungsbaums in Abbildung 2.31 erläutert. Der erste Ansatz ist die Ersetzung eines Unterbaums (engl. 'subtree replacement') durch einen einzigen Blattknoten. Im Beispiel wird der Unterbaum mit den Attributen b und d durch einen einzelnen Blattknoten mit der Klasse Q ersetzt. Die Klasse Q folgt aus einem Mehrheitsentscheid der Klassen K, M und N. Der zweite Ansatz ist das Verschieben eines Unterbaums (engl. 'subtree raising') in Richtung des Wurzelknotens. Im Beispiel wird der Knoten *Attribut c* entfernt und der Unterbaum des Knotens *Attribut e* nach oben verschoben. Anschließend muss der so erstellte Teilbaum erneut klassifiziert werden. Das Verschieben ist demnach aufwändiger als das Ersetzen, kann sich jedoch lohnen wenn beispielsweise die Anzahl der Objekte in Klasse L gering ist. Prinzipiell können Entscheidungsbäume sowohl bei ihrer Erstellung (engl. 'pre-pruning') als auch in deren Anschluss (engl. 'post-pruning') zurückgeschnitten werden.

Der C4.5-Algorithmus schneidet die von ihm erstellten Bäume erst zurück, nachdem diese komplett erstellt sind. Für das Zurückschneiden werden lediglich die Trainingsobjekte verwendet (für weitere Details siehe Quinlan (1993)).

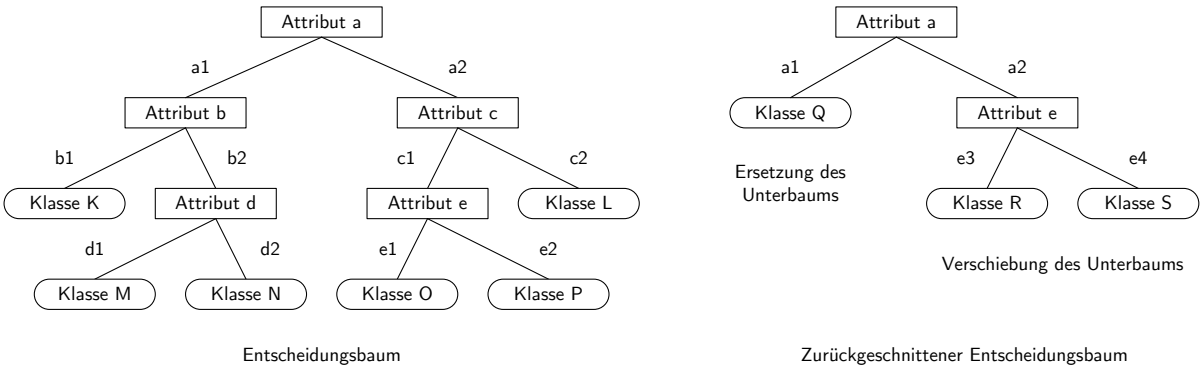


Abbildung 2.31: Beispiel für das Zurückschneiden eines Entscheidungsbaums

Eine Erweiterung der Nutzung eines einzelnen Entscheidungsbaums zur Klassifikation ist das *Random Forest* Verfahren, bei denen die Klassifikationen mehrerer Entscheidungsbäume zu einem Mehrheitsentscheid kombiniert werden. Für die Erstellung von Random Forests existieren mehrere Ansätze, die unter anderem in Han u. a. (2011) beschrieben werden. Die Klassifikationsgüte von Random Forests liegt üblicherweise über der eines einzelnen Entscheidungsbaums. Da die Mehrzahl an verwendeten Entscheidungsbäumen jedoch auch zu einer deutlich höheren Komplexität der Regeln führt, werden Random Forests für die Analyse der Datensätze in den Kapiteln Kapitel 6 und 7 nicht verwendet.

OneRule

Eine Vereinfachung des Entscheidungsbaums stellt der *OneRule*-Algorithmus dar, der mit einer einzigen Entscheidung auskommt. Der entsprechende Baum hat die Tiefe 1 und somit keine internen Blattknoten, sondern lediglich einen Wurzelknoten und mehrere Blattknoten. In Abbildung 2.32 ist ein Beispiel für das Ergebnis des OneRule-Algorithmus dargestellt, der damit einer Vereinfachung des Entscheidungsbaums aus Abbildung 2.31 entspricht. Der OneRule-Algorithmus wird in Holte (1993) vorgestellt. Als Maß für die Attributselektion verwendet der Algorithmus den Informationsgewinn.

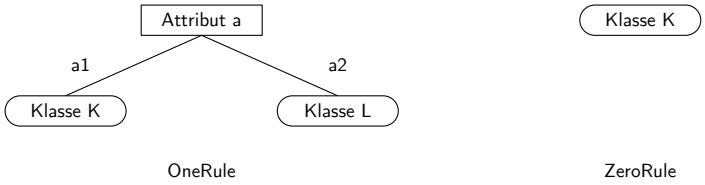


Abbildung 2.32: Beispiel für die Ergebnisse des OneRule- und ZeroRule-Algorithmus

In Holte (1993) werden 16 häufig verwendete Testdatensätze des maschinellen Lernens sowohl mit dem C4-Algorithmus, einem Vorgänger des C4.5-Algorithmus, als auch mit dem OneRule-Algorithmus klassifiziert. Die Anzahl der Objekte pro Datensatz reicht dabei von etwa 50 bis über 8100. In einem ersten Experiment wird jeweils nur das beste Attribut ausgewählt und anhand dessen der Entscheidungsbaum aufgebaut. Für die 16 Datensätze ist dabei die Klassifikationsgüte des OneRule-Algorithmus um durchschnittlich 6 % schlechter als die des C4.5-Algorithmus. Der Abstand der beiden Algorithmen verringert sich jedoch auf 3 %, wenn die zwei schlechtesten Datensätze nicht berücksichtigt werden.

In einem zweiten Experiment werden im Training mehrere Attribute ausgewählt und für jedes dieser Attribute ein eigener Entscheidungsbaum der Tiefe 1 aufgebaut. Als Maß für die Attributselektion wird dabei der Informationsgewinn verwendet. Beim Vergleich mit dem Testdatensatz wird dann nur der beste Entscheidungsbaum für die Ermittlung der Klassifikationsgüte herangezogen. Der Unterschied der Klassifikationsgüte zwischen den zwei Algorithmen verringert sich dadurch auf durchschnittlich 2 % für alle 16 Datensätze und sogar auf durchschnittlich 0 % für 15 der Datensätze. Diese Variante des OneRule-Algorithmus wird auch in dieser Arbeit verwendet.

Zwei Einschränkungen für die Anwendung des OneRule-Algorithmus werden in Holte (1993) erwähnt. Erstens eignet sich der einfache Algorithmus nicht für bewiesen schwere Klassifikationsprobleme[17], wie beispielsweise die Klassifikation der Struktur von Proteinen. Zweitens ist der Algorithmus in seiner maximalen Klassifikationsgüte beschränkt, wenn die Anzahl der Klassen höher ist als die maximale Anzahl an unterschiedlichen Werten eines Attributs. Ein Beispiel für diese Beschränkung ist in Tabelle 2.14 aufgelistet. Den sechs Klassen A bis F stehen lediglich drei unterschiedliche Attributwerte a_1 bis a_3 gegenüber. Selbst wenn die drei Klassen mit der höchsten Anzahl an Objekten (A, B und C) korrekt klassifiziert werden, so müssen doch die verbleibenden Klassen einer einzigen Klasse (D) über einen Mehrheitsentscheid zugeordnet werden. Dadurch ist im Beispiel die maximal erreichbare Klassifikationsgüte auf 80 % beschränkt.

Klasse	Anzahl Objekte	Klassifiziert mit	Klassifiziert als	Klassifikationsgüte [%]
A	300	a_1	A	100
B	200	a_2	B	100
C	200	a_3	C	100
D	100	—	D	100
E	100	—	D	0
F	100	—	D	0
A – F	1000	a_1 – a_3	A – D	80

Tabelle 2.14: Beispiel für die maximale Klassifikationsgüte beim OneRule-Algorithmus

Holte (1993) sieht den OneRule-Algorithmus besonders dann im Vorteil gegenüber komplexeren Algorithmen, wenn die geringe Komplexität wichtiger ist als die Erreichbarkeit der maximal möglichen Klassifikationsgüte. Dies gilt beispielsweise für den Fall, dass der Klassifikationsprozess durch einen Benutzer einfach nachvollziehbar sein soll. Holte (1993) fasst dies in dem Paradigma 'erstmal einfach' (engl. 'simplicity first') zusammen.

ZeroRule

Eine weitere Vereinfachung des OneRule-Algorithmus stellt der *ZeroRule*-Algorithmus dar, dessen Anzahl an Entscheidungen gleich Null ist. Demnach besteht bei diesem Algorithmus der Entscheidungsbaum lediglich aus einem einzigen Blattknoten, dessen zugeordnete Klasse durch einen Mehrheitsentscheid bestimmt wird. In Abbildung 2.32 ist ein Beispiel für das Ergebnis des ZeroRule-Algorithmus dargestellt, der damit einer maximal möglichen Vereinfachung des Entscheidungsbaums aus Abbildung 2.31 entspricht.

Mit dem ZeroRule-Algorithmus lässt sich die minimal zu erwartende Klassifikationsgüte bestimmen. Insbesondere bei Datensätzen die von einer Klasse dominiert werden, erinnert der Algorithmus daran die absolute Klassifikationsgüte eines komplexeren Algorithmus in Relation zu den Ausgangsdaten zu setzen. Beispielsweise erscheint die Klassifikationsgüte eines komplexen Entscheidungsbaums von 96 % auf den ersten Blick beeindruckend. Gehören jedoch 95 % der Objekte einer einzigen Klasse an, dann ist der Informationsgewinn marginal.

2.6.5 Regression

Der funktionale Zusammenhang zwischen den Werten von zwei oder mehr Attributen lässt sich durch eine Regressionsanalyse bestimmen. Im Folgenden wird der lineare Zusammenhang zwischen zwei Variablen untersucht und entsprechend den Ausführungen in Han u. a. (2011) vorgestellt. Die Verallgemeinerung auf mehrere Variablen und nicht-lineare Zusammenhänge ist beispielsweise in Han u. a. (2011) beschrieben.

Bei der linearen Regression wird der Zusammenhang zweier Attribute durch eine ausgleichende Gerade nach der Methode der kleinsten Quadrate geschätzt. Für zwei Attribute x und y wird eine Gerade allgemein durch die Gleichung 2.30 beschrieben, wobei die zwei Regressionskoeffizienten m und b die Steigung und den y-Achsenabschnitt bezeichnen.

$$y = m \cdot x + b \tag{2.30}$$

[17] Bewiesen schwere Probleme beziehen sich dabei auf NP-vollständige Problemstellungen, die nichtdeterministisch in polynomieller Zeit lösbar sind.

Für das in Abschnitt 2.5 eingeführte Beispiel der Flächen und Bevölkerung der 16 Bundesländer Deutschlands ist in Abbildung 2.33 die entsprechende Regressionsgerade dargestellt. Diese wird als zusätzliche Information in den Scatter-Plot eingezeichnet (vgl. Abbildung 2.28). Dabei wird die Regressionsgerade durch die Gleichung 2.31 eindeutig bestimmt. Die Regressionsgerade weist jedoch hohe Standardabweichungen von 49 für die Steigung und von 1,4 Millionen für den y-Achsenabschnitt auf, die auf die Streuung der Daten zurückzuführen sind. Aus der Fläche eines Bundeslandes lässt sich demnach mit einer Regressionsgerade nur bedingt die Bevölkerung ableiten.

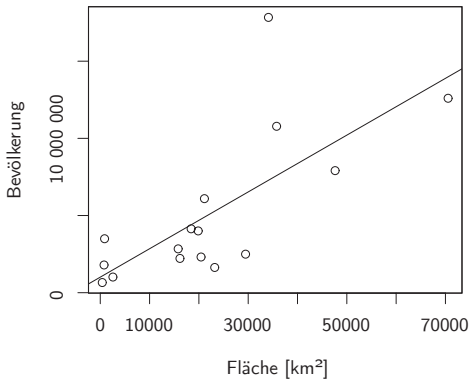


Abbildung 2.33: Scatter-Plot mit Regressionsgerade der Fläche und der Bevölkerung des Beispiels

$$\textit{Bevölkerung} = 184 \cdot \textit{Fläche} + 1008047 \tag{2.31}$$

2.6.6 Spatial Data Mining

In Analogie zur Definition des Data Mining bezeichnet *Spatial Data Mining* den Prozess der Entdeckung von interessanten Mustern und Wissen in umfangreichen räumlichen Daten. In Miller (2008) wird der Begriff des KDD entsprechend auf das sogenannte *Geographic Knowledge Discovery* (GKD) übertragen.

Durch die Verwendung von räumlichen Daten ergeben sich nach Miller (2008) einige Änderungen im Vergleich zu nicht-räumlichen Daten. Räumliche Objekte sind implizit durch die drei elementaren Relationen *Distanz*, *Orientierung* und *Topologie* miteinander verbunden. Diese müssen beim Data Mining entsprechend berücksichtigt werden. Zudem spielen sowohl die räumliche Abhängigkeit als auch die räumliche Heterogenität eine wichtige Rolle. Die *räumliche Abhängigkeit* bezieht sich darauf, dass Objekte die räumlich benachbart sind oft eine höhere Ähnlichkeit aufweisen als räumlich entfernte Objekte. Die *räumliche Heterogenität* bezieht sich darauf, dass die Eigenschaften der Objekte in Bezug auf den gesamten geographischen Raum variieren.

Die Verfahren des Spatial Data Mining können ebenso wie in Unterabschnitt 2.6.2 in fünf Gruppen unterteilt werden. Im Folgenden werden die Änderungen, die durch die Verwendung räumlicher Daten entstehen, entsprechend den Ausführungen in Miller (2008) sowie Guo und Mennis (2009) vorgestellt:

- *Räumliche Klassen- / Konzeptbeschreibung*: Bei der Beschreibung der Klassen können nun auch räumliche Eigenschaften beziehungsweise Attribute wie Form, Distanz, Orientierung und Topologie berücksichtigt werden.
- *Räumliche Klassifikation und räumliche Regression*: Bei der räumlichen Klassifikation werden die räumlichen Eigenschaften der Objekte berücksichtigt, wobei auch benachbarte Objekte und deren Beziehungen in das Modell einfließen können. Dazu ähnlich ist die räumliche Regression, bei der die Attributwerte eines Objekts aus den Attributwerten der benachbarten Objekte abgeleitet werden.
- *Häufige räumliche Muster, Beziehungen und Korrelationen*: Auch hier können die räumlichen Eigenschaften beispielsweise als zusätzliche Attribute berücksichtigt werden.
- *Analyse von räumlichen Clustern*: Cluster-Verfahren für nicht-räumliche Daten können durch die Berücksichtigung der Form, Distanz, Orientierung und Topologie auch für räumliche Daten genutzt werden. Dadurch können Objekte aufgrund ihrer Ähnlichkeit in Cluster partitioniert werden, wobei Objekte desselben Clusters nicht zwingend benachbart sein müssen. Ein Beispiel für diesen Ansatz ist in Werder u. a. (2010) zu finden, wo Gebäude anhand ihrer geometrischen und topologischen Eigenschaften automatisch in zehn Cluster unterteilt werden. Anschließend werden diese durch eine visuelle Analyse zu fünf Gruppen

zusammengefasst. Exemplarische Gebäude und deren Zuweisung zu einer der fünf Gruppen sind in Abbildung 2.34 dargestellt. Beispielsweise ist die zweite Gruppe dadurch gekennzeichnet, dass die Gebäude dieser Gruppe häufig Teil einer homogenen Gebäudereihe oder eines Gebäudeblocks sind. Dagegen sind Gebäude der ersten Gruppe meist freistehend. In Werder u. a. (2010) werden die Ergebnisse des Clustering anschließend genutzt um durch von Straßen begrenzte Blöcke bezüglich ihrer Nutzung zu klassifizieren.

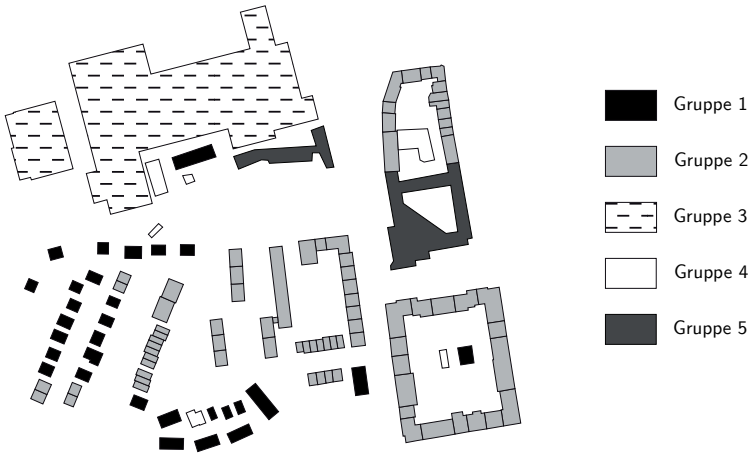


Abbildung 2.34: Clustering von Gebäuden nach Werder u. a. (2010)

Neben den partitionierenden Cluster-Verfahren können für die Bildung räumlicher Cluster auch hierarchische, dichte-, gitter- und modellbasierte Verfahren verwendet werden.

– *Räumliche Ausreißeranalyse*: Selbst wenn die Attributwerte eines Objekts in Bezug auf den Gesamtdatensatz konsistent sind, so können diese doch in Bezug auf die räumlich benachbarten Objekte signifikant abweichen. Durch die Analyse der räumlichen Nachbarschaft von Objekten können somit mögliche Ausreißer identifiziert werden.

2.6.7 Auswertung der Muster

Als Ergebnis des Data Mining können eine Vielzahl an Mustern entstehen. Um diejenigen Muster identifizieren zu können, die im Sinne des Data Mining interessant sind, müssen die entstandenen Muster anhand geeigneter Maße verglichen werden. Für diese Arbeit ist es wichtig, die Ergebnisse verschiedener Klassifikationen vergleichend quantitativ bewerten zu können. Ein entsprechendes Verfahren wird im Folgenden entsprechend den Ausführungen in Han u. a. (2011) beschrieben.

Die Güte einer Klassifikation lässt sich durch mehrere Maße beschreiben, die ausdrücken wie genau das erstellte Modell die Klasse eines Objekts prädizieren kann. Die Maße basieren dabei auf disjunkten Teilmengen der Objekte, die im Folgenden zuerst vorgestellt werden.

Für eine bestimmte Klasse K können die Objekte eines Datensatzes in zwei disjunkte Teilmengen unterteilt werden. Die Teilmenge der *positiven Objekte* zeichnet sich dadurch aus, dass die prädizierten Klassen der Objekte gleich der Klasse K sind. Die Teilmenge der *negativen Objekte* bildet die dazu komplementäre Menge, d.h. die prädizierten Klassen der Objekte sind ungleich der Klasse K. Werden die Objekte des Datensatzes nun klassifiziert, so lassen sich diese in vier disjunkte Teilmengen unterteilen:

– *Richtig positiv* (RP) (engl. 'true positives'): Anzahl der korrekt klassifizierten positiven Objekte, d.h. die prädizierte Klasse K' ist gleich der tatsächlichen Klasse K.

– *Richtig negativ* (RN) (engl. 'true negatives'): Anzahl der korrekt klassifizierten negativen Objekte, d.h. die prädizierte Klasse K' ist ungleich K und die tatsächliche Klasse ist auch ungleich K.

– *Falsch positiv* (FP) (engl. 'false positives'): Anzahl der falsch klassifizierten negativen Objekte, d.h. die prädizierte Klasse K' ist gleich K, obwohl die tatsächliche Klasse ungleich K ist.

– *Falsch negativ* (FN) (engl. 'false negatives'): Anzahl der falsch klassifizierten positiven Objekte, d.h. die prädizierte Klasse K' ist ungleich K, obwohl die tatsächliche Klasse gleich K ist.

Die vier Teilmengen können für beliebig viele Klassen in einer sogenannten *Konfusionsmatrix* zusammengefasst werden. Für eine binäre Klassifikation ist die Konfusionsmatrix in Tabelle 2.15 dargestellt. Die zwei Elemente auf der Diagonalen bezeichnen die korrekt klassifizierten Objekte, wohingegen die zwei verbleibenden Elemente die falsch klassifizierten Objekte bezeichnen. Zusätzlich ist die Gesamtzahl der positiven (P) und negativen (N) Objekte in der Tabelle angegeben.

Prädizierte Klasse		K	Nicht K	Gesamt
Tatsächliche Klasse	K	RP	FN	P
	Nicht K	FP	RN	N

Tabelle 2.15: Beispiel für eine Konfusionsmatrix

Aus den Größen lassen sich nun Maße für die Güte einer Klassifikation bestimmen. Die *Erkennungsrate* (engl. 'recognition rate') gibt den prozentualen Anteil der korrekt klassifizierten Objekte an und kann nach Gleichung 2.32 bestimmt werden.

$$Erkennungsrate = \frac{RP + RN}{P + N} \tag{2.32}$$

Die *Genauigkeit* (engl. 'precision') ist ein Maß für die Exaktheit der Klassifikation und gibt prozentual an, wieviele der als positiv prädizierten Objekte tatsächlich positiv sind. Die *Sensitivität* (engl. 'recall') ist ein Maß für die Vollständigkeit der Klassifikation und gibt prozentual an, wieviele der tatsächlich positiven Objekte auch als positiv prädiziert werden. Die Berechnung der beiden Maße ist in Gleichung 2.33 angegeben. Die beiden Maße Genauigkeit und Sensitivität ergänzen sich dabei in ihren Aussagen, da die Genauigkeit keine Aussage über die Anzahl der falsch klassifizierten positiven Objekte (FN) und die Sensitivität keine Aussage über die Anzahl der falsch klassifizierten negativen Objekte (FP) enthält.

$$Genauigkeit = \frac{RP}{RP + FP}, \quad Sensitivität = \frac{RP}{RP + FN} = \frac{RP}{P} \tag{2.33}$$

Eine wichtige Kombination der beiden Maße Genauigkeit und Sensitivität bildet das *F-Maß*, das nach Gleichung 2.34 als harmonisches Mittel der beiden Maße bestimmt werden kann.

$$F = \frac{2 \cdot Genauigkeit \cdot Sensitivität}{Genauigkeit + Sensitivität} \tag{2.34}$$

2.6.8 Wissensrepräsentation

Das durch das Data Mining erzeugte Wissen über die Daten muss schließlich geeignet aufbereitet werden, um dieses dem Benutzer visualisieren zu können oder um das Wissen geeignet zu repräsentieren, beispielsweise um dieses maschinell verarbeiten zu können. Für diese Arbeit ist die Wissensrepräsentation von Klassifikationen wichtig, die im Folgenden anhand der Ausführungen in Han u. a. (2011) beschrieben wird.

Das Wissen der in Unterabschnitt 2.6.4 vorgestellten Klassifikation mit Entscheidungsbäumen sowie dessen Vereinfachungen kann auf mehrere Arten präsentiert werden. Die einfachste Art ist die direkte Darstellung des Entscheidungsbaums mit seinen Knoten und Ästen, wie beispielsweise in Abbildung 2.30. Aus einem Entscheidungsbaum lassen sich jedoch auch einfach Wenn-Dann-Regeln (engl. 'if-then rules') ableiten. Dazu wird für jeden möglichen Pfad vom Wurzelknoten zu einem der Blattknoten eine eigene Regel erstellt, d.h. die Anzahl der Regeln entspricht der Gesamtzahl der Blattknoten. Der Wenn-Teil wird durch eine logische Und-Verknüpfung der einzelnen Teilungskriterien entlang des Pfads aufgestellt. Der Dann-Teil enthält die prädizierte Klasse. Für den im Beispiel in Abbildung 2.30 aufgestellten Entscheidungsbaum lassen sich so die in Tabelle 2.16 beschriebenen Wenn-Dann-Regeln ableiten.

Die aus einem Entscheidungsbaum abgeleiteten Wenn-Dann-Regeln schließen sich gegenseitig aus (engl. 'mutually exclusive'), da für jedes Objekt nur genau ein Pfad zutrifft, der sich aus den Attributwerten des Objekts ableitet. Somit stehen die einzelnen Wenn-Dann-Regeln niemals in Konflikt zueinander. Zudem sind die Regeln vollständig (engl. 'exhaustive'), da für jede der möglichen Attribut-Wert-Kombinationen eines Objekts eine Klasse prädiziert wird.

Teilunkgskriterien		Prädizierte Klasse
Wenn Material = Stein	Und Seitenlänge $\leq 15\,\text{cm}$	Dann Klasse = Grenzpunkt
Wenn Material = Stein	Und Seitenlänge $> 15\,\text{cm}$	Dann Klasse = Trigonometrischer Punkt
Wenn Material = Metall	Und Ausrichtung = vertikal	Dann Klasse = Grenzpunkt
Wenn Material = Metall	Und Ausrichtung = horizontal	Dann Klasse = Höhenfestpunkt

Tabelle 2.16: Wenn-Dann-Regeln für einen einfachen Entscheidungsbaum zur Klassifikation von Vermessungspunkten

2.7 Parallelisierung

Zwei seit längerem beobachtbare Trends in der Informatik bilden die Motivation für die parallele Verarbeitung von Daten. Erstens steigt sowohl die Anzahl an verfügbaren Datensätzen als auch deren Umfang rapide an. Die *steigende Anzahl* lässt sich auf die einfache und weltweite Verfügbarkeit von Datensätzen über das Internet sowie die Verwendung eines einzelnen Datensatzes durch viele unterschiedliche Benutzer zurückführen. Der *steigende Umfang* basiert auf der Verfügbarkeit detaillierter Erfassungsmethoden, wie beispielsweise der schnellen Erfassung umfangreicher 3D Punktwolken durch Laserscanner mit mehreren Millionen von Punkten, sowie auf der Datenerfassung mit günstigen Sensoren durch viele freiwillige Benutzer (engl. 'crowd sourcing'), wie beispielsweise durch die in vielen Mobiltelefonen verbauten Global Positioning System (Gps) Sensoren. Zweitens steigt die *Komplexität der Algorithmen* beziehungsweise Operatoren an, die auf die Daten angewendet werden. Die steigende Komplexität lässt sich beispielsweise auf die Verwendung komplexerer Modelle zur Interpretation und Interaktion der Daten zurückführen.

In diesem Abschnitt werden einige allgemeine Aspekte der Parallelisierung vorgestellt, während in Abschnitt 7.1 speziell auf die Parallelisierung der Verarbeitung von Geodaten eingegangen wird. Umfassendere Informationen zur parallelen Programmierung finden sich beispielsweise in Rauber und Rünger (2012).

Daten- und Task-Parallelität

Die Konzepte der Daten- und Task-Parallelität bilden die Grundlage für die parallele Verarbeitung von Daten. Die Definition der beiden Konzepte nach Rauber und Rünger (2012) wird im Folgenden vorgestellt. Um die Berechnungen eines Programms oder Algorithmus zu parallelisieren, müssen diese in kleinste parallel ausführbare Einheiten, sogenannte *Tasks*, zerlegt werden und Abhängigkeiten zwischen diesen Tasks bestimmt werden. Ein Task kann dabei aus einer einzelnen Anweisung bestehen, kann aber ebenso auch einem kompletten Funktionsaufruf entsprechen. *Datenparallelität* liegt vor, wenn derselbe Task auf unterschiedliche Elemente einer Datenstruktur oder eines Datensatzes angewendet wird und die einzelnen Tasks dabei voneinander unabhängig sind. Ein Beispiel für die Datenparallelität ist die Addition von zwei Vektoren gleicher Größe, bei der die Addition der Elemente mit dem jeweils gleichen Index parallel erfolgen kann. Ein weiteres Beispiel ist die paarweise Bestimmung von Distanzen zwischen allen Vektoren eines Datensatzes. Dahingegen liegt *Task-Parallelität* vor, wenn unterschiedliche und voneinander unabhängige Tasks parallel ausgeführt werden. Task-Parallelität wird auch als Funktionsparallelität bezeichnet, da bei dieser Form der Parallelität Tasks oft kompletten Funktionsaufrufen entsprechen. Bei der Task-Parallelität können die einzelnen Tasks auf dieselben oder auf unterschiedliche Daten zugreifen. Ein Beispiel für die Task-Parallelität ist die parallele Bestimmung der Addition und Multiplikation zweier Vektoren. Die Daten- und Task-Parallelität können miteinander kombiniert werden, und so in einem Programm gleichzeitig oder ergänzend verwendet werden.

Abbildung der Tasks auf Prozesse und Prozessoren

Nach Rauber und Rünger (2012) werden die Tasks in zwei weiteren Schritten zuerst zu Prozessen zusammengefasst und schließlich auf physikalische Prozessoren abgebildet. Ein Prozess entspricht einem Kontrollfluss, der mehrere Tasks zusammenfasst, die dann von einem Prozessor sequentiell ausgeführt werden können. Um eine optimale Lastverteilung zu erreichen, werden bei der Zusammenfassung sowohl die Berechnungszeiten als auch notwendige Zugriffe auf gemeinsame Daten berücksichtigt. Die Zuordnung der einzelnen Tasks an die Prozesse wird auch als Scheduling bezeichnet.

Parallele Leistungsmaße

Die Parallelisierung von Programmen soll deren Laufzeit reduzieren und die Verarbeitung umfangreicherer Datensätze ermöglichen. Um den Effekt der Parallelisierung bewerten zu können werden Leistungsmaße verwendet,

die auf einem Vergleich der sequentiellen und parallelen Laufzeit eines Programms basieren. Im Folgenden werden einige Leistungsmaße entsprechend den Definitionen in Rauber und Rünger (2012) vorgestellt.

Die Laufzeit eines parallelen Programms setzt sich nach Rauber und Rünger (2012) aus vier Zeiten zusammen, die im Folgenden mit absteigendem Einfluss auf die Gesamtlaufzeit angegeben sind:

- Zeit für die Durchführung von lokalen Berechnungen auf jedem einzelnen Prozessor
- Zeit für den Austausch von Daten
- Wartezeiten, beispielsweise wegen ungleicher Lastverteilungen
- Zeit für die Synchronisation aller Prozessoren

Für den Vergleich der Laufzeiten müssen zuerst einige Definitionen getroffen werden. Sei $T_p(n)$ die Laufzeit eines parallelen Programms unter der Verwendung von p Prozessoren und einer Problemgröße beziehungsweise eines Datensatzes vom Umfang n. Sei $T^*(n)$ die Laufzeit des schnellsten sequentiellen Programms, dessen Ergebnis dem Ergebnis des parallelen Programms entspricht[18].

Der relative Geschwindigkeitsvorteil der Parallelisierung wird als *Speedup* bezeichnet und ergibt sich aus dem Verhältnis der Laufzeiten des parallelen und des entsprechenden sequentiellen Programms nach Gleichung 2.35. Nach Rauber und Rünger (2012) gilt für den Speedup in der Regel $S_p(n) < p$, d.h. es wird kein linearer Speedup mit $S_p(n) = p$ erreicht. Dies lässt sich auf den zusätzlichen Aufwand zum Austausch von Daten und zur Synchronisation zurückführen. Beispielsweise bedeutet ein Speedup von vier, dass das parallele Programm ein Viertel der Laufzeit des sequentiellen Programms aufweist. Für diese Verringerung der Laufzeit sind jedoch in der Regel mehr als vier Prozessoren notwendig.

$$S_p(n) = \frac{T^*(n)}{T_p(n)} \tag{2.35}$$

Der maximal theoretisch erreichbare Speedup hängt von drei Größen ab. Erstens ist in der Regel die Anzahl an verfügbaren Prozessoren p begrenzt. Zweitens spielt die Problemgröße n eine Rolle. Drittens lässt sich ein Programm in der Regel nicht komplett parallelisieren, d.h. es verbleibt immer ein sequentieller Anteil. Sequentielle Anteile entstehen beispielsweise durch die Synchronisation zwischen Prozessen oder durch den Zugriff auf gemeinsame Daten. Soll beispielsweise das Ergebnis eines parallelen Programms in eine einzelne Datei auf einer einzelnen Festplatte geschrieben werden, so wird die Geschwindigkeit des Schreibzugriffs nicht durch die Verwendung mehrerer Prozesse erhöht, da die maximale Schreibgeschwindigkeit physikalisch limitiert ist.

Das *Amdahlsche Gesetz* gibt den Einfluss der Laufzeit des sequentiellen Anteils auf den Speedup bei variabler Anzahl an verfügbaren Prozessoren und konstanter Problemgröße an. Sei f mit $0 \leq f \leq 1$ der konstante sequentielle Bruchteil eines Programms. Dann setzt sich die Laufzeit des parallelen Programms aus der Laufzeit des sequentiellen Anteils $f \cdot T^*(n)$ und der Laufzeit des parallelen Anteils zusammen. Diese wäre bei linearem Speedup nach Gleichung 2.35 $(1-f)/p \cdot T^*(n)$, ist aber in der Regel länger. Der maximal erreichbare Speedup ergibt sich dann durch das Amdahlsche Gesetz nach Gleichung 2.36.

$$S_p(n) = \frac{T^*(n)}{f \cdot T^*(n) + \frac{1-f}{p} \cdot T^*(n)} = \frac{1}{f + \frac{1-f}{p}} \tag{2.36}$$

Der Grenzwert des Speedups nach dem Amdahlschen Gesetz ergibt sich für $p \to \infty$ nach Gleichung 2.37.

$$\lim_{p\to\infty} S_p(n) = \frac{1}{f + \lim\limits_{p\to\infty} \frac{1-f}{p}} = \frac{1}{f} \tag{2.37}$$

Parallele Leistungsmaße: Beispiele

Im Amdahlschen Gesetz wird von einer festen Problemgröße n und einem konstanten sequentiellen Bruchteil f ausgegangen. Soll die Laufzeit des parallelen Programms verringert werden, dann kann der Speedup nur

[18] Nach Rauber und Rünger (2012) lässt sich der schnellste sequentielle Algorithmus oft nur schwierig ermitteln, da dieser unbekannt sein kann, die Verwendung eines parallelen Algorithmus erst ab einer bestimmten Problemgröße sinnvoll sein kann oder der sequentielle Algorithmus nur schwer zu implementieren ist. In der Praxis wird deshalb oft die sequentielle Ausführung des parallelen Algorithmus auf nur einem Prozessor als Vergleich herangezogen.

durch zwei Maßnahmen erhöht werden. Mit der ersten Maßnahme wird einfach die Anzahl der Prozessoren p erhöht und damit die Laufzeit des parallelen Anteils verringert. Je nach Größe des sequentiellen Bruchteils f entfaltet diese Maßnahme jedoch nur eine geringe Wirkung. Deutlich wird dies in Abbildung 2.35, in der der Speedup in Abhängigkeit vom sequentiellen Bruchteil f und der Anzahl der Prozessoren p dargestellt ist. Mit $f = 20\,\%$ beträgt der maximale Speedup 5, egal wie viele Prozessoren verwendet werden (durchgezogene Linie in Abbildung 2.35). Das Amdahlsche Gesetz zeigt demnach eine schnelle Sättigung des Speedups mit steigender Anzahl der Prozessoren auf, wenn der sequentielle Bruchteil f entsprechend hoch ist. Die zweite Maßnahme ist entsprechend komplexer, da diese auf die Verringerung des konstanten Bruchteils f zielt und damit grundlegende Änderungen im sequentiellen Teil eines Programms notwendig macht. Kann f beispielsweise von 20 % auf 10 % verringert werden, dann kann grob von einer Verdopplung des Speedup gesprochen werden.

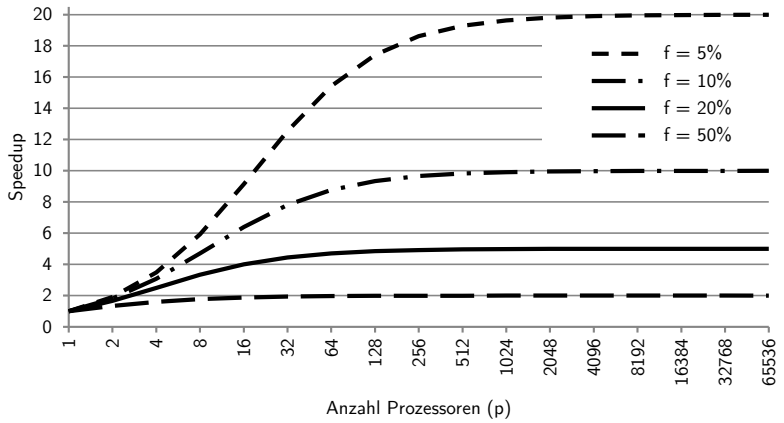


Abbildung 2.35: Speedup nach dem Amdahlschen Gesetz in Abhängigkeit vom sequentiellen Bruchteil f

Als Beispiel für den Effekt der ersten Maßnahme sind in Tabelle 2.17 die Laufzeiten und Werte für den Speedup eines Programms angegeben. Die Laufzeiten sind mit einer Genauigkeit von einer Minute und die Werte für den Speedup mit einer Genauigkeit von einer Nachkommastelle angegeben. Mit $f = 20\,\%$ beträgt der maximale Speedup 5. Bei einer sequentiellen Laufzeit von $T^*(n) = 60\,\text{min}$ hat das entsprechende parallele Programm mit $p = 8$ Prozessoren eine Laufzeit von $T_p(n) = 12\,\text{min} + (48\,\text{min}/8) = 18\,\text{min}$ mit einem Speedup von $S_p(n) = 3{,}3$. Eine Vervierfachung der Anzahl der Prozessoren auf $p = 32$ verringert die Laufzeit lediglich auf $T_p(n) = 14\,\text{min}$ mit einem Speedup von $S_p(n) = 4{,}4$. Bei $p = 512$ ist die Laufzeit $T_p(n) = 12\,\text{min}$ mit einem Speedup von $S_p(n) = 5$. Für das Beispiel erscheint demnach die Verwendung von mehr als 8 Prozessoren nur eingeschränkt sinnvoll.

Würde stattdessen der sequentielle Anteil auf $f = 10\,\%$ verringert, dann hätte das entsprechende parallele Programm mit $p = 32$ Prozessoren eine Laufzeit von nur $T_p(n) = 6\,\text{min} + 2\,\text{min} = 8\,\text{min}$ mit einem Speedup von $S_p(n) = 7{,}8$.

Parallelisierung	Anzahl Prozessoren	Größe Datensatz	Seq. Laufzeit [min]	Laufzeit seq. Anteil [min]	Laufzeit par. Anteil [min]	Par. Laufzeit [min]	Speedup
Sequentiell	1	n	60	12	48	60	1,0
Amdahl	8	n	60	12	6	18	3,3
	32				2	14	4,4
	512				0	12	5,0

Tabelle 2.17: Beispiel für die parallelen Leistungsmaße auf Basis des Amdahlschen Gesetzes

Architekturen zur Parallelisierung

Für die Parallelisierung eines Programms stehen für die Implementierung und Ausführung viele mögliche Architekturen zur Auswahl. Diese unterscheiden sich unter anderem bezüglich der verwendeten Soft- und Hardware sowie der Kommunikation und Synchronisation zwischen den parallelen Programmteilen. Detaillierte Informationen finden sich beispielsweise in Rauber und Rünger (2012). Im Folgenden werden kurz drei Architekturen vorgestellt, die Parallelisierung auch auf vergleichsweise kostengünstiger Hardware ermöglichen.

Die Parallelisierung kann durch die *gleichzeitige Nutzung mehrerer Prozessorkerne* der Central Processing Unit (CPU) eines Rechners erfolgen. Die als Mehrkernprozessoren (engl. 'multi-core prozessor') bezeichneten Prozessoren vereinen dabei mehrere voneinander unabhängige Prozessorkerne auf einem einzelnen Chip. Derzeit vereinen kostengünstige Mehrkernprozessoren in der Regel zwischen zwei und acht Kernen. Normale Desktop-Rechner verfügen meist auf ihrer Hauptplatine (engl. 'motherboard', 'mainboard') über einen einzelnen Sockel zur Aufnahme einer CPU, wohingegen die Hauptplatinen von Servern oft zwei oder mehr Sockel zur Aufnahme mehrerer CPUs anbieten.

Ebenso kann die Parallelisierung durch die *gleichzeitige Nutzung mehrerer Grafikprozessorkerne* der Graphic Processing Unit (GPU) eines Rechners erfolgen. Die Grafikkarte eines Rechners wird dabei nicht zur Erzeugung von Bildern für die Ausgabe auf einem Bildschirm, sondern für allgemeine Berechnungen genutzt. Derzeit verfügen Grafikkarten über mehrere Hundert Prozessoren, jedoch nur über eine begrenzte Menge an Arbeitsspeicher. Abhängig von der Aufgabenstellung liegt die Leistungsfähigkeit einer GPU bei vergleichbarem Preis teilweise weit über der Leistungsfähigkeit einer CPU, da Grafikkarten hochgradig auf Datenparallelität optimiert sind. Es lassen sich jedoch nicht alle Aufgabestellungen effizient auf einer Grafikkarte lösen, beispielsweise weil der verfügbare Arbeitsspeicher nicht ausreicht. Zur Programmierung auf einer Grafikkarte muss eine darauf ausgerichtete Programmiersprache verwendet werden, die teilweise spezifisch für die Grafikkarten eines Herstellers optimiert beziehungsweise ausgerichtet ist.

Die dritte Architektur zur Parallelisierung ist das *verteilte Rechnen* (engl. 'distributed computing') auf mehreren Rechnern, die über ein Netzwerk miteinander verbunden sind. Der wichtigste Vorteil des verteilten Rechnens ist die einfache Skalierbarkeit, indem einfach weitere Rechner zum Netzwerk hinzugefügt werden. Die Anzahl der Rechner kann dabei beispielsweise für einfache Aufgaben einstellig sein, kann aber genauso für umfangreiche oder eine Vielzahl an Aufgaben fünfstellig sein. Die wichtigsten Architekturen für das verteilte Rechnen sind Cluster, Grid und Cloud. In dieser Arbeit wird das verteilte Rechnen in einem Cluster genutzt (siehe Abschnitt 2.8). Die Architekturen Grid- und Cloud-Computing werden von Foster (2002) beziehungsweise Armbrust u. a. (2010) definiert und diskutiert. Auf Aspekte der Parallelisierung in einer Grid-Umgebung wird zudem in Werder und Krüger (2009) eingegangen.

Workflows und Orchestrierung

Für die Parallelisierung eines Programms ist auch dessen Unterteilung in logische aufeinander folgende Prozesse wichtig. Dadurch entsteht ein Workflow, dessen Zusammensetzung und Ablaufkontrolle als Orchestrierung bezeichnet werden. Diese Aspekte werden für die Datenintegration heterogener Datensätze in Werder (2010) diskutiert und im Folgenden verallgemeinert.

In Abbildung 2.36 sind drei mögliche Softwarearchitekturen für ein Programm dargestellt. In einer *monolithischen Softwarearchitektur* ist die gesamte Logik in einem einzelnen Programm enthalten. Änderungen oder Erweiterungen des Programms sind somit nur möglich, wenn der Quelltext des Programms verfügbar ist. In einer *Plug-in Softwarearchitektur* können kleine Programme, sogenannte Plug-ins, in ein größeres Programm bei Bedarf integriert werden. Diese Plug-ins können in den meisten Fällen nur in ein einziges Programm integriert werden. Die vielseitigste Softwarearchitektur ist die *Modularisierung*. Die Programmlogik wird dabei in kleine Module unterteilt. Der Programmfluss, d.h. die Abfolge der Module, als auch der Datenfluss werden von einer leichtgewichtigen Workflow-Komponente gesteuert. Die Modularisierung bietet zwei wichtige Vorteile gegenüber den beiden anderen vorgestellten Softwarearchitekturen. Erstens können die Module in anderen Programmen wiederverwendet werden. Sind die Module generisch genug, so können diese sogar in Programmen aus anderen Anwendungsgebieten eingesetzt werden. Zweitens können die Module bei kompatiblem Format der Ein- und Ausgabedaten einfach ausgetauscht werden, beispielsweise durch eine verbesserte Programmversion oder eine Version einer anderen Organisation.

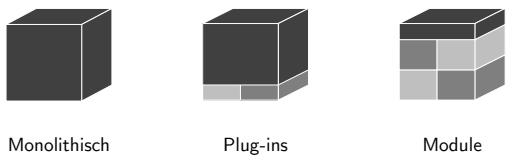


Abbildung 2.36: Mögliche Softwarearchitekturen für ein Programm

Die Modularisierung der Datenintegration nach Werder (2010) ist in Abbildung 2.37 dargestellt. Die Aufgabenstellung der Datenintegration wird in mehrere Module unterteilt, die dann zu einem Workflow zusammengestellt werden. Anhand des Workflows lassen sich mehrere Aspekte der Modularisierung diskutieren. Erstens können einzelne Module in einem Workflow mehrfach auftreten. Ein Beispiel dafür ist das Modul der Vorverarbeitung,

das sowohl auf die Referenz- als auch auf die Fachdaten angewendet wird. Zweitens ergibt sich aus dem Workflow explizit dessen Parallelisierung. So können die Referenz- und Fachdaten mit mehreren Modulen erst einmal unabhängig voneinander parallel bearbeitet werden, bevor diese zusammengeführt werden. Zudem muss die Ausführung des Moduls zur Bestimmung der Datenqualität sowohl auf die Ergebnisse der Prüfung der Datenintegrität als auch auf die Ergebnisse der Objektzuordnung warten. Drittens können, wie in Werder (2010) aufgezeigt, einzelne Module nochmals feingranularer unterteilt und in einem eigenen Workflow zusammengesetzt werden. Viertens können Module auch iterativ in einem Workflow aufgerufen werden, beispielsweise indem eine zuerst grobe Berechnung anschließend iterativ verfeinert wird.

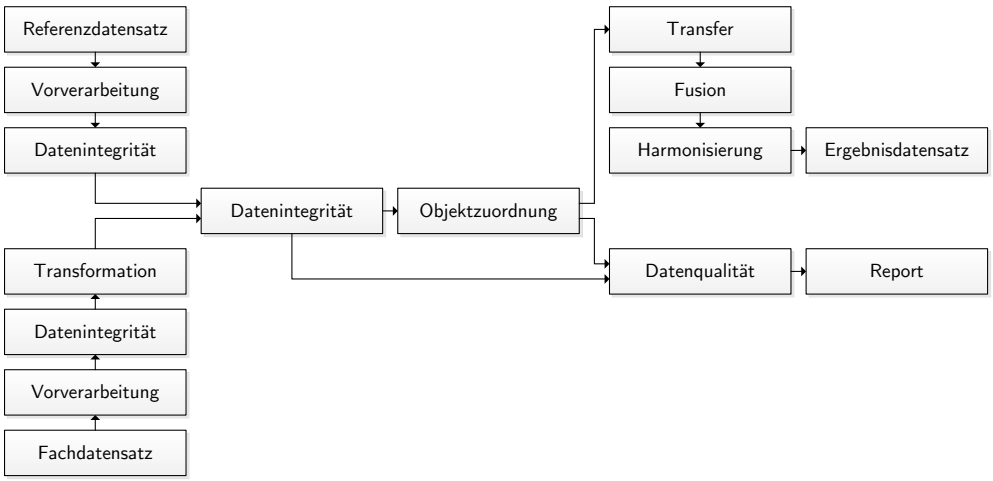


Abbildung 2.37: Workflow der Datenintegration nach Werder (2010)

Aufwand

Der Aufwand zur Parallelisierung existierender sequentieller Programme kann sehr unterschiedlich ausfallen. Rauber und Rünger (2012) unterscheiden dabei mehrere Komplexitätsstufen. Im einfachsten Fall wird das existierende Programm nicht geändert und die Parallelisierung wird implizit durch einen parallelisierenden Compiler durchgeführt. Dieser untersucht automatisch den Quelltext nach Abhängigkeiten in den Berechnungen und parallelisiert deren Ausführung. Im kompliziertesten Fall muss die komplette Parallelisierung inklusive der notwendigen Kommunikation und Synchronisation manuell im Quelltext des Programms umgesetzt werden. Gegebenenfalls muss dabei der Workflow angepasst oder Algorithmen ausgetauscht werden.

Eine Parallelisierung lohnt sich demnach nicht für jedes Programm. Für eine Abschätzung müssen sowohl der Gewinn als auch der Aufwand der Parallelisierung quantifiziert werden. Der *Gewinn* lässt sich durch die vorgestellten Metriken oder durch die Kosten für den Einsatz von Hard- und Software sowie Kosten für den Transport von Daten quantifizieren. Der *Aufwand* beziehungsweise die Investition lässt sich durch die benötigte Arbeitszeit zur Parallelisierung des Programms oder deren Kosten quantifizieren. In Anlehnung an das in der Finanzwelt verwendete Maß der Anlagenrendite beziehungsweise Investitionsrentabilität (engl. 'Return on Investment', kurz RoI) kann dann berechnet werden ob sich die Parallelisierung lohnt. Das Maß RoI wird nach Gleichung 2.38 als Verhältnis des Gewinns zur Investition berechnet, wobei ein gewisser Zeitraum zur Ermittlung beider Werte angesetzt wird (Feibel, 2003).

$$RoI\,[\%] = \frac{Gewinn\ oder\ Verlust}{Investition} \cdot 100 \tag{2.38}$$

Für einige Programme stellt sich nicht die Frage ob eine Parallelisierung notwendig ist, sondern nur wie diese umgesetzt kann. Dies ist der Fall wenn die Ressourcen eines einzelnen Computersystems nicht ausreichen um das Programm sequentiell abzuarbeiten, beispielsweise weil nicht genügend Arbeits- oder Festplattenspeicher zur Verfügung gestellt werden können. Mit dem Maß RoI kann dann beispielsweise berechnet werden, ob es sich lohnt zusätzliche Computersysteme oder Ressourcen anzukaufen oder anzumieten.

2.8 MapReduce

MapReduce ist ein eleganter Ansatz um die Prozessierung umfangreicher Datenmengen in einem Cluster, beziehungsweise bei angemieteten Servern in einer Cloud, zu parallelisieren. Durch die Verwendung von MapReduce werden Details der Parallelisierung, der Fehlertoleranz sowie der Daten- und Lastverteilung abstrahiert. Der Benutzer muss lediglich eine sequentielle Map-Funktion und eine sequentielle Reduce-Funktion implementieren, die dann vom MapReduce-Framework automatisch parallel ausgeführt werden. Die Eingabedaten werden parallel durch nebenläufige Tasks verarbeitet, welche unabhängig voneinandener ausgeführt werden. Diese Form der Parallelität wird als offensichtlich oder *hochgradig parallel* bezeichnet (engl. 'embarrassingly parallel', 'massively parallel') (Foster, 1995).

MapReduce wurde bei Google entwickelt um die Komplexität der eingesetzten Programme zur Verarbeitung umfangreicher Datenmengen zu vereinfachen. MapReduce verwendet zur Datenhaltung ein skalierbares und verteiltes Dateisystem, das auch bei Google entwickelt wurde. Mit Hadoop existiert eine quelloffene Implementierung von MapReduce und verteiltem Dateisystem.

Im Folgenden werden in Unterabschnitt 2.8.1 Google MapReduce und in Unterabschnitt 2.8.2 das verteilte Dateisystem von Google vorgestellt. Deren Architektur und Implementierung in Hadoop werden anschließend in Unterabschnitt 2.8.3 vorgestellt. In Unterabschnitt 2.8.4 wird die Serialisierung von Daten im Kontext von Hadoop beschrieben. Schließlich wird in Unterabschnitt 2.8.5 die Optimierung der Gesamtlaufzeit von MapReduce-Jobs diskutiert.

2.8.1 Google MapReduce

Nach Dean und Ghemawat (2004) ist *MapReduce* ein Programmiermodell und eine dazugehörige Implementierung für die Prozessierung und Erzeugung umfangreicher Datensätze. Im Folgenden werden sowohl das Modell als auch die Rahmenbedingungen der Implementierung entsprechend den Ausführungen in Dean und Ghemawat (2004) vorgestellt.

Ein Beispiel für die Verwendung von MapReduce bei Google ist das Indizierungssystem welches die Datenstrukturen für die Google-Websuche erzeugt. Das System prozessiert gespeicherte Webseiten mit einem Datenumfang von mehr als 20 TiB in fünf bis zehn aufeinanderfolgenden MapReduce-Durchläufen. Der Quelltext einer dieser Durchläufe konnte durch die Verwendung von MapReduce von 3800 auf 700 Zeilen reduziert werden und ist dadurch verständlicher und einfacher zu warten.

Programmiermodell

Der Name MapReduce lehnt sich an die Benennung der map und reduce Primitive funktionaler Programmiersprachen, wie beispielsweise Lisp, an[19]. Die Ein- und Ausgabedaten der Map- und Reduce-Funktionen sind Paare der Form (*Schlüssel, Wert*) (engl. $(key, value)$, abgekürzt als (k, v)) und Listen dieser Paare. Sowohl die Map- als auch die Reduce-Funktion müssen vom Benutzer definiert werden. Im Folgenden werden die Ein- und Ausgaben beider Funktionen zuerst vorgestellt und dann anhand eines Beispiels erläutert.

Die *Map-Funktion* erhält als Eingabe jeweils ein Paar (k_1, v_1), welches einem einzelnen Element der Eingabedaten entspricht. Die Ausgabe der Map-Funktion ist jeweils eine Liste von Paaren (k_2, v_2), die als Zwischenergebnis angesehen werden können. Unterschiedliche Indizes i bei den Schlüsseln k_i oder Werten v_i bedeuten dabei, dass die Parameter unterschiedliche Datentypen aufweisen können. Nachdem alle Eingabepaare prozessiert sind, gruppiert das MapReduce-Framework alle Werte v_2 mit demselben Schlüssel k_2 zu einer gemeinsamen Liste. Die *Reduce-Funktion* erhält als Eingabe jeweils einen Schlüssel und dessen gemeinsame Liste $(k_2, list(v_2))$. Die Ausgabe der Reduce-Funktion $(k_2, list(v_2))$ entspricht bezüglich des Formats deren Eingabe, wobei die Werte in der Eingabeliste jedoch durch die Reduce-Funktion vereint werden. Meist ist die Länge der Ausgabeliste

[19] In Lämmel (2008) sind detailliert die Unterschiede der beiden Modelle dargestellt. Für den Vergleich wird die funktionale Programmiersprache Haskell verwendet. Der sogenannte map-Kombinator wendet eine Funktion auf alle Elemente einer Liste an. Wird beispielsweise mit einem map-Kombinator eine Funktion zur Verdopplung der numerischen Werte auf alle Elemente einer Liste angewendet, dann wird mit dem Ausdruck `map ((*) 2) [1,2,3]` das Ergebnis `[2,4,6]` erzeugt. Der sogenannte foldl-Kombinator, der in etwa einem reduce in Lisp entspricht, wendet einen binäre Operation auf eine Liste an und 'faltet' diese zu einem Wert zusammen. Wird beispielsweise mit einem foldl-Kombinator die binäre Operation `+` mit dem Initialwert `0` auf eine Liste angewendet, dann wird mit dem Ausdruck `foldl (+) 0 [1,2,3]` das Ergebnis `6` erzeugt. Die 'Faltung' beziehungsweise die Reduktion kann durch das entsprechende Setzen von Klammern veranschaulicht werden `(((0 + 1) + 2) + 3)`.

einer Reduce-Funktion somit kleiner als deren Eingabeliste. In Tabelle 2.18 sind die Ein- und Ausgabedaten der beiden Funktionen nochmals zusammengefasst.

Funktion	Eingabe	Ausgabe
map	(k_1, v_1)	$list\,(k_2, v_2)$
reduce	$(k_2, list\,(v_2))$	$(k_2, list\,(v_2))$

Funktion	Eingabe	Ausgabe
map	(*Zeilennummer*, *Textzeile*)	$list\,(Wort, 1)$
reduce	$(Wort, list\,(1, \ldots, 1))$	(*Wort*, *Anzahl*)

Tabelle 2.18: Schlüssel und Werte der Google Map- und Reduce-Funktionen sowie Beispiel für MapReduce

Als Beispiel für den Datenfluss in Map- und Reduce-Funktionen ist zudem in Tabelle 2.18 das Zählen von absoluten Worthäufigkeiten in einem Textdokument dargestellt. Die Map-Funktion erhält als Eingabe jeweils für jede Zeile des Dokuments ein Paar (*Zeilennummer*, *Textzeile*). Der Schlüssel wird nicht weiter verwendet. Für jedes Wort in der Zeile wird als Ausgabe ein Paar $(Wort, 1)$ erzeugt, wobei mehrfach vorkommende Worte entsprechend mehrere Einträge in der Liste erzeugen. Das MapReduce-Framework gruppiert nun alle Werte mit demselben Wort zu einer gemeinsamen Liste, die in diesem Beispiel einer Menge der Werte 1 entspricht. Die Reduce-Funktion erhält jeweils ein Paar $(Wort, 1, \ldots, 1)$, summiert die Werte auf und gibt schließlich ein Paar (*Wort*, *Anzahl*) aus, nämlich das Wort und dessen absolute Worthäufigkeit im Dokument.

Implementierung

Nach Dean und Ghemawat (2004) sind viele unterschiedliche Implementierungen des MapReduce-Modells möglich, die sich an der gegebenen Infrastruktur ausrichten sollten. Die Infrastruktur bei Google besteht aus Clustern mit mehreren hundert oder tausend Servern, die bezüglich ihrer Hardware auf einfache und günstige Massenprodukte (engl. 'commodity') setzen. Die Rechenleistung wird also nicht durch wenige besonders leistungsfähige Server bereitgestellt, sondern durch eine Vielzahl von Servern. Durch die Verwendung günstiger Massenprodukte erhöht sich jedoch auch die Ausfallwahrscheinlichkeit einzelner Hardwarekomponenten und damit kompletter Server.

MapReduce ist bei Google als Bibliothek in der Programmiersprache C++ implementiert. Alle Schlüssel und Werte sind vom Datentyp String und müssen durch den Benutzer in den Map- und Reduce-Funktionen explizit in die gewünschten Datentypen umgewandelt werden (engl. 'cast').

2.8.2 Google File System

Nach Ghemawat u. a. (2003) ist das Google File System (GFS) ein skalierbares und verteiltes Dateisystem, das für verteilte datenintensive Anwendungen mit vielen Benutzern konzipiert ist. Das Dateisystem wird auf den in Unterabschnitt 2.8.1 beschriebenen Clustern betrieben. Es bildet zudem die Basis für die Datenhaltung in der MapReduce-Implementierung von Google. Im Folgenden werden einige Anforderungen an das GFS und deren Umsetzung entsprechend den Ausführungen in Ghemawat u. a. (2003) vorgestellt.

Die erste Anforderung ist eine möglichst hohe *Performanz* des Dateisystems. Dabei werden sowohl die Eigenschaften der Daten als auch die Zugriffsmuster auf diese Daten berücksichtigt. Das GFS ist für die Speicherung von vielen großen Dateien optimiert, die üblicherweise größer als 100 MiB sind und oft mehrere GiB umfassen. Große Datenmengen werden meist fortlaufend als Datenstrom (engl. 'streaming read') ausgelesen, wohin durch wahlfreie Zugriffe (engl. 'random read') meist nur kleine Datenmengen ausgelesen werden. Werden Daten geschrieben, dann werden meist komplette Dateien sequentiell geschrieben oder Daten an bereits existierende Dateien angehängt (engl. 'append').

Aus diesen Gründen werden im GFS Dateien in 64 MiB große Blöcke (engl. 'chunk') zerlegt, was bedeutend mehr ist als die üblicherweise in Dateisystemen verwendeten Blockgrößen von einigen KiB. Die Blöcke im GFS werden wiederum als einzelne Dateien im darunterliegenden Dateisystem gespeichert. Ist ein Block nur zur Hälfte mit Daten gefüllt, dann belegt dieser entsprechend nur die Hälfte an physikalischem Speicherplatz (engl. 'lazy allocation'). Die 64 MiB großen Blöcke bieten mehrere wichtige Vorteile. Erstens müssen weniger Informationen über die Position der Blöcke an Clients übertragen werden. Zweitens werden durch die großen Blöcke Netzwerkverbindungen effizienter ausgelastet, da weniger Verbindungen erzeugt werden müssen. Drittens wird der Umfang der Metadaten erheblich verringert, so dass diese selbst für viele Dateien performant vorgehalten werden können.

Die zweite Anforderung an das GFS ist die *Skalierbarkeit*. Das System besteht aus einem Master und mehreren sogenannten Chunk-Servern. Der Master verwaltet die Metainformationen zum Dateisystem und kontrolliert

die Chunk-Server, welche die Daten speichern. Soweit möglich werden zur Entlastung des Masters alle Datentransfers direkt zwischen den Clients und den Chunk-Servern beziehungsweise zwischen den Chunk-Servern abgewickelt. Werden neue Chunk-Server hinzugefügt, so melden sich diese beim Master an und werden dann bei Bedarf automatisch mit Daten befüllt.

Die dritte Anforderung ist die *Zuverlässigkeit*. Fehler in der Datenspeicherung werden durch die Erzeugung und Überprüfung von Prüfsummen (engl. 'check sum') vermieden beziehungsweise erkannt. Jede Datei wird zudem auf mehreren Chunk-Servern, im GFS üblicherweise dreifach, gespeichert. Durch diese Replikation kann der Ausfall einzelner Chunk-Server abgefangen werden. Zudem steigert die Replikation die Performanz, da angeforderte Daten von in der Netzwerktopologie naheliegenden Servern abgefragt werden können.

Die vierte Anforderung an das GFS ist die *Verfügbarkeit*. Der Master überprüft periodisch den Status der Chunk-Server und gleicht Ausfälle durch Replikation der Daten auf andere Chunk-Server aus. Außerdem werden die Informationen des Masters redundant auf zusätzliche Server gesichert.

2.8.3 Hadoop

Hadoop[20] ist eine Implementierung von Google MapReduce (siehe Unterabschnitt 2.8.1) und des Google File Systems (siehe Unterabschnitt 2.8.2). Hadoop entwickelte sich aus dem Projekt Nutch, einer quelloffenen Implementierung einer Suchmaschine für Webseiten. Die Entwicklungen an Nutch begannen 2002, jedoch wurde die Skalierbarkeit der Suchmaschine erst durch die Implementierung des Google File Systems sowie von Google MapReduce entscheidend verbessert. Anfang 2006 wurden diese Teile dann in das Hadoop-Projekt ausgegliedert und die Entwicklung unter anderem durch ein eigenes Entwicklerteam bei Yahoo! forciert. Seit 2008 ist Hadoop ein sogenanntes Top-Level-Projekt der Apache Software Foundation. Mehr Details zur geschichtlichen Entwicklung von Hadoop sind beispielsweise in White (2012) wiedergegeben.

Nach eigenen Angaben in Hadoop (2012) wird Hadoop von etlichen Firmen und Forschungseinrichtungen produktiv eingesetzt. Prominente Beispiele sind Yahoo! und Facebook. Yahoo! verfügt über mehrere Hadoop-Cluster mit insgesamt mehr als 40 000 Servern, welche unter anderem für die Websuche, die Planung und Optimierung von Werbeeinblendungen sowie für die Filterung unerwünschter Email-Nachrichten eingesetzt werden. Facebook verwendet Hadoop zur Speicherung der Nutzerdaten mit einem Umfang von mehr als 100 Petabyte (Facebook, 2012) und für Analysen und maschinelles Lernen auf diesen Daten.

Im Folgenden wird das Programmiermodell sowie die Implementierung der derzeit aktuellen Versionsreihe 1.x von Hadoop vorgestellt. Die Ausführungen lehnen sich dabei an die Publikationen von White (2012) bezüglich Hadoop sowie von Ghemawat u. a. (2003) und Dean und Ghemawat (2004) bezüglich des Google File Systems und Google MapReduce an.

Programmiermodell

Bezüglich des Programmiermodells von MapReduce weicht Hadoop von der ursprünglichen Definition nach Dean und Ghemawat (2004) ab. In Hadoop besteht die Ausgabe der Reduce-Funktion jeweils aus einer Liste von Paaren (k_3, v_3), anstatt wie in der Definition von Google aus einem einzelnen Paar $(k_2, \mathit{list}(v_2))$. In Tabelle 2.19 sind die Ein- und Ausgabedaten der Hadoop Map- und Reduce-Funktion nochmals zusammengefasst.

Funktion	Eingabe	Ausgabe
map	(k_1, v_1)	$\mathit{list}(k_2, v_2)$
reduce	$(k_2, \mathit{list}(v_2))$	$\mathit{list}(k_3, v_3)$

Tabelle 2.19: Schlüssel und Werte der Hadoop Map- und Reduce-Funktionen

Diese Abweichung entspricht keiner Beschränkung, sondern einer Erweiterung des Programmiermodells. Der Beweis dieser Aussage lässt sich in zwei Teile gliedern. Lässt sich die Ausgabe der Reduce-Funktion nach der Definition von Google mit der Ausgabe nach der Definition von Hadoop abbilden, dann stellt die Abweichung keine Beschränkung dar. Enthält die Liste $\mathit{list}(k_3, v_3)$ nur ein Element, dann ist das Ergebnis ein Paar (k_3, v_3). Der Schlüssel k_3 kann vom selben Datentyp wie der Schlüssel k_2 sein. Der Wert v_3 kann in Hadoop einem beliebigen Objekt entsprechen, also auch einer Liste $\mathit{list}(v_2)$. Die Ausgabe lässt sich demnach abbilden und

[20] Doug Cutting, der Gründer des Hadoop-Projektes, benannte das Projekt nach der Bezeichnung seines Sohnes für dessen gelben Stoffelefanten.

stellt somit keine Beschränkung dar. Die Abweichung stellt eine Erweiterung dar, wenn sich mit der Definition von Hadoop mächtigere Ausgaben erstellen lassen als mit der Definition von Google. Sowohl der Schlüssel k_3 als auch der Wert v_3 können von einem beliebigen Datentyp sein und sind nicht auf die Datentypen der Eingabe der Reduce-Funktion beschränkt. Zudem kann eine beliebige Anzahl an Paaren in der Ausgabeliste enthalten sein. Die Abweichung ist demnach eine Erweiterung.

Implementierung: Architektur

Die Architektur eines MapReduce-Clusters basiert auf zwei Prinzipien. Das erste Prinzip ist die Klassifizierung von Servern in *Master und Slaves*. Ein einzelner Master überwacht viele ihm zugeordnete Slaves und weist ihnen Aufgaben zu. Der Master erkennt Ausfälle einzelner Server und reagiert dynamisch auf diese, indem er die betroffenen MapReduce-Tasks und Daten neu verteilt. Werden neue Slaves zum Netzwerk hinzugefügt, so melden sich diese beim Master an und werden ab diesem Zeitpunkt bei der Verteilung von MapReduce-Tasks und Daten berücksichtigt.

Im Kontext von MapReduce erfolgt jede Kommunikation ausschließlich zwischen dem Master und den Slaves. Die einzelnen MapReduce-Tasks auf einem Slave benötigen während ihrer Laufzeit keine Synchronisation mit dem Zustand anderer Slaves. Der Master erkennt wenn ein Task abgeschlossen ist und stößt entsprechend die nachlaufenden Prozesse an. Im Kontext des verteilten Dateisystems erfolgt die Kommunikation auch zwischen den Slaves, jedoch nur aufgrund von Informationen die der Master bereitstellt. So wird die Replikation von Daten linear in einer Kette von Slaves durchgeführt. Bei einem Replikationsfaktor von drei repliziert beispielsweise der erste Slave die Daten direkt zu einem zweiten Slave, welcher nachdem er die Daten vollständig erhalten hat diese wiederum zu einem dritten Slave repliziert. Durch das Verfahren der direkten Weitergabe von Daten zwischen den Slaves wird die Bandbreite des Netzwerkes effizient ausgenutzt. Die Daten müssen somit nicht über den Master transportiert werden, da dieser das Dateisystem nur verwaltet, selbst aber keine Dateien besitzt.

Die entscheidende Schwachstelle (engl. 'single point of failure') der Master-Slave-Architektur ist der einzelne Master. Fällt der Master aus, so ist das gesamte System bis zur Wiederherstellung des Masters oder bis zur Bereitstellung eines neuen Masters funktionsunfähig. Dieser Schwachstelle wird mit zwei Maßnahmen begegnet. Die erste Maßnahme ist die regelmäßige Sicherung der Informationen des Masters auf andere Server, um mit diesen Informationen schnellstmöglich ein neuen Master bereitstellen zu können[21]. Die zweite Maßnahme ist der Einsatz hochverfügbarer und redundant ausgelegter Hardware für den Master, um beispielsweise beim Ausfall eines Netzteils ohne Unterbrechung auf das zweite Netzteil im Server umschalten zu können. Die Hardware des Masters unterscheidet sich also signifikant von der Hardware der Slaves.

Das zweite Prinzip der Architektur eines MapReduce-Clusters ist die Berücksichtigung der *Lokalität* (engl. 'locality'). Da die Daten sehr viel größer sind als die MapReduce-Tasks welche die Daten verarbeiten, werden die MapReduce-Tasks unter anderem so verteilt, dass möglichst wenig Daten übertragen werden müssen. Dieses Paradigma wird auch als engl. 'move-code-to-data' bezeichnet. Die Daten werden üblicherweise dreifach repliziert, deshalb kann der Master einen der drei Slaves wählen, auf dem die Daten auf den lokalen Festplatten gespeichert sind. Ist keiner dieser Slaves für die Ausführung von MapReduce-Tasks verfügbar, so wird ein in der Netzwerktopologie naheliegender Slave gewählt, auf den die Daten dann übertragen werden.

In Abbildung 2.38 sind die MapReduce-Architekturen von Google und Hadoop mit ihren jeweiligen Bezeichnungen gegenübergestellt. Im Folgenden wird die MapReduce-Architektur von Hadoop vorgestellt und die entsprechenden Bezeichnungen der Google MapReduce-Architektur sind jeweils in Klammern angegeben. Das verteilte Dateisystem besteht aus einem einzelnen Master, der *NameNode* (Master), und vielen Slaves, den *DataNodes* (Chunkserver). Die DataNodes speichern die Daten auf ihren lokalen Festplatten, während die NameNode das Dateisystem verwaltet. Die Informationen der NameNode werden periodisch auf einen *Secondary NameNode* (im GFS nicht benannt) gesichert. Das MapReduce-Framework besteht aus einem einzelnen Master, dem *JobTracker* (Master), und vielen Slaves, den *Task-Trackern* (Worker). Die Task-Tracker führen MapReduce-Tasks aus, während der JobTracker diese verwaltet. Aufgrund des Prinzips der Lokalität hat ein einzelner Slave-Server gleichzeitig zwei Rollen, nämlich die einer DataNode und die eines TaskTrackers. Der Server kann dadurch MapReduce-Tasks auf den lokal verfügbaren oder lokal bereitgestellten Daten ausführen.

Implementierung: Verteiltes Dateisystem

Das Hadoop Distributed File System (HDFS) ist das native Dateisystem von Hadoop. Es werden jedoch von Hadoop auch andere Dateisysteme unterstützt, wie beispielsweise das File Transfer Protocol. Das verteilte

[21] Dabei entsteht zwangsläufig ein geringer Informationsverlust, da Sicherungen nur zu fest definierten Zeitpunkten erfolgen. Alle Informationen die seit der letzten Sicherung beim Master angefallen sind gehen unwiederbringlich verloren.

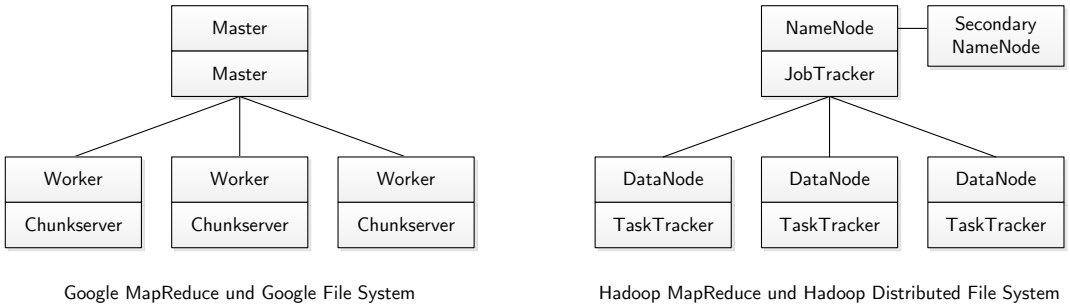


Abbildung 2.38: MapReduce-Architekturen von Google und Hadoop

Dateisystem ist eine Implementierung des in Unterabschnitt 2.8.2 beschriebenen GFS. Im Folgenden wird der Zugriff auf das Dateisystem sowie die Erstellung von Blöcken im Dateisystem vorgestellt. Weitere Details zum HDFS sind unter anderem in White (2012) beschrieben.

Der Zugriff auf das Dateisystem kann über mehrere Varianten erfolgen. Über die Kommandozeile können Dateien zwischen dem lokalen Dateisystem des Benutzers und dem HDFS ausgetauscht werden. Durch die Angabe zusätzlicher Parameter kann beispielsweise der Replikationsfaktor explizit für einzelne zu kopierende Dateien festgelegt werden. Innerhalb eines MapReduce-Programms können mit der bereitgestellten Programmierschnittstelle in der Programmiersprache Java Dateien gelesen und geschrieben werden. Der Zugriff auf das HDFS kann jedoch auch über das Hypertext Transfer Protocol (HTTP) sowie über eine Bibliothek in der Programmiersprache C erfolgen.

Ebenso wie das GFS setzt auch das HDFS auf große Blöcke für die Speicherung von Daten. Die Standardgröße von Blöcken ist im HDFS auf 64 MiB voreingestellt. Diese Einstellung kann jedoch global für das gesamte Dateisystem oder für jede einzelne Datei separat definiert werden. Durch die Speicherung von Dateien in Blöcken können außerdem Dateien gespeichert werden, die größer sind als jede einzelne Festplatte im Cluster.

Einige wichtige Aspekte der Verteilung von Dateien auf Blöcke sind in den Beispielen in Abbildung 2.39 zusammengefasst. Die Datei A hat eine Größe von 128 MiB und wird entsprechend für die Speicherung im HDFS automatisch in zwei jeweils 64 MiB große Blöcke $A.1$ und $A.2$ aufgeteilt. Mit einem Replikationsfaktor von drei wird jeder Block exakt dreimal auf den verfügbaren DataNodes im Cluster gespeichert, wobei das HDFS automatisch die Lokalität und die Auslastung der einzelnen DataNodes berücksichtigt. Die Datei B hat ebenso eine Größe von 128 MiB, jedoch wird beim kopieren der Datei in das HDFS vom Benutzer eine Blockgröße von lediglich 32 MiB vorgegeben. Dadurch werden im verteilten Dateisystem vier Blöcke $B.1$ bis $B.4$ erzeugt. Die Speicherung vieler kleiner Dateien im HDFS ist ineffizient, wie die fünf Dateien C bis G aufzeigen, da jede einzelne Datei einen gesamten Block im HDFS belegt. Dies führt beispielsweise bei der Speicherung vieler kleiner Bilddateien zu einem starken Anstieg der Metadaten, die dann von der NameNode verwaltet werden müssen. Als Alternative können zur Steigerung der Effizienz viele kleine Dateien in einem Containerformat zusammengefasst werden, wie die fünf Dateien H bis L aufzeigen. Durch das befüllen des Containers, einer sogenannten Sequenzdatei (engl. 'sequence file'), entstehen weniger und größere Blöcke im HDFS.

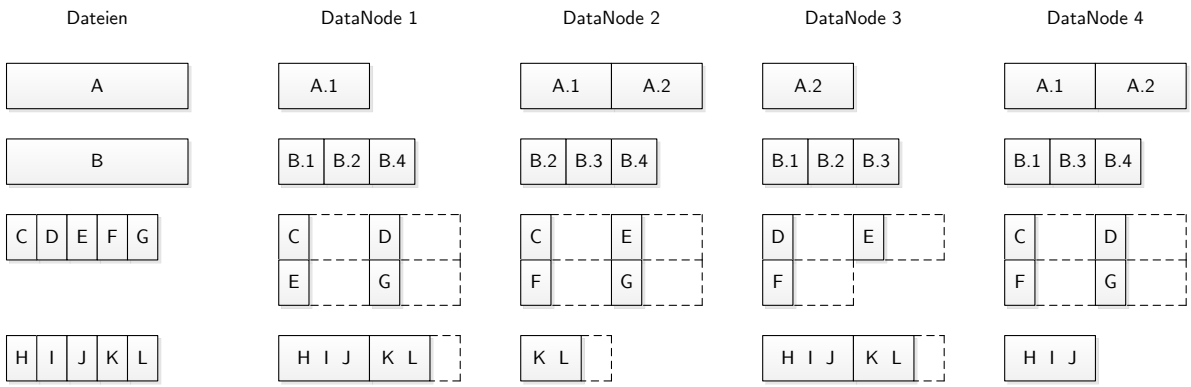


Abbildung 2.39: Beispiel für die Teilung von Dateien in Blöcke des Hadoop Distributed File Systems

Implementierung: MapReduce

Hadoop MapReduce ist eine Implementierung des Google MapReduce Programmiermodells und lehnt sich an dessen Implementierung bei Google an (siehe Unterabschnitt 2.8.1). Hadoop ist jedoch in der Programmiersprache Java implementiert.

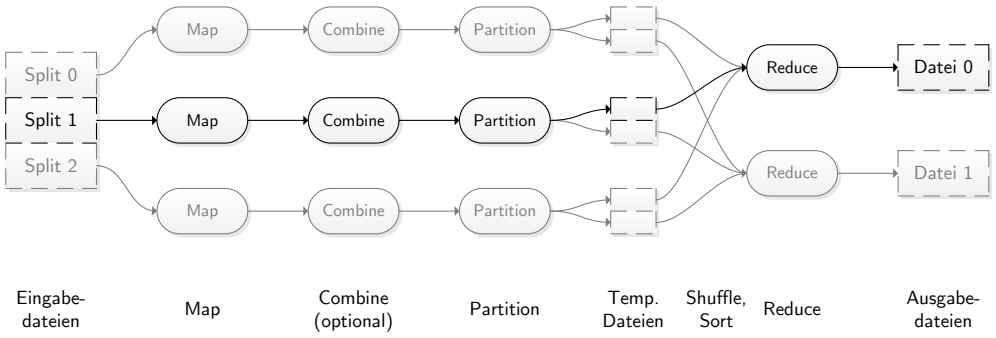


Abbildung 2.40: Workflow MapReduce

Der in Abbildung 2.40 dargestellte Workflow von Hadoop MapReduce lässt sich in fünf bis sechs aufeinanderfolgende Schritte gliedern:

1. *Unterteilung der Eingabedateien*: Einem MapReduce-Job können als Eingabedaten einzelne Dateien oder alle Dateien eines Verzeichnisses im HDFS übergeben werden. Diese werden automatisch in sogenannte Splits unterteilt. In der Regel entspricht ein Split genau einem Block im HDFS. Jedoch können Eingabeformate auch selbst implementiert und so die Größen eines Splits angepasst werden. Beispielsweise existiert für Textdateien bereits ein Eingabeformat, bei dem die Anzahl der Zeilen pro Split angegeben werden kann.

   Werden zu viele Splits erzeugt, dann führt dies zu einem erheblichen Mehraufwand für die Erzeugung der Splits und die Verwaltung der Map-Tasks. Werden zu wenige Splits erzeugt, dann wird gegebenenfalls der Cluster nicht optimal ausgelastet, da wenige Server die gesamte Last schultern müssen.

2. *Map*: Für jeden der insgesamt M Splits wird ein eigener Map-Task erzeugt. Nach dem Prinzip der Lokalität wird ein Map-Task bevorzugt auf demjenigen DataNode-Server erzeugt, auf dem auch die Daten des Splits liegen.

   Die Map-Funktion erhält jeweils ein Datenelement (engl. 'record') beziehungsweise Paar (k_1, v_1) des Splits als Eingabe und gibt jeweils eine Liste von Paaren (k_2, v_2) als Ergebnis aus. Entgegen der generischen Implementierung von Google sind die Datentypen der Schlüssel und Werte in Hadoop nicht vom Typ String. Für primitive Datentypen in Java existieren Wrapper, die spezielle Funktionalitäten zur Serialisierung und zu Binärvergleichen bereitstellen. Beispielsweise entspricht einem Integer in Java der IntWritable-Datentyp in Hadoop. Eigene Datentypen können einfach durch die Implementierung einer Schnittstelle (engl. 'interface') erstellt werden. Durch diesen Ansatz werden weniger manuelle Umwandlungen (engl. 'cast') benötigt und eine größere Typsicherheit gewährleistet.

3. *Combine (optional)*: Der Combine-Schritt ist optional und auch nicht für alle individuellen MapReduce-Workflows sinnvoll. Er entspricht einem Reduce (siehe sechster Schritt), der jedoch lokal die Ergebnisdaten eines einzelnen Map-Tasks zusammenfasst. Durch diese Optimierung kann die zu übertragende Datenmenge verringert werden.

4. *Partition*: Während die Anzahl M der Map-Tasks automatisch aus der Anzahl der Splits abgeleitet wird, muss die Anzahl R der Reduce-Tasks explizit durch den Benutzer angegeben werden. Das Ziel der Partitionierung ist in der Regel die gleichmäßige Verteilung der Ausgaben aller M Map-Tasks auf die R Reduce-Tasks. In der Standardimplementierung werden die Schlüssel k_2 anhand ihrer Hashwerte durch eine Modulo-Operation in R Partitionen unterteilt. Durch die Implementierung einer eigenen Partitionierungsfunktion kann anhand des Schlüssels und Werts jedes Paars (k_2, v_2) die Nummer der zugehörigen Partition festgelegt werden.

   Jede Partition wird als temporäre Datei auf die lokale Festplatte des Servers geschrieben, auf dem der Map-Task ausgeführt wird.

5. *Shuffle und Sort*: Durch den Shuffle-Schritt werden die einzelnen Partitionen der Map-Tasks den Reduce-Tasks zugewiesen[22]. Ein Reduce-Task kopiert parallel die temporären Dateien abgeschlossener Map-Tasks auf die lokale Festplatte beziehungsweise in den Arbeitsspeicher des Servers, auf dem der Reduce-Task ausgeführt wird. Die einzelnen Dateien werden anschließend zusammengeführt. Dabei wird sichergestellt, dass die Paare $(k_2, list(v_2))$ nach ihrem Schlüssel sortiert sind.

6. *Reduce*: Für jede der R Partitionen wird ein eigener Reduce-Task erzeugt.

   Die Reduce-Funktion erhält jeweils ein Paar $(k_2, list(v_2))$ als Eingabe und gibt jeweils eine Liste von Paaren (k_3, v_3) als Ergebnis aus. Die Datentypen sind wie bei der Map-Funktion auch hier typisiert.

   Jeder Reduce-Task schreibt eine eigene Ausgabedatei in das HDFS. Die Dateien können anschließend von weiteren MapReduce-Tasks verarbeitet oder von dort auf das lokale System des Benutzers kopiert werden. Wird als Ergebnis des MapReduce-Workflows eine einzelne Datei gewünscht, so kann entweder die Anzahl der Reduce-Tasks auf eins festgelegt werden oder die einzelnen Ausgabedateien einem zusätzlichen Schritt zusammengeführt werden.

Ein MapReduce-Workflow kann wie in Abbildung 2.40 aus Map- und Reduce-Tasks bestehen. Falls lediglich Daten parallel prozessiert werden sollen, dann kann der Workflow jedoch auch nur aus Map-Tasks bestehen.

Da Hadoop in *Java* implementiert ist, ist die primäre Programmiersprache für die Entwicklung von MapReduce-Programmen mit Hadoop entsprechend Java. Jedoch können unter Verwendung des Hadoop-Frameworks auch mit anderen Programmiersprachen MapReduce-Programme entwickelt werden. Durch das sogenannte 'Streaming' können alle Programmiersprachen, die aus der Standardausgabe lesen und in die Standardausgabe schreiben können für die Entwicklung von MapReduce-Workflows verwendet werden. Zu diesen Sprache zählen beispielsweise nahezu alle Skriptsprachen, wie Python oder Perl. Durch sogenannte 'Pipes' stellt Hadoop zudem eine direkte Schnittstelle für die Programmiersprache C++ bereit.

Hadoop-Anwendungen mit Raumbezug

Umfangreiche Daten mit Raumbezug und komplexe Analysen auf diesen Daten bilden eine sinnvolle Anwendung für das Hadoop-Framework. Die folgende Auflistung gibt einen Überblick über einige Typen von räumlichen Analysen auf Vektordaten, für die das Hadoop-Framework eingesetzt wird. Die Auflistung enthält dabei auch Referenzen auf Arbeiten in denen diese Analysen beschrieben und verwendet werden.

- *Selektion*: Auswahl von Objekten anhand räumlicher und nicht-räumlicher Filterkriterien – Wang u. a. (2011), Aji u. a. (2012)
- *Aggregation und Statistik*: Räumliche Aggregation von Objekten und Berechnung statistischer Werte für aggregierte Objekte wie beispielsweise Summe oder Mittelwert von Attributen – Wang u. a. (2011)
- *Objektzuordnung und Fusion zweier Datensätze*: Gegenseitige Objektzuordnung beziehungsweise Fusion zweier Datensätze anhand räumlicher Kriterien wie beispielsweise Flächenschnitt – Zhang u. a. (2009), Wang u. a. (2010), Wang u. a. (2011), Aji u. a. (2012)
- *Räumliche Nähe von Objekten*: Bestimmung der räumlichen Nachbarn von Objekten – Wang u. a. (2010), Aji u. a. (2012)

2.8.4 Serialisierung mit Avro

An die Datentypen der Schlüssel und Werte, die als Ein- und Ausgabe der Map- und Reduce-Funktion dienen, werden durch das Hadoop-Framework bestimmte Anforderungen gestellt. Die verwendeten Datentypen müssen die Serialisierung von Instanzen unterstützen, d.h. eine Instanz eines Objekts muss als binäre Repräsentation gespeichert und wieder geladen werden können. Diese Anforderung ist essentiell für die effiziente Übertragung der Daten im Hadoop-Framework, da die Daten durch die binäre Serialisierung kompakt abgebildet werden können. Die in Java eingebaute Serialisierung von Objekten erfüllt jedoch nicht die Anforderungen des Hadoop-Frameworks. Nach White (2012) ist diese nicht ausreichend kompakt, schnell, erweiterbar und interoperabel.

[22] Der engl. Begriff 'shuffle' kann im Kontext von MapReduce mit dem mischen von Spielkarten assoziiert werden. Die Spielkarten entsprechen den einzelnen Partitionen der M Map-Tasks. Diese werden jedoch nicht zufällig gemischt, sondern entsprechend der Partitionierungsregel auf R Stapel aufgeteilt und anschließend den jeweiligen Reduce-Tasks zugeordnet. Weitere Details zum Shuffle und zur Sortierung sind in White (2012) ausgeführt.

Eine weitere Anforderung an die Datentypen ist die Unterstützung von Binärvergleichen. Zwei binär repräsentierte Instanzen eines Objekts können somit direkt binär verglichen werden. Damit wird die Performanz bei der Verarbeitung von Objekten erhöht, beispielsweise bei der Sortierung der Paare $(k_2, list(v_2))$ nach ihrem Schlüssel, die vor der Ausführung der Reduce-Funktion erfolgt.

Wie in Unterabschnitt 2.8.3 bei der Beschreibung des MapReduce-Workflows ausgeführt ist, werden durch das Hadoop-Framework Wrapper für primitive Datentypen in Java bereitgestellt. Zudem können beliebige eigene Datentypen erstellt werden. Diese müssen jedoch eine entsprechende Schnittstelle implementieren, welche die Serialisierbarkeit und Binärvergleiche für Instanzen der Datentypen bereitstellt.

Die Einschränkungen dieses Ansatzes werden im Folgenden anhand eines Beispiels zur Messung von Umweltwerten an einer Messstation erläutert. Eine einzelne Messung umfasst die in Programm 2.7 angeführten Attribute, die beispielsweise den Mittelwert des Messintervalls umfassen. Das Konzept der Messung kann als eigener Datentyp beziehungsweise als eigene Klasse in Java implementiert werden, wobei die entsprechende Schnittstelle für die Serialisierbarkeit und Binärvergleiche zusätzlich implementiert werden muss. Dieser neue Datentyp kann dann als Schlüssel oder Wert in den Map- und Reduce-Funktionen verwendet werden. Wird der Datentyp beispielsweise als Schlüssel oder Wert der Ausgabe der Reduce-Funktion verwendet, so entsteht abstrakt formuliert eine Datei mit Objekten der Klasse Messung. Diese Datei kann dann jedoch nur von Programmen in der Programmiersprache Java weiter verarbeitet werden. Programme in anderen Programmiersprachen, wie beispielsweise C++ oder Python, können die Ergebnisse des MapReduce-Durchlaufs somit nicht auswerten.

Eine Lösung für diese mangelhafte Interoperabilität ist die Reduktion der Ausgabe auf den kleinsten gemeinsamen Nenner der unterschiedlichen Programmiersprachen. Für das Beispiel bedeutet dies, dass die einzelnen Messungen nicht als Objekte serialisiert werden, sondern vorher in eine textuelle Repräsentation umgewandelt werden. Diese Repräsentation kann von allen Programmiersprachen verarbeitet werden, jedoch ist diese weder kompakt noch typsicher[23]. Einen eleganten alternativen Lösungsansatz bieten Frameworks zur Serialisierung, bei denen Datentypen unabhängig von der Programmiersprache definiert werden können. Bei diesen Frameworks wird der Datentyp durch eine sogenannte Interface Description Language (IDL) definiert. Mit entsprechenden Werkzeugen können aus dieser Beschreibung automatisch Datentypen für mehrere Programmiersprachen generiert werden. So kann beispielsweise eine Instanz des Datentyps Messung mit einem Java-Programm erstellt und serialisiert werden, welches anschließend von einem Programm in C++ deserialisiert und weiterverarbeitet werden kann. Beispiele für solche Frameworks zur Serialisierung sind Protobuf (2012) von Google sowie Thrift (2012) und Avro (2012) von der Apache Foundation. Von diesen wird im Folgenden basierend auf den Beschreibungen in Avro (2012) und White (2012) das Framework Avro detaillierter vorgestellt, da dieses speziell für die Zusammenarbeit mit Hadoop ausgelegt ist und einige Besonderheiten bietet, die für diese Arbeit relevant sind.

Das Avro Serialisierungssystem wurde vom gleichen Entwickler gegründet wie das Hadoop-Projekt. Momentan kann das Datenformat von Avro mit den Programmiersprachen Java, C, C++, C#, PHP, Python und Ruby prozessiert werden. Die sprachunabhängige Beschreibung eines Datentyps erfolgt bei Avro in einem Schema, das üblicherweise im Datenformat der JavaScript Object Notation (JSON) definiert wird. JSON (2012) ist ein leichtgewichtiges Format zum Datenaustausch, das sowohl maschinenlesbar ist als auch einfach von Menschen gelesen und erstellt werden kann. JSON basiert auf Schlüssel-Wert-Paaren sowie auf Listen von Werten. Schlüssel und Werte sind in JSON mit einem Doppelpunkt voneinander getrennt. Auflistungen werden in JSON durch eckige Klammern und Objekte durch geschweifte Klammern gekennzeichnet. Die Daten werden bei Avro üblicherweise in einem effizienten Binärformat codiert, sie können jedoch auch lesbar als JSON ausgegeben werden.

Der Aufbau eines Avro-Schemas wird am bereits diskutierten Beispiel der Messung von Umweltwerten deutlich, das im Programm 2.7 definiert ist. Der Datentyp Messung wird im Schema durch den komplexen Avro-Datentyp Record modelliert, der aus einer Auflistung von Feldern (engl. 'fields') besteht. Die Felder sind jeweils benannt ('name') und über einen primitiven Avro-Datentyp ('type') definiert. Die im Programm 2.7 verwendeten primitiven Datentypen sind Ganzzahlen (int), boolesche Werte (boolean), Texte (string), Ganzzahlen (double) und Bytesequenzen (bytes). Avro stellt weitere einfache und komplexe Datentypen bereit, die in Avro (2012) detailliert beschrieben sind. Der Datentyp selbst ist auch benannt und kann eine optionale Dokumentation (doc) enthalten. Aus diesem Schema können nun mit einem von Avro bereitgestellten Werkzeug entsprechende Klassenbeschreibungen in den unterstützten Programmiersprachen erstellt werden.

Avro unterstützt zudem Änderungen am Schema, d.h. das Schema mit dem die Daten gelesen werden kann von dem Schema abweichen mit dem die Daten geschrieben wurden. Eine einfache und zugleich mächtige Änderungsmöglichkeit ist die sogenannte Projektion, bei der das zum lesen verwendete Schema weniger Felder enthält

[23]Die fehlende Typsicherheit drückt sich dadurch aus, dass die einzelnen textuell kodieren Attribute manuell in entsprechende Datentypen der Programmiersprache konvertiert werden müssen.

```
{
  "type": "record",
  "name": "Messung",
  "doc": "Informationen zur Messung von Umweltwerten an einer Station.",
  "fields": [
    { "name": "stationsId", "type": "int" },
    { "name": "stationZustandStabil", "type": "boolean" },
    { "name": "zeitpunkt", "type": "string" },
    { "name": "mittelwert", "type": "double" },
    { "name": "rohdaten", "type": "bytes" }
  ]
}
```

Programm 2.7: Avro-Schema des Datentyps Messung

als das beim schreiben verwendete Schema. Für das Beispiel des Datentyps Messung ist eine Projektion des Schemas aus Programm 2.7 im Programm 2.8 angegeben. Wird dieses Schema zum lesen der Daten verwendet, so steht die Information aus den beiden Feldern stationZustandStabil und rohdaten nicht zur Verfügung. Eine komplexere Änderung ist die Evolution, bei der dem Schema das zum lesen verwendet wird weitere Felder hinzugefügt werden, die mit initialen Werten vorbelegt werden. Zudem können in einem Avro-Schema Namen durch Aliase umbenannt werden.

```
{
  "type": "record",
  "name": "Messung",
  "doc": "Informationen zur Messung von Umweltwerten an einer Station.",
  "fields": [
    { "name": "stationsId", "type": "int" },
    { "name": "zeitpunkt", "type": "string" },
    { "name": "mittelwert", "type": "double" },
  ]
}
```

Programm 2.8: Avro-Schema des Datentyps Messung mit Projektion

Drei Eigenschaften von Avro sind im Kontext von Hadoop essentiell und werden so von keinem anderen Framework zur Serialisierung bereitgestellt. Erstens spezifiziert Avro ein Container-Format in dem eine Sequenz von Objekten als einzelne Datei abgespeichert werden kann. Dieses Format ist selbstbeschreibend, da in den Metadaten der Datei das Schema der Daten enthalten ist. Zudem sind diese Avro-Dateien teilbar und ermöglichen die Verwendung von Kompressionsverfahren zur Verringerung der Dateigröße. Durch die Teilbarkeit kann Hadoop eine einzelne Avro-Datei in eine entsprechende Anzahl an Splits unterteilen, die dann jeweils einem Map-Task als Eingabe zugeordnet werden. Zweitens sind für Avro-Objekte und Avro-Dateien spezielle Ein- und Ausgabetypen für die Map- und Reduce-Funktionen von Hadoop vorhanden. Diese Typen vereinfachen die Verarbeitung der serialisierten Daten erheblich, da keine expliziten Typumwandlungen der Schlüssel-Wert-Paare erforderlich sind. Drittens ist die Erzeugung von spezifischen Datentypen aus einem Avro-Schema optional, d.h. für ein Schema muss nicht zwingend Quelltext erzeugt werden um entsprechende Avro-Objekte verarbeiten zu können. Ist beim verarbeiten eines Avro-Objekts das zugehörige Schema bekannt, dann kann das Objekt über einen sogenannten Avro GenericRecord (Gr) verarbeitet werden. Die spezifische Umsetzung des Datentyps Messung aus dem Programm 2.8 in die Programmiersprache Java ist im Programm 2.9 angegeben. Die entsprechende generische Umsetzung ist im Programm 2.10 aufgelistet.

```
Messung messung = new Messung();
messung.stationsId = 1000;
messung.zeitPunkt = "2012-10-20.14:32";
messung.mittelwert = 6.2d;
```

Programm 2.9: Spezifischer Datentyp Messung in Java

```
GenericRecord messung = new GenericData.Record(schema);
messung.set("stationsId", 1000);
messung.set("zeitPunkt", "2012-10-20.14:32");
messung.set("mittelwert", 6.2d);
```

Programm 2.10: Generischer Datentyp Messung in Java

2.8.5 Optimierung der Gesamtlaufzeit eines MapReduce-Jobs

Die Gesamtlaufzeit eines MapReduce-Jobs wird maßgeblich von den Laufzeiten der jeweils langsamsten Map- und Reduce-Tasks beeinflusst, da diese die weitere Abarbeitung des Workflows blockieren. In Abbildung 2.41 sind beispielhaft die Laufzeiten einiger Tasks abgebildet. Wenige Tasks, sogenannte Nachzügler (engl. 'straggler'), weisen eine überproportional lange Laufzeit auf und können entsprechend als Ausreißer deklariert werden. Das Ziel der Optimierung der Gesamtlaufzeit eines MapReduce-Jobs ist es diese Nachzügler zu vermeiden beziehungsweise deren Laufzeit zu verbessern. Als Ursachen für die verlängerte Laufzeit von Nachzüglern kommen Abweichungen zwischen den einzelnen Tasks bezüglich der Hardware, dem Datenumfang sowie der Komplexität der Daten in Betracht. Im Folgenden werden Ansätze vorgestellt, die darauf abzielen diese Abweichungen zu minimieren.

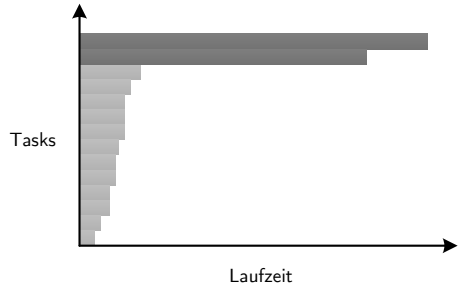


Abbildung 2.41: Laufzeit von Tasks als Motivation für die spekulative Ausführung

Spekulative Ausführung

Die zur Verfügung gestellte Hardware für einen Map- oder Reduce-Task kann sich erheblich zwischen mehreren parallel ausgeführten Tasks unterscheiden. Ein Grund dafür sind beschädigte, aber durch Korrekturverfahren behebbare, Schäden der Hardware eines Servers. Beispielsweise ist der Zugriff auf eine Festplatte erheblich langsamer, wenn diese defekte Sektoren aufweist die behoben werden müssen. Ein zweiter Grund ist eine erheblich verminderte Performanz eines Servers, die beispielsweise auf eine hohe Auslastung der Ressourcen durch Tasks anderer MapReduce-Jobs zurückzuführen ist.

Durch die *spekulative Ausführung* (engl. 'speculative execution') in Hadoop, beziehungsweise Backup Tasks in Google MapReduce, wird ein als Nachzügler identifizierter Task automatisch mit identischen Eingabedaten parallel auf einem anderen Server ausgeführt. Ist einer der beiden Tasks abgeschlossen, wird automatisch der zweite noch laufende Task beendet und dessen Daten verworfen. Die spekulative Ausführung ist nur dann sinnvoll, wenn die erzeugten Ausgabedaten der beiden Tasks identisch und eindeutig sind.

Partitionierung

Die spekulative Ausführung ist dagegen nach Kwon u. a. (2012) ineffizient, wenn ein einzelner Task deshalb langsam ist, weil seine Eingabedaten vergleichsweise umfangreich oder vergleichsweise aufwändig zu prozessieren sind. Die spekulative Ausführung des Tasks führt dann zu keiner signifikanten Laufzeitverbesserung, da die Eingabedaten nicht verändert werden. Dies betrifft sowohl die Splits als Eingabedaten für Map-Tasks als auch die Partitionen als Eingabedaten für Reduce-Tasks.

Kwon u. a. (2012) stellen vier Ansätze zum Umgang mit der ungleichen Verteilung der Eingabedaten auf einzelne Tasks beziehungsweise deren Partitionen vor. Erstens können Tasks so implementiert beziehungsweise Partitionen so gebildet werden, dass diese weitgehend *resistent gegen die ungleiche Verteilung* der Eingabedaten sind. Dieser Ansatz benötigt zusätzlichen Aufwand bei der Entwicklung. Der zweite Ansatz basiert auf einer *extrem feingranularen Partitionierung* der Eingabedaten, wodurch die Laufzeit jedes einzelnen Tasks deutlich reduziert wird. Jedoch führt dieser Ansatz zu einem deutlichen Mehraufwand für die Verwaltung der Tasks

und den Austausch der Daten. Der dritte Ansatz eignet sich für zwei aufeinanderfolgende Tasks. Die *Ausgabe des ersten Tasks wird stichprobenartig untersucht* um daraus eine geeignete Partitionierung der Daten für den zweiten Task abzuleiten. Der vierte Ansatz basiert auf einer *automatischen Repartitionierung* der Daten des jeweils langsamsten Map- oder Reduce-Tasks und wird in Kwon u. a. (2012) ausführlich dargestellt. Das System ist so konzipiert, dass es einen kompatiblen Ersatz zur parallelen Jobausführung mit Hadoop bereitstellt. Befindet sich ein Knoten nach der Ausführung seiner ihm zugewiesenen Tasks im Leerlauf, dann identifiziert das System den Task mit der voraussichtlich längsten Restlaufzeit. Dessen unprozessierte Eingabedaten werden unter Berücksichtigung der jeweiligen voraussichtlichen Restlaufzeiten der anderen Knoten auf momentan oder bald verfügbare Knoten aufgeteilt. Dieser Ansatz wird wiederholt bis alle Tasks abgeschlossen sind.

Räumliche Partitionierung

Durch geeignete räumliche Partitionierungsverfahren können räumliche Eingabedaten möglichst gleichmäßig auf einzelne Tasks beziehungsweise deren Partitionen verteilt werden. Das Ziel der im Folgenden vorgestellten Verfahren ist die Bildung möglichst resistenter Partitionen. Diese greifen die Ideen der extrem feingranularen Partitionierung sowie der stichprobenartigen Untersuchung der Ausgabedaten auf. In den Betrachtungen wird der Begriff 'räumliche Partition' für die Unterteilung raumbezogener Daten auf Basis der Objektgeometrien verwendet. Der Begriff 'MapReduce-Partition' wird dagegen für die Eingabedaten eines Map- beziehungsweise Reduce-Tasks verwendet.

Zum Vergleich der Verfahren ist in Abbildung 2.42 und Tabelle 2.20 ein Beispiel für die räumliche Partitionierung angegeben. Die in Tabelle 2.20 zusammengefassten Verteilungen sind explizit nur exemplarisch zu sehen, da das Beispiel nur wenige Objekte enthält. Als Referenz dient die *einfache* räumliche Partitionierung entsprechend einem gleichmäßigen räumlichen Gitter. Für diese räumliche Partitionierung ist keine Vorprozessierung der Daten notwendig, da die räumlichen Partitionen direkt aus deren Abmessungen und dem umschließenden Rechteck um die gesamten Eingabedaten abgeleitet werden können. Diese unterteilt die 16 Objekte in Abbildung 2.42 (graue Quadrate) in vier gleich große MapReduce-Partitionen (A, B, C, D). In der zugehörigen Tabelle 2.20 weisen die MapReduce-Partitionen eine sehr ungleichmäßige Verteilung der Eingabedaten auf, so steht neun Objekten in MapReduce-Partition B lediglich ein Objekt in MapReduce-Partition C gegenüber. Die einfache räumliche Partitionierung wird beispielsweise in Wang u. a. (2011) sowie Aji u. a. (2012) verwendet.

Das erste Verfahren zur Bildung resistenter MapReduce-Partitionen basiert auf räumlichen Indexstrukturen. Für den Aufbau der räumlichen Indexstruktur ist eine Vorprozessierung der Eingabedaten notwendig. Im Beispiel und in Zhong u. a. (2012) wird dafür ein *Quadtree* (siehe Unterabschnitt 2.1.5) verwendet, da dieser überlappungsfreie räumlichen Partitionen definiert. In Abbildung 2.42 ist ein Quadtree mit maximal vier Objekten pro Blattknoten dargestellt, was zu einer weiteren Unterteilung der Partition B des einfachen Verfahrens und damit zu einer gleichmäßigeren Verteilung der Objekte auf die MapReduce-Partitionen führt.

Das zweite Verfahren zur Bildung resistenter MapReduce-Partitionen wird in Zhang u. a. (2009) vorgestellt. Für dieses Verfahren wird eine extrem feingranulare räumliche Partitionierung verwendet, um eine annähernd gleichmäßige Verteilung der Objekte eines Datensatzes auf die entsprechenden MapReduce-Partitionen zu erreichen. Dazu werden die Eingabedaten in regelmäßige räumliche Partitionen unterteilt, deren Anzahl sehr viel größer ist als die Anzahl der MapReduce-Partitionen. In Zhang u. a. (2009) stehen beispielsweise 16 384 räumlichen Partitionen lediglich 8 MapReduce-Partitionen gegenüber. Danach werden die räumlichen Partitionen auf Basis einer Z-Kurve (siehe Unterabschnitt 2.1.6) durchnummeriert und nach dem Round-Robin-Verfahren den einzelnen Reduce-Partitionen zugeteilt (*Z-Kurve Round-Robin*). Jedes Objekt wird schließlich genau den MapReduce-Partitionen zugeteilt, deren räumliche Partitionen das miminal umschließende Rechteck des Objekts überlappen. Die beschriebene Partitionierung kann mit einem Map-Task erzeugt werden. Für das Verfahren Z-Kurve Round-Robin ist in Abbildung 2.42 die Aufteilung in die vier MapReduce-Partitionen A bis D dargestellt. Die Objekte sind auch bei diesem Verfahren gleichmäßiger als bei der einfachen räumlichen Partitionierung verteilt.

Das dritte Verfahren zur Bildung resistenter MapReduce-Partitionen wird in Cary u. a. (2009) vorgestellt. Das Verfahren stellt eine räumliche Partitionierungsfunktion in einem MapReduce-Durchlauf auf. Auch bei diesem Verfahren wird eine extrem feingranulare räumliche Partitionierung verwendet. In der Map-Funktion wird jedoch lediglich eine Stichprobe der Objekte des InputSplits verwendet und deren eindimensionaler Z-Wert auf Basis einer Z-Kurve bestimmt. Die Stichproben umfassen nur einen geringen Anteil der Daten und beschleunigen dadurch das Verfahren. In Cary u. a. (2009) wird ein Anteil von sowohl ein als auch drei Prozent genannt. Ein einzelner Reducer sortiert die Z-Werte aller Objekte und bestimmt schließlich Teilungspunkte für die MapReduce-Partitionierung (*Z-Kurve Teilungspunkte*). Diese sind so gewählt, dass jede MapReduce-

Partition annähernd gleich viele Objekte enthält. Im Beispiel ist die Verteilung der 16 Objekte auf die vier MapReduce-Partitionen bei diesem Verfahren am optimalsten.

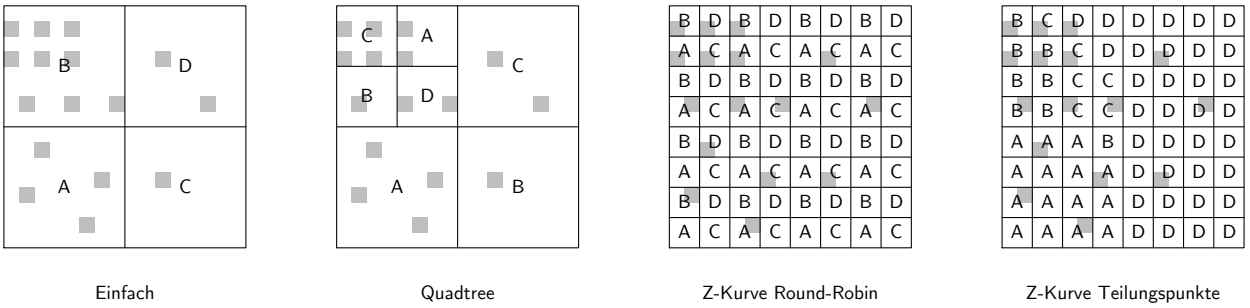


Abbildung 2.42: Beispiel für die räumliche Partitionierung

	Anzahl Objekte pro Partition			
Partition	Einfach	Quadtree	Z-Kurve Round-Robin	Z-Kurve Teilungspunkte
A	4	6	6	4
B	9	2	3	4
C	1	6	5	4
D	2	2	2	4

Tabelle 2.20: Beispiel für die räumliche Partitionierung

Prozessierung räumlicher Partitionen

Bei der Prozessierung räumlicher Partitionen werden häufig Abfragen benötigt, die Objekte anhand geometrischer Kriterien filtern können. Für die Beschleunigung dieser Abfragen werden üblicherweise räumliche Indexstrukturen verwendet (siehe Unterabschnitt 2.1.5).

In Aji u. a. (2012) und Wang u. a. (2011) werden räumliche Indexstrukturen erst bei Bedarf (engl. 'on-the-fly') erzeugt. Der Mehraufwand für die einmalige Erzeugung und Speicherung des Index für jede Datei ist nach Aji u. a. (2012) nur für einen Bruchteil der Gesamtlaufzeit eines Programms verantwortlich. Wang u. a. (2010) und Zhang u. a. (2009) argumentieren dagegen, dass die Eingabedaten im MapReduce-Framework meist dynamisch sind, da diese beispielsweise das Ergebnis einer räumlichen Abfrage sind. Da nach ihrer Ansicht die Erstellung eines räumlichen Index zeitaufwändig ist, setzen sie auf Verfahren welche die räumlichen Partitionen so klein machen, dass alle benötigten Daten in den verfügbaren Arbeitsspeicher eines MapReduce-Tasks passen.

In den bisher vorgestellten Arbeiten werden räumliche Indexstrukturen entweder nicht verwendet oder lokal für jede MapReduce-Partition erzeugt. Diese lassen sich jedoch auch verteilt erzeugen und abspeichern, wie die drei folgenden Arbeiten aufzeigen. Das MapReduce Framework wird von Cary u. a. (2009) verwendet MapReduce um die Erstellung eines R-Baums zu parallelisieren, indem viele parallel berechnete Teilbäume zu einem globalen Baum zusammengesetzt werden. Die Erstellung erfordert zwei MapReduce-Durchläufe und die anschließende Ausführung eines sequentiellen Programms. Die verteilte Speicherung eines R-Baums als einzelne Datei im HDFS wird in Liao u. a. (2010) diskutiert. Für den Benutzer ist die verteilte Speicherung transparent, da er über eine Programmierschnittstelle auf den R-Baum zugreift. In Zhong u. a. (2012) wird ein Dateiformat beschrieben, das auf einem verteilten räumlichen Index basiert. Global werden die Indexknoten eines Quadtree gespeichert, dessen Blattknoten jeweils einer Datei im HDFS entsprechen. Lokal sind die Objekte in einer solchen Datei entsprechend ihrer Reihenfolge in einer Hilbert-Kurve (siehe Unterabschnitt 2.1.6) abgespeichert.

3 Integritätsbedingungen: Anwendung, Klassifikation und Formalisierung

In diesem Kapitel werden auf Basis einer umfangreichen Analyse von publizierten Arbeiten mit dem Thema Integritätsbedingungen drei Fragen beantwortet. Der erste Abschnitt gibt einen Überblick über die praktischen Anwendungen die in den Arbeiten die Motivation, Datenmodelle und Daten für die Aufstellung von Integritätsbedingungen liefern. Abschnitt 3.1 beantwortet damit die Frage nach dem *warum*. Der zweite Abschnitt stellt mehrere Klassifikationen von Integritätsbedingungen vor und gibt so einen Eindruck über deren Bandbreite und Bedeutung. In Abschnitt 3.2 wird also die Frage nach dem *was* eine Integritätsbedingung fordert beziehungsweise einschränkt beantwortet. Im dritten und letzten Abschnitt werden existierende Ansätze zur Formalisierung von Integritätsbedingungen vorgestellt. Damit wird in Abschnitt 3.3 die Frage beantwortet *wie* Integritätsbedingungen formal und damit eindeutig aufgestellt werden können.

3.1 Anwendung

Integritätsbedingungen finden in den unterschiedlichsten Bereichen Anwendung. Die referenzierten Arbeiten decken ein breites Spektrum an Anwendungen ab, welche deren Diversität veranschaulichen. In diesem Abschnitt werden einige Aspekte dieser Diversität genauer diskutiert.

3.1.1 Amtliche Geobasisdaten

Die Integrität *amtlicher Geobasisdaten* ist von besonderer Wichtigkeit, da Geobasisdaten die Grundlage vieler Entscheidungen auf politischer und wirtschaftlicher Ebene bilden. In Tabelle 3.1 sind die Anwendungsbeispiele aus den referenzierten Arbeiten aufgelistet. Die Diversität der Anwendungsbeispiele ergibt sich in Bezug auf drei Kriterien. Das erste Kriterium für die Diversität ist die Größe des administrativen Gebietes, das vom Gebiet einer Stadt bis zur Geodateninfrastruktur eines Staates reicht. Das zweite Kriterium ergibt sich daraus, dass sowohl Daten des Katasters als auch der Topographie abgedeckt werden. Das dritte Kriterium für die Diversität ist die weltweite Bedeutung von Integritätsbedingungen. In den Anwendungsbeispielen sind Geobasisdaten der Staaten Brasilien, Dänemark, Frankreich, Italien und den Niederlanden repräsentiert. Jeder Staat modelliert in der Regel seine Geobasisdaten entsprechend seiner Anforderungen in einem speziell dafür entworfenen Datenmodell. Als Ausnahme dieser Regel ist in den Anwendungsbeispielen das Land Administration Domain Model (LADM) enthalten. Das LADM ist ein Modell für Daten der Landesvermessung und des Katasters und ist als International Organization for Standardization (ISO) 19152 Standard veröffentlicht.

3.1.2 Fachdaten

Anforderungen an die Integrität von *Fachdaten* sind meist vom Anwendungsgebiet der Daten geprägt. Entsprechend wichtig ist es das umfangreiche Expertenwissen eines Anwendungsgebietes explizit abzubilden. In Tabelle 3.2 sind die Anwendungsbeispiele aus den referenzierten Arbeiten aufgelistet. Die Diversität zeigt sich unter anderem darin, auf welchen Daten die Anwendungen aufbauen. Erstens können diese auf Geobasisdaten aufbauen, wie beispielsweise die Anwendung aus dem Gesundheitsmanagement zeigt. Zweitens können diese auf einer eigenen Datenbasis aufbauen, die parallel zu den Geobasisdaten existiert, wie beispielsweise die Anwendung aus der Fahrzeugnavigation zeigt. In dieser Anwendung werden Daten verwendet, die unter anderem durch Befahrungen erhoben werden. Drittens können diese auf einer unabhängigen Datenbasis aufbauen, wie beispielsweise die Anwendung mit dem Versorgungsnetzwerk aus Rohrleitungen aufzeigt.

3.2 Klassifikation

Eine *Klassifikation* bezeichnet die Einteilung beziehungsweise Einordnung von Objekten anhand ihrer Merkmale in aufgestellte Klassen (Duden, 2011). Durch die Einteilung werden Objekte unterschiedlicher Klassen voneinander abgegrenzt. Durch die Einordnung in eine hierarchische Klassenstruktur können Objekte durch ihre über- und untergeordneten Klassen charakterisiert werden.

Referenz(en)	Anwendung	Anwendungsbeispiel: Objektarten
Servigne u. a. (2000)	Geodatenbank der französischen Stadt Lyon	Gemeinde, Block, Flurstück, Gebäude
Borges u. a. (2001)	Geodatenbank des städtischen Katasters der brasilianischen Stadt Belo Horizonte	Stadt, Block, Flurstück, Gebäude, Adresse, Straßensegment, Straßenkreuzung, Höhenlinie
Belussi u. a. (2006), Pelagatti u. a. (2009)	Kerndatendank der italienischen Geodateninfrastruktur	—
Christensen (2007)	Geodatenbank der topographischen Grundkarte Dänemarks	Gebäude
	—	Gemeinde, Wohngebiet, Industriegebiet, Gebäude, Adresse, Straßensegment, Firma
van Oosterrom (2006)	Geodatenbank des niederländischen Katasters	Flurstück
	Geodatenbank der topographischen Datenbasis der Niederlande	Straße, Schiene, Gewässer
	Notarielle Teilung von Flurstücken	Flurstück
van Bennekom-Minnema (2008)	Land Administration Domain Model	Flurstück, Vermessungspunkt, (Mess-) Protokoll, Gebäudefläche

Tabelle 3.1: Anwendungsbeispiele von Integritätsbedingungen für Geobasisdaten

Referenz(en)	Anwendung	Anwendungsbeispiel: Objektarten
Cockcroft (2001)	Verwaltung von Jagdrevieren	Jagdrevier, Reservat für Eingeborene, Hütte, Küstenlinie, Tiere, Jäger
Cockcroft (2004)	Versorgungsnetzwerk aus Rohrleitungen	Rohr, Anschlussstück, Flansch, Absperrklappe, Pumpe, Reservoir, Abfluss, Tank, Schacht
Borges u. a. (2002)	Gesundheitsmanagement	Gemeinde, Block, Flurstück, Adresse, Straßensegment, Straßenkreuzung, Routensegment, Orthofoto, Gesundheitsbezirk, Gesundheitsbeauftragter, Hausbesuch, Familie
Casanova u. a. (2000)	Fahrzeugnavigation: Bewegungsabfolgen in Routen (bevorzugt, verboten, eingeschränkt), Dienstleistungen und deren Zugangspunkte	Straßenelement, Kreuzung, Bewegungsabfolge (Manöver), Dienstleistung, Zugangspunkt für Dienstleistung
Kang u. a. (2004)	Ad-hoc Drahtlosnetzwerk aus stationären und mobilen Kommunikationsknoten	Dorf, Straße, Gebäude, Mobiler Vermittler, Kommunikationsknoten
Pinet u. a. (2005), Bejaoui (2009), Bejaoui u. a. (2010)	Landwirtschaftliches Informationssystem: Ausbringung von Klärschlamm	Stadt, Flurstück, Bauernhof, See, Wasserlauf
Duboisset u. a. (2005a,b)	—	Stadt, Rathaus, Straßensegment, Gebäudeblock, Gebäude, Touristische Attraktion
Louwsma (2004), Louwsma u. a. (2006)	Landschaftsarchitektur: Simulation von Pflanzenwachstum, interaktive Planung in einer virtuellen Realität	Pflanze, Gras, Straßenbelag, Gewässer, Brücke
Xu (2011)	Geodatenbank des Campus der TU Delft für die Simulation von Wetter und Umwelt	Gebäude, Straße, Baum, Gewässer, Terrain, Sensor
Mäs u. a. (2005), Wang und Reinhardt (2007)	Mobiles GIS für die Erfassung von Hangrutschungen	Straße, Graben, Dehnungsmessgerät (Extensometer)
Reeves u. a. (2006)	Straßenmarkierungen: Haltelinien, Fußgängerüberwege, Sperrflächen etc.	Straßenmarkierung, Erhöhtes Straßenhindernis
Salehi (2009)	Katastrophenschutz	Katastrophenereignis, Stadt, Straße, Kreisverkehr, Krankenhaus, Einsatzfahrzeug
Salehi u. a. (2011)	Landwirtschaftliche Datenbank	Provinz, Feld, Feldfrucht, Bewässerungsfläche, Wind
Watson (2007)	Flughafen	Flughafen, Start- und Landebahn, Stoppbahn, Rollbahn, Vorfeldfläche, Hubschrauberlandeplatz, Positionsleuchte

Tabelle 3.2: Anwendungsbeispiele von Integritätsbedingungen für Fachdaten

Die Klassifikation von Integritätsbedingungen wird in dieser Arbeit aus zwei Gründen vorgestellt. Erstens erlaubt sie die Einteilung und Einordnung bereits identifizierter Bedingungen, die beispielsweise von Experten für ein konkretes Datenmodell aufgestellt wurden. Für die Klassifikation müssen die Integritätsbedingungen dann systematisch anhand ihrer Merkmale untersucht werden. So werden unter anderem Gemeinsamkeiten und Unterschiede deutlich. Sind an der Aufstellung von Integritätsbedingungen mehrere Akteure beteiligt, so wird durch die Festlegung auf eine gemeinsame Klassifikation auch die zu verwendende Terminologie in der Gruppe eindeutig festgelegt. Der zweite Grund für die Vorstellung der Klassifikationen ist die Vorstellung unterschiedlicher Sichtweisen auf Integritätsbedingungen. Dadurch können die für ein konkretes Datenmodell bereits aufgestellten Integritätsbedingungen um noch nicht berücksichtigte Aspekte beziehungsweise Klassen von Bedingungen erweitert werden.

Integritätsbedingungen werden in mehreren der referenzierten Arbeiten klassifiziert. In diesem Abschnitt werden davon diejenigen Klassifikationen vorgestellt, die entweder besonders detailliert sind oder Aspekte hervorheben die in anderen Klassifikationen fehlen. Die wichtigste Klassifikation von nicht-räumlichen Integritätsbedingungen bezieht sich auf Bedingungen in Datenbanksystemen und wird in Unterabschnitt 3.2.1 vorgestellt. Aus der Klassifikation in Datenbanksystemen leiten sich einige der anderen Klassifikationen ab, indem sie diese um Aspekte des Raumbezugs erweitern. Für diese Arbeit sind insbesondere die Klassifikationen von Integritätsbedingungen für die Sicherung der Datenintegrität wichtig, von denen insgesamt vier in Unterabschnitt 3.2.2 vorgestellt werden. Abschließend wird in Unterabschnitt 3.2.3 auf Klassifikationen aus dem Gebiet der Generalisierung eingegangen. Aus diesen wird deutlich, dass erstens Integritätsbedingungen für viele Gebiete sinnvoll angewendet werden können und zweitens viele Parallelen zwischen den Bedingungen der einzelnen Gebiete vorhanden sind.

3.2.1 Integritätsbedingungen in Datenbanksystemen

Integritätsbedingungen in nicht-räumlichen Datenbanksystemen werden im Standardwerk 'Fundamentals of Database Systems' von Elmasri und Navathe (2011) als wichtiger Bestandteil von Datenbanksystemen hervorgehoben und in diesem Unterabschnitt vorgestellt. Die in Unterabschnitt 3.2.2 vorgestellten abgeleiteten Klassifikationen beziehen sich dabei alle auf Elmasri und Navathe.

Die meisten aktuell eingesetzten Datenbanken basieren auf dem relationalen Datenmodell[1]. In einem relationalen Modell werden Daten in Tabellen (*Relationen*) gespeichert, die aus Zeilen (*Tupel*) und Spaltenköpfen (*Attribute*) aufgebaut sind. Die möglichen Werte, die ein Attribut in einer Spalte annehmen kann, werden durch einen Wertebereich eingeschränkt. Ein *Wertebereich* entspricht einer Menge von atomaren Werten und wird durch einen Namen, einen Datentyp und ein Format definiert. Für numerische Wertebereiche ist es sinnvoll zusätzlich deren Maßeinheit anzugeben.

Die Struktur einer Relation wird durch ihr *Schema* definiert, das aus einem Namen und einer Liste von Attributen besteht. Ein *relationales Datenbankschema* wird darauf aufbauend als eine Menge von relationalen Schemata und deren Integritätsbedingungen definiert. Die konkrete Ausprägung einer Relation wird als *Zustand einer Relation* bezeichnet. Ein *gültiger Zustand einer relationalen Datenbank* bezeichnet schließlich eine Menge von Zuständen von Relationen, die alle Integritätsbedingungen der Schemata erfüllen.

Elmasri und Navathe (2011) klassifizieren Integritätsbedingungen in Datenbanksystemen hierarchisch (siehe Abbildung 3.1). *Statische* Integritätsbedingungen definieren Bedingungen, die ein gültiger Zustand einer Datenbank erfüllen muss. Eine statische Bedingung ist beispielsweise 'das Alter eines Mitarbeiter muss mindestens 16 Jahre sein'. *Dynamische* Integritätsbedingungen beschränken gültige Übergänge (engl. 'transition') zwischen den einzelnen Zuständen einer Datenbank und damit die erlaubten Modifikationen an den Daten. Ein Übergang tritt immer dann auf, wenn Daten erstellt (engl. 'create'), aktualisiert (engl. 'update') oder gelöscht (engl. 'delete') werden. Eine entsprechende dynamische Bedingung ist beispielsweise 'das Alter eines Mitarbeiter kann nur erhöht (und nicht verringert) werden'.

Die Klasse der statischen Bedingungen ist weiter unterteilt. *Implizite* Integritätsbedingungen sind durch die Struktur und Restriktionen des Datenmodells definiert. So sind beispielsweise keine Duplikate von Tupeln erlaubt. Um die Anforderungen des Datenmodells zu erfüllen, garantiert das Datenbanksystem dass diese Bedingungen jederzeit erfüllt sind. *Explizite* Integritätsbedingungen werden als Teil des Datenbankschemas in der sogenannten Data Definition Language (DDL) formuliert. *Anwendungsbasierte* Integritätsbedingungen können nicht direkt im Schema formuliert werden. Sie beziehen sich auf die Bedeutung (Semantik) und das Verhalten

[1] Elmasri und Navathe (2011) geben an, dass weniger als fünf Prozent des Datenbankmarktes objektorientierten Datenbanken zugeordnet werden kann.

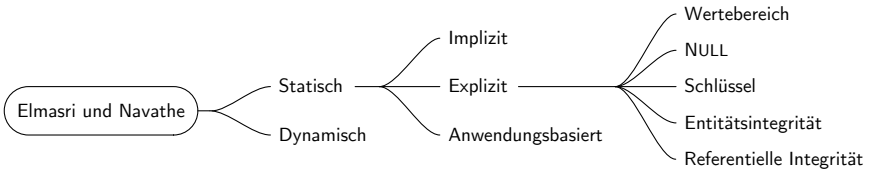


Abbildung 3.1: Klassifikation von Integritätsbedingungen nach Elmasri und Navathe (2011)

von Attributen und Relationen. Anwendungsbasierte Integritätsbedingungen werden üblicherweise als Teil eines Anwendungsprogramms umgesetzt, welches auf die Datenbank zugreifen kann.

Innerhalb der expliziten Integritätsbedingungen spezifizieren Bedingungen an den *Wertebereich* (engl. 'domain') den Datentyp und das Format der möglichen Werte eines Attributs. Beispielsweise kann das Alter eines Mitarbeiters durch einen ganzzahligen Datentyp kodiert werden und zusätzlich durch das Format die Bedingung aufgestellt werden, dass der Wert zwischen 16 und 67 liegen muss. Die *NULL* Bedingung gibt an, ob ein Attribut den speziellen Wert NULL annehmen darf und damit ein Wert als unbekannt, nicht verfügbar oder nicht definiert gekennzeichnet werden kann. Integritätsbedingungen an den *Schlüssel* (engl. 'key') definieren, dass ein oder mehrere Attributwerte eindeutig für jedes Tupel einer Relation sind, d.h. es existiert kein Tupel mit den gleichen Werten. Identifiziert ein Schlüssel in einer Relation eindeutig jedes Tupel, dann wird dieser als Primärschlüssel bezeichnet. Um jedes Tupel in einer Relation eindeutig identifizieren zu können sichert die *Entititätsintegrität* (engl. 'entity integrity') zu, dass kein Wert eines Primärschlüssels NULL sein darf. Als Entität wird dabei ein Objekt oder Konzept der realen Welt bezeichnet. Verweist ein Tupel in einer Relation auf ein anderes Tupel derselben oder einer anderen Relation, dann sichert die *referentielle Integrität* (engl. 'referential integrity') zu, dass das referenzierte Tupel existiert. Das Konzept der referentiellen Integrität wird in einer Datenbank durch sogenannte Fremdschlüssel umgesetzt.

3.2.2 Integritätsbedingungen in der Datenintegrität

In diesem Unterabschnitt werden vier Klassifikationen von Integritätsbedingungen für die Sicherung der Datenintegrität vorgestellt. Die Klassifikationen von Louwsma u. a. (2006) und van Oosterrom (2006) sowie von Mäs und Reinhardt (2009) leiten sich dabei aus der Klassifikation von Bedingungen in nicht-räumlichen Datenbanksystemen ab.

Servigne u.a. (2000)

Servigne u. a. (2000) unterteilen Fehler in der Konsistenz räumlicher Daten in drei Klassen (siehe Abbildung 3.2). *Strukturelle Fehler* entstehen wenn die verwendeten Datenstrukturen zur Speicherung der Daten nicht den Anforderungen des Datenmodells entsprechen. Sind beispielsweise Polygone mit Löchern zwar im Datenmodell aber nicht in den Datenstrukturen vorgesehen, dann können diese nicht korrekt abgespeichert werden. *Geometrische Fehler* beziehen sich auf die Lage und Form der Geometrie von Objekten. An Geometrien werden Anforderungen wie Eindeutigkeit, Geschlossenheit und korrekte Orientierung gestellt. Bedingungen bezüglich der Beziehung von Geometrien fordern vor allem die korrekte Struktur topologischer Netzwerke. *Topologisch-semantische Fehler* berücksichtigen die Bedeutung von Objekten und deren topologische Relationen. So dürfen beispielsweise Straßengeometrien niemals Gebäudegeometrien schneiden.

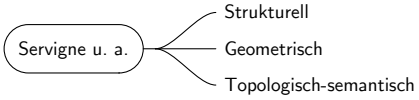


Abbildung 3.2: Klassifikation von Integritätsbedingungen nach Servigne u. a. (2000)

Louwsma u. a. (2006) und van Oosterom (2006)

Mit der Klassifikation von Louwsma u. a. (2006) und van Oosterrom (2006) werden jeder Integritätsbedingung sechs Klassen zugeordnet, die unterschiedliche Aspekte der Bedingung berücksichtigen (siehe Abbildung 3.3). Der erste Aspekt dient zur Unterscheidung in *statische* und *dynamische* Integritätsbedingungen entsprechend deren Definition in Datenbanksystemen. Der zweite Aspekt gibt die *Anzahl der Objekte und Objektklassen* an, auf die in einer Integritätsbedingung Bezug genommen wird. Der dritte Aspekt berücksichtigt welche *Objekteigenschaften und Beziehungen* Teil einer Integritätsbedingung sind. Dieser Aspekt ist weiter unterteilt in die

fünf Unterklassen *metrisch* (Distanz oder Winkel zwischen Objekten), *topologisch* (Nachbarschaft und Enthalten sein), *zeitlich*, *thematisch* (semantisch, ohne räumlichen Bezug) und *komplex*. Letztere bildet eine Kombination aus den anderen Unterklassen. Die *Dimension* ist der vierte Aspekt und kann *räumlich* (2D, 3D), *zeitlich*, *gemischt räumlich und zeitlich* (bis zu 4D) sein oder auf einer *thematischen Skala*[2] basieren. Die beiden Aspekte Dimension und Objekteigenschaften und Beziehungen sind miteinander verwandt. So kann die metrische Bedingung eines Mindestabstandes in den zwei räumlichen Dimensionen 2D und 3D spezifiziert werden. Der fünfte Aspekt unterscheidet die *Art des Ausdrucks* in 'muss immer' und 'darf nie'. Der sechste und letzte Aspekt unterscheidet schließlich die *Natur der Bedingung* nach dem Kriterium, ob diese auf einem *Theorem basiert*, beispielsweise weil sie physikalisch unmöglich ist, oder ob sie auf dem speziellen *Entwurf der Daten basiert*.

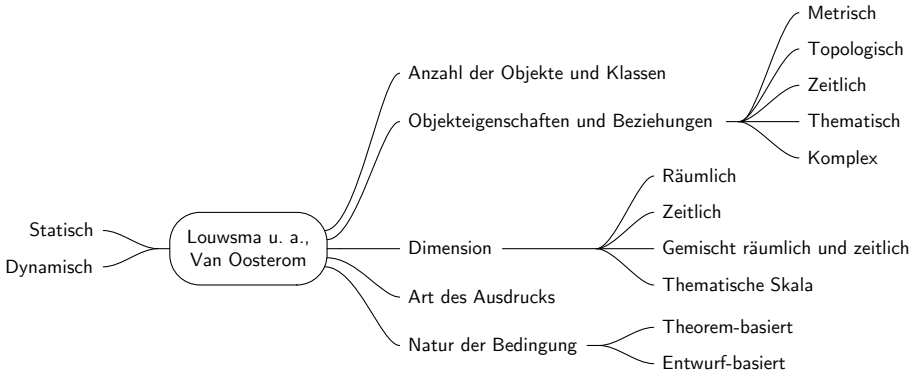


Abbildung 3.3: Klassifikation von Integritätsbedingungen nach Louwsma u. a. (2006) und van Oosterrom (2006)

Mäs und Reinhardt (2009)

Mäs und Reinhardt (2009) erweitern die Klassifikationen von Integritätsbedingungen in Datenbanksystemen und von Servigne u. a. (2000) (siehe Abbildung 3.4). Integritätsbedingungen die den *Wertebereich* einschränken erweitern die entsprechende Bedingung der Datenbanksysteme um geometrische, topologische und zeitliche Primitive. Für die Definition der Primitive sowie der entsprechenden Integritätsbedingungen verweisen Mäs und Reinhardt (2009) auf die beiden Standards ISO 19107 (2003) für geometrische und topologische Primitive sowie ISO 19108 (2002) für zeitliche Primitive. Die Klasse der Bedingungen an *Schlüssel und Beziehungen* entsprechen denjenigen in Datenbanksystemen. *Allgemeine semantische* Bedingungen schränken die Eigenschaften von Objekten und Beziehungen zwischen Objekten ein und sind weiter untergliedert.

Die Klasse der *räumlichen* Bedingungen ist in vier Unterklassen unterteilt. *Metrische* Bedingungen beschränken Distanzen und Größen von Objekten. *Topologische* Relationen werden sowohl zwischen Objekten als auch in Bezug auf ein einzelnes Objekt definiert. Innerhalb eines Objektes kann beispielsweise die Anzahl der Komponenten einer Geometrie oder die Präsenz und Anzahl von Löchern eines Polygons eingeschränkt werden. *Richtungsabhängige* Bedingungen können sich auf ein festes Referenzsystem beziehen, beispielsweise 'nördlich von', oder auf die Ordnung von Objekten im Raum, beispielsweise 'links von'. Mäs und Reinhardt (2009) weisen darauf hin, dass richtungsabhängige Bedingungen mit einem festen Bezugssystem selten verwendet werden, da diese zwar geeignet zur Beschreibung spezifischer Objekte sind, jedoch meist nicht verallgemeinert werden können. Bedingungen an die *Form* von Objekten sind nach Mäs und Reinhardt (2009) schwer zu quantifizieren und für viele Objektklassen nicht verallgemeinerbar. *Zeitliche* Bedingungen basieren auf den Relationen von Allen (1983).

Thematische Bedingungen schränken mögliche Attributwerte eines Objektes ausgehend von den Werten anderer Attribute desselben Objektes ein. Beispielsweise muss für Straßen vom Typ 'Autobahn' die Anzahl der Fahrspuren mindestens zwei sein. *Komplexe semantische* Bedingungen bilden eine Kombination der anderen Klassen von Bedingungen und enthalten zudem Teil – Ganzes Bedingungen. Die Beschränkung von *Änderungen* bezieht sich auf dynamische und auf zeitliche Sequenzen von dynamischen Bedingungen. Bedingungen bezüglich der *Repräsentation* beziehen sich auf Objekte die mehrere Geometrien oder mehrere thematische Attributwerte unterschiedlichen Detailgrades aufweisen, beispielsweise aufgrund einer kartographischen Generalisierung. Bedingungen bezüglich Änderung und Repräsentation können sich auch auf die anderen Klassen von Bedingungen beziehen. Mäs und Reinhardt (2009) weisen darauf hin, dass die Zuweisung einer Integritätsbedingung zu einer

[2] Im Kontext einer thematischen Skala bezieht sich die Dimension auf die thematische Aggregation von Objekten.

der Klassen nicht immer eindeutig ist. In diesem Fall soll die Bedingung dem abstrakteren Konzept zugeordnet werden.

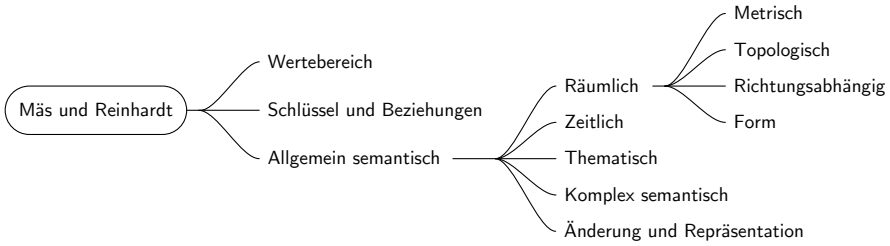


Abbildung 3.4: Klassifikation von Integritätsbedingungen nach Mäs und Reinhardt (2009)

Salehi (2009) und Salehi u. a. (2011)

Nach Salehi (2009) und Salehi u. a. (2011) basieren Integritätsbedingungen auf den drei Konzepten Raum, Zeit und Thematik sowie deren Kombinationen. *Raum* (engl. 'space') bezieht sich auf die Form und Lage, *Zeit* (engl. 'time') bezieht sich auf die Dauer und *Thematik* (engl. 'themes') bezieht sich auf die anwendungsspezifische Bedeutung. Die Konzepte und deren Kombinationen werden in einem 3-Venn-Diagramm dargestellt (siehe Abbildung 3.5). Zusätzlich zu den drei Konzepten wird die Kombination aus *Raum – Zeit* als viertes Konzept eingeführt. Dies wird damit begründet, dass manche Konzepte effizienter als Kombination aus Raum und Zeit abgebildet werden, wie beispielsweise ein sich bewegender Punkt, eine sich ausdehnende Fläche oder der Begriff Geschwindigkeit.

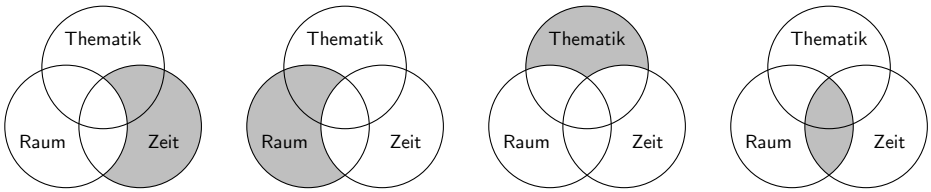


Abbildung 3.5: 3-Venn Diagramm nach Salehi (2009) und Salehi u. a. (2011)

Die Aufteilung in die vier Konzepte basierend auf dem 3-Venn-Diagramm findet sich in der Klassifikation von Integritätsbedingungen wieder (siehe Abbildung 3.6). Diese werden anhand von zwei Aspekten klassifiziert. Der erste Aspekt unterscheidet zwischen elementaren und thematischen Bedingungen, jedoch nur für die Klassen räumlich, zeitlich und räumlich – zeitlich des zweiten Aspekts. *Elementare* Bedingungen weisen keinen Bezug zur Thematik auf und entsprechen damit genau den Bereichen im 3-Venn-Diagramm die sich nicht mit dem Konzept Thematik überschneiden. *Thematische* Bedingungen bilden das Komplement zu den elementaren Bedingungen.

Der zweite Aspekt der Klassifikation entspricht den vier identifizierten Konzepten. *Räumliche* Integritätsbedingungen sind in fünf Klassen unterteilt. *Primäre* Bedingungen definieren räumliche Primitive, beispielsweise muss ein Polygon aus mindestens drei Liniensegmenten zusammengesetzt sein. Aber auch die Bedingung, dass eine Straße durch eine Linie repräsentiert wird, ist eine primäre räumliche Bedingung. *Metrische* Bedingungen definieren unter anderem Mindestabstände oder Mindestbreiten, aber auch richtungsabhängige Bedingungen, da diese auf einem räumlichen Referenzsystem basieren. *Topologische* Bedingungen basieren auf den binären topologischen Relationen. *Ordnungsbedingungen* haben immer auch eine inverse Bedingung, beispielsweise 'vor' und 'hinter' in Bezug auf ein Objekt. *Gemischte* Bedingungen enthalten zwei oder mehr der anderen räumlichen Klassen. *Zeitliche* Bedingungen sind in drei Unterklassen unterteilt. *Primäre* zeitliche Bedingungen definieren die zeitlichen Primitive Zeitpunkt, Intervall und Zyklus. *Metrische* zeitliche Bedingungen entsprechen Distanzen im zeitlichen Referenzsystem, beispielsweise repräsentiert durch die Dauer eines Intervalls. *Topologische* zeitliche Bedingungen beschreiben die Abfolge und Beziehungen zwischen zeitlichen Primitiven. *Räumlich – zeitliche* Bedingungen sind in drei Unterklassen unterteilt. *Inhärente* Bedingungen enthalten nur gemischt räumlich – zeitliche Primitive, jedoch keine nur räumlichen oder nur zeitlichen Primitive. Beispiele für inhärente Bedingungen sind 'die Ausbreitungsgeschwindigkeit eines Waldbrandes ist kleiner als $100\,\mathrm{m}^2/\mathrm{min}$' und 'falls die Beschleunigung eines bewegten Punktes positiv ist, dann muss sich dessen Geschwindigkeit erhöhen'. *Zusammengesetzte* Bedingungen können räumliche, zeitliche und räumlich – zeitliche Konzepte enthalten, wie beispielsweise in der Bedingung 'die Entfernung zwischen einer Schule und einer Tankstelle muss seit dem Jahr 1970 mehr als 300 Meter betragen'. *Gemischte* Bedingungen entsprechen wiederum eine Kombination aus den beiden anderen Unterklassen. *Thematische* Bedingungen werden nicht weiter unterteilt, da für thematische Konzepte keine

generell akzeptierte und allumfassende Klassifizierung existiert. Ein Beispiel für eine Bedingung, die sich lediglich auf thematische Konzepte bezieht, ist 'die Nutzung eines Gebäudes muss entweder Wohnen, Handel und Dienstleistung oder Industrie entsprechen'.

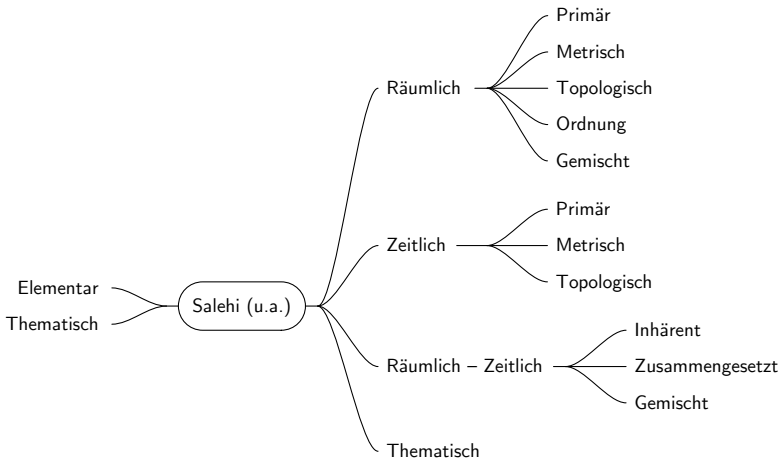


Abbildung 3.6: Klassifikation von Integritätsbedingungen nach Salehi (2009) und Salehi u. a. (2011)

Nach Salehi (2009) und Salehi u. a. (2011) sind die meisten Klassifizierungen von Integritätsbedingungen nicht präzise genug, da die Definition der Bedingungen nur in natürlicher Sprache erfolgt. Deshalb basiert die Formalisierung der Integritätsbedingungen in Salehi (2009) und Salehi u. a. (2011) auf mathematischer Logik. Durch die Formalisierung werden die Klassen überlappungsfrei und damit eineindeutig definiert.

3.2.3 Integritätsbedingungen in der Generalisierung

Neben der Datenintegrität ist die Generalisierung eine weitere wichtige Anwendungsdomäne für Integritätsbedingungen. Der Zusammenhang der beiden Domänen wird in Werder (2009) detaillierter dargestellt. Im Kontext der Klassifikation von Integritätsbedingungen zeichnen sich vor allem die Arbeiten des AGENT-Projekts (AGENT, 1998, 1999) sowie die Arbeiten von Steiniger und Weibel (2005) und Steiniger (2007) durch ihren hohen Detailgrad aus.

Exemplarisch wird im Folgenden die Klassifikation von Steiniger und Weibel (2005) und Steiniger (2007) kurz vorgestellt (siehe Abbildung 3.7). Integritätsbedingungen werden dabei in fünf Klassen unterteilt, die wiederum mehrere Unterklassen enthalten. *Geometrische* Bedingungen schränken die mögliche Größe, Position, Form und Orientierung von Objektgeometrien ein. *Topologische* Bedingungen beziehen sich auf die binäre topologische Relation von Objektgeometrien, aber ebenso auf die Struktur, die Nachbarschaftsordnung und auf Ringkonfigurationen. Die Struktur unterscheidet beispielsweise wiederum in Inselpolygone, d.h. alleinstehende Polygone, und Landschaftsmosaike und zeigt so die enge Kopplung der Bedingungen an den Prozess der Generalisierung. *Statistische und dichte-basierte* Bedingungen beziehen sich auf die Ermittlung statistischer Kenngrößen, aber auch auf Wahrscheinlichkeiten und Verteilungen von bestimmten Objekten und Objektklassen sowie auf Diversitätsmetriken, die beispielsweise auch in der Landschaftsökologie angewendet werden. *Semantische* Bedingungen enthalten Anforderungen zu Ähnlichkeiten von Objektklassen, Prioritäten, Abstoßung und Anziehung von Objekten sowie Abhängigkeiten zwischen unterschiedlichen Klassen. Semantische Bedingungen sind dabei wieder stark vom Prozess der Generalisierung geprägt. *Strukturelle* Bedingungen umfassen schließlich Aspekte der Wahrnehmung, der Natur des Entstehungsprozesses, Orientierungsmuster und Strukturen auf unterschiedlichen Maßstabsebenen.

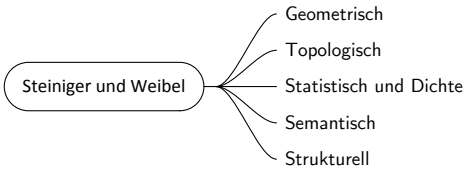


Abbildung 3.7: Klassifikation von Integritätsbedingungen nach Steiniger und Weibel (2005) und Steiniger (2007)

Integritätsbedingungen in der Generalisierung und Integritätsbedingungen in der Datenintegrität sind grundlegend ähnlich, was sich darin zeigt, dass sich die Klassen aus Steiniger und Weibel (2005) und Steiniger (2007) ebenso in den anderen vorgestellten Klassifikationen wiederfinden. So gelten die in dieser Arbeit vorgestellten Betrachtungen bezüglich der Anforderungen in Kapitel 4 und der Formalisierung in Kapitel 5 nahezu unverändert auch für die Generalisierung.

3.3 Formalisierung

Für die formale Spezifikation von Integritätsbedingungen sowie für die Prüfung von Daten bezüglich dieser Bedingungen werden viele unterschiedliche Ansätze in der Literatur vorgestellt. Die verschiedenen Ansätze spezifizieren die Integritätsbedingungen dabei auf teilweise sehr unterschiedlichen Abstraktionsebenen. Die vier möglichen Abstraktionsebenen sind dabei in Abbildung 3.8 angegeben, wobei die Domäne das höchste Abstraktionsniveau aufweist und das physikalische Modell das niedrigste Abstraktionsniveau. Diametral zur Abstraktion verhält sich die Konkretisierung, d.h. umso weniger abstrakt ein Modell ist, desto konkreter ist es.

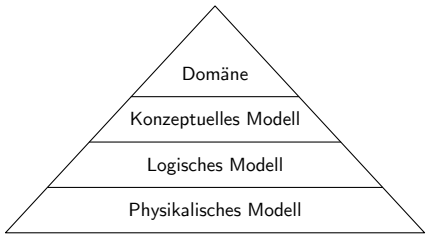


Abbildung 3.8: Abstraktionsebenen der Formalisierung

Die vier Abstraktionsebenen werden im Folgenden anhand des Beispiels einer Katasteranwendung vorgestellt. Die *Domäne* beschreibt die allgemeinen Rahmenbedingungen eines Modells und die darin vorkommenden Entitäten und deren Beziehungen. Die Domäne wird im englischen auch als 'universe of discourse' bezeichnet. Bildlich gesprochen wird ein leeres Universum um genau die Entitäten und Beziehungen zwischen den Entitäten erweitert, die für ein Modell beziehungsweise eine Anwendung benötigt werden. Für das Beispiel der Katasteranwendung werden in Fließtext die Entitäten Flurstück und Gebäude eingeführt und deren Eigenschaften und Beziehungen allgemein beschrieben. Im *konzeptuellen Modell* werden die Entitäten formal und plattformunabhängig beschrieben. Im Sinne der Model Driven Architecture (MDA) wird das konzeptuelle Modell eindeutig mit der Unified Modeling Language (UML) spezifiziert (siehe Unterabschnitt 2.3.1). Für die Katasteranwendung werden die Entitäten und deren Beziehungen dabei eindeutig in einem UML-Klassendiagramm beschrieben. Das *logische Modell* überführt das konzeptuelle Modell in ein plattformspezifisches Modell. Im Sinne der MDA kann dies beispielsweise durch die Überführung des UML-Modells in Ausdrücke in der Structured Query Language (SQL) erfolgen, die dann für die Erstellung eines Datenbankschemas verwendet werden können. Die Entitäten und Beziehungen der Katasteranwendung werden so in SQL-Ausdrücke zur Erstellung von zwei Tabellen Flurstueck und Gebaeude und weiteren Konstrukten zur Modellierung der Beziehungen zwischen den Tabellen überführt. Das *physikalische Modell* bezeichnet schließlich die Umsetzung beziehungsweise Implementierung auf genau einer Plattform. Das physikalische Modell entspricht dabei dem Code der MDA. Für die Katasteranwendung entspricht das physikalische Modell der Anpassung und Ausführung der SQL-Ausdrücke des logischen Modells für eine spezifische Datenbank, wie beispielsweise PostgreSQL (2013).

Im Folgenden werden die verschiedenen Ansätze in der Literatur in drei Gruppen unterteilt. In Unterabschnitt 3.3.1 wird die erste Gruppe vorgestellt, bei der räumliche objektorientierte Datenmodelle direkt um Integritätsbedingungen erweitert werden. In der zweiten Gruppe in Unterabschnitt 3.3.2 wird die Object Constraint Language (OCL) verwendet um Integritätsbedingungen zu spezifizieren. Die dritte Gruppe in Unterabschnitt 3.3.3 enthält schließlich eine Auflistung weiterer Ansätze, die nicht durch die beiden anderen Gruppen abgedeckt werden.

3.3.1 Räumliche objektorientierte Datenmodelle

Die zwei im Folgenden vorgestellten Modelle Perceptory und OMT-G erweitern die UML jeweils um geometrische Klassen beziehungsweise Typen. Für diese sind Integritätsbedingungen definiert, so dass Bedingungen einfach grafisch modelliert werden können.

Perceptory

Das in Bédard u. a. (2004) beschriebene Modellierungswerkzeug Perceptory verwendet vereinfachte UML-Klassendiagramme um die Struktur räumlicher Datenbanken zu definieren. So sind in den Klassendiagrammen beispielsweise keine Sichtbarkeiten und Datentypen enthalten. In Perceptory werden räumliche und zeitliche Datentypen durch entsprechende Piktogramme gekennzeichnet. Im Beispiel in Abbildung 3.9 sind die Klasse Straße sowie das Attribut fahrspuren Geometrien vom Typ Linie, wohingegen die Klasse Flurstück durch eine Fläche repräsentiert wird. Die räumliche Assoziation des Zugangs wird dabei wie eine reguläre Assoziation gekennzeichnet.

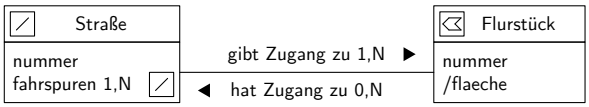


Abbildung 3.9: Räumliche Klassen und Assoziationen in Bédard u. a. (2004)

OMT-G

Das in Borges u. a. (1999, 2001, 2002) vorgestellte räumliche objektorientierte Datenmodell OMT-G basiert auf einer Erweiterung der UML[3] um räumliche Klassen, Beziehungen und Integritätsbedingungen. Als diskrete räumliche Klassen werden Punkte, Linien und Flächen aber auch uni- und bidirektionale Linien (Kanten) und Knoten definiert. Als kontinuierliche Klassen werden Isolinien, Tesselierung im Sinne eines räumlichen Gitters, planare Unterteilung in Polygone, Dreiecksvermaschungen sowie Stichproben definiert. Die Bedingungen sind Teil der Klassendefinition, so werden Isolinien als geschlossene Linien ohne Überschneidungen mit einheitlichem Wert definiert, wohingegen Linien nicht geschlossen sein dürfen. Die Integritätsbedingung für Isolinien nach Borges u. a. (2002) ist in Tabelle 3.3 wiedergegeben.

In OMT-G werden räumliche Beziehungen durch eine einfache und Netzwerkbeziehungen durch eine doppelte gestrichelte Linie dargestellt (siehe Abbildung 3.10). Konnektivitätsbedingungen für Netzwerke sind nicht für Klassen, sondern für Beziehungen definiert. Das Datenmodell unterstützt zudem räumliche Aggregation und konzeptuelle Generalisierung. Die Integritätsbedingungen für die Konnektivitätsbedingung von Kanten und für die räumliche Aggregation sind ebenfalls in Tabelle 3.3 augeführt.

	Integritätsbedingung
Klasse Isolinie	Sei F ein kontinuierliches Objekt. Seien v_0, v_1, ..., v_n $n+1$ Punkte in der Ebene. Seien $a_0 = \overline{v_0 v_1}$, $a_1 = \overline{v_1 v_2}$, ..., $a_{n-1} = \overline{v_{n-1} v_n}$ n Segmente die die Punkte verbinden. Diese Segmente formen eine Isolinie L dann und nur dann wenn (1) der Schnitt adjazenter Liniensegmente in L der einzige Extrempunkt ist, den die Segmente teilen ($a_i \cap a_{i+1} = v_{i+1}$) (2) nicht-adjazente Segmente sich nicht schneiden ($a_i \cap a_{i+1} = \varnothing$ für alle i, j mit $j \neq i+1$), und (3) der Wert von F an jedem Punkt P mit $P \in a_i, 0 \leq i \leq n-1$ konstant ist.
Beziehung zwischen Kante – Kante	Sei $G = \{A\}$ eine Netzwerkstruktur, zusammengesetzt aus einer Menge an Kanten $A = \{a_0, a_1, ..., a_q\}$. Dann muss folgende Bedingung erfüllt sein: (1) Jede Kante $a_k \in A$ muss eine Beziehung zu mindestens einer anderen Kante $a_i \in A$ mit $k \neq i$ aufweisen.
Räumliche Aggregation	Sei $P = \{P_0, P_1, ..., P_n\}$ eine Menge von räumlichen Objekten. Dann formt P ein anderes Objekt W durch räumliche Aggregation dann und nur dann wenn (1) $P_i \cap W = P_i$ für alle i mit $0 \leq i \leq n$, (2) $(W \cap \bigcup_{i=1}^{n} P_i) = W$ und (3) $((P_i \; touch \; P_j) \vee (P_i \; disjoint \; P_j)) = TRUE$ für alle i, j mit $i \neq j$.

Tabelle 3.3: Integritätsbedingungen in OMT-G nach Borges u. a. (2002)

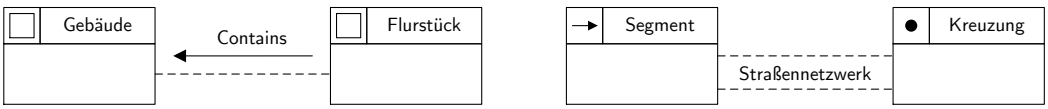


Abbildung 3.10: Räumliche und Netzwerkbeziehungen in OMT-G nach Borges u. a. (2002)

[3] In Borges u. a. (1999) basiert OMT-G noch auf der Object-Modeling Technique (OMT), einem Vorgänger der UML.

3.3.2 Object Constraint Language

In den im Folgenden vorgestellten Arbeiten wird die in Unterabschnitt 2.3.3 vorgestellte OCL um Konstrukte, Datentypen und Operationen erweitert, um unter anderem räumliche Integritätsbedingungen auf Basis der OCL ausdrücken zu können.

Integritätsbedingungen an mehrere Klassen

Bedingungen können in der OCL lediglich in Bezug auf den aktuellen Kontext definiert werden, d.h. sie sind beispielsweise auf die Instanzen einer einzelnen Klasse beschränkt. In Casanova u. a. (2000) wird eine Erweiterung der OCL vorgestellt, mit der sich eine Invariante auf mehrere Klassen des Modells beziehen kann. Im Programm 3.1 verweist die Invariante beispielsweise nicht nur auf den eigenen Kontext RestrictedManoeuvre, sondern zusätzlich auf die Instanzen einer zweiten Klasse ProhibitedManoeuvre.

```
context RestrictedManoeuvre
  inv: RestrictedManoeuvre.viaJunction = ProhibitedManoeuvre.viaJunction implies
      RestrictedManoeuvre.from <> ProhibitedManoeuvre.from
```

Programm 3.1: Integritätsbedingung an mehrere Klassen nach Casanova u. a. (2000)

Generische Integritätsbedingungen

Sogenannte parametrische Bedingungen erlauben in Casanova u. a. (2000) die Wiederverwendung einer Integritätsbedingung für mehrere Klassen, ähnlich den Templates in der Programmiersprache C++ oder Generics in Java. Als Beispiel dient im Programm 3.2 die Bedingung, dass Objekte der Klasse Service einen EntryPoint haben müssen. Diese parametrische Bedingung kann dann konkret für die Bedingung an die Klassen Restaurant wiederverwendet werden. Der parametrische Typ kann dabei hinsichtlich seiner Oberklasse(n), beispielsweise Service, oder seines Namens eingeschränkt werden.

```
entryPointBelongsToService<@param> where @param->supertypes->includes(Service):
  ServiceBelongingToService.services->select(s1 | s1->oclIsTypeOf(@param))->notEmpty() implies
  ServiceBelongingToService.services->select(s2 | s2->oclIsTypeOf(EntryPoint))->notEmpty()

-- Verwendung mit der Klasse Restaurant als @param:
context Restaurant
  inv: %entryPointBelongsToService<Restaurant>
```

Programm 3.2: Generische Integritätsbedingung nach Casanova u. a. (2000)

Geometrische Integritätsbedingungen

In Stoter u. a. (2008) werden geometrische Bedingungen definiert, die an Standards der ISO und des Open Geospatial Consortium (OGC) angelehnt sind. Es werden jedoch keine weiteren Details genannt, wie diese Standards umgesetzt sind. Im Programm 3.3 ist ein Beispiel einer Bedingung angegeben, die fordert dass Gebäude in der Karte einen Mindestabstand von 0,2 mm zueinander haben. Der OCL-Ausdruck ist jedoch nicht valide, da g2 sich auf eine einzelne Instanz der Klasse Gebaeude bezieht und nicht auf alle Instanzen der Klasse. Neben der Distanzberechnung wird in Stoter u. a. (2008) auch die Berechnung der Fläche eines Polygons in entsprechenden OCL-Ausdrücken verwendet.

```
context top50::Gebaeude
  inv: Distance(self.geometry, g2.geometry) >= 0.2 -- [mm], Gebaeude g2
```

Programm 3.3: Geometrische Integritätsbedingung nach Stoter u. a. (2008)

Xu (2011) erweitert die OCL um die in der ISO 19107 (2003) standardisierte Operation distance. Ein Beispiel für die Verwendung dieser Operation ist im Programm 3.4 angegeben. Dort wird gefordert, dass Gebäude mehr als 2 m von Bäumen entfernt sind.

Jedoch ist sowohl die in Stoter u. a. (2008) als auch die in Xu (2011) angegebene Invariante syntaktisch falsch, da die Distanzoperation des ISO 19107 (2003) Standards nicht korrekt verwendet wird. Der Unterschied der beiden Varianten wird im Programm 3.5 aufgezeigt.

```
context Gebaeude
  inv: Baum.allInstances()->forAll(b | distance(b.geometry, self.geometry) > 2)
```

Programm 3.4: Geometrische Integritätsbedingung nach Xu (2011)

```
context Gebaeude
  inv: Baum.allInstances()->forAll(b | distance(b.geometry, self.geometry) > 2) -- falsch
  inv: Baum.allInstances()->forAll(b | b.geometry.distance(self.geometry) > 2) -- korrekt, da die
      Operation als GM_Object::distance(geometry: GM_Object): Distance definiert ist
```

Programm 3.5: Valide und invalide geometrische Integritätsbedingung entsprechend des ISO 19107 Standards

Topologische Integritätsbedingungen

Eine Erweiterung der Basistypen und Operationen der OCL (siehe Tabelle 2.3) um geometrische Basistypen und topologische Operationen wird in Kang u. a. (2004), Duboisset u. a. (2005b) und Pinet (2010) vorgestellt. Die sogenannte $\text{OCL}_{9\text{IM}}$ erweitert die OCL um die drei geometrischen Primitive Punkt, Linie und Fläche sowie die entsprechenden topologischen Operationen der 9 Intersection Method (9IM) (siehe Tabelle 3.4). Zudem unterstützt die $\text{OCL}_{9\text{IM}}$ auch zusammengesetzte Geometrien (engl. 'composite geometries'), die als Liste von Geometrien in OCL abgebildet werden. Auf die Elemente der Liste kann dann mit den in OCL definierten Operationen zugegriffen werden (siehe Tabelle 2.3).

Typen	Topologische Operationen auf den Typen
Point, Polyline, Polygon	`->disjoint()`, `->contains()`, `->inside()`, `->equal()`, `->meet()`, `->covers()`, `->coveredBy()`, `->overlaps()`

Tabelle 3.4: Erweiterte Basistypen und topologische Operationen nach Duboisset u. a. (2005b)

Im Programm 3.6 sind zwei Beispiele für die Definition von Integritätsbedingungen in der $\text{OCL}_{9\text{IM}}$ angegeben. Die erste Bedingung fordert, dass die Geometrie eines Rathauses innerhalb der Geometrie der ihr zugeordneten Stadt liegen muss. Die zweite Bedingung verwendet zusammengesetzte Geometrien. Objekte der Klasse Schule können aus mehreren einzelnen Geometrien zusammengesetzt sein. Die Bedingung fordert, dass jede der Geometrien in einem Sektor enthalten sein muss.

In Stoter u. a. (2008) werden auch topologische Bedingungen definiert, für die jedoch auch keine weiteren Details genannt werden. Im Programm 3.7 ist ein Beispiel einer Bedingung angegeben, die fordert dass Gebäude und Straßen sich nicht überschneiden. Der OCL-Ausdruck ist wiederum nicht valide, da st sich auf eine einzelne Instanz der Klasse Strasse bezieht und nicht auf alle Instanzen der Klasse.

Die OCL-Bedingungen in van Bennekom-Minnema (2008) verwenden Ausdrücke auf dem Abstraktionsniveau des physikalischen Modells. Im Programm 3.8 ist ein Beispiel für eine Bedingung aus van Bennekom-Minnema (2008) angegeben, die fordert dass Flurstücksgeometrien überlappungsfrei sind. Diese verwendet den topologischen Befehl ST_Intersects, der spezifisch für die räumliche Datenbank PostGIS (2013) ist.

In Xu (2011) wird die Operation 'inside' der Oracle Spatial Datenbank als zusätzliche Operation für die OCL verwendet. Diese basiert auf der 9IM (siehe Abschnitt 2.1). Im Programm 3.9 ist ein Beispiel für eine entsprechende Integritätsbedingung angegeben, die fordert dass die dreidimensionale Geometrie eines Solarpanels nicht innerhalb eines Gebäudes sein darf. Der Ausdruck ist wiederum auf dem Abstraktionsniveau des physikalischen Modells, da das Ergebnis der Operation anstatt eines Wahrheitswerts ein Text ist, was spezifisch für die verwendete Oracle Datenbank ist.

Defizite der geometrischen und topologischen Integritätsbedingungen

Die vorgestellten Ansätze zur Modellierung geometrischer und topologischer Integritätsbedingungen mit Erweiterungen der OCL weisen einige Defizite auf, die im Folgenden zusammengefasst sind.

- Die geometrischen Bedingungen umfassen lediglich die Berechnung der Fläche und der Distanz.
- Die geometrischen Bedingungen in Stoter u. a. (2008) und Xu (2011) sind syntaktisch falsch.
- Die topologischen Bedingungen der in Kang u. a. (2004), Duboisset u. a. (2005b) und Pinet (2010) vorgestellten $\text{OCL}_{9\text{IM}}$ basieren auf Datentypen, die nicht standardisiert sind.

```
context Rathaus
  inv: self.geometry->inside(self.stadt.geometry)

context Schule
  inv: self.geometry->forAll(g | Sector.allInstances()->exists(sec | sec.geometry->contains(g)))
```

Programm 3.6: Integritätsbedingung in der OCL9IM nach Kang u. a. (2004) und Duboisset u. a. (2005b)

```
context top50::Gebaeude
  inv: Disjoint(self.geometry, st.geometry) -- Strasse st
```

Programm 3.7: Topologische Integritätsbedingung nach Stoter u. a. (2008)

– Die topologischen Bedingungen in Stoter u. a. (2008) sind syntaktisch falsch.

– Die topologischen Bedingungen in van Bennekom-Minnema (2008) und Xu (2011) nutzen Ausdrücke auf dem Abstraktionsniveau des physikalischen Modells und widersprechen damit dem Anspruch der OCL plattformunabhängig zu sein.

Zusammenfassend lässt sich feststellen, dass die vorgestellten Ansätze lediglich wenige Bedingungen unterstützen, nicht ausreichend auf räumlichen Standards basieren und die Grundprinzipien und Syntax der OCL nicht vollständig berücksichtigen. Die in Kapitel 5 vorgestellte GeoOCL weist diese Defizite dagegen nicht auf. Sie unterstützt eine Vielzahl geometrischer und topologischer Bedingungen, basiert auf den räumlichen Standards der OGC und der ISO und berücksichtigt die Grundprinzipien der OCL sowie deren Syntax.

Implementierung

Die Konvertierung von Bedingungen in der $\text{OCL}_{9\text{IM}}$ in Datenbank-Trigger (siehe Unterabschnitt 3.3.3) wird in Duboisset u. a. (2005b) beschrieben. Die Konvertierung wird mit dem frei verfügbare Werkzeug OCL2SQL durchgeführt, das in Demuth u. a. (2001) vorgestellt wird. Der Workflow der Konvertierung ist in Abbildung 3.11 dargestellt. Neben den Bedingungen besteht die Eingabe in den Generator aus dem UML Klassendiagramm, mit dem das den Bedingungen zugrundeliegende Modell definiert wird. Das Klassendiagramm wird dabei im Extensible Markup Language (XML) Metadata Interchange (XMI)-Format übergeben, einem von der Object Management Group (OMG) definierten Standard zum Austausch von (UML)-Modellen (OMG, 2011a). Die vom Generator erstellten Trigger sind auf dem Abstraktionsniveau des physikalischen Modells, d.h. sie sind spezifisch für eine bestimmte räumliche Datenbank, wie beispielsweise Oracle Spatial oder PostGIS. Mit demselben Workflow werden auch die Integritätsbedingungen in van Bennekom-Minnema (2008) in entsprechende Trigger konvertiert.

In Stoter u. a. (2008) werden die Integritätsbedingungen manuell in Datenbanksichten (engl. 'view') konvertiert. Eine Datenbanksicht ist eine logische Relation, die Abfragen als virtuelle Tabellen abbildet. In Stoter u. a. (2008) wird die Datenbanksicht so erstellt, dass diese alle Objekte enthält die eine entsprechende Integritätsbedingung verletzen.

Integritätsbedingungen werden in Xu (2011) manuell als Datenbank-Trigger umgesetzt. Komplexere Integritätsbedingungen werden zusätzlich durch Programmlogik in der proprietären prozeduralen Programmiersprache PL/SQL von Oracle definiert und dann in entsprechenden Triggern referenziert.

3.3.3 Weitere Ansätze

In der Literatur werden einige weitere Ansätze zur Definition von Integritätsbedingungen vorgestellt. Diese werden jedoch selten angewendet oder sind einem niedrigen Abstraktionsniveau zuzuordnen und damit nur

```
context Flurstueck
  inv: polygon->notEmpty() implies not exists(f: Flurstueck | ST_Intersects(self.polygon, f.polygon
      ))
```

Programm 3.8: Topologische Integritätsbedingung nach van Bennekom-Minnema (2008)

```
context StationaererSensor
  inv: Gebaeude.allInstances()->forAll(g | self.sensorTyp = 'Solarpanel' implies inside(self.
      geometry, g.geometry) = 'FALSE')
```

Programm 3.9: Topologische Integritätsbedingung nach Xu (2011)

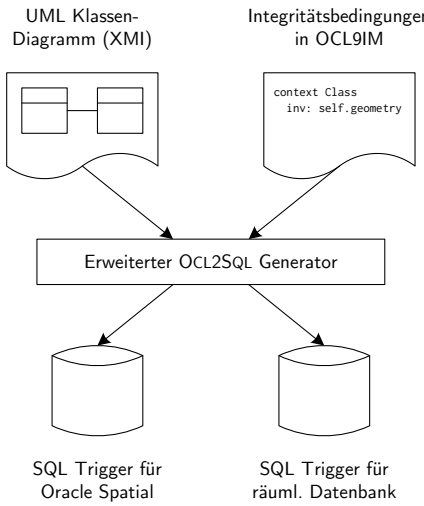


Abbildung 3.11: Konvertierung der Integritätsbedingungen in Datenbanktrigger nach Duboisset u. a. (2005b)

bedingt übertragbar. Im Folgenden werden Ansätze vorgestellt, die auf räumlichen Datenbanken, formalen Sprachen, der XML oder sonstigen Methoden basieren.

Räumliche Datenbanken

Integritätsbedingungen können in Datenbanken mit Assertions oder Triggern ausgedrückt werden. Mit Assertions können die typischen Bedingungen in Datenbanksystemen (siehe Unterabschnitt 3.2.1) um eigene Bedingungen erweitert werden (Elmasri und Navathe, 2011). Ein Beispiel für eine Assertion ist im Programm 3.10 aufgelistet. Die einzuhaltende Bedingung wird nach dem Schlüsselwort check aufgestellt und entspricht einer SQL-Abfrage, die für alle Zustände der Datenbank gültig sein muss. Im Beispiel müssen alle Elemente der Tabelle baeume einen Mindestabstand von 2 m von allen Elementen der Tabelle gebaeude aufweisen. Die Assertion kann dabei jedoch nur für räumliche Datenbanken erstellt werden, da sie sowohl einen Geometrietyp als auch eine räumliche Operation verwendet. Alternativ können Bedingungen auch mit Triggern ausgedrückt werden. Ein Trigger definiert welche Aktionen durchgeführt werden sollen, wenn ein Ereignis auftritt (Elmasri und Navathe, 2011). Typischerweise werden als Ereignisse die grundlegenden Datenbankoperationen 'create' (Element anlegen), 'read' (Element lesen), 'update' (Element aktualisieren) und 'delete' (Element löschen) verwendet. Als Aktion kann dann beispielsweise der Abbruch der Operation oder die Ausgabe einer Meldung an den Benutzer definiert werden.

```
CREATE ASSERTION min_dist_baum_gebaeude CHECK (NOT EXISTS (
  SELECT * FROM baeume b, gebaeude g WHERE ST_DISTANCE(b.geom, g.geom) < 2
));
```

Programm 3.10: Assertion

Assertions oder Trigger werden für die Aufstellung und Überprüfung von Integritätsbedingungen sowohl für relationale Datenbanken (Louwsma (2004), Louwsma u. a. (2006), van Oosterrom (2006)) als auch für objektorientierte Datenbanken (Ditt u. a. (1997)) verwendet. Der entscheidende Nachteil von Assertions und Triggern ist jedoch, dass diese dem Abstraktionsniveau des physikalischen Modells zuzuordnen sind, da diese für räumliche Bedingungen für genau eine Plattform erstellt werden müssen. So wird die Berechnung der räumlichen Distanz im Programm 3.10 mit der Operation ST_DISTANCE durchgeführt, die für das Datenbanksystem PostgreSQL (2013) mit der Erweiterung PostGIS (2013) definiert ist. Die gleiche Assertion müsste beispielsweise für das Datenbanksystem Oracle Spatial mit der dort definierten Operation SDO_WITHIN_DISTANCE aufgestellt werden.

Formale Sprachen

Hadzilacos und Tryfona (1992) definieren Integritätsbedingungen in einer formalen Pseudosprache die auf Prädikatenlogik basiert. Die verwendete Prädikatenlogik basiert auf den Operationen Negation, Konjunktion, Disjunktion sowie der universellen und existentiellen Quantifizierung.

Integritätsbedingungen werden in Christensen (2007) in einer sogenannten High Level Constraint Language (HLCL) spezifiziert. Die Basiskonstrukte der HLCL sind Bedingungen der Form 'alle müssen' (engl. 'all-must') und 'keine dürfen' (engl. 'no-may'). Weitere Sprachkonstrukte sind Quantifikatoren (alle, nur, mindestens, maximal, exakt), benutzerdefinierte Variablen, Vergleichsoperatoren (kleiner als, größer als, gleich, ist Element einer Auflistung) und benutzerdefinierte Prädikate. Letztere bilden räumliche Informationen ab, die nur implizit in den Daten enthalten sind, beispielsweise metrische oder topologische Beziehungen zwischen Objekten.

Die verwendeten formale Sprachen eignen sich für die Definition von Integritätsbedingungen als Teil des konzeptuellen Modells. Diese konkurrieren theoretisch mit der OCL. Sie weisen jedoch weder die Bekanntheit noch die Maturität von OCL auf, weshalb sie für diese Arbeit nicht berücksichtigt werden.

XML

Für die Aufstellung von Integritätsbedingungen im Kontext von Webservices verwenden Mäs u. a. (2005) und Watson (2007, 2008) die Semantic Web Rule Language (SWRL), die in W3C (2004) definiert wird. Die SWRL entstammt dem Forschungsbereich des Semantic Web und beschreibt Bedingungen an die Elemente einer Ontologie.

XML-basierte Ansätze zur Aufstellung von Integritätsbedingungen sind dem Abstraktionsniveau des logischen Modells zuzuordnen, da XML plattformspezifisch ist. Zudem sind umfangreiche XML-Dateien durch die Vielzahl der Auszeichnungselemente häufig unübersichtlich, und damit manuell schwerer interpretierbar.

Sonstige

In Ubeda (1997), Ubeda und Egenhofer (1997) und Servigne u. a. (2000) werden topologische Integritätsbedingungen im Format

Bedingung = (Klasse A, Relation, Klasse B, Spezifikation)

definiert. Die Relation entspricht dabei einer Relation aus der 9IM. Die Spezifikation gibt die Anzahl der Objekte an und kann aus den Begriffen 'verboten', 'mindestens n mal', 'maximal n mal' und 'exakt n mal' gewählt werden. Die Bedingungen werden vom Benutzer über eine grafische Benutzeroberfläche erstellt.

Ein Repositorium (engl. 'repository') bildet die Grundlage der Prüfung von Integritätsbedingungen in Cockcroft (2004). Es enthält neben den Bedingungen auch das vollständige Datenmodell, Informationen zur Visualisierung der Daten sowie Metadaten bezüglich der Datenqualität. Bedingungen werden vom Benutzer über eine grafische Benutzeroberfläche erstellt und im Repositorium gespeichert. Das Repositorium übersetzt dann Bedingungen an nicht-räumliche Attribute in Anweisungen in der DDL, die vom Datenbanksystem Microsoft Access überprüft werden. Bedingungen an räumliche Attribute werden dagegen in Abfragen an das Geoinformationssystem (GIS) MapInfo übersetzt, in dem auch die räumlichen Daten repräsentiert werden.

	Beschreibung	Spezifikation
Ereignis	KlasseA.Methode	Wahr / falsch
Bedingung	KlasseA.räumlicheBeziehung.KlasseB KlasseA.attribut(Attributname)	Wert mit mathematischem Operator (=, >, ! =, ...)
Aktion	KlasseA.Methode	Wahr / falsch

Tabelle 3.5: Entscheidungstabelle für Integritätsbedingungen nach Wang und Reinhardt (2007)

Regeln werden in Wang und Reinhardt (2007) in Entscheidungstabellen für Integritätsbedingungen (engl. 'constraint decision table') ausgedrückt. Diese implementieren sogenannte Event-Condition-Action Regeln, in denen das Ereignis (die Ursache der Prüfung, engl. 'event'), die zu prüfenden Bedingungen inklusive deren Resultat (engl. 'condition') sowie mögliche darauf folgende Aktionen (engl. 'action') kodiert werden. Der Aufbau einer solchen Entscheidungstabelle ist in Tabelle 3.5 abgebildet. Die Methoden einer Klasse aktualisieren die Geometrie eines Objektes, erzeugen oder löschen Objekte und erlauben den lesenden und schreibenden Zugriff auf deren Attribute. Räumliche Beziehungen umfassen topologische Relationen, die metrischen Relationen Distanz und Richtung, sowie die Teil – Ganzes Beziehung. Die Beschreibung enthält den spezifischen Ausdruck des

Ereignisses, der Bedingung und der Aktion. Die Spezifikation zeigt schließlich das Ergebnis des Ausdrucks. Zusätzlich können Ereignisse, Bedingungen und Aktionen aus mehreren Elementen zusammengesetzt werden, die durch die logischen Relationen 'und' sowie 'oder' verknüpft sind.

4 Integritätsbedingungen: Anforderungen

An Integritätsbedingungen werden vielseitige Anforderungen gestellt, die beispielsweise vom verwendeten Datenmodell abhängen. In diesem Kapitel wird das breite Spektrum von Anforderungen an Integritätsbedingungen vorgestellt und in einem Anforderungskatalog mit insgesamt 27 Anforderungen zusammengefasst. Die Aufstellung basiert auf einer umfassenden und detaillierten Auswertung zahlreicher Arbeiten und ermöglicht so erstmals einen vollständigen Überblick über alle wichtigen Aspekte von Integritätsbedingungen.

Die Aufstellung des Anforderungskatalogs verfolgt dabei mehrere Ziele. Erstens soll sie als strukturierte Referenz für Experten dienen, die Integritätsbedingungen für ein bestimmtes Datenmodell aufstellen und überprüfen wollen. Dabei sind für eine konkrete Anwendung voraussichtlich nicht alle 27 Anforderungen von Bedeutung, jedoch ermöglicht die detaillierte Beschreibung und Diskussion der Anforderungen eine fundierte Entscheidung über die tatsächlich benötigten Anforderungen. Zweitens soll durch die Aufstellung des Anforderungskatalogs der aktuelle Stand der Forschung strukturiert zusammengefasst, gegenübergestellt und diskutiert werden. Die Aufstellung kann somit zum Vergleich von unterschiedlichen Arbeiten herangezogen werden und als Grundlage für die Identifikation offener Forschungsfragen dienen. Drittens soll die Aufstellung die Zusammenhänge zwischen den Anforderungen aufzeigen und dadurch ein strukturierteres Verständnis von Integritätsbedingungen ermöglichen. Schließlich bilden die Anforderungen die Grundlage für die Entwicklung einer formalen Sprache zur Definition von Integritätsbedingungen, siehe Kapitel 5.

Die im Folgenden vorgestellten 27 Anforderungen des Anforderungskatalogs sind in fünf Gruppen unterteilt, welche die unterschiedlichen Schritte bei der Aufstellung, Prüfung und Modifikation von Integritätsbedingungen umfassen (siehe Abbildung 4.1). Die Entscheidung darüber, welche Integritätsbedingungen für eine Anwendung wichtig sind, wird bereits durch die Modellierung der Daten (Abschnitt 4.1) getroffen. Sind beispielsweise kontinuierliche Objekte im Modell vorgesehen, dann können für diese entsprechende Bedingungen aufgestellt werden. Für die Definition von Integritätsbedingungen (Abschnitt 4.2) werden einige grundlegende Mechanismen benötigt, die beispielsweise definieren auf welche Daten eines Modells zugegriffen werden kann. Bei der Aufstellung einzelner Integritätsbedingungen (Abschnitt 4.3) sind einige Aspekte bezüglich deren Formulierung und Interpretation zu berücksichtigen. Sind alle Integritätsbedingungen für ein Datenmodell aufgestellt, dann kann die Prüfung von Daten auf Übereinstimmung mit den Bedingungen (Abschnitt 4.4) erfolgen. Schließlich können als Ergebnis der Prüfung sowohl die Daten als auch die Integritätsbedingungen geändert werden (Abschnitt 4.5).

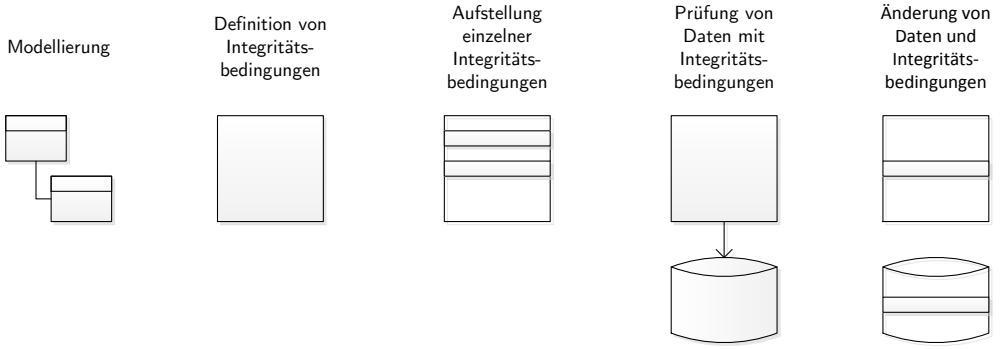


Abbildung 4.1: Gruppierung der Anforderungen an Integritätsbedingungen

4.1 Modellierung

Integritätsbedingungen müssen sich am verwendeten Datenmodell einer Anwendung orientieren. Grundlegende Entscheidungen bei der Erstellung eines Modells, wie beispielsweise ob Daten zwei- oder dreidimensional modelliert werden, führen zu grundlegend unterschiedlichen Bedingungen. Es ist deshalb wichtig, bereits bei der Modellierung die entsprechenden Bedingungen und deren Bedeutung zu berücksichtigen.

Die acht im Folgenden vorgestellten Anforderungen der Modellierung an Integritätsbedingungen decken das breite Spektrum von Möglichkeiten zur Modellierung räumlicher Daten ab. So können räumliche Objekte diskret

oder kontinuierlich modelliert werden (Anforderung 1). Integritätsbedingungen können nicht nur für einzelne Zustände der Daten, sondern auch für den Übergang zwischen diesen Zuständen aufgestellt werden (Anforderung 2). Die Dimension der Objekte spielt eine weitere wichtige Rolle. Diese bezieht sich auf die Unterscheidung zwischen Daten in 2D, 2,5D und 3D (Anforderung 3) sowie auf die Zeit, die sogenannte vierte Dimension (Anforderung 4).

Auch für komplexere Aspekte der Modellierung räumlicher Daten können entsprechende Integritätsbedingungen aufgestellt werden. Dazu gehören topologische Netzwerke (Anforderung 5), zusammengesetzte Geometrien und die Aggregation von Objekten (Anforderung 6), die konzeptuelle Generalisierung (Anforderung 7) sowie unscharfe Geometrien (Anforderung 8).

Anforderung 1: Diskrete und kontinuierliche Objekte

> Unterstützung von diskreten Objekten (Punkt, Linie, Fläche, ...) und kontinuierlichen Objekten (Isolinien, TIN, Raster, ...)

Friis-Christensen u. a. (2001) weisen auf die Wichtigkeit der Unterscheidung zwischen diskreten und kontinuierlichen Objekten hin, da diese sich fundamental in ihren Eigenschaften unterscheiden.

Ausgehend von der Semantik diskreter und kontinuierlicher Objekte und ihrer Definition stellen Borges u. a. (2002) Integritätsbedingungen für diese auf. Unterstützte kontinuierliche Objektklassen sind Isolinien, Tesselierung im Sinne eines räumlichen Gitters, planare Unterteilung in Polygone, Dreiecksvermaschungen sowie Stichproben. Die Bedingungen sind dabei fester Teil der Klassendefinition.

Diskussion Es unterscheiden sich nicht nur die Eigenschaften von diskreten und kontinuierlichen Objekten fundamental, sondern auch deren Integritätsbedingungen.

Die strikte Definition der kontinuierlichen Objektklassen weist einige Besonderheiten auf. Beispielsweise müssen Dreiecksvermaschungen nicht nur flächendeckend sein, sondern es werden auch oft zusätzliche Kriterien wie das Delaunay-Kriterium oder Bedingungen an die Innenwinkel der Dreiecke gestellt. Die Bedingung der Flächendeckung kann nur geprüft werden falls die gesamte Ausdehnung der Daten bekannt ist, beispielsweise durch die Angabe eines umschließenden Polygons. So können Lücken innerhalb der kontinuierlichen Daten und an deren Rand aufgedeckt werden. Aus Sicht der objektorientierten Datenmodellierung stellt ein kontinuierliches Objekt eine Aggregation (siehe Anforderung 6) aus diskreten Objekten dar, beispielsweise besteht ein räumliches Gitter aus einer Aggregation von Gitterzellen.

Kontinuierliche Daten ermöglichen zudem spezielle Operationen auf den Daten und damit mögliche Ansatzpunkte für Integritätsbedingungen. So lassen sich für einige Objektklassen Gradienten bilden, Interpolationen durchführen oder andere Daten ableiten. Interpolationen sind beispielsweise wichtig für Stichproben, um an räumlich nicht verprobten Koordinaten Attributwerte durch geostatistische Verfahren interpolieren zu können. Ein Beispiel für die Ableitung von Daten ist die Erstellung eines Voronoi-Diagramms aus einer Dreiecksvermaschung.

Anforderung 2: Statisch und dynamisch

> Unterstützung von statischen, dynamischen und dynamisch sequentiellen Integritätsbedingungen

Die Definition statischer und dynamischer Integritätsbedingungen basiert auf deren Definition in nicht-räumlichen Datenbanksystemen (siehe Unterabschnitt 3.2.1).

Fahrner u. a. (1997) definieren darüberhinaus dynamisch sequentielle Bedingungen[1], welche mögliche Sequenzen von Übergängen einschränken. Als Beispiel für eine dynamisch sequentielle Bedingung wird in Cockcroft (1997) die Bedingung 'wenn ein Mitarbeiter entlassen wird, dann kann danach dessen Gehalt nicht mehr erhöht werden' angeführt.

In Xu (2011) wird eine Integritätsbedingung für mobile Sensoren aufgestellt, die als dynamisch sequentiell bezeichnet werden kann. Aufgrund der Ungenauigkeit der Positionsbestimmung des Sensors können Messungen in der Nähe von Gebäuden fehlerhaft bezüglich der Topologie von Sensor und Gebäude sein. Der Sensor ist dann fälschlicherweise innerhalb des Gebäudes, obwohl er außerhalb sein müsste und umgekehrt. Xu (2011) bezeichnet

[1] In Fahrner u. a. (1997) werden diese als 'dynamisch' bezeichnet. Da dieser Begriff jedoch schon im Kontext nicht-räumlicher Datenbanksysteme belegt ist, wird der entsprechend angepasste Begriff in dieser Arbeit verwendet.

dieses Verhalten als springen (engl. 'jump'). Die Bedingung prüft die Topologie von Sensor und Gebäude mit einem gleitenden Fenster aus drei Punkten. Ist diese in der Reihenfolge 'innen, außen, innen' oder umgekehrt 'außen, innen, außen', dann wird der zweite Punkt jeweils als Fehler interpretiert.

In Mäs und Reinhardt (2009) werden Bedingungen erwähnt, die Änderungen zwischen mehreren konsekutiven Versionen eines Objektes oder Gruppen von Objekten beschränken. Diese können auch als dynamisch sequentiell bezeichnet werden.

Diskussion Statische Bedingungen beziehen sich auf exakt einen Zustand der Daten und sind damit am einfachsten zu prüfen.

Dynamische Bedingungen beziehen sich auf zwei Zustände, nämlich den aktuellen Zustand der Daten und den Zustand der Daten nach Durchführung der geplanten Modifikationen. Der Übergang lässt sich dabei als Transaktion im Sinne einer Datenbank oder als Operation im Sinne der objektorientierten Entwicklung abbilden. Der aktuelle Zustand und die Menge an Modifikationen, die auch als Delta bezeichnet werden, bilden die Eingabe der Operation. Die Operation wendet die Modifikationen auf den aktuellen Zustand an und ermittelt so den Zielzustand der Daten. Dieser kann schließlich auf dieselbe Art wie ein statischer Zustand geprüft werden. Ist der Zielzustand gültig, dann werden die Modifikationen übernommen. Ist der Zielzustand hingegen nicht gültig, dann werden die Modifikationen verworfen und der Benutzer entsprechend informiert.

Dynamisch sequentielle Bedingungen werden selten für Daten aufgestellt, da oft statische und dynamische Bedingungen zur Prüfung der Datenintegrität ausreichen und diese Klasse von Bedingungen außerdem besondere Anforderungen an die Daten und deren Modellierung stellt. Dynamisch sequentielle Bedingungen benötigen für ihre Umsetzung eine Historisierung der Daten und damit expliziten Zeitbezug (siehe Anforderung 4). Die Historisierung ermöglicht den Zugriff auf frühere Zustände eines Objektes und dessen Beziehungen, um so mögliche Änderungen der Daten überprüfen zu können. Im Allgemeinen kann diese Historisierung Bestandteil eines Objektes sein, d.h. es werden alle Zustände eines Objektes explizit gespeichert, oder sie kann auf einem Änderungsprotokoll basieren, das dann bei Bedarf zeitlich rückwärts abgearbeitet wird. Gegebenenfalls reicht es dabei aus nur eine feste Anzahl an früheren Zuständen zu speichern.

Anforderung 3: 2,5D und 3D

Unterstützung von Objektgeometrien mit 2,5 und 3 Dimensionen

Integritätsbedingungen für dreidimensionale Objekte von Gebäuden, Bäumen, Straßen, Gras, Wasser, Terrain und Sensoren werden in Xu (2011) aufgestellt. Durch die Berücksichtigung der dritten Dimension müssen einige zusätzliche Aspekte beachtet werden. Geometrien von Objekten die über ein Volumen definiert sind, wie beispielsweise Gebäude, können entweder durch die Kombination eines oder mehrerer Körper (engl. 'solid') definiert werden oder durch die Kombination mehrerer Flächen (engl. 'surface') als sogenannte Randbeschreibung. Zudem müssen geometrische und topologische Beziehungen auf die dritte Dimension erweitert werden, wie beispielsweise die Abstandsberechnung oder die Ermittlung der binären topologischen Relation zweier Objekte. So definiert Xu (2011) die topologische Relation 'strong touch', die ausdrückt dass sich zwei Objekte nur an ihren Außenflächen beziehungsweise Rändern berühren. Demnach ist beispielsweise ein Gebäudekörper der auf einer Straßenfläche steht, also deren Inneres berührt, nicht zulässig, wohingegen ein Gebäudekörper den Rand einer Straßenfläche berühren darf. Bei der Bedeutung topologischer Relationen ist auch die Semantik beziehungsweise Modellierung von Objekten wichtig. So dürfen sich zwei Gebäude nicht überschneiden, wohingegen sich die Kronen zweier Bäume durchaus überschneiden dürfen. Eine zusätzliche Komplexität entsteht in Xu (2011) durch die Verwendung von CityGml, einem Standard zur Erstellung von 3D-Stadtmodellen. Die OGC CityGML (2012) erlaubt die Modellierung von Objekten in fünf diskreten Skalenbereichen beziehungsweise Detailgraden, sogenannten Level of Detail (LoD). Ein Gebäude mit Satteldach wird im LoD 1 als Quader ohne Dachstruktur repräsentiert. Erst im LoD 2 werden differenzierte Dachstrukturen modelliert. Die Überprüfung ob eine in der Nähe eines Gebäudes verlaufende Freileitung dessen Dach schneidet kann so unterschiedliche Ergebnisse in den beiden genannten Skalenbereichen liefern.

Diskussion Die Verwendung von Objektgeometrien mit 2,5 und 3 Dimensionen erweitern sowohl die Modellierung der Daten als auch deren Beziehungen. Die Berücksichtigung der dritten Dimension eignet sich sowohl für diskrete als auch kontinuierliche Objekte (siehe auch Anforderung 1).

Auch die Vielfalt der möglichen Methoden beziehungsweise Operationen auf den Daten wird durch die zusätzliche Dimension erweitert. Beispiele dafür sind Sichtbarkeitsanalysen, die Berechnung von Gradienten sowie die physikalische Modellierung der Ausbreitung von Flüssigkeiten und Wind.

Anforderung 4: Zeit

Unterstützung der Zeit als Dimension

Aspekte zeitlicher Integritätsbedingungen werden unter anderem in Christensen (2007), Pinet (2012), Louwsma (2004), van Oosterrom (2006), Xu (2011), Mäs und Reinhardt (2009), Bédard u. a. (2004), Salehi (2009) und Salehi u. a. (2011) behandelt. Dort sind auch Beispiele für deren Nutzung angegeben.

Diskussion Die Zeit, die oft als auch die vierte Dimension bezeichnet wird, basiert auf den drei Konzepten Zeitpunkt, Intervall und Zyklus.

Zeitliche Intervalle entsprechen einem linearen Modell der Zeit. Die Beziehungen von zeitlichen Intervallen können durch die sieben von Allen (1983) identifizierten Relationen und deren Inversen abgebildet werden (siehe Unterabschnitt 2.1.4). Als Ergänzung zum linearen Modell der Zeit werden in Hornsby u. a. (1999) zeitliche Zyklen und deren Relationen diskutiert, um beispielsweise jahreszeitliche Phänomene modellieren zu können.

Ein Aspekt der Modellierung der Zeit ist die Frage, ob nur auf Zustände eines Objekts zu diskreten Zeitpunkten zugegriffen werden kann, oder ob zwischen den Zuständen interpoliert werden darf. Dürfen Zustände interpoliert werden, dann muss eine Interpolationsvorschrift für die Bestimmung von Attributwerten zwischen vorhandenen Zeitpunkten angegeben werden. Ein Beispiel für eine einfache Interpolationsvorschrift ist die Aufstellung einer Geradengleichung durch die Attributwerte der beiden angrenzenden Zeitpunkte mit nachfolgender Interpolation am gewünschten Zeitpunkt. Die Interpolationsvorschrift kann jedoch auch bedeutend komplexer werden, wie beispielsweise durch die Angabe detaillierter Funktionen oder durch die Interpolation über mehrere angrenzende Zeitpunkte.

Die Anforderung nach der Unterstützung der Zeit als Dimension ist eng mit der Anforderung nach dynamischen und dynamisch sequentiellen Integritätsbedingungen verwandt (Anforderung 2).

Anforderung 5: Topologische Netzwerke

Unterstützung von topologischen Netzwerken

Im Datenmodell OMT-G von Borges u. a. (2002) sind den topologischen Primitiven Knoten und uni- sowie bidirektionale Kante die geometrischen Primitive Punkt und Linie zugeordnet. Ein Objekt der Klasse Knoten hat demnach auch einen zugeordneten Punkt und damit eine Koordinate. Die Konnektivitätsbedingungen fordern dass jedem Knoten mindestens eine Kante und jeder Kante exakt zwei Knoten zugeordnet sind. In OMT-G können topologische Netzwerke auch ohne Knoten aufgebaut werden, indem Verweise zwischen den Kanten definiert werden. Die Konnektivitätsbedingung fordert dann, dass jeder Kante mindestens eine andere Kante zugeordnet ist.

Christensen (2007) gibt an, dass topologische Integritätsbedingungen Konnektivitätsregeln in Netzwerken ausdrücken können. Als Beispiel wird die Bedingung angegeben, dass Straßensegmente in einem Straßennetzwerk keine Pseudoknoten aufweisen dürfen, also eine Kreuzung der Schnittpunkt von mindestens drei Kanten sein muss.

Diskussion Durch Konnektivitätsbedingungen kann nicht nur gefordert werden, dass Verbindungen zwischen den topologischen Elementen bestehen, sondern auch welche Objektklassen miteinander verbunden werden dürfen. Wie oft diese Verbindungen auftreten dürfen, kann durch Bedingungen an den Grad von Knoten definiert werden. Durch Hinzunahme gerichteter Kanten können zudem Integritätsbedingungen definiert werden, die den korrekten Fluss im Netzwerk fordern.

Repräsentiert ein topologisches Netzwerk geometrische Primitive, wie beispielsweise Straßensegmente und Kreuzungen, dann sind auch geometrische Integritätsbedingungen wichtig um die Konnektivität des gesamten Netzwerkes zu sichern. Diese können beispielsweise Fehler wie unter- und überschneidende (engl. 'undershoot' und 'overshoot') Liniensegmente sowie zu geringe Abstände zwischen Knoten aufdecken.

Weitere Typen von Integritätsbedingungen für topologische Netzwerke ergeben sich aus den Algorithmen die auf diesen arbeiten, beispielsweise die Suche nach kürzesten oder optimalen Routen zwischen zwei Knoten im Netzwerk.

Anforderung 6: Zusammengesetzte Geometrien und Aggregation

Unterstützung von zusammengesetzten Geometrien und der Aggregation von Objekten

Die Anforderungen nach der Unterstützung zusammengesetzter Geometrien und der Aggregation von Objekten sind miteinander verwandt. Zusammengesetzte Geometrien treten auf, wenn ein Objekt mehrere Geometrien aufweist, beispielsweise wenn räumlich getrennte Gebäude eines Gebäudekomplexes als eine einzige Objektgeometrie abgebildet werden. Durch die Aggregation von Objekten werden mehrere Objekte zu einem neuen Objekt zusammengefasst. Eine besondere Ausprägung einer Aggregation ist dabei die Komposition, die auch als Teil – Ganzes Beziehung bezeichnet wird. In einer Komposition ist jede Geometrie eines Teils vollständig in der Geometrie des Ganzen enthalten. Zusätzlich überlappen sich die Geometrien der Teile nicht und die Geometrie des Ganzen wird vollständig von den Geometrien der Teile überdeckt. Die Aggregation von Stadtbezirken zu einer Stadt bildet beispielsweise eine Komposition.

Das Datenmodell von Kang u. a. (2004) und Pinet u. a. (2005) unterstützt die Angabe einer Menge von Geometrien für ein einzelnes Objekt. Die Menge darf zudem aus unterschiedlichen geometrischen Primitiven zusammengesetzt sein. Sogenannte Typbedingungen (engl. 'type constraints') schränken für eine Klasse die möglichen Geometrietypen und deren Anzahl ein. Die Definition erfolgt in Textform, wobei die drei Operatoren XOR (Polygon XOR Punkt), AND (Polygon AND Polylinie) sowie MULT m (Polygone MULT(1..5)) verwendet werden. Die Operatoren können auch zu komplexeren Ausdrücken kombiniert werden ((Punkt AND Polygon) MULT(1..∗)).

Durch die Aggregation von Geometrien für einzelne Objekte können nach Kang u. a. (2004) binäre topologische Relationen zwischen zwei Objekten zu vier verschiedenen Konstellationen führen, die jeweils unterschiedliche Anzahlen für die beteiligten Geometrien der Objekte A (schwarze Geometrien) und B (graue Geometrien) aufweisen (siehe Tabelle 4.1).

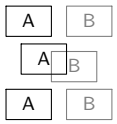
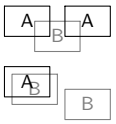
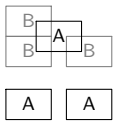
Anzahlen	1..∗ von A 1..∗ von B	Alle von A 1..∗ von B	1..∗ von A Alle von B	Alle von A Alle von B
Beispiel				

Tabelle 4.1: Konstellationen für die binäre topologische Relation 'overlap' nach Kang u. a. (2004)

In der formalen Sprache von Duboisset u. a. (2005a,b) können binäre topologische Relationen auch für Objekte mit zusammengesetzten Geometrien definiert werden. Dazu wird in Duboisset u. a. (2005a) eine Erweiterung von topologischen Relationen durch die zusätzliche Angabe von Adverbien vorgestellt, die auf den Definitionen von Claramunt (2000) basiert. Soll allgemein die topologische Relation der Geometrien eines Objektes mit m Geometrien und eines zweiten Objektes mit n Geometrien bestimmt werden, so ergibt sich insgesamt eine Matrix mit $8^{m \times n}$ Elementen bei der Verwendung der 9 Intersection Method (9IM)[2]. In Claramunt (2000) wird zur Reduktion der Komplexität die Angabe eines einzelnen Adverbs anstatt der Matrix von Relationen vorgestellt. Für die binäre topologische Relation zweier Objekte mit zusammengesetzten Flächen werden insgesamt sechs Adverbien definiert, welche die räumliche Konfiguration der Objekte beschreiben. Die sechs Adverbien des Adverb-Modells sind in ihrer allgemeinen Form in Tabelle 4.2 definiert. In Tabelle 4.3 sind die sechs Adverbien für die Relation 'meets' zweier Objekte A (schwarze Geometrien) und B (graue Geometrien) dargestellt.

Das Datenmodell OMT-G von Borges u. a. (2002) unterstützt die räumliche Aggregation zwischen zwei räumlichen Klassen. Das Klassendiagramm selbst gibt keine Auskunft darüber ob die Teile automatisch aus dem Ganzen oder das Ganze automatisch aus den Teilen abgeleitet werden kann.

In Belussi u. a. (2006) wird das Datenmodell GeoUML vorgestellt, das zwei Klassen zur Definition von Aggregationen vorsieht. Die Klasse GM_Aggregate stellt keine Anforderungen an die Menge der enthaltenen Klassen, wohingegen die Klasse GM_Complex einer Komposition von geometrischen Primitiven entspricht. Nach Belussi u. a. (2006) müssen für alle Kompositionen sogenannte strukturelle Bedingungen definiert werden, welche die Klassen des Ganzen und der Teile explizit festlegen. Dabei werden beide Richtungen der Beziehung festgelegt. Für alle Objekte der Klasse Teil wird gefordert, dass sie Teil von mindestens einem Objekt der Klasse Ganzes sind. Dadurch werden lose, d.h. nicht referenzierte Objekte, der Klasse Teil vermieden. Entsprechend wird für alle Objekte der Klasse Ganzes gefordert, dass sie vollständig aus Objekten der Klasse Teil zusammengesetzt

[2] Selbst bei der Verwendung von Schnittmatrizen anstatt der benannten topologischen Relationen ergibt sich eine Matrix mit $m \times n$ Elementen.

Adverb	Beschreibung
Mostly	Alle B haben Beziehung zu A, einige A können keine Beziehung zu B haben
MostlyRev	Alle A haben Beziehung zu B, einige B können keine Beziehung zu A haben
Completely	Alle A haben Beziehung zu B und alle B haben Beziehung zu A
Partially	Mindestens ein A hat Beziehung zu B und alle anderen Beziehungen sind 'disjoint'
Occasionally	Mindestens ein A hat Beziehung zu B
Entirely	Alle A haben Beziehung zu B
Never	Kein A hat Beziehung zu B

Tabelle 4.2: Adverb-Modell nach Claramunt (2000)

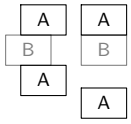
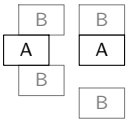
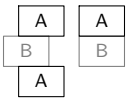
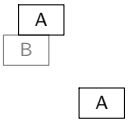
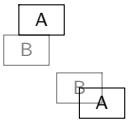
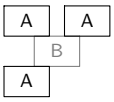
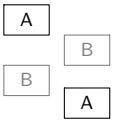
Adverb	Mostly	MostlyRev	Completely	Partially	Occasionally	Entirely	Never
Beispiel							

Tabelle 4.3: Adverb-Modell für die binäre topologische Relation 'meet' nach Duboisset u. a. (2005a)

sind. Durch die strukturellen Bedingungen wird beispielsweise verhindert, dass Objekte der Klasse Straße aus Objekten der Klasse Schienenelement zusammengesetzt sind.

In Rodríguez (2005) werden verschachtelte (engl. 'nested') Aggregationen als grundlegender Abstraktionsmechanismus für die Modellierung räumlicher Phänomene bezeichnet. Eine verschachtelte Aggregation enthält mehrere hintereinander ausgeführte Aggregationen, die so mehrere Ebenen bilden. Ein Beispiel für eine verschachtelte Aggregation ist die Aggregation von Gemeinden zu Landkreisen, von Landkreisen zu Bundesländern und schließlich von Bundesländern zu einem Staat.

Diskussion Ein neuer Typ von Integritätsbedingungen ist notwendig um die Zusammensetzung von Geometrien und Objekten zu überprüfen. Dieser schränkt die erlaubten geometrischen Primitive und deren Anzahl für zusammengesetzte Geometrien ein. Analog werden die erlaubten Klassen in einer Aggregation eingeschränkt. Auch hier ist eine Angabe der Anzahl, in diesem Fall der Objekte der Teile und des Ganzen, sinnvoll (siehe Anforderung 9).

Durch zusammengesetzte Geometrien und durch die Aggregation von Objekten verändern sich auch die Beziehungen zwischen Objekten, da ein Objekt nun mehrere Geometrien umfassen kann. Mögliche Lösungen für topologische Beziehungen werden von Kang u. a. (2004) und Pinet u. a. (2005) mit der Angabe von Kardinalitäten sowie von Duboisset u. a. (2005a) mit der Verwendung des Adverb-Modells angegeben. Aber auch andere Arten von Beziehungen werden dadurch beeinflusst. Beispielsweise lässt sich die Distanz zwischen zwei Objekten mit jeweils mehreren Geometrien als Abstand der zueinander nächstliegenden Geometrien definieren. Ebenso möglich ist aber auch die Angabe des Abstandes der beiden Schwerpunkte der Geometrien beider Objekte, oder ein gewichteter Mittelwert basierend auf den Eigenschaften der Objekte, wie beispielsweise der Flächeninhalt der Geometrien.

Die Aggregation hat als starke Beziehung zwischen Objekten Verknüpfungen zu zwei weiteren Anforderungen. Erstens zur Anforderung der funktionalen Abhängigkeit (Anforderung 12), da beispielsweise die Gesamtfläche des Ganzen direkt aus der arithmetischen Summe der Flächen der Teile folgt und damit keine Berechnung basierend auf der Geometrie des Ganzen erfolgen muss. Die zweite Anforderung ist die automatische Aktualisierung (Anforderung 25), da beim erstellen, aktualisieren oder beim löschen eines Teils automatisch das Ganze nachgeführt werden kann. Die beiden Anforderungen können insbesondere bei verschachtelten Aggregationen zur Sicherung der Datenintegrität beitragen.

Anforderung 7: Konzeptuelle Generalisierung

Unterstützung der konzeptuellen Generalisierung von Geometrien

Im objektorientierten räumlichen Datenmodell OMT-G kann ein einzelnes Objekt durch mehrere räumliche Primitive repräsentiert werden (Borges u. a., 2002) (siehe Abbildung 4.2). Das Objekt wird durch eine Oberklasse

ohne expliziten Raumbezug und mehrere Unterklassen abgebildet. Die Unterklassen erben die nicht-räumlichen Attribute der Oberklasse und ergänzen diese um den Raumbezug und optional um weitere nicht-räumliche Attribute. Im Beispiel in Abbildung 4.2 wird die Klasse Fluss durch eine Linie (Achse, Ränder), als Fläche und als Netzwerk (Segment) repräsentiert. Durch diesen Ansatz können räumliche Beziehungen für jede einzelne Repräsentationen explizit definiert werden.

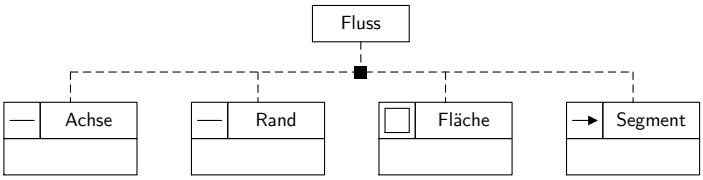


Abbildung 4.2: Klassendiagramm für die konzeptuelle Generalisierung in OMT-G nach Borges u. a. (2002)

Borges u. a. (2002) berücksichtigen neben der konzeptuellen Generalisierung basierend auf der Form auch die konzeptuelle Generalisierung basierend auf dem Maßstab. Als Beispiel wird die Repräsentation einer Stadt als Punkt sowie als Fläche angegeben. Diese Repräsentationen gelten dann jeweils nur für einen bestimmten Maßstabsbereich.

Diskussion Die konzeptuelle Generalisierung führt zu drei Anforderungen an die Modellierung und Umsetzung von Integritätsbedingungen. Erstens muss es möglich sein für jede Repräsentation eines Objekts eine eigene Menge an Bedingungen definieren zu können. Zweitens müssen die Bedingungen so gewählt werden, dass diese konsistent über alle Repräsentationen sind (siehe Anforderung 19). Beispielsweise muss eine Integritätsbedingung an die minimale Breite eines Flusses sowohl für die geometrische Repräsentation der Ränder als auch für die Fläche definiert werden. Drittens können durch Integritätsbedingungen teilweise die Beziehungen der geometrischen Repräsentationen untereinander zugesichert werden. Im Beispiel des Objekts Fluss lässt sich gegebenenfalls die Geometrie des Rands aus der Fläche ableiten und wiederum die Achse aus der Geometrie des Rands (siehe Anforderung 11).

Anforderung 8: Unscharfe Geometrie

Unterstützung des Konzeptes unscharfer Geometrien und der damit verbundenen Topologie

Friis-Christensen u. a. (2001) ordnen die Unschärfe als Bestandteil der Objekte ein. Die Unschärfe bezieht sich dabei sowohl auf geometrische als auch thematische Attribute. Die Attribute sollen jedoch mit einer quantifizierbaren Wahrscheinlichkeit exakten Koordinaten sowie thematischen Klassen zuordenbar sein.

In Bejaoui (2009) und Bejaoui u. a. (2010) werden Integritätsbedingungen für sogenannte unscharfe Flächen (engl. 'regions with broad boundaries') aufgestellt. Die Unschärfe entsteht dabei durch die Schwierigkeit ein Objekt von seinen Nachbarn abzugrenzen. Unschärfe kann prinzipiell für alle Geometrietypen definiert werden, also auch für Punkte und Linien. In Bejaoui (2009) und Bejaoui u. a. (2010) werden jedoch nur Integritätsbedingungen für unscharfe Flächen aufgestellt. Beispiele für unscharfe Flächen sind Waldbestände, Gewässer mit Überflutungsbereichen und Flächen mit Umweltverschmutzung. Das verwendete Konzept der unscharfen Flächen basiert auf der Egg-Yolk-Theorie nach Cohn und Gotts (1996). Darin wird allgemein eine unscharfe Fläche (engl. 'vague', auch 'non-crisp') durch mehrere konzentrische Teilflächen (diese sind exakt, engl. 'crisp') repräsentiert, die den Grad der Zugehörigkeit zum Objekt modellieren. Im einfachsten Fall von zwei Teilflächen wird die minimale Fläche als Eigelb (engl. 'egg yolk') und die maximale Fläche als Eiweiß (engl. 'egg white') definiert. Beide bilden zusammen das Ei (engl. 'egg'). Eigelb und Eiweiß können dabei als minimale und maximale geometrische Ausdehnung interpretiert werden. Ein Beispiel für die beiden Begriffe ist die normale Ausdehnung eines Gewässers und die Ausdehnung desselben Gewässers bei Überflutung. In Tabelle 4.4 entsprechen die gefüllten Quadrate den minimalen Flächen und die umgebenden Rechtecke den maximalen Flächen.

Bejaoui (2009) und Bejaoui u. a. (2010) erweitern die binären topologischen Relationen nach Egenhofer und Herring (1990) auf unscharfe Flächen. Von zwei Flächen A und B werden die vier topologischen Relationen ihrer minimalen und maximalen Ausdehnung berechnet. Die Matrix drückt die Zugehörigkeit der unscharfen Flächen A und B zu einem von acht Clustern aus, die den binären topologischen Relationen nach Egenhofer und Herring (1990) entsprechen ('disjoint', 'covers', ...). Der Grad der Zugehörigkeit wird durch eines der Adverbien 'schwach' (engl. 'weakly'), 'einigermaßen' (engl. 'fairly'), 'stark' (engl. 'strongly'), und 'vollständig' (engl. 'completely') ausgedrückt. Die Adverbien drücken aus wie viele der vier Elemente der Matrix einer definierten topologischen Relation entsprechen (siehe Tabelle 4.4). Sie sind dabei in ansteigender Reihenfolge von einem bis vier entsprechenden Elementen sortiert.

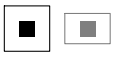
	Zugehörigkeit			
	Vollständig	Stark	Einigermaßen	Schwach
Beispiel				
Relation	$\begin{pmatrix} Disjoint & Disjoint \\ Disjoint & Disjoint \end{pmatrix}$	$\begin{pmatrix} Disjoint & Disjoint \\ Disjoint & Overlap \end{pmatrix}$	$\begin{pmatrix} Disjoint & Disjoint \\ Contains & Overlap \end{pmatrix}$	$\begin{pmatrix} Disjoint & Overlap \\ Contains & Overlap \end{pmatrix}$

Tabelle 4.4: Topologische Relation 'disjoint' zwischen zwei unscharfen Geometrien nach Bejaoui u. a. (2010)

Diskussion Unscharfe Geometrien stellen eine Erweiterung scharfer Geometrien dar. Integritätsbedingungen müssen für die Verwendung unscharfer Geometrien bezüglich des Geometriemodells, beispielsweise nach der Egg-Yolk Theorie, und bezüglich der verfügbaren Operatoren, beispielsweise der binären topologischen Relationen, erweitert werden. Zudem müssen die Wahrscheinlichkeiten der Objektzugehörigkeit entsprechend berücksichtigt werden. Prinzipiell sind die Integritätsbedingungen unscharfer und scharfer Geometrien jedoch vergleichbar.

4.2 Definition von Integritätsbedingungen

Werden Integritätsbedingungen aufgestellt, so werden dafür einige grundlegende Mechanismen benötigt. Diese definieren auf welche Daten eines Modells zugegriffen werden kann und welche Methoden auf diesen Daten ausgeführt werden können.

Die fünf im Folgenden vorgestellten Anforderungen der Definition an Integritätsbedingungen enthalten wichtige Werkzeuge für die Definition von Bedingungen. Diese können sich auf mehrere Attributwerte, Attribute, Objekte sowie auf mehrere Klassen beziehen (Anforderung 9). Die vielfältigen Möglichkeiten zur Definition topologischer Relationen erfordern eine flexible Definition der Topologie (Anforderung 10). Zudem kann durch Methoden für die Ableitung von Geometrien (Anforderung 11) nicht nur auf die Originalgeometrie eines Objektes zugegriffen werden, sondern auch auf abgeleitete Geometrien, wie das umschließende Rechteck. Die Beziehung zwischen den Attributen eines Objekts kann mit Hilfe der funktionalen Abhängigkeit (Anforderung 12) berücksichtigt werden. Schließlich ermöglichen generische Bedingungen (Anforderung 13) die Wiederverwendbarkeit von Integritätsbedingungen für mehrere Objektklassen sowie unterschiedliche Datenmodelle.

Anforderung 9: Anzahl der Attributwerte/Attribute/Objekte/Klassen

> Unterstützung von Bedingungen, die mehrere Attributwerte, Attribute, Objekte und Klassen referenzieren

Die Anzahl der in einer einzelnen Bedingung verwendeten Attribute, Objekte und (Objekt-) Klassen wird in vielen der referenzierten Arbeiten zur Klassifizierung von Integritätsbedingungen verwendet. Im Folgenden werden aus Gründen der Übersichtlichkeit nur einige ausgewählte Aspekte der Arbeiten vorgestellt. In der Diskussion der Anforderung sind jedoch alle Arbeiten berücksichtigt.

In Kang u. a. (2004) und Pinet u. a. (2005) kann ein einzelnes Objekt eine zusammengesetzte Geometrie aufweisen (siehe Anforderung 6). An die Anzahl der Geometrien eines Objekts können somit Bedingungen gestellt werden. In Pinet u. a. (2005) ist als Beispiel die Bedingung angegeben, dass die Anzahl der einzelnen Geometrien eines Gebäudeverbunds dem numerischen Attribut AnzahlGebaeude entsprechen muss.

Sowohl Louwsma (2004) als auch Duboisset u. a. (2005a,b) geben die Anzahl der Objekte einer Klasse als Dichte an. In Duboisset u. a. (2005a,b) wird die Anzahl der Objekte einer Klasse relativ zur Anzahl der Objekte einer anderen Klasse festgelegt. Ein Beispiel dafür ist die Bedingung 'die Anzahl der Städte muss kleiner oder gleich der Anzahl der Rathäuser sein'. In Louwsma (2004) wird eine Bedingung an die 'maximale Anzahl an Gebäuden in einem Wohngebiet' und damit an die Dichte von Objekten in Bezug auf die Geometrien anderer Objekte gestellt. Die Bezugsgeometrie kann nach Louwsma (2004) auch der Gesamtfläche der Daten entsprechen, wie aus der exemplarischen Bedingung 'es gibt nur einen Turm im Modell' ersichtlich ist.

Um in einer Integritätsbedingung die Anzahl der Objekte der Klasse A, die mit Objekten der Klasse B in Beziehung stehen, quantifizieren zu können, werden sogenannte Kardinalitäten oder Multiplizitäten verwendet. Die Kardinalität in Datenbanksystemen gibt nach Elmasri und Navathe (2011) für binäre Beziehungen die minimale und maximale Anzahl der Objekte an, die mit einem bestimmten Objekt in Beziehung stehen. Die

Multiplizität in Klassendiagrammen der Unified Modeling Language (Uml) wird nach Hitz u. a. (2005) zur Angabe der Anzahl der Werte die ein Attribut aufnehmen kann und zur Angabe der beteiligten Objekte einer Assoziation oder Aggregation verwendet (siehe Abschnitt 2.3).

Kardinalitäten (beziehungsweise Multiplizitäten) werden für die Definition von Integritätsbedingungen unter anderem von Fahrner u. a. (1997), Borges u. a. (2002), Friis-Christensen u. a. (2001), Kang u. a. (2004), Mäs u. a. (2005) und van Bennekom-Minnema (2008) verwendet. Sowohl in Servigne u. a. (2000) als auch in Christensen (2007) wird die Angabe der Kardinalität dadurch vereinfacht, dass nur ein einzelner Schwellwert festgelegt wird. So kann entweder nur die minimale, maximale oder exakte Anzahl an Objekten der Klasse B angegeben werden. In Mäs (2010) werden spezielle Kardinalitäten, die sogenannten mathematischen Relationen, zum logischen Schließen zwischen Beziehungen verwendet. Diese sind in Tabelle 4.5 zusammengefasst. Bis auf die eineindeutige Relation schließt jeweils eine der Relationen die andere nicht notwendigerweiße aus, d.h. die Relationen können kombiniert werden. Durch die Kombination kann für die Beziehung zwischen zwei Klassen A und B sowohl die Kardinalität der Beziehung $A \rightarrow B$ als auch die Kardinalität der Beziehung $B \rightarrow A$ quantifiziert werden. Für die Beziehung zwischen Objektklassen werden in Mäs (2010) insgesamt 17 mögliche Kombinationen vorgestellt. Davon stellen sieben Kombinationen zusätzliche Anforderungen an die Anzahl der Elemente zweier Klassen A und B, beispielsweise $|A| < |B|$ oder $|A| > |B| + 1$.

Name	Aussagenlogik	Kardinalität	Bedeutung
linkstotal	$\forall a \in A \; \exists \; b \in B : (a, b) \in R$	$1 \longrightarrow 1..*$	Jedes Element aus A hat mindestens einen Partner in B
rechtstotal / surjektiv	$\forall b \in B \; \exists \; a \in A : (a, b) \in R$	$1..* \longleftarrow 1$	Jedes Element aus B hat mindestens einen Partner in A
linkseindeutig / injektiv	$\forall a, c \in A \; \forall \; b \in B : (a, b) \in R \wedge (c, b) \in R \Rightarrow a = c$	$0..1 \longleftarrow 1$	Kein Element aus B hat mehr als einen Partner in A
rechtseindeutig / funktional	$\forall a \in A \; \exists \; b, c \in B : (a, b) \in R \wedge (c, b) \in R \Rightarrow b = c$	$1 \longrightarrow 0..1$	Kein Element aus A hat mehr als einen Partner in B
eineindeutig / bijektiv	$\forall b \in B \; \exists! \; a \in A : (a, b) \in R$	$1 \longleftarrow 1$	Jedes Element aus B hat genau einen Partner in A

Tabelle 4.5: Mathematische Relationen

Diskussion Die Anforderung nach der Unterstützung der Anzahl wird in den referenzierten Arbeiten im Vergleich zu den anderen identifizierten Anforderungen am häufigsten genannt.

Zusammenfassend lässt sich die Anzahl der Attributwerte/Attribute/Objekte/Klassen wie folgt unterscheiden:

- Anzahl Attributwerte: Umfasst ein Attribut mehr als einen Wert, dann können Bedingungen an alle oder einzelne Werte gestellt werden. Zudem lässt sich die Anzahl der anzunehmenden Werte relativ oder absolut einschränken. Dabei kann das Attribut auch aus einer zusammengesetzten Geometrie bestehen (siehe Anforderung 6).
- Anzahl Attribute: Bedingungen können an einzelne oder mehrere Attribute gestellt werden. Ein Beispiel für Bedingungen zwischen mehreren Attributen bildet die funktionale Abhängigkeit von Attributen (siehe Anforderung 12).
- Anzahl Objekte: Bedingungen können an ein Objekt oder mehrere Objekte gestellt werden. Die Anzahl der Objekte kann dabei exakt durch die Angabe der Kardinalität oder Multiplizität festgelegt werden und reicht von keinem Objekt bis zu allen Objekten einer Menge oder Klasse. Die Anzahl der Objekte eines Datensatzes lässt sich sowohl relativ zu einer anderen Anzahl (Duboisset u. a., 2005a; Mäs, 2010) als auch absolut (Louwsma, 2004) einschränken. Die absolute Einschränkung entspricht dabei der Angabe einer Dichte von Objekten, bezogen entweder auf das gesamte Modell oder auf eine Referenzfläche (Louwsma, 2004). Zudem kann auf Aggregationen von Objekten Bezug genommen werden, beispielsweise auf die Vereinigung der Geometrien aller Objekte einer Menge (Belussi u. a., 2006).
- Anzahl Klassen: Bedingungen können an ein oder mehrere Objekte einer oder mehrerer Klassen gestellt werden.

In einer einzelnen Integritätsbedingung kann eine beliebige Kombination der Anzahlen an Attributwerten, Attribute, Objekte und Klassen enthalten sein.

Durch die Verwendung von nicht-räumlichen und räumlichen Filtern beziehungsweise durch Selektionen können spezifische Untermengen von Attributen, Objekten und Klassen in einer Bedingung verwendet werden. So kann auch auf ein spezifisches Objekt anstatt auf alle Objekte einer Klasse Bezug genommen werden (Belussi u. a., 2006; Wang und Reinhardt, 2007).

Anforderung 10: Flexible Topologie

> Unterstützung von Schnittmatrizen und benutzerdefinierten Namen für (Kombinationen von) binären topologischen Relationen

Hadzilacos und Tryfona (1992) erweitern die binären topologischen Relationen von Egenhofer und Herring (1990) auf Relationen zwischen allen geometrischen Basistypen. Sie identifizieren insgesamt 16 mögliche binäre topologische Relationen und ihre entsprechenden Schnittmatrizen. Diese sind einfach von r_1 (Schnittmatrix 0000) bis r_{16} (Schnittmatrix 1111) durchnummeriert. Den Relationen wird aus zwei Gründen keine Bedeutung zugewiesen. Erstens können diese mit bereits vorhandenen Namen im Widerspruch stehen. Wie in Unterabschnitt 2.1.3 gezeigt, existieren mehrere alternative Benennungen für topologische Relationen. Zweitens wird darauf hingewiesen, dass die Bedeutung einer Relation beziehungsweise Schnittmatrix zwischen den verschiedenen geometrischen Basistypen unterschiedlich interpretiert werden kann. Beispielsweise bedeutet die Relation r_3 (1100) 'equal' für die Kombination aus Fläche – Fläche, für die Kombination aus Linie – Fläche hingegen wäre eine entsprechende Bezeichnung etwa intern verbunden (engl. 'internally connects') (siehe Tabelle 4.6). Um schließlich eine benannte Relation für die Verwendung in einer Integritätsbedingung zu erstellen, kann eine Kombination aus den 16 Relationen ausgewählt und mit einem Namen versehen werden.

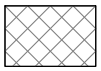
Relation	Fläche – Fläche	Fläche – Linie
r_3 (1100)		

Tabelle 4.6: Topologische Relation r_3 nach Hadzilacos und Tryfona (1992)

Den Benutzern vertraute kulturelle oder semantische Konzepte machen nach Borges u. a. (2002) eine Erweiterung der Relationen nach Clementini u. a. (1993) notwendig. Neben den bereits in anderen topologischen Modellen abgebildeten Relationen 'coincide' (entspricht 'equals') und 'contain' werden 'adjacent to' und 'near'[3] definiert (siehe Tabelle 4.7). So definiert 'adjacent to' beispielsweise dass zwei Flächen oder eine Fläche und eine Linie aneinander angrenzen, sich also nur in einer Linie schneiden.

Name	Bedeutung
'Adjacent to'	Fläche – Fläche oder Fläche – Linie 'touch' und dim(Schnitt)=1 (Linie)
'Near'	Zu vergleichende Geometrie nicht disjoint zu Puffer mit Radius R

Tabelle 4.7: Auswahl an topologischen Relationen nach Borges u. a. (2002)

Im gleichen Kontext verwenden Duboisset u. a. (2005a,b) zwei verschiedene Ansätze um topologische Integritätsbedingungen auszudrücken. Duboisset u. a. (2005a) nehmen Bezug auf die 9IM und Duboisset u. a. (2005b) tauschen dieses durch den Region Connected Calculus (RCC) aus.

Diskussion Es gibt mehrere etablierte Varianten um binäre topologische Relationen anzugeben (siehe Unterabschnitt 2.1.3). Am mächtigsten sind Schnittmatrizen mit Angabe der Dimensionen der Schnittgeometrien, da sich daraus erschöpfend die möglichen Relationen angeben lassen. Wichtig ist vor allem, dass die Definition der binären topologischen Relationen exakt der Definition dieser Relationen durch den Benutzer entspricht. Gegegebenenfalls müssen dazu Namen der Relationen geändert oder neue benannte Relationen eingeführt werden.

Anforderung 11: Methoden für die Ableitung von Geometrien

> Bereitstellung von Methoden für die Ableitung, Erzeugung und Zerlegung von Objektgeometrien

[3] Die Definition der Relation 'near' in Borges u. a. (2002) ist streng genommen keine topologische Relation, sondern eine metrische Relation.

In ihrem Datenmodel definieren Hadzilacos und Tryfona (1992) sogenannte elementare geometrische Methoden, welche lediglich die Geometrie als Parameter verwenden. Diese können den Konzepten der Mengenlehre, Geometrieerzeugung und Geometriezerlegung zugeordnet werden. Die für die Anforderung relevanten Methoden sind in Tabelle 4.8 aufgelistet[4]. In Hadzilacos und Tryfona (1992) wird jedoch kein Beispiel für die Verwendung der Methoden vorgestellt.

Typ	Eingabetyp Geometrie	Methode	Ausgabe
Mengenlehre	Zwei Mengen des gleichen Basistyps	Union	Vereinigung der zwei Mengen
		SetDifference	Differenz der zwei Mengen
Geometrieerzeugung	Menge von Stützpunkten	ConstructLine	Linie
		ConstructArea	Fläche
Geometriezerlegung	Linie	EndPoints	Menge mit zwei Endpunkten
	Fläche	Nodes	Menge von Stützpunkten

Tabelle 4.8: Auswahl von elementaren geometrischen Methoden nach Hadzilacos und Tryfona (1992)

Der OGC Simple Feature (2011) Standard sieht mehrere Methoden beziehungsweise Operationen für Geometrien vor (siehe Tabelle 2.5).

Diskussion Integritätsbedingungen können sich auf Elemente beziehungsweise Ableitungen einer Geometrie beziehen. Dies führt zu der Forderung dass eine Geometrie in ihre Bestandteile zerlegbar sein muss. Eine Fläche besteht beispielsweise aus einem Umring der durch die Verbindung mehrerer Stützpunkte zu einer Linie repräsentiert wird. Die Integritätsbedingung nach der Geschlossenheit des Umrings benötigt Zugriff auf den Start- und Endpunkt dieser Linie, erfordert also eine Geometriezerlegung. Aus den Bestandteilen einer Geometrie lassen sich aber auch neue Geometrien ableiten, beispielsweise durch die Generalisierung des Umrings einer Fläche mit dem Douglas-Peucker-Algorithmus.

Anforderung 12: Funktionale Abhängigkeit

> Unterstützung der funktionalen Abhängigkeit von Attributen eines Objektes und Attributen von Objekten in einer Aggregation

In Datenbanksystemen kann nach Elmasri und Navathe (2011) die Beziehung zwischen zwei Mengen von Attributen X und Y durch eine Integritätsbedingung an deren funktionale Abhängigkeit festgelegt werden. Die Werte von X bestimmen dann eindeutige Werte für Y in allen Zuständen der Relation. Die funktionale Abhängigkeit wird auch in der Form $X \rightarrow Y$ angegeben. Die funktionale Abhängigkeit basiert auf der Semantik der Attribute und muss explizit definiert werden.

Das objektorientierte räumliche Datenbankmodell von Ditt u. a. (1997) unterstützt die automatische Aktualisierung von Attributwerten beim erstellen, aktualisieren oder löschen von Objekten. So wird zum Beispiel beim einfügen einer Stadt mit einer Einwohnerzahl größer einer Million automatisch das Attribut Metropole auf wahr gesetzt. Aber auch die Abhängigkeit von Attributen innerhalb einer Aggregation kann berücksichtigt werden. Beispielsweise wird bei der Änderung eines Staates automatisch überprüft ob dessen Einwohnerzahl gleich der Summe der Einwohnerzahlen der zugeordneten Städte ist und gegebenenfalls entsprechend korrigiert. Ähnliche Funktionalitäten finden sich auch in van Bennekom-Minnema (2008) und Bravo und Rodríguez (2012) wieder. So wird beispielsweise in van Bennekom-Minnema (2008) die automatische Berechnung des Attributs Fläche aus der Geometrie des Objektes gefordert.

Funktionale Abhängigkeit wird in Christensen (2007) und Mäs und Reinhardt (2009) genutzt, um ausgehend vom Wert eines Attributs den möglichen Wertebereich anderer Attribute einzuschränken. Als Beispiel wird in Christensen (2007) der mögliche Wertebereich der Postleitzahlen eines Gebäudes durch dessen Zugehörigkeit zu einer definierten Gemeinde eingeschränkt. In Mäs und Reinhardt (2009) folgt beispielsweise aus dem Straßentyp 'Autobahn' dass die Anzahl der Fahrspuren größer gleich zwei sein muss.

Reeves u. a. (2006) definieren sogenannte parametrische geometrische Integritätsbedingungen in Anlehnung an parametrische Modelle des Computer Aided Design (CAD). Eine Instanz des parametrischen Modells, eine

[4] Die Methoden wurden im Vergleich zu Hadzilacos und Tryfona (1992) umbenannt um die konsistente Benennung von geometrischen Primitiven zu gewährleisten und um selbsterklärend zu sein.

sogenannte parametrische Instanz, besteht aus einer Menge an Parametern, einer Menge an geometrischen Primitiven und einer parametrischen Funktion pro Primitiv. Letztere kann aus den gegebenen Parametern und den anderen geometrischen Elementen die Geometrie des Elements ableiten. Beispielsweise kann ein Gebäude aus einem Quader mit den Parametern Länge, Breite und Höhe sowie einem Spitzdach mit den Parametern Länge, Breite und Firsthöhe zusammengesetzt werden. Dabei wird zusätzlich durch Funktionen sichergestellt, dass die Elemente überlappungsfrei und korrekt orientiert angeordnet sind. Parametrische Methoden können vier Klassen von Definitionen enthalten. Auf Integritätsbedingungen basierende Definitionen enthalten räumliche Beziehungen zwischen Elementen, wie beispielsweise die Orthogonalität von Ebenen. Numerische und boolesche Ausdrücke beschreiben Zusammenhänge zwischen Parametern und geometrischen Elementen im Modell. Grafisch-numerische Ausdrücke ermöglichen die Verwendung geometrischer Elemente als Argumente parametrischer Methoden, beispielsweise um die Distanz zwischen zwei Elementen berechnen zu können. Virtuelle auf Integritätsbedingungen basierende Definitionen enthalten beispielsweise die Projektion eines Punktes auf eine Linie.

Diskussion Prinzipiell lässt sich unterscheiden ob das referenzierte Attribut in einer funktionalen Abhängigkeit räumlich oder nicht-räumlich ist. Ist es räumlich, so muss das referenzierende Attribut eine Eigenschaft der Geometrie beschreiben, beispielsweise die Länge oder Krümmung einer Linie. Zur effizienteren Bearbeitung werden solche Attribute in der Regel so selten wie möglich berechnet, da deren Ableitung aus der Geometrie rechnerisch teuer ist. Effizient ist die Ableitung der von der Geometrie abhängigen Attribute eines Objektes beim Erstellen des Objektes und bei nachfolgenden Aktualisierungen der Objektgeometrie.

Die funktionale Abhängigkeit lässt sich auf Objekte in einer Aggregation (siehe Anforderung 6) erweitern. Beispielsweise besteht ein Bundesland aus mehreren Gemeinden und der Wert des Attributs Fläche des Bundeslandes entspricht der Summe der Flächen aller Gemeinden, ist also funktional abhängig. Invertiert man die Bedingung, dann muss die Fläche jeder Gemeinde kleiner als die Gesamtfläche des Bundeslandes sein.

Parametrische geometrische Bedingungen nach Reeves u. a. (2006) haben ihre Wurzeln in der Konstruktion von Geometrien und eröffnen dabei eine neue Sichtweise auf Integritätsbedingungen. Einerseits beschreiben sie wiederkehrende Ansätze zur Konstruktion, wie beispielsweise Rechtwinkelbedingungen. Zweitens beschreiben sie die Abhängigkeit von Parametern in Integritätsbedingungen von räumlichen und nicht-räumlichen Attributen des gleichen Objektes beziehungsweise von Objektteilen (Aggregation siehe Anforderung 6) sowie von deren Nachbarn. Die funktionale Abhängigkeit von Attributwerten reicht also über das Objekt selbst hinaus.

Anforderung 13: Generische Bedingungen

> Unterstützung von generischen Bedingungen zur Steigerung der Flexibilität und Wiederverwendbarkeit

Sogenannte parametrische Bedingungen erlauben in Casanova u. a. (2000) die Wiederverwendung einer Integritätsbedingung für mehrere Klassen, ähnlich den Templates in der Programmiersprache C++ oder Generics in Java (siehe Unterabschnitt 3.3.2).

Generische Bedingungen beziehen sich in Xu (2011) auf geometrische Basistypen in 3D, die im verwendeten Modell Punkte, Flächen und Körper (Volumina) umfassen. Die auch als abstrakt bezeichneten Regeln können dann für alle Klassen mit dem definiertem geometrischen Basistyp verwendet werden. So kann die Bestimmung der topologischen Relation zwischen zwei Körpern beispielsweise für die Klassen Gebäude – Gebäude, Gebäude – Baum sowie Baum – Baum analog erfolgen.

Diskussion Generische Bedingungen erhöhen die Wiederverwendbarkeit. So kann eine Bedingung für mehrere Klassen verwendet werden, was die Duplikation verringert und damit die Gefahr der damit verbundenen Fehler reduziert.

Dieser Ansatz lässt sich hinsichtlich zweier Aspekte erweitern. Erstens kann nicht nur eine sondern eine Menge von Integritätsbedingungen generisch definiert und verwendet werden. Zweitens können generische Bedingungen nicht nur innerhalb eines Datenmodells wiederverwendet werden, sondern durch ihre Flexibilität auch in anderen ähnlichen Datenmodellen eingesetzt werden. So können beispielsweise typische Integritätsbedingungen für Straßen(-netzwerke) aufgestellt werden, die dann in allen kompatiblen Datenmodellen einfach durch Angabe der entsprechenden Klassen- und Attributnamen verwendet werden können.

4.3 Aufstellung einzelner Integritätsbedingungen

Bei der Aufstellung einzelner konkreter Integritätsbedingungen sind einige Aspekte zu berücksichtigen. Diese betreffen wie diese formuliert und interpretiert werden.

Die fünf im Folgenden vorgestellten Anforderungen der Aufstellung einzelner Integritätsbedingungen enthalten wichtige Aspekte, die bei der Aufstellung jeder einzelnen Bedingung berücksichtigt werden sollten. Bedingungen können so formuliert werden, dass sie fordern oder dass sie einschränken (Anforderung 14). Der Wertebereich eines Attributs kann durch verschiedene Ansätze eingeschränkt werden (Anforderung 15). Bei der Aufstellung von Integritätsbedingungen sollte sinnvollerweise auch die Genauigkeit von Attributwerten (Anforderung 16) berücksichtigt werden. Ebenso ist der richtige Umgang mit fehlenden Werten (Anforderung 17) wichtig, da Datensätze immer unvollständig sein können. Durch die Angabe eines Schweregrads (Anforderung 18) können einzelnen Bedingungen unterschiedliche Gewichte zugewiesen werden. Schließlich ist die gegenseitige Konsistenz der Integritätsbedingungen (Anforderung 19) ein wichtiges Kriterium für deren Anwendung.

Anforderung 14: Fordern vs. Einschränken

> Jede Regel kann so formuliert werden, dass entweder die Menge der gültigen Instanzen (fordern) oder die Menge der ungültigen Instanzen (einschränken) definiert wird. Von den beiden Formulierungen ist die mit weniger Vergleichswerten zu wählen.

Nach Louwsma (2004) können Bedingungen entweder fordern (engl. 'force') oder einschränken (engl. 'restrict'). In natürlicher Sprache entspricht dies 'muss immer' und 'darf nie'. Als Beispiel wird die Bedingung 'Gras muss immer grün sein' angegeben, die in einschränkender Formulierung alle anderen Farben ausschließen müsste. Die Anzahl der Vergleichswerte kann dabei in die drei Klassen 'ein Wert', 'viele (aber abzählbare) Werte' und 'unendlich viele Werte' eingeteilt werden. Ist der Vergleichswert ein einziger Schwellwert, so macht es keinen Unterschied, in welcher Variante die Regel formuliert wird. Soll die Formulierung einer Bedingung mit (unendlich) vielen Werten umgekehrt werden, so kann dies durch eine Negation erreicht werden. Beispiele für Formulierungen von Regeln in beiden Varianten sind in Tabelle 4.9 angegeben.

Fordern (Anzahl Vergleichswerte)	Einschränken (Anzahl Vergleichswerte)
Alleinstehende Gebäude müssen mindestens zehn Meter von anderen Gebäuden entfernt sein (1)	Alleinstehende Gebäude dürfen nie weniger als zehn Meter von anderen Gebäuden entfernt sein (1)
Wohngebäude müssen immer innerhalb von Flurstücken mit der Nutzung 'Wohnen' sein (1)	Wohngebäude dürfen nie innerhalb von Flurstücken mit der Nutzung 'Straße', 'Wald', 'Industrie', ... sein (n)
Der Ort Greenwich muss die Länge 0 aufweisen (1)	Der Ort Greenwich darf nicht die Länge 1, 2, ... aufweisen (∞)

Tabelle 4.9: Beispiele für die Formulierung, angepasst nach Louwsma (2004)

Christensen (2007) verwenden zur Formulierung von Integritätsbedingungen die zwei Basiskonstrukte 'alle müssen' (engl. 'all-must') und 'keine dürfen' (engl. 'no-may').

Diskussion Alternative Benennungen für Fordern vs. Einschränken sind Erlauben vs. Verbieten und Regel vs. Ausnahmen.

Die Forderung nach der Einfachheit beziehungsweise geringeren Anzahl an Vergleichswerten einer Regel führt sowohl zu einer besseren Verständlichkeit durch den Benutzer als auch zu einer beschleunigten Prüfung durch einen Rechner. Hat ein Attribut n verschiedene Ausprägungen und ist davon nur ein Wert der Vergleichswert, dann stehen $(n-1)$ Prüfungen bei der einschränkenden Formulierung genau einer Prüfung bei der fordernden Formulierung entgegen.

Anforderung 15: Wertebereich

> Unterstützung der Einschränkung des möglichen Wertebereiches für Attribute

Der Wertebereich kann in Christensen (2007) durch kodierte Werte (Auflistung), Intervalle oder funktionale Abhängigkeit eingeschränkt werden (siehe Anforderung 12). Integritätsbedingungen bezüglich des Formats schränken die möglichen Werte ebenso ein.

In van Bennekom-Minnema (2008) werden mehrere Bedingungen an den Wertebereich vorgestellt. Diese umfassen alpha-numerische Bereiche, Auflistungen, und Formate wie beispielsweise die Großschreibung von Texten.

Diskussion Die Einschränkung des Wertebereiches ist eine einfache und doch mächtige Integritätsbedingung. Die Einschränkung kann über die Angabe eines Schwellwerts oder eines Intervalls für quantitative Attribute, Auflistungen für qualitative Attribute, oder auch komplexere Einschränkungen, wie reguläre Ausdrücke, Formate, Grammatiken oder Formeln, erfolgen.

Anforderung 16: Genauigkeit

Berücksichtigung der Genauigkeit von Attributwerten

Diskussion Die Genauigkeit von Attributwerten bezieht sich hier auf die Anzahl der signifikanten Nachkommastellen der Werte, beispielsweise der Nachkommastellen der Koordinaten der (Stütz-) Punkte von Geometrien. Diese Genauigkeit wird sowohl bei räumlichen als auch nicht-räumlichen Attributen durch drei Faktoren begrenzt. Erstens durch die Genauigkeit der Erfassung, so kann beispielsweise mit einem Maßband maximal auf Millimeter genau gemessen werden. Zweitens durch die geforderte Genauigkeit des Datenmodells beziehungsweise der entsprechenden Regelwerke zur Erfassung. So werden beispielsweise die Seiten von Gebäudegrundrissen in der Regel zentimetergenau erfasst. Drittens wird die Genauigkeit durch die Speicherung der Attributwerte auf einem Rechner und damit durch die interne Repräsentation der Werte bestimmt, beispielsweise durch die Speicherung eines Gleitkommawertes im Format Double.

Die Genauigkeit der Speicherung ist sinnvollerweise höher als die Genauigkeit der Erfassung. Im Beispiel in Abbildung 4.3 sind die Koordinaten der Schrägstrecken ohne Nachkommastellen angegeben, der Schnittpunkt der beiden Strecken muss jedoch sinnvollerweise mit einer zu definierenden Anzahl von Nachkommastellen angegeben werden.

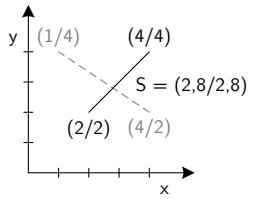


Abbildung 4.3: Beispiel zur Anforderung Genauigkeit

Die Genauigkeit von abgeleiteten Werten spielt auch eine Rolle. Ist beispielsweise der Grundriss eines Gebäudes zentimetergenau erfasst worden, so kann dessen Fläche auf Quadratzentimeter genau angegeben werden. Für eine bestimmte Anwendung kann es dagegen sinnvoll sein, die Fläche lediglich in Quadratmetern anzugeben.

Anforderung 17: Fehlende Werte und NULL

Berücksichtigung fehlender und nicht definierter (Null) Attributwerte

In Fahrner u. a. (1997), Cockcroft (1997) und van Bennekom-Minnema (2008) wird die nicht Null Bedingung angeführt.

Diskussion Fehlende und nicht definierte Attributwerte sind ein Spezialfall der Anforderung an den Wertebereich (siehe Anforderung 15). Integritätsbedingungen werden in den referenzierten Arbeiten so aufgestellt, dass fehlende und nicht definierte Attribute nicht erlaubt sind. Wie solche Daten dennoch interpretiert werden können ist in Abschnitt 2.6 (Data Mining) aufgeführt.

Anforderung 18: Schweregrad

Unterstützung von unterschiedlichen Schweregraden

Cockcroft (1997) gibt an, dass der Schweregrad (engl. 'severity') der Verletzung einer Integritätsbedingung abhängig von deren Typ ist. Vom Schweregrad lassen sich die Art der Warnung an den Benutzer sowie mögliche Aktionen zur Behandlung der Bedingung ableiten. Bedingungen mit dem Schweregrad 'fundamental' sollen den Benutzer dazu zwingen die entsprechenden Objekte zu bereinigen, bevor weitere Eingaben akzeptiert werden. Andere weniger wichtige Bedingungen können dagegen einfach durch die Bestätigung einer Warnung außer Kraft

gesetzt werden. Nach Cockcroft (1997) sind die Anforderungen Schweregrad und Sicherheit (Anforderung 27) miteinander verwandt.

In Mäs u. a. (2005) ist die Festlegung des Schweregrads einer Integritätsbedingung ein Teil deren Definition. Der Schweregrad wird in drei Werte unterteilt. Wird eine Bedingung als 'strikt' (engl. 'strict') definiert, so müssen die entsprechenden Objekte vom Benutzer geändert werden. Die beiden anderen Werte 'vermeide Verletzung' (engl. 'avoid violation') und 'verletze Bedingung nicht ohne guten Grund, Reaktion des Benutzers erforderlich' (engl. 'apply with caution, users reaction necessary') überlassen es dem Nutzer, welche Konsequenzen aus der Verletzung der Integritätsbedingung gezogen werden sollen. Die Reaktion des Benutzers kann dabei die Eintragung eines Kommentars sein, der die Umstände der Verletzung der Bedingung beschreibt, oder beispielsweise die Signalisierung einer gefährlichen Situation durch Warnhinweise vor Ort.

Salehi u. a. (2007) unterscheiden zwischen harten (engl. 'hard') und weichen (engl. 'soft') Integritätsbedingungen. Harte Bedingungen werden strikt interpretiert. Weiche Bedingungen verwenden einen Schwellwert um Abweichungen zu tolerieren die auf Entscheidungsprozesse keinen Einfluss haben. Eine weiche Bedingung ist beispielsweise 'Straßen dürfen Gebäude überschneiden, jedoch soll die Schnittfläche weniger als zehn Prozent der Fläche des Gebäudes betragen'. Weiche Bedingungen können zudem die Generalisierung von Geometrien bei der Verwendung von Daten unterschiedlichen Maßstabs berücksichtigen. Als Beispiel führen Salehi u. a. (2007) die Bedingung an, dass 'die Vereinigung aller Bundesstaaten mindestens 95 Prozent der Fläche des entsprechenden Staates überdecken muss'.

Diskussion Die Unterscheidung von Typen von Integritätsbedingungen mit unterschiedlichen Schweregraden ist sinnvoll. Beispielsweise sind topologische Bedingungen meist hart, wohingegen metrische Bedingungen wie die Distanz zwischen zwei Objekten auch weich sein können. Prinzipiell wäre es möglich jeder Bedingung ein eigenes Gewicht entsprechend ihrer Bedeutung zuzuweisen.

Die beiden Anforderungen nach Schweregrad und Ausmaß der Verletzung einer Integritätsbedingung (Anforderung 21) sind miteinander verbunden, jedoch weisen sie eine fest definierte Reihenfolge auf. Ist der Schweregrad so definiert dass Abweichungen nicht toleriert werden, dann ist das Ausmaß nicht von Relevanz, da selbst kleinste Abweichungen nicht erlaubt sind. Für alle anderen Fälle des Schweregrads muss das Ausmaß dagegen bestimmt werden.

In den Arbeiten von Cockcroft (1997) und Mäs u. a. (2005) bestimmt der Schweregrad das weitere Vorgehen, d.h. die weiteren Schritte sind unabhängig vom Ausmaß. Entweder muss der Fehler korrigiert werden oder der Fehler wird als Ausnahme deklariert (siehe Anforderung 26).

In der Arbeit von Salehi u. a. (2007) werden dagegen Schweregrad und Ausmaß kombiniert. Das maximal tolerierte Ausmaß wird jedoch beschränkt, was einer relaxierteren Definition der Integritätsbedingung entspricht.

Anforderung 19: Konsistenz

Unterstützung von automatischen Konsistenzprüfungen aller definierten Integritätsbedingungen

Um Konflikte zwischen Integritätsbedingungen zu vermeiden sieht Louwsma (2004) die Notwendigkeit für eine Überprüfung der Konsistenz sobald mehr als zwei Integritätsbedingungen formuliert werden. Die Überprüfung wird in der konzeptionellen Phase manuell durchgeführt. Eine automatische Prüfung wird jedoch empfohlen, die zusätzlich auch bei der Änderung von Bedingungen erfolgen sollte.

Die Konsistenz binärer topologischer Relationen zwischen den Objekten dreier Klassen A, B und C wird in Mäs (2010) untersucht. Die Relationen werden dabei nicht auf Objektebene sondern auf Klassenebene analysiert. Ausgehend von zwei bekannten topologischen Relationen r(A, B) und r(B, C) kann die dritte Relation r(A, C) abgeleitet werden, oder es können zumindest einige Relationen ausgeschlossen werden. Zu einem gewissen Grad können auch Redundanzen in den Relationen aufgedeckt werden. Die Konsistenzprüfung ist in einem Prototyp implementiert, der auf dem wissensbasierten System Protégé aufbaut.

Bravo und Rodríguez (2012) stellen zwei Typen von Problemen beim logischen Schließen über topologische Relationen vor. Das Problem der topologischen Konsistenz (engl. 'topological consistency problem') beschreibt den Fall, dass eine Menge von topologischen Relationen, die auf einer Menge von Objekten definiert sind, Widersprüche aufweisen, d.h. es können nicht alle topologischen Relationen gleichzeitig erfüllt werden. Das Problem der konjunktiven Realisierbarkeit (engl. 'conjunctive realizability problem') beschreibt den Fall, dass zwar alle topologischen Relationen für eine Menge von Objekten konsistent sein können, die Objekte jedoch nicht in einer Ebene repräsentierbar sind. Beispiele für dieses Problem bilden unter anderem planare Netzwerke. Die Konsistenzprüfung zwischen einer beliebigen Anzahl an Klassen mit beliebigen topologischen Relationen

zwischen den Klassen ist nach Bravo und Rodríguez (2012) prinzipiell NP-schwer, d.h. das Problem lässt sich vermutlich nicht effizient lösen. Bravo und Rodríguez (2012) identifizieren jedoch einige Konstellationen in denen das Problem in polynomischer Laufzeit lösbar ist.

Widersprüchliche Integritätsbedingungen werden in Xu (2011) diskutiert. Eine Menge an Bedingungen wird als inkonsistent bezeichnet, falls keine Objekte in der Realität existieren, die alle Bedingungen gleichzeitig erfüllen. Zusätzliche Regeln (engl. 'extra-check rule') sollen sicherstellen, dass es von jeder Klasse auch Objekte geben kann, die alle Bedingungen erfüllen. Die Lösung der Erfüllbarkeit darf also nicht sein, dass die Menge der Objekte einer Klasse leer sein muss. Als möglichen Ansatz zur Überprüfung der Konsistenz von Regeln wird deren Formulierung in einer Prädikatenlogik erster Stufe (engl. 'first order logic') vorgeschlagen. Die Integritätsbedingungen müssen dazu in Prädikate umgeschrieben werden, die dann von einem automatischen System, einem sogenannten Theorembeweiser, überprüft werden können.

Diskussion Die automatische Konsistenzprüfung von Integritätsbedingungen ist ein komplexes Problem, das nur bedingt effizient lösbar ist. Dagegen ist die manuelle Konsistenzprüfung zwar jederzeit möglich, aber auch nur für eine überschaubare Anzahl an Integritätsbedingungen sinnvoll. Ein komplementärer Ansatz zur automatischen und manuellen Überprüfung der Konsistenz ist die Anwendung der Bedingungen auf einen umfangreichen Datensatz und die anschließende systematische Auswertung der als fehlerhaft identifizierten Objekte.

Die Konsistenzprüfung lässt sich in die drei Bereiche Redundanzen, Widersprüche und Abhängigkeiten unterteilen (siehe Tabelle 4.10). Redundanzen und Widersprüche lassen sich am einfachsten aufdecken. Dagegen können Abhängigkeiten beliebig komplex werden und sich über mehrere Attribute, Klassen und Integritätsbedingungen erstrecken.

Konsistenzprüfung	Beispiel für Geometrie	Beispiel für Topologie
Redundanzen	Länge $> 10\,\text{m}$, Länge $> 10\,\text{m}$	disjoint(A, B), disjoint(A, B)
Widersprüche	Länge $> 10\,\text{m}$, Länge $< 8\,\text{m}$	disjoint(A, B), intersects(A, B)
Abhängigkeiten	Länge $< 2\,\text{m}$, Breite $< 3\,\text{m}$, Fläche $> 20\,\text{m}^2$	covers(A, B), covers(B, C), disjoint(A, C)

Tabelle 4.10: Beispiele für die Konsistenzprüfung

4.4 Prüfung von Daten mit Integritätsbedingungen

Wenn alle Integritätsbedingungen für ein Datenmodell aufgestellt sind, dann können Datensätze auf ihre Übereinstimmung mit den Bedingungen geprüft werden. Dabei können Anforderungen an die Methodik der dazu verwendeten Algorithmen und Anforderungen an das Ergebnis der Überprüfung gestellt werden.

Die vier im Folgenden vorgestellten Anforderungen der Prüfung von Integritätsbedingungen sollten für jeden Datensatz individuell berücksichtigt werden. Die Toleranz der für die Prüfung der Daten eingesetzten Algorithmen (Anforderung 20) sollte sich an der Genauigkeit der Daten und ihrer internen Verarbeitung in den Algorithmen ausrichten. Für die Quantifizierung der Abweichung der Daten von den vorgegebenen Sollwerten kann das Ausmaß (Anforderung 21) der Verletzung einer Integritätsbedingung bestimmt werden. Ebenso kann das globale Ausmaß (Anforderung 22) aller Verletzungen der Bedingungen ermittelt werden. Für die Prüfung umfangreicher Datensätze müssen sowohl beim Aufstellen von Integritätsbedingungen als auch bei deren Implementierung einige Aspekte berücksichtigt werden (Anforderung 23).

Anforderung 20: Toleranz

Berücksichtigung der Toleranz von Algorithmen

Nach Servigne u. a. (2000) verwenden viele Algorithmen zur Datenüberprüfung einen Toleranzwert. Dieser Wert gibt im Allgemeinen eine maximale Distanz zwischen zwei Punkten an, kann aber beispielsweise auch eine Winkeldifferenz sein. Der Toleranzwert ist dabei ein fester Schwellwert, da alle Elemente die innerhalb der Toleranz liegen als korrekt angesehen werden und alle anderen als inkorrekt. Servigne u. a. (2000) geben an, dass die Festlegung der Toleranz problematisch ist, da abhängig von der Wahl des Wertes fehlerhafte Objekte unentdeckt bleiben können und korrekte Objekte fälschlicherweise als fehlerhaft identifiziert werden können. Deshalb soll der Toleranzwert entsprechend der zu überprüfenden Objekte und der Genauigkeit des Datensatzes gewählt werden.

Diskussion Die Toleranz sollte bei allen Algorithmen berücksichtigt werden die räumliche und nicht-räumliche Attributwerte verarbeiten. Die Algorithmen haben prinzipiell nur Zugriff auf die rechnerinterne Repräsentation der Werte, die jedoch bei Gleitkommawerten von den Sollwerten geringfügig abweichen kann. Weist beispielsweise der zentimetergenau gemessene Sollwert der Distanz zweier Punkte den exakten Wert 3,00 m auf, so kann dieser intern im Rechner mit einem Wert von 3,000 000 01 oder auch 2,999 999 98 abgespeichert sein. Wird nun in einem Algorithmus der Wert geprüft, so sollte eine Toleranz von beispielsweise 0,0005 m verwendet werden um Fehler zu vermeiden.

Anforderung 21: Ausmaß

Unterstützung des Ausmaßes der Verletzung von Bedingungen

Um sukzessive Änderungen topologischer Relationen erklären zu können, untersuchen Egenhofer und Al-Taha (1992) die konzeptuelle Nachbarschaft topologischer Relationen für Paare von Flächen (siehe Abbildung 4.4). Während kontinuierliche topologische Transformationen die topologische Relation zweier Objekte erhalten, da sie auf beide Objekte angewendet werden, unterscheiden sich sogenannte sukzessive Änderungen (engl. 'gradual changes') beziehungsweise Deformationen dadurch, dass die Transformation nur auf eines der beiden Objekte angewendet wird. Durch diese kann die topologische Relation zweier Objekte verändert werden. Die Nähe zweier topologischer Relationen der 9IM wird in Egenhofer und Al-Taha (1992) durch die Summe der sich voneinander unterscheidenden Elemente der Schnittmatrix des 9IM definiert. Die Summe ist im Intervall 0 (gleiche Relation) bis 7 ('covers' zu 'inside', 'contains' zu 'coveredBy'). Die Übergänge zwischen den topologischen Relationen, die in Abbildung 4.4 durch Pfeile gekennzeichnet sind, können durch die Deformationen Skalierung, Rotation oder Transformation einer der beiden beteiligten Flächen erreicht werden. Beispiele für die Übergänge sind in Egenhofer und Al-Taha (1992) aufgeführt. Liegen für zwei Objekte zu zwei Zeitpunkten die topologischen Relationen vor, so kann aus der konzeptuellen Nachbarschaft auf die Art der Deformation zwischen diesen beiden Zeitpunkten geschlossen werden. Teilweise kann auch die nächste Änderung aufgrund einer Deformation vorausgesagt werden, beispielsweise folgt 'disjoint' immer 'meet'. Mäs (2010) referenziert weitere Arbeiten zur konzeptuellen Nachbarschaft zwischen den Kombinationen von Flächen – Linien und von Linien – Linien.

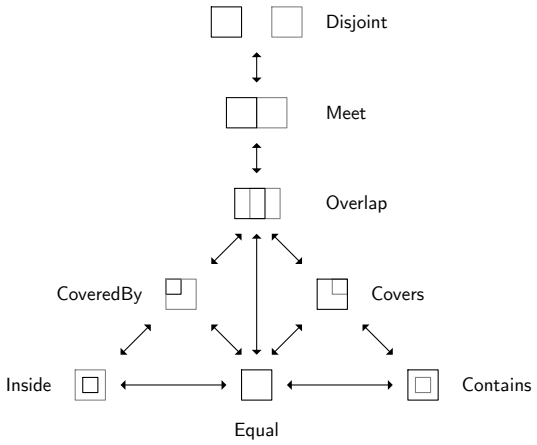


Abbildung 4.4: Konzeptueller Nachbarschaftsgraph nach Egenhofer und Al-Taha (1992)

In Rodríguez u. a. (2010) werden Maße vorgestellt um die Inkonsistenz von topologischen Integritätsbedingungen zu quantifizieren. Das Ausmaß der Verletzung der Integritätsbedingungen (engl. 'constraint violation measure') wird durch den Vergleich der tatsächlichen und erwarteten topologischen Relation zweier Geometrien bestimmt. Bei der Bestimmung des Ausmaßes werden zwei Aspekte berücksichtigt. Das Ausmaß selbst drückt aus wie sehr sich die beiden topologischen Relationen unterscheiden, indem die Kosten für die Korrektur der Relation ermittelt werden. Beispielsweise sind die Kosten proportional zur Distanz zweier Geometrien falls diese die Relation 'touch' erfüllen sollen, jedoch 'disjoint' sind. In Tabelle 4.11 sind die Basistransformationen zur Korrektur der genannten topologischen Relationen für den Geometrietyp Fläche aufgelistet. Der zweite Aspekt, die Relevanz der Objekte, drückt aus wie wichtig diese im Vergleich zu allen anderen Objekten sind. Die Relevanz wird als relatives Gewicht ausgedrückt. Prinzipiell ist die Relevanz anwendungsabhängig. Sie kann beispielsweise über die Größe der Objekte definiert werden. Als Beispiel wird in Rodríguez u. a. (2010) die Fläche von Objekten verwendet, d.h. je größer die Fläche desto wichtiger das Objekt. Überschneiden sich zwei kleine Flächen mit einen festen Prozentsatz, dann ist diese Verletzung weniger relevant als wenn sich zwei große Flächen mit demselben Prozentsatz überschneiden.

Von	Nach: 'touch'
'disjoint'	Eliminiere Abstand
'overlap', 'within', 'equal'	Eliminiere überlappende Fläche

Tabelle 4.11: Auswahl an Basistransformationen zur Änderung topologischer Relationen nach Rodríguez u. a. (2010)

Für topologische Relationen zwischen den geometrischen Primitiven Fläche – Fläche und Linie – Linie werden in Rodríguez u. a. (2010) Formeln für die Berechnung des Ausmaßes der Verletzung aufgestellt. Dabei werden sowohl das Ausmaß als auch die Relevanz im Intervall [0..1] normiert. Durch die Normierung sind beide Werte abhängig von den verwendeten Daten. Das Ausmaß der Verletzung ist dann das Produkt aus Ausmaß und Relevanz. Am Beispiel der erwarteten Relation 'touch' werden in Tabelle 4.12 die Maße für die Relation zwischen den Primitiven Fläche – Fläche dargestellt.

Von	Nach: 'touch'	Wert	Beschreibung
'disjoint'	$\frac{d_{me}}{c_1 + c_2 + d_{me}} \cdot \frac{c_1 + c_2}{2P}$	c_i	Umfang einer Geometrie g_i
'overlap'	$\frac{c_S}{c_1 + c_2 + c_S} \cdot \frac{c_1 + c_2}{2P}$	P	Maximaler Umfang aller Geometrien
'within'	$\frac{c_1 + d_{mi}}{2c_1 + c_2 + d_{mi}} \cdot \frac{c_1 + c_2}{2P}$	d_{me}, d_{mi}	Maximale externe/interne Distanz zweier Geometrien
'equal'	$\frac{c_1}{3c_1} \cdot \frac{c_1 + c_2}{2P}$	c_S	Umfang von $g_1 \cap g_2$

Tabelle 4.12: Formeln für die Berechnung des Ausmaßes für Fläche – Fläche nach Rodríguez u. a. (2010)

Diskussion Die konzeptuelle Nachbarschaft topologischer Relationen gibt einen Überblick über die Ähnlichkeit von zwei topologischen Konstellationen, beispielsweise einer Soll- und Ist-Konstellation. Dadurch kann deren Unterschied qualitativ spezifiziert werden.

Die Grundlagen der konzeptuellen Nachbarschaft eignen sich auch für die Definition von dynamischen und dynamisch sequentiellen Bedingungen (siehe Anforderung 2). Außerdem kann durch die Nachbarschaft auf die notwendigen Korrekturen beziehungsweise Deformationen geschlossen werden, die notwendig sind um eine unerwünschte topologische Relation zu korrigieren (siehe Anforderung 24).

Die Kombination von Ausmaß und Relevanz in Rodríguez u. a. (2010) ist sinnvoll, jedoch sollte für die Relevanz mehr die Semantik der Objekte und weniger deren Geometrie bestimmend sein. Zusätzlich zu rein numerischen Größen für das Ausmaß wie beispielsweise die Größe der Schnittfläche zweier Flächen sollte auch deren Form und Lage berücksichtigt werden. So unterscheidet sich in Abbildung 4.5 beispielsweise die absolute Größe der Schnittfläche der beiden Geometrien A und B nicht. Während sich im ersten Fall jedoch die beiden Geometrien lediglich am Rand schneiden, ragt die Geometrie von B im zweiten Fall weit in die Geometrie von A hinein.

Abbildung 4.5: Beispiel zur Anforderung Ausmaß

Allgemein sind die Anforderungen nach Schweregrad (Anforderung 18) und Ausmaß mit der Anforderung nach der Korrektur von Fehlern (Anforderung 24) verwandt. Das Ausmaß liefert die Grundlage für die Sortierung der Korrekturen, beispielsweise absteigend nach Ausmaß. Das Ausmaß kann auch die Grundlage für die Auswahl von Korrekturalgorithmen bilden, um beispielsweise unterschiedliche Algorithmen für geringe und umfangreiche Fehler auszuwählen.

Anforderung 22: Globales Ausmaß

Unterstützung für die Bestimmung globaler Maße für die Einhaltung und Verletzung von Integritätsbedingungen

Zusätzlich zur lokalen Ermittlung des Ausmaßes auf Basis binärer topologischer Relationen werden in Rodríguez u. a. (2010) auch zwei Maße für die globale Datenqualität beziehungsweise Inkonsistenz vorgestellt. Eine wichtige Größe ist dabei die Anzahl der überprüften topologischen Relationen (CTR, engl. 'checked topological relations'). Diese gibt an wie viele binäre topologische Relationen für eine Integritätsbedingung tatsächlich überprüft werden müssen. So müssen für eines von n Objekten eines Datensatzes bei einer 'disjoint' Relation nicht alle Relationen zu den $(n-1)$ anderen Objekten überprüft werden, sondern nur alle Relationen zu den Objekten in der direkten Nachbarschaft. Die Streuung der Inkonsistenzen (VS, engl. 'violation spread') kann dann nach Gleichung 4.1 bestimmt werden.

$$VS = \frac{Anzahl\ der\ Verletzungen}{CTR} \tag{4.1}$$

Das zweite Maß für die globale Datenqualität berücksichtigt das jeweilige Ausmaß der Verletzung jeder überprüften binären topologischen Relation. Die globale Erfüllung (GF, engl. 'global fulfillment') kann dann nach Gleichung 4.2 berechnet werden. Der Wert liegt im Intervall [0..1] wobei 0 auf eine maximale und 1 auf eine minimale Inkonsistenz der Daten hinweist.

$$GF = \frac{\sum_{1}^{CTR} (1 - Ausmaß\ der\ Verletzung)}{CTR} \tag{4.2}$$

Rodríguez u. a. (2010) vergleichen ihre Definition der globalen Erfüllung mit zwei alternativen Berechnungsmethoden. Der erste Ansatz berücksichtigt die semantische Distanz in Form der konzeptuellen Nachbarschaft topologischer Relationen. In die Formel zur Berechnung der globalen Erfüllung geht die normierte semantische Distanz zweier topologischer Relationen als Wert SD_i ein (siehe Gleichung 4.3). Der zweite Ansatz basiert auf der Berechnung der Distanz von zugeordneten Stützpunkten der Ränder zweier Geometrien. Dazu muss jedoch eine fehlerbereinigte Version der beiden Geometrien vorliegen, was nicht immer für alle Integritätsbedingungen möglich ist (siehe Anforderung 24).

$$GF_{sd} = \frac{\sum_{1}^{CTR} (1 - SD_i)}{CTR} \tag{4.3}$$

Diskussion Die für topologische Relationen definierten globalen Maße nach Rodríguez u. a. (2010) lassen sich auch auf andere Integritätsbedingungen übertragen. So lassen sich beispielsweise für metrische Bedingungen ebenso die Anzahl der Verletzungen und das jeweilige Ausmaß der Verletzung (siehe Anforderung 21) bestimmen. Der berechnete Wert für die globalen Erfüllung ist jedoch wie die Berechnung des Ausmaßes vom Datensatz abhängig, da dieser die Normierung der Werte bestimmt. Daraus folgt, dass der Wert für die globale Erfüllung nicht zwischen mehreren Datensätzen vergleichbar ist, es sei denn es werden die gleichen Werte für die Normierung herangezogen.

Eine Alternative zur Bestimmung globaler Maße für die Einhaltung und Verletzung von Integritätsbedingungen bildet die deskriptive Statistik, wie beispielsweise Anzahl, Minimum, Maximum, Mittelwert, Median und Standardabweichung der Verletzungen (siehe Unterabschnitt 2.5.1).

Anforderung 23: Umfangreiche Datensätze

Unterstützung für die Überprüfung von Integritätsbedingungen für umfangreiche Datensätze

Diskussion Sowohl beim Aufstellen von Integritätsbedingungen als auch bei der Implementierung der Überprüfung sollten Anforderungen, die durch die Verwendung umfangreicher Datensätze entstehen, berücksichtigt werden. Beim Aufstellen der Bedingungen betrifft dies vor allem die Komplexität der Algorithmen und damit deren Laufzeit. Soll beispielsweise der mittlere Abstand eines Objekts zu allen Objekten im Umkreis von 10 km berechnet werden, dann müssen viele Objekte zueinander in Beziehung gesetzt werden. Wird dagegen nur eine Integritätsbedingung für den minimalen Abstand zum nächstliegenden Objekt aufgestellt, dann müssen deutlich weniger Objekte prozessiert werden. Bei der Implementierung der Bedingungen müssen gegebenenfalls Aspekte der Parallelisierung berücksichtigt werden, die sich beispielsweise darauf beziehen welche Integritätsbedingungen parallel prozessiert werden können und wie eine räumliche Partitionierung sinnvoll erstellt werden kann (siehe auch Abschnitt 7.1).

4.5 Änderung von Daten und Integritätsbedingungen

Wird bei der Prüfung der Integritätsbedingungen für einen Datensatz festgestellt, dass einige Objekte Bedingungen verletzen, dann existieren mehrere Möglichkeiten mit diesen Objekten umzugehen. Eine Möglichkeit ist die Veränderung der Daten, so dass diese die Bedingungen nicht mehr verletzen. Eine zweite Möglichkeit besteht in der Überarbeitung der Bedingungen.

Die vier im Folgenden vorgestellten Anforderungen der Änderung von Daten und von Integritätsbedingungen umfassen wichtige Aspekte, die nach der Prüfung der Bedingungen berücksichtigt werden können. So können Fehler teilweise (semi-) automatisch korrigiert werden (Anforderung 24). Wird ein Objekt verändert, das mit anderen Objekten über Integritätsbedingungen verbunden ist, dann können die verbundenen Objekte gegebenenfalls automatisch aktualisiert werden (Anforderung 25). Statt einer Korrektur der Daten können auch die Bedingungen dahingehend modifiziert werden, dass Ausnahmen (Anforderung 26) explizit definiert werden. Prinzipiell sind alle Änderungen an Integritätsbedingungen unter dem Aspekt der Sicherheit (Anforderung 27) zu sehen, da in einer Organisation klar definiert sein sollte, wer welche Integritätsbedingungen in welchem Umfang ändern darf.

Anforderung 24: Korrektur von Fehlern

(Semi-) Automatische Korrektur von Fehlern

Die Korrektur von geometrischen und topologisch-semantischen Fehlern wird in Ubeda und Egenhofer (1997), Ubeda (1997) und Servigne u. a. (2000) diskutiert. Die Korrektur von Geometrien basiert dabei vor allem auf der Korrektur von (Stütz-) Punkten. Diese können hinzugefügt, gelöscht, verschoben, verschmolzen oder auf ein Liniensegment projiziert werden. Topologisch-semantische Fehler zwischen zwei Klassen müssen nach Ubeda (1997) manuell korrigiert werden, wenn die Anzahl der an einer Beziehung teilnehmenden Objekte der Klassen spezifiziert ist (siehe Anforderung 9). Dazu müssen die über- oder unterzähligen Objekte identifiziert werden. Wird eine topologische Beziehung zwischen zwei Klassen hingegen verboten, dann kann die Korrektur der Fehler semi-automatisch erfolgen. Dazu können fehlerhafte Objekte verschoben, verformt, gelöscht oder geteilt werden. Für jeden Fehler wird eine Liste von möglichen Korrekturszenarien berechnet. Es wird vorgeschlagen, dass diese Liste anschließend gefiltert und sortiert wird. Die Filterung soll beispielsweise Korrekturen aussortieren, bei denen ein Objekt weiter als ein gewisser Schwellwert verschoben werden müsste. Die Sortierung der Korrekturen soll nach einem vom Benutzer auswählbaren Kriterium erfolgen. Zu den Kriterien zählen unter anderem der Flächenerhalt, minimale Verschiebungen oder gültige alternative topologische Relationen.

Servigne u. a. (2000) geben zwei zusätzliche Anforderungen an die Konsistenz der Korrektur von Fehlern an. Erstens dürfen die Korrekturen keine neuen Fehler erzeugen. Zweitens sollen die Korrekturen in einer geordneten Reihenfolge durchgeführt werden. Zuerst sollen Fehler bereinigt werden, die sich nur auf ein Objekt beziehen, und danach erst Fehler, die sich auf mehrere Objekte beziehen.

In Mäs u. a. (2005) werden für Integritätsbedingungen eindeutige Instruktionen zur Korrektur von Fehlern in natürlicher Sprache angegeben.

Die Korrektur von Fehlern wird in Wang und Reinhardt (2007) durch die Angabe einer Methode definiert. Die Methoden einer Klasse aktualisieren die Geometrie eines Objektes, erzeugen oder löschen Objekte und erlauben den lesenden und schreibenden Zugriff auf deren Attribute. Wie jedoch die Geometrie eines Objektes im Detail verändert werden muss bleibt dem Benutzer überlassen.

Rodríguez u. a. (2008) stellen eine Semantik zur Reparatur der topologischen Beziehungen von Objektgeometrien vor. Die Geometrien weisen dabei topologische Beziehungen auf, die nicht erlaubt sind und somit korrigiert werden müssen. Rodríguez u. a. (2008) sehen lediglich zwei Operationen zur Korrektur der Geometrien vor, die beide darauf ausgelegt sind nur minimale Veränderungen zu erzeugen. Erstens kann die Geometrie eines Objektes verkleinert werden. Um zwei überlappende Geometrien zu korrigieren, kann eine der Geometrien um die gemeinsame Schnittfläche verkleinert werden. Um zwei sich berührende Geometrien zu korrigieren, kann eine der Geometrien um einen Puffer verkleinert werden. Zweites kann ein Objekt auch komplett gelöscht werden um Duplikate zu entfernen. In Rodríguez u. a. (2010) wird auf die Beschränkungen des entwickelten Ansatzes hingewiesen. So können topologische Bedingungen nicht gefordert, sondern nur verboten werden. Außerdem sind die Reparaturen nur für kleine Ausschnitte der Daten vorgesehen. Die Angabe minimaler Reparaturen für einen kompletten Datensatz ist nach Rodríguez u. a. (2010) nur schwer oder unlösbar.

Diskussion Die Korrektur von Fehlern ist nur für einzelne Objekte beziehungsweise für eine kleine Menge von Objekten automatisierbar. Soll jedoch ein gesamter Datensatz optimal automatisch korrigiert werden, dann

wird die Korrektur schwer bis unlösbar. Zu dieser Problematik tragen mehrere Aspekte bei. Erstens sind die Korrekturen voneinander abhängig, d.h. wird ein Objekt verändert, dann ändern sich dadurch auch alle mit dem Objekt in Beziehung stehenden Objekte. Anschaulich lassen sich die jeweils möglichen Entscheidungen zur Korrektur in einem Baum darstellen. Alleine die Auswahl des zu verändernden Objektes erzeugt viele Äste, die zusätzlich mit der Anzahl der möglichen Korrekturen für das jeweils ausgewählte Objekt eine weitere Ebene von Ästen hinzufügen. Dies führt sowohl bei iterativen Verfahren als auch bei einer simultanen Lösung zu einer hohen Komplexität. Zweitens ist die Auswahl der Optimierungskriterien wie auch die Suche nach der optimalen Korrektur des Datensatzes sehr komplex. Mögliche Optimierungskriterien sind beispielsweise die minimale Änderung der Form oder der Fläche von Objektgeometrien und der Erhalt der topologischen Beziehungen (siehe auch Ubeda (1997)).

Zwei weitere Anforderungen sind mit der Korrektur von Fehlern verbunden. Konsistenzprüfungen können (teilweise) Widersprüche und Redundanzen in einer Menge von Integritätsbedingungen aufdecken (siehe Anforderung 19). Zudem können Fehler in den Daten auch als Ausnahmen deklariert werden (siehe Anforderung 26).

Anforderung 25: Automatische Aktualisierung

> Automatische Aktualisierung von Objekten, die mit dem aktualisierten Objekt über Integritätsbedingungen in Beziehung stehen

Belussi u. a. (2000) zeigen die automatische Aktualisierung von Objekten am Beispiel von Straßensegmenten mit Verkehrsinseln auf. Eine Verkehrsinsel hat dabei die gleiche Länge wie das zugehörige Straßensegment und liegt in deren Mitte (siehe Abbildung 4.6). Wird das Straßensegment verlängert oder gekürzt, dann wird auch automatisch die Verkehrsinsel entsprechend angepasst. Die Abbildung 4.6 zeigt links die Ausgangssituation, in der Mitte die Verkürzung des Straßensegments und rechts die automatische Aktualisierung der Verkehrsinsel auf das veränderte Straßensegment. Veränderungen an Objekten werden dabei durch erzeugen oder löschen einer Fläche, durch hinzufügen, teilen oder verschmelzen von Segmenten einer Fläche und durch löschen von Stützpunkten durchgeführt.

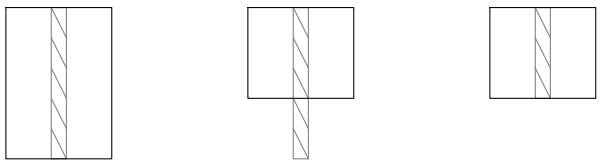


Abbildung 4.6: Automatische Aktualisierung der Objekte Straßensegment und Verkehrsinsel nach Belussi u. a. (2000)

Die interaktive Modifikation von Objekten mit Bedingungen wird in Brenner (2005) diskutiert. Am Beispiel der Verschiebung von einem der Stützpunkte eines Rechtecks werden vier Möglichkeiten zur Änderung der Geometrie angegeben (siehe Abbildung 4.7). Die einfachste Möglichkeit ist die Änderung der Koordinaten des verschobenen Stützpunkts. Die zweite Möglichkeit besteht darin, den diagonal gegenüberliegenden Stützpunkt festzuhalten und nur die Breite und Länge des Rechtecks anzupassen. Bei der dritten Möglichkeit bleibt der Mittelpunkt des Rechtecks unverändert. Die gleichzeitige Rotation und Skalierung des Rechtecks bildet schließlich die vierte Möglichkeit. Sind mehrere Objekte über Bedingungen miteinander verbunden, dann stellt sich nach Brenner (2005) die Frage nach einem geeigneten Optimierungskriterium. Durch die Methode der kleinsten Quadrate werden an vielen Objekten viele kleine Korrekturen angebracht. Alternativen, wie die Minimierung der Anzahl der modifizierten Objekte, führen jedoch zu nicht eindeutigen Lösungen.

Abbildung 4.7: Automatische Aktualisierung eines Rechtecks nach Brenner (2005)

Diskussion Die automatische Aktualisierung von Objekten auf Basis der Nachführung von Integritätsbedingungen geht einen Schritt weiter als die Korrektur von Fehlern (siehe Anforderung 24). Die beiden Anforderungen weisen jedoch beide eine hohe Komplexität auf und die damit verbundenen Einschränkungen hinsichtlich ihrer Lösbarkeit und Automatisierbarkeit.

Es existieren mehrere sinnvolle Awendungsszenarien für die automatische Aktualisierung. Sind Objekte über eine Aggregation miteinander verbunden (siehe Anforderung 6), dann kann die Veränderung eines der Objekte bei

den anderen Objekten nachgeführt werden. Der einfachste Fall dieses Szenarios ist die Teil – Ganzes Beziehung. Wird beispielsweise die Geometrie eines Teils verändert, dann sollte sich automatisch die Geometrie des Ganzen entsprechend mit ändern.

Parametrische geometrische Integritätsbedingungen in Anlehnung an parametrische Modelle des Computer Aided Design (siehe Anforderung 12) beschreiben explizit die Beziehungen zwischen den einzelnen Geometrien eines Objektes. Wird also eine Geometrie verändert, dann können automatisch die davon funktional abhängigen Attribute und Geometrien aktualisiert werden.

Ein weiteres Anwendungsszenario für die automatische Aktualisierung ist in Planungsszenarien zu sehen. So können mehrere Szenarien für die Erzeugung, Modifikation und Löschung von Objekten vom Benutzer durchgeführt werden und die Auswirkungen auf andere Objekte und die Erfüllung der Integritätsbedingungen studiert werden.

Anforderung 26: Ausnahmen

> Keine Regel ohne Ausnahme: Unterstützung von Ausnahmen sowohl für einzelne Objekte als auch für Objektgruppen bzw. (Unter-) Klassen

Die Hypothese, dass tatsächlich alle Objekte eines Datensatzes einer Regel folgen müssen, ist nach Servigne u. a. (2000) zu strikt. Es werden zwei Ansätze zur Definition von Ausnahmen für topologische Beziehungen vorgestellt. Der erste Ansatz ist iterativ. Er beginnt mit der Überprüfung der strikten Regel und erlaubt dann die sukzessive Hinzunahme von Ausnahmen aus eindeutig identifizierbaren Objektpaaren, welche die Regel nicht erfüllen. Der zweite Ansatz ermöglicht die Definition der Ausnahmen bereits bei der Erstellung der Regel. Dabei können Unterklassen von der Überprüfung a priori ausgenommen werden. Beispiele für die beiden Ansätze sind in Tabelle 4.13 aufgelistet.

Ansatz	Beispiel
1	(Straße, 'cross', Gebäude, verboten) mit Ausnahme (S.7,G.112), (S.8,G.114), ...
2	(Straße, 'cross', Gebäude, verboten) mit Ausnahme (Tunnel, Gebäude)

Tabelle 4.13: Definition von Ausnahmen nach Servigne u. a. (2000)

Räumliche Beziehungen sollen nach Friis-Christensen u. a. (2001) Ausnahmen erlauben. Für die exemplarische Bedingung dass Häuser keine Gewässer schneiden dürfen, geben sie Häuser auf Pfeilern als Ausnahme an.

Borges u. a. (2002) sehen sowohl für Integritätsbedingungen als auch für Konsistenzprüfungen die Notwendigkeit für einen Mechanismus, der Ausnahmen für spezielle Fälle ermöglicht. Als Beispiel geben sie Verbindungsbrücken zwischen Gebäuden an. Obwohl diese zu Gebäuden zählen, dürfen sie Straßen schneiden.

Um Integritätsbedingungen nutzbarer und realistischer zu machen, führt Xu (2011) Ausnahmen für unübliche und nicht einheitliche Fälle ein. Regeln sind entweder der Klasse 'strikt verboten' oder der Klasse 'unüblich, aber akzeptabel' zugeordnet. In natürlicher Sprache verwenden Regeln analog die Verben muss und kann oder die Verben sollte und darf. Ausnahmen werden auf Basis einzelner Instanzen definiert, die explizit als solche gekennzeichnet werden. Der Ablauf besteht dabei aus mehreren Schritten. Zuerst werden die Daten geprüft und alle fehlerhaften Instanzen getrennt von den fehlerfreien Instanzen gespeichert. Diese erhalten als weitere Attribute einen booleschen Wert, der angibt ob die Instanz eine Ausnahme ist, sowie als Text die Regel welche die Instanz als fehlerhaft identifiziert hat. Der boolesche Wert kann dann vom Benutzer geändert und damit die Instanz als Ausnahme markiert werden. Diese wird schließlich zusätzlich bei den fehlerfreien Instanzen gespeichert.

Diskussion Die Unterstützung von Ausnahmen ist eine existentielle Anforderung, da diese in nahezu jedem Datensatz auftreten. Ausnahmen sollten sowohl für einzelne Objekte beziehungsweise Objektpaare definiert werden können als auch für Objektgruppen und (Unter-) Klassen. Der entsprechende Filter sollte also so mächtig wie möglich sein, um Ausnahmen exakt spezifizieren zu können.

Bei der Definition von Ausnahmen in Servigne u. a. (2000) fehlt die Verbindung beider Ansätze, d.h. sowohl Ausnahmen aus eindeutig identifizierbaren Objekten als auch Ausnahmen anhand von Objektgruppen und (Unter-) Klassen. Xu (2011) erlaubt Ausnahmen nur für einzelne Objekte. Bei beiden Ansätzen fehlt der Dokumentation darüber, warum bestimmte Objekte als Ausnahmen deklariert werden. Dies ist wichtig um die Nachvollziehbarkeit zu gewährleisten.

Werden Ausnahmen anhand eindeutig identifizierbarer Objekte definiert, dann scheint es sinnvoll nach der Definition aller Ausnahmen die Menge der Objekte auf Gemeinsamkeiten zu untersuchen. Gegebenenfalls führt dies zu einer verbesserten Klassifikation der Objekte oder zur vereinfachten Definition der Ausnahmen auf Basis von Objektgruppen oder (Unter-) Klassen.

Wichtig bei der Definition von Ausnahmen ist auch, dass der Vorgang der Identifikation von Ausnahmen iterativ ist. Durch markieren eines Objektes als Ausnahme können sich Attribute und Beziehungen anderer Objekte ändern, die dann entweder keine Ausnahme mehr sind, keine Ausnahme bleiben oder zu einer neuen Ausnahme führen. Dieses iterative Vorgehen basiert auf einer Strategie zur Abarbeitung der als fehlerhaft identifizierten Objekte, die beispielsweise mit der größten Abweichung beginnt und dann iterativ absteigend nach der Größe des Fehlers vorgeht.

Die Anforderung nach Ausnahmen hat Querbezüge zu anderen Anforderungen. Ist es effizienter die Ausnahmen zu kodieren als die Regel, dann ist die Anforderung 14 wichtig. Zudem stellt sich die Frage welche Benutzer Ausnahmen definieren dürfen beziehungsweise in welchem Umfang. Dieser Aspekt bezieht sich auf die Sicherheit (Anforderung 27).

Anforderung 27: Sicherheit

> Unterstützung von Benutzerrechten für das Anlegen, Ändern und Löschen von Integritätsbedingungen

Der Aspekt der Sicherheit dient nach Cockcroft (1997) zur Kontrolle darüber, welche Benutzer welche Klassen von Integritätsbedingungen erstellen oder ändern dürfen. Die entsprechende Rechteverwaltung basiert auf der Authentifizierung der Benutzer innerhalb ihrer Organisation.

Louwsma (2004) verwendet den Begriff Interaktivität um auszudrücken, ob eine Integritätsbedingung durch den Benutzer verändert werden kann oder unveränderlich ist. Zur Veränderung zählt auch die Erstellung neuer Bedingungen um beispielsweise individuelle Richtlinien bezüglich der Daten abzubilden. Unveränderliche Bedingungen umfassen beispielsweise auf Theoremen basierte Bedingungen wie 'ein Haus kann nicht schweben'. Louwsma (2004) schlägt die Speicherung der Veränderlichkeit als zusätzliches Attribut einer Regel vor.

Diskussion Die Frage, wer Integritätsbedingungen erstellen, ändern und löschen darf, hängt damit zusammen wann dies erfolgt. Exemplarisch wird im Folgenden ein Arbeitsablauf skizziert. Experten stellen ein Datenmodell mit den entsprechenden Integritätsbedingungen auf. Daten werden dann in dieses Modell von einem Datenproduzenten importiert. Benutzer verändern diese Daten, fügen neue Daten hinzu oder löschen Daten.

Die Anforderung nach Sicherheit steht mit zwei weiteren Anforderungen in enger Beziehung. Durch die Definition von Ausnahmen (Anforderung 26) können Integritätsbedingungen dahingehend geändert werden, dass sonst ungültige Instanzen akzeptiert werden. Deshalb muss bei der Definition der Nutzerrechte zur Änderung von Integritätsbedingungen auch definiert werden, ob die Erstellung von Ausnahmen zugelassen wird. Die zweite Anforderung betrifft den Schweregrad und das Ausmaß (Anforderungen 18 und 21). Durch den Schweregrad kann kodiert werden wer eine Integritätsbedingung modifizieren darf. Das Ausmaß der Verletzung einer Integritätsbedingung gibt an, um welchen Wert beispielsweise der Schwellwert einer Integritätsbedingung geändert beziehungsweise relaxiert werden müsste, damit eine Instanz wieder gültig wäre.

5 Integritätsbedingungen: Formalisierung und Implementierung

Die formale Spezifikation von Integritätsbedingungen bildet die Grundlage für die informationstechnische Verarbeitung der Bedingungen und die automatische Überprüfung von Daten bezüglich ihrer Konformität zu den Bedingungen. Sie bildet aber auch die Grundlage dafür, dass Bedingungen zwischen mehreren Akteuren eindeutig interpretiert werden und sich somit Diskussionen mit dem Inhalt und nicht der Form von Bedingungen befassen können.

Die in diesem Kapitel vorgestellte Formalisierung basiert auf den Grundlagen der Model Driven Architecture (MDA), die in Abschnitt 2.3 ausführlich dargestellt ist. Insbesondere wird die Object Constraint Language (OCL) erweitert, mit der Bedingungen an Elemente in Unified Modeling Language (UML)-Modellen eindeutig spezifiziert werden können. Die GeoOCL genannte Erweiterung ermöglicht insbesondere die Aufstellung geometrischer und topologischer Integritätsbedingungen durch die Berücksichtigung entsprechender Standards des Open Geospatial Consortium (OGC) und der International Organization for Standardization (ISO) (siehe Abschnitt 2.4). Die GeoOCL berücksichtigt darüber hinaus die in Kapitel 4 aufgestellten Anforderungen an Integritätsbedingungen und zeigt auf, wie diese entsprechend formalisiert werden können.

Im Folgenden wird in Abschnitt 5.1 zuerst die Formalisierung von Integritätsbedingungen auf Basis der OCL und der GeoOCL detailliert vorgestellt. Anschließend werden in Abschnitt 5.2 kurz einige Aspekte der Implementierung entsprechender Integritätsbedingungen diskutiert.

5.1 Formalisierung

Für die Formalisierung von Integritätsbedingungen wird in dieser Arbeit die OCL als Basis verwendet um Bedingungen eindeutig, plattformunabhängig, formal korrekt und dennoch verständlich modellieren zu können. Die GeoOCL erweitert die OCL dabei nicht nur um die Möglichkeit räumliche Integritätsbedingungen aufstellen zu können, sondern auch um weitere Sprachkonstrukte mit der die Ausdrucksfähigkeit der OCL verbessert wird.

Im Folgenden wird in Unterabschnitt 5.1.1 erst diskutiert, wieso eine Erweiterung der OCL den geeignetsten Ansatz zur Formalisierung von Integritätsbedingungen darstellt. Dabei werden die möglichen Alternativen kurz hinsichtlich ihrer Schwächen untersucht. In Unterabschnitt 5.1.2 wird dann die Erweiterung der OCL zur GeoOCL vorgestellt, mit der zusätzlich zur OCL auch geometrische und topologische Integritätsbedingungen aufgestellt werden können. In Unterabschnitt 5.1.3 wird schließlich untersucht, inwieweit die Anforderungen aus Kapitel 4 in Integritätsbedingungen in der GeoOCL umgesetzt werden können und welche zusätzlichen Erweiterungen für bestimmte Anforderungen notwendig sein könnten.

5.1.1 Ansatz

Eine Erweiterung der OCL ist aus mehreren Gründen der geeignetste Ansatz zur Formalisierung von Integritätsbedingungen. In Bezug auf die Abstraktionsebenen der Formalisierung (siehe Abschnitt 3.3) ist die OCL dem konzeptuellen Modell zuzuordnen[1], in dem Entitäten formal und plattformunabhängig beschrieben werden. Die OCL, und damit auch eine Erweiterung der OCL, bietet durch ihr hohes Abstraktionsniveau somit die drei wichtigen Vorteile der Portabilität, Interoperabilität und Wiederverwendbarkeit[2].

Mit der OCL formalisierte Integritätsbedingungen sind portabel, da aus den Bedingungen Implementierungen für verschiedene Plattformen abgeleitet werden können. So können dieselben Integritätsbedingungen beispielsweise in ein Datenüberprüfungsmodul eines Geoinformationssystem (GIS) oder in entsprechende Konstrukte zur Überprüfung von Daten in einer Datenbank überführt werden. Die Interoperabilität von mit der OCL formalisierten Integritätsbedingungen zeigt sich darin, dass zwei Systeme, die dasselbe plattformunabhängige Modell verwenden, zusammenarbeiten können. Konkret manifestiert sich die Interoperabilität darin, dass, wenn

[1] Da die OCL eine formale Sprache ist um Ausdrücke in UML-Modellen beschreiben zu können, ist sie demselben Abstraktionsniveau wie die UML zuzuordnen.

[2] Portabilität, Interoperabilität und Wiederverwendbarkeit sind allgemein die drei Vorteile der MDA, deren grundlegende Konzepte in Unterabschnitt 2.3.1 beschrieben sind.

Objekte eines Quell-Systems in ein Ziel-System übertragen werden, diese im Ziel-System dieselben Integritätsbedingungen erfüllen müssen wie im Quell-System. Die Wiederverwendbarkeit von mit der OCL formalisierten Integritätsbedingungen zeigt sich darin, dass Bedingungen aus einem Modell in ein anderes Modell übernommen werden können, sofern beide Modelle mit der UML spezifiziert sind. Beispielsweise können Integritätsbedingungen die an Straßengeometrien im Datenmodell eines Landes gestellt werden, mit gegebenenfalls nur geringfügigen Änderungen auf die Straßengeometrien im Datenmodell eines anderen Landes übertragen werden.

Alternative Ansätze zur Formalisierung von Integritätsbedingungen, wie sie in Unterabschnitt 3.3.3 vorgestellt werden, bieten diese wichtigen Vorteile nicht. Die Aufstellung von Integritätsbedingungen in Datenbanken mit Assertions oder Triggern ist nicht plattformunabhängig, sondern spezifisch für eine Plattform und damit nur bedingt portabel, interoperabel und wiederverwendbar. Gleiches gilt für die Formalisierung von Integritätsbedingungen in der Extensible Markup Language (XML). Die weiteren vorgestellten Ansätze basierend auf selbst definierten formalen Sprachen beziehungsweise Konstrukten sind auch nur bedingt geeignet, da diese nicht standardisiert sind und damit die genannten Vorteile ebenso wenig erfüllen können.

Die in Unterabschnitt 3.3.1 vorgestellten räumlichen objektorientierten Datenmodelle sind zwar prinzipiell geeignet, sie sind jedoch weder standardisiert noch verbreitet. Zudem ist die Mächtigkeit der grafischen Modellierung von Integritätsbedingungen hinsichtlich der Komplexität der Bedingungen und deren übersichtlich darstellbaren Anzahl beschränkt.

Aus diesen Gründen ist eine Erweiterung der OCL der geeignetste Ansatz zur Formalisierung von Integritätsbedingungen. Die GeoOCL genannte Erweiterung wird in den weiteren Unterabschnitten vorgestellt und hinsichtlich ihrer Anwendbarkeit und Ausdrucksfähigkeit untersucht.

5.1.2 GeoOCL

Die GeoOCL stellt eine Erweiterung der OCL dar[3]. Der größte und wichtigste Teil der Erweiterung bilden die Grundlage für geometrische und topologische Integritätsbedingungen. Weitere kleinere Zusätze erweitern die Ausdrucksfähigkeit der Sprache im Allgemeinen, wodurch komplexere Integritätsbedingungen aufgestellt werden können. Die einzelnen Erweiterungen der OCL zur GeoOCL werden im Folgenden vorgestellt.

Geometrie und Topologie

Die Basistypen der OCL umfassen lediglich alphanumerische Typen und Auflistungen. Der einfache und elegante Ansatz der Erweiterung der OCL zur GeoOCL besteht nun darin, dass die Basistypen der OCL um zusätzliche geometrische und topologische Basistypen erweitert werden. Die zusätzlichen Basistypen der GeoOCL entsprechen dabei den standardisierten Datentypen des OGC Simple Feature Standards (OGC Simple Feature, 2011) und des räumlichen Schemas der ISO 19107 (2003).

In Abbildung 5.1 ist exemplarisch die Erweiterung der Basistypen der OCL um die Basistypen des OGC Simple Feature Standards skizziert, wobei einige Unterklassen aus Gründen der Übersichtlichkeit nur angedeutet sind. Das UML-Klassendiagramm stellt dabei eine Kombination der entsprechenden Klassendiagramme aus Abbildung 2.19 und Abbildung 2.20 dar. Die Klasse Geometry sowie deren Unterklassen befinden sich dabei auf derselben Ebene der Vererbungshierarchie wie die restlichen Basistypen der OCL. Dadurch verfügen diese Klassen auch über die Operationen der Klasse OclAny, die beispielsweise Objektvergleiche ermöglichen. Die GeoOCL übernimmt ebenso die entsprechenden Operationen des OGC Simple Feature Standards, die in Tabelle 2.5 aufgelistet sind.

Die Klasse GeometryCollection bedarf dabei einer gesonderten Betrachtung. Laut OGC Standard modelliert die Klasse eine Auflistung von Geometrien ohne jegliche Anforderungen an die Elemente der Auflistung (siehe auch Unterabschnitt 2.4.1). So sind beispielsweise Duplikate ebenso erlaubt wie die Mischung mehrerer Geometrietypen in einer Auflistung. Die Operationen der Klasse GeometryCollection erlauben zudem den wahlfreien Zugriff auf deren Elemente durch Angabe des gewünschten Index in der Auflistung. Die Klasse GeometryCollection wird deshalb in der GeoOCL als eine Unterklasse der Klasse Sequence behandelt, die wiederum eine Unterklasse der Klasse Collection in der OCL ist. Die Klasse Sequence ermöglicht im Gegensatz zu den Klassen Set und OrderedSet Duplikate und die Elemente der Auflistung sind im Gegensatz zur Klasse Bag sortiert (siehe auch Unterabschnitt 2.3.3). So ermöglicht lediglich die Klasse Sequence mit der Operation at(i: Integer) den wahlfreien Zugriff auf Elemente der Auflistung basierend auf ihrem Index.

[3] In Werder (2009) wird der Begriff GeoOCL erstmals geprägt und ein rudimentärer Stand vorgestellt.

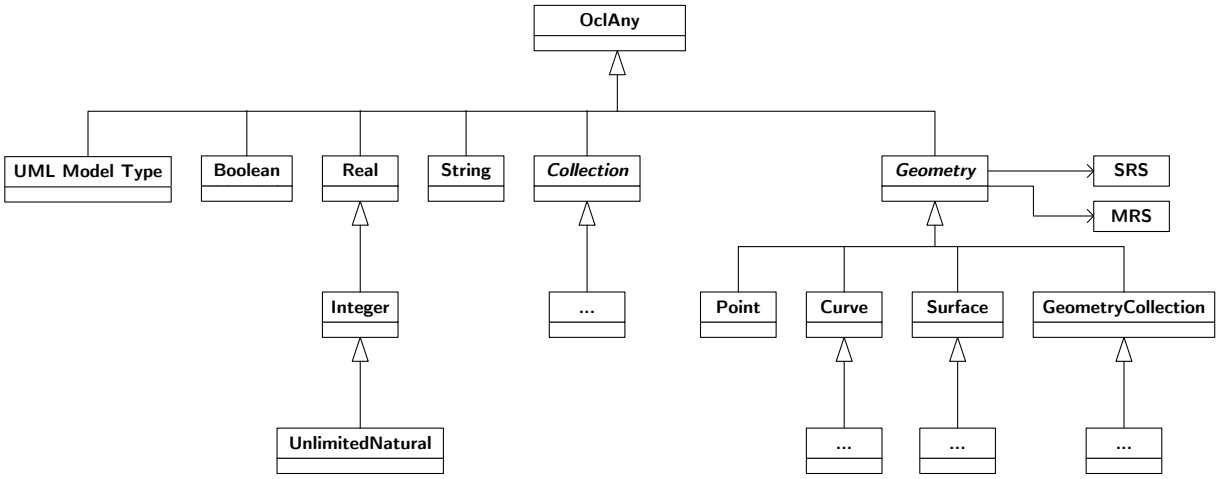


Abbildung 5.1: Prinzip der Erweiterung der Basistypen für die GeoOCL

Die Erweiterung der OCL um die geometrischen und topologischen Basistypen des ISO 19107 (2003) Standards sowie deren Operationen erfolgt analog, d.h. die Oberklasse Object des Geometriemodells (siehe Abbildung 2.21) und die Oberklasse Primitive des Topologiemodells werden auf der gleichen Ebene der Vererbungshierarchie eingefügt wie die Klasse Geometry des OGC Simple Feature Standards. In dieser Arbeit wird jedoch die Erweiterung um die entsprechenden Basistypen des ISO 19107 (2003) Standards nicht weiter eingesetzt, da die verwendeten Daten geeignet durch das Geometriemodell des OGC Simple Feature Standards modelliert werden können.

Die bis hier vorgestellte Erweiterung ermöglicht bereits die Aufstellung typischer geometrischer und topologischer Integritätsbedingungen. Als Beispiel werden im Folgenden Integritätsbedingungen für die drei Klassen Gebaeude, Baum und Strasse aufgestellt, deren UML-Klassendiagramm in Abbildung 5.2 dargestellt ist. Die drei Klassen weisen jeweils ein Attribut namens geom auf, das eine punktuelle, linien- oder flächenhafte Geometrie entsprechend des OGC Simple Feature Standards modelliert.

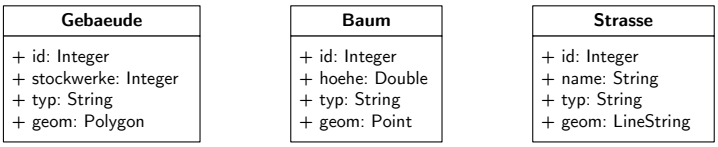


Abbildung 5.2: UML-Klassendiagramm für die Formulierung exemplarischer Integritätsbedingung in der GeoOCL

Für die drei Klassen sind im Programm 5.1 einige Integritätsbedingungen in der GeoOCL aufgestellt. Die Invariante Einfach stellt sicher, dass sich die Geometrien von Gebäuden oder Straßen nicht selbst überschneiden. Die Operation isSimple ist dabei eine Operation der Oberklasse Geometry, die auch für die Unterklassen LineString und Polygon definiert ist. Die Bedingung MinQuadrat fordert, dass der Umring jeder Gebäudegeometrie mindestens vier Stützpunkte aufweist. Die Integritätsbedingung verwendet dabei eine Verkettung von zwei Operationen. Von der Operation exteriorRing der Klasse Polygon wird ein LineString zurückgegeben, dessen Operation numPoints wiederum die Anzahl der Stützpunkte liefert. Die Invarianten MinFlaeche der Klasse Gebaeude und MinLaenge der Klasse Strasse nutzen wiederum die entsprechenden Operationen der Geometrietypen. Die Invariante Freistehend der Klasse Baum ist schließlich die komplexeste Integritätsbedingung. Sie fordert, dass falls ein Objekt der Klasse Baum vom Typ freistehend ist, die minimale Distanz aller anderen Objekte der Klasse zu diesem Objekt nicht unterschritten wird.

Wie im Beispiel gezeigt, können mit der auf den Standards der OGC und ISO basierenden Erweiterung der OCL bereits vielfältige geometrische und topologische Integritätsbedingungen aufgestellt werden. Für komplexere Integritätsbedingungen sind jedoch noch einige kleinere Zusätze notwendig, die im Folgenden vorgestellt werden.

Zusätzliche Erweiterungen

Die zusätzlichen Erweiterungen basieren teilweise auf den in Unterabschnitt 3.3.2 vorgestellten Erweiterungen der OCL und teilweise auf Erweiterungen aufgrund der Anforderungen aus Kapitel 4 und der Anwendung der Bedingungen in den Kapiteln 6 und 7.

```
context Gebaeude
  inv Einfach: geom.isSimple()
  inv MinQuadrat: geom.exteriorRing().numPoints() >= 4
  inv MinFlaeche: geom.area() >= 40 -- [Quadratmeter]

context Baum
  inv Freistehend: typ = 'freistehend' implies Baum.allInstances()->forAll(b | b.id <> id implies b
      .geom.distance(geom) >= 15.0) -- [Meter]

context Strasse
  inv Einfach: geom.isSimple()
  inv MinLaenge: geom.length() >= 2 -- [Meter]
```

Programm 5.1: Exemplarische Integritätsbedingung in der GeoOCL

Von Casanova u. a. (2000) werden für die GeoOcl die Erweiterung von Invarianten auf mehrere Klassen des Modells sowie die generischen Integritätsbedingungen übernommen. Dadurch können Invarianten aufgestellt werden, die sich auf den Kontext mehrerer Klassen beziehen. Für das Beispiel aus Abbildung 5.2 kann so im Programm 5.2 die Invariante MinDistanzZuBaum definiert werden, die fordert dass Bäume mindestens zwei Metern von Gebäuden entfernt sind.

```
context Gebaeude
  inv MinDistanzZuBaum: Baum.allInstances()->forAll(b | b.geom.distance(geom) >= 2) -- [Meter]
```

Programm 5.2: Integritätsbedingung über mehrere Klassen in der GeoOCL

Die in Unterabschnitt 3.3.2 definierten geometrischen und topologischen Integritätsbedingungen können durch die GeoOcl bereits standardkonform und syntaktisch korrekt definiert werden.

Für die Aufstellung von Bedingungen, die sich auf Elemente einer Geometrie beziehen, werden zusätzliche Operationen benötigt. Dazu werden die Operationen des Ogc Simple Feature Standards (siehe Tabelle 2.5) um weitere Operationen ergänzt. Diese sind in Tabelle 5.1 aufgelistet. So können beispielsweise die Stützpunkte einer Linie durch die Operation points abgefragt werden, die eine Auflistung von Punkten zurückgibt. Die Operation ergänzt somit sinnvoll die bereits vorgesehenen Operationen numPoints und pointN, die im OGC Simple Feature (2011) Standard vorgesehen sind[4]. In Tabelle 5.1 sind entsprechend für die inneren Ringe von Polygonen, die Patches von PolyhedralSurface und die einzelnen Geometrien einer GeometryCollection weitere Operationen für die Abfrage als Auflistung definiert.

In Tabelle 5.1 sind weitere Operationen aufgelistet, mit denen zusätzliche geometrische und topologische Integritätsbedingungen aufgestellt werden können. Der Typ Geometry wird um zwei Operationen ergänzt, mit denen die Schnittmatrix zwischen zwei Geometrien ermittelt und getestet werden kann. Die Schnittmatrix wird dabei als String angegeben (siehe Unterabschnitt 2.1.3). Zudem können das minimal umschließende Rechteck (engl. 'minimum enclosing rectangle'), das minimal umschließende angepasste Rechteck (engl. 'minimum enclosing adapted rectangle') und das achsenparallele umschließende Rechteck (engl. 'bounding box') einer Geometrie mit den entsprechend benannten Operationen ermittelt werden (siehe auch Unterabschnitt 2.1.2). Der Rückgabetyp der drei Operationen ist ein Rechteck (engl. 'rectangle'), das als zusätzliche Unterklasse von Polygon dem Geometriemodell hinzugefügt wird (siehe Abbildung 5.3). Die Klasse Rectangle weist damit die gleichen Operationen wie die Klasse Polygon auf und wird zusätzlich um drei Operationen ergänzt. Die Operationen width und length geben die Länge und die Breite des Rechtecks zurück, wobei die Länge immer größer gleich der Breite ist. Die Operation elongation gibt schließlich das Verhältnis von Länge zu Breite des Rechtecks zurück. Die Elongation ist in Unterabschnitt 6.2.2 näher erläutert.

Die Bedingungen an Geometrien des Typs Rectangle sind im Programm 5.3 in der GeoOcl definiert. Insgesamt muss eine Geometrie fünf Integritätsbedingungen erfüllen, damit sie ein valides Rechteck bilden kann. Erstens darf ein Rechteck im Gegensatz zu einem Polygon keine inneren Ringe aufweisen. Zweitens muss ein Rechteck aus genau vier Liniensegmenten bestehen. Die dritte Bedingung fordert, dass zwei jeweils gegenüberliegende Segmente gleich lang sind. Viertens müssen alle Innenwinkel des äußeren Rings rechtwinklig sein, d.h. genau einen Wert von 90° aufweisen. Schließlich muss die Länge des Rechtecks größer gleich dessen Breite sein.

[4] Die Operation points ergänzt die Operationen numPoints und pointN nicht nur, sondern kann diese auch komplett ersetzen. Die Anzahl der Stützpunkte kann nämlich ebenso über die Größe der Auflistung abgefragt werden, und der wahlfreie Zugriff auf einen beliebigen Stützpunkt der Auflistung ist eine grundlegende Operation der Klasse Sequence der Ocl.

Typ	Zusätzliche Operationen auf dem Typ
Geometry	`intersectionMatrix(o: Geometry): String, intersectionMatrix(o: Geometry, m: String): Boolean, minimumEnclosingRectangle(): Rectangle, minimumEnclosingAdaptedRectangle(): Rectangle, boundingBox(): Rectangle`
LineString	`points(): Sequence(Point), segments(): Sequence(LineString), numSegments(): Integer, segmentN(n: Integer): LineString, leftAngles(): Sequence(Real), numLeftAngles(): Integer, leftAngleN(n: Integer): Real, rightAngles(): Sequence(Real), numRightAngles(): Integer, rightAngleN(n: Integer): Real, interiorAngles(): Sequence(Real), numInteriorAngles(): Integer, interiorAngleN(n: Integer): Real, exteriorAngles(): Sequence(Real), numExteriorAngles(): Integer, exteriorAngleN(n: Integer): Real, rectangularity(tolerance: Real): Real`
Surface	`compactnessSquare(): Real, compactnessCircle(): Real, fractalDimension(): Real`
Polygon	`interiorRings(): Sequence(LineString)`
PolyhedralSurface	`patches(): Sequence(Polygon)`
GeometryCollection	`geometries(): Sequence(Geometry)`
Rectangle	`width(): Real, length(): Real, elongation(): Real`

Tabelle 5.1: Erweiterung der Operationen der OGC Simple Feature (2011)

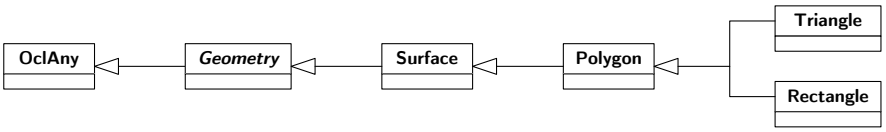


Abbildung 5.3: UML-Klassendiagramm für die Erweiterung um die Klasse Rectangle in der GeoOCL

Für den Typ LineString sind weitere Operationen für den Zugriff auf die einzelnen Segmente einer Linie definiert, die analog zu den Operationen für den Zugriff auf die einzelnen Stützpunkte einer Linie benannt sind. Zudem können die Winkel zwischen aufeinanderfolgenden Liniensegmenten abgefragt werden. Mit der Operation leftAngles werden die in Digitalisierungsrichtung links liegenden Innenwinkel zurückgegeben, wohingegen mit der Operation rightAngles die rechtsliegenden Winkel zurückgegeben werden. Ist die Linie geschlossen, beispielsweise als Umring eines Polygons, dann geben die Operationen interiorAngles und exteriorAngles entsprechend die innen und außen liegenden Winkel an. Ist die Linie nicht geschlossen, dann gibt die Operation interiorAngles die links und die Operation exteriorAngles die rechts liegenden Innenwinkel an. Mit der Operation rectangularity kann schließlich die Rechtwinkligkeit einer Linie berechnet werden. Die Berechnung ist in Unterabschnitt 7.4.2 detailliert ausgeführt.

Zusätzliche Operationen des Typs Surface sind ebenso in Tabelle 5.1 aufgelistet. Die Operationen ermöglichen die Berechnung der Kompaktheit (engl. 'compactness'), sowohl bezogen auf ein Quadrat als auch auf einen Kreis als Referenzfläche, sowie die Bestimmung der fraktalen Dimension (engl. 'fractal dimension'). Die drei Maße sind detailliert in Unterabschnitt 6.2.2 erläutert.

Um in Integritätsbedingungen der GeoOcl statistische Kenngrößen verwenden zu können, wird die Ocl um entsprechende Operationen und Datentypen erweitert (siehe Tabelle 5.2). Für Auflistungen von Werten, die im Datentyp Collection der Ocl gespeichert sind, können für die Werte mit der Operation statistics Lage-

```
context Rectangle
  inv KeineInnerenRinge: geom.numInteriorRings() = 0
  inv VierLiniensegmente: geom.exteriorRing().numSegments() = 4
  inv Raute: let segmente: Sequence(LineString) = geom.exteriorRing().segments() in segmente->at(0)
      .length() = segmente->at(2).length() and segmente->at(1).length() = segmente->at(3).length()
  inv Rechtwinklig: geom.exteriorRing().interiorAngles()->forAll(a | a = 90) -- Alle Innenwinkel
      sind 90°
  inv LaengeBreite: let segmente: Sequence(LineString) = geom.exteriorRing().segments() in if
      segmente->at(0).length() >= segmente->at(1).length() then length() = segmente->at(0).length
      () and width() = segmente->at(1).length() else width() = segmente->at(0).length() and length
      () = segmente->at(1).length() endif
```

Programm 5.3: Integritätsbedingungen des Typs Rectangle in der GeoOCL

und Streuungsparameter der deskriptiven Statistik ermittelt werden (siehe auch Unterabschnitt 2.5.1). Als Rückgabewert der Operation wird ein Objekt der Klasse Statistics erstellt, das mit seinen Operationen Zugriff auf das arithmetische Mittel, den Median, die α-Quantile (mit α im Intervall $[0,1]$), die Extremwerte und die Spannbreite sowie die Varianz und die Standardabweichung ermöglicht. Ebenso kann mit der Operation fiveNumberSummary das 5-Zahlenmaß der explorativen Datenanalyse für die Werte einer Auflistung bestimmt werden (siehe auch Unterabschnitt 2.5.3). Dieses stellt wiederum über seine Operationen den Median, die Extremwerte und die Spannbreite, den unteren und oberen hinge sowie den hinge-Abstand bereit.

Typ	Zusätzliche Operationen auf dem Typ
Collection	`statistics(): Statistics, fiveNumberSummary(): FiveNumberSummary`
Statistics	`mean(): Real, median(): Real, alphaQuantile(alpha: Real): Real, min(): Real, max(): Real, range(): Real, variance(): Real, standardDeviation(): Real,`
FiveNumberSummary	`median(): Real, min(): Real, max(): Real, range(): Real, lowerHinge(): Real, upperHinge(): Real, hSpread(): Real`

Tabelle 5.2: Erweiterung der Operationen der OGC Simple Feature (2011)

5.1.3 Analyse der Anforderungen

Aus der Analyse der Anforderungen in Kapitel 4 folgen mögliche zusätzliche Erweiterungen der GeoOcl. Im Folgenden werden deshalb die aufgestellten Anforderungen hinsichtlich ihrer Umsetzbarkeit in Integritätsbedingungen und damit Invarianten der GeoOcl analysiert. Dadurch werden sowohl die Möglichkeiten als auch die Grenzen der Formalisierung auf Basis der GeoOcl aufgezeigt.

Modellierung von Integritätsbedingungen

Die acht in Abschnitt 4.1 aufgestellten Anforderungen der Modellierung an Integritätsbedingungen können wie folgt in Invarianten der GeoOcl beziehungsweise durch eine zusätzliche Erweiterung der GeoOcl ausgedrückt werden.

Anforderung 1: Diskrete und kontinuierliche Objekte Integritätsbedingungen für diskrete Objekte wie Punkte, Linien und Flächen sind bereits durch die GeoOcl abgedeckt (siehe Unterabschnitt 5.1.2). Einige der in Anforderung 1 vorgestellten kontinuierlichen Objekte werden zudem bereits durch die Objektmodelle des Ogc Simple Feature Standards beziehungsweise durch das räumliche Schema der ISO 19107 (2003) abgedeckt. Dazu zählen beispielsweise die planare Unterteilung in Polygone und die Dreiecksvermaschung, die den Klassen PolyhedralSurface und TIN der Ogc Simple Features entsprechen (siehe Unterabschnitt 2.4.1). Weitere in Anforderung 1 aufgezählte kontinuierliche Objekte lassen sich entweder direkt durch die GeoOcl oder durch eine Erweiterung der Sprache formulieren. Als Beispiel werden im Folgenden die Integritätsbedingungen für Isolinien vorgestellt. Weitere Operationen für kontinuierliche Objekte, wie beispielsweise zur Berechnung von Gradienten oder für räumliche Interpolationen, könnten der GeoOcl ebenso hinzugefügt werden.

Für Isolinien sind die entsprechenden Integritätsbedingungen im Programm 5.4 aufgelistet. Die Invariante Linie fordert, dass die Geometrie vom Typ LineString oder einer Unterklasse von LineString ist. Mit den Invarianten Einfach und Geschlossen wird gefordert, dass sich die Linie nicht selbst überschneidet und geschlossen ist, d.h. dass Start- und Endpunkt der Linie identisch sind. Die letzte Invariante fordert schließlich, dass die Höhen z aller Stützpunkte der Linie gleich der Höhe z des ersten Stützpunktes der Linie sind.

```
context Isolinie
  inv Linie: geom.oclIsKindOf(LineString)
  inv Einfach: geom.isSimple()
  inv Geschlossen: geom.isClosed()
  inv KonstanteHoehe: geom.points()->forAll(p | p.z = geom.pointN(0).z)
```

Programm 5.4: Integritätsbedingungen für Isolinien in der GeoOCL

Anforderung 2: Statisch und dynamisch Statische Bedingungen lassen sich bereits durch die GeoOcl abbilden. Dynamische Bedingungen lassen sich einfach durch Vor- und Nachbedingungen der Ocl umsetzen, wie sie beispielsweise im Programm 2.5 definiert sind. Dynamisch sequentielle Bedingungen sind selten, sie könnten

der OCL jedoch durch eine Erweiterung von Vor- und Nachbedingungen hinzugefügt werden. In Tabelle 5.3 ist eine mögliche Erweiterung des Schlüsselworts @pre skizziert.

Schlüsselwort	Bedeutung
@pre	Zustand des Attributs direkt vor der Ausführung der Operation
@pre[-2]	Zustand des Attributs vor der vorangegangen Ausführung der Operation
@pre[2013-01-31]	Zustand des Attributs am 31.1.2013

Tabelle 5.3: Mögliche Erweiterung der OCL für dynamisch sequentielle Bedingungen

Anforderung 3: 2,5D und 3D Objekte mit 2,5 und 3 Dimensionen werden bereits durch den OGC Simple Feature Standard (OGC Simple Feature, 2011) sowie durch das räumliche Schema der ISO 19107 (2003) abgedeckt. Dazu zählen auch entsprechende Operationen auf den jeweiligen Objekten. Weitere notwendige Operationen könnten der GeoOCL einfach hinzugefügt werden, beispielsweise basierend auf Sichtbarkeitsanalysen. In Bezug auf Objekte in mehreren Detailgraden, wie beispielsweise den Level of Detail des OGC CityGML (2012) Standards, könnten für jeden Detailgrad unterschiedliche Integritätsbedingungen aufgestellt werden. Eine mögliche Umsetzung ist in der Integritätsbedingung im Programm 5.5 angedeutet, in der Freileitungen eine Mindestdistanz von 5 m zu Gebäuden aufweisen müssen, jedoch nur falls der Level of Detail des Gebäudes größer gleich 2 ist.

```
context Gebaeude
  inv FreileitungAbLoD2: geom.loD() >= 2 implies Freileitung.allInstances()->forAll(f | f.geom.
      distance(geom) >= 5) -- [Meter]
```

Programm 5.5: Integritätsbedingungen basierend auf dem Level of Detail in der GeoOCL

Anforderung 4: Zeit Prinzipiell entspricht die Berücksichtigung der Zeit einer Erweiterung der OCL um die Datentypen Zeitpunkt, Intervall und Zyklus, sowie um die entsprechenden Operationen auf diesen Datentypen. Die Operationen für Intervalle sind dabei bereits durch die von Allen (1983) identifizierten Relationen gegeben. Der Zugriff auf Attributwerte von Objekten basierend auf deren Zeitstempel könnte entsprechend der in Tabelle 5.3 beschriebenen Erweiterung erfolgen.

Anforderung 5: Topologische Netzwerke Auf Basis des topologischen Modells der ISO 19107 (2003) könnten der GeoOCL weitere Operationen hinzugefügt werden, um beispielsweise Integritätsbedingungen an den Grad eines Knotens aufstellen zu können. Zudem ist eine Verbindung von Geometrie und Topologie in einem einzelnen Objekt einfach dadurch zu erreichen, dass für die entsprechende Klasse neben einem Attribut für die Geometrie, beispielsweise vom Typ GM_Point, ein weiteres Attribut für die Topologie, beispielsweise vom Typ GM_Node, definiert wird.

Anforderung 6: Zusammengesetzte Geometrien und Aggregation Eine Klasse weist eine zusammengesetzte Geometrie auf, wenn deren geometrisches Attribut einer Auflistung von Geometrien entspricht. Im OGC Simple Feature (2011) Standard ist das geometrische Attribut der Klasse dann vom Typ GeometryCollection. Durch deren Unterklassen lassen sich die Elemente der Auflistung einschränken, so sind beispielsweise für den Typ MultiPolygon nur Elemente der Klasse Polygon erlaubt. Es lassen sich jedoch auch komplexere Einschränkungen für Geometrien vom Typ GeometryCollection in der GeoOCL formulieren, wie im Programm 5.6 aufgezeigt wird. Dort wird gefordert, dass Elemente der Auflistung entweder vom Typ Point oder LineString sind und mindestens drei Punkte in der Auflistung enthalten sind.

```
context Klasse
  inv Auflistung: geom.oclIsTypeOf(GeometryCollection)
  inv PunktLinie: geom.geometries()->forAll(g | g.oclIsTypeOf(Point) or g.oclIsTypeOf(LineString))
  inv Min3Punkte: geom.geometries()->select(g | g.oclIsTypeOf(Point))->size() >= 3
```

Programm 5.6: Einschränkung der möglichen Elemente einer GeometryCollection in der GeoOCL

Die topologische Beziehung zwischen Objekten mit zusammengesetzten Geometrien kann entweder direkt mit den vorhandenen Operationen abgebildet werden, oder wie von Kang u. a. (2004) und Duboisset u. a. (2005a) vorgestellt mit Multiplizitäten beziehungsweise Adverbien. Im Programm 5.7 sind für jeweils zwei Klassen mit flächenhafter Geometrie entsprechende topologische Integritätsbedingungen angegeben. Die Objekte der

Klassen A und B dürfen sich niemals überlappen. Dagegen müssen von der Klasse D mindestens zwei Objekte entsprechende Objekte der Klasse C überlappen. Die Angabe der Multiplizität wird dabei durch die select Operation der OCL ermöglicht, mit der diejenigen Objekte aus der Auflistung entfernt werden, die nicht die geforderte Bedingung erfüllen. Die GeoOCL könnte zudem dahingehend erweitert werden, dass Adverbien für topologische Beziehungen angegeben werden können, wie beispielsweise in der Integritätsbedingung für die Objekte der Klassen E und F. Die Adverbien sind dabei als Enumeration mit den Namen TopologyAdverbs eindeutig definiert.

```
context KlasseA
  inv ImmerOverlap: KlasseB.allInstances()->forAll(b | b.geom.overlaps(geom))

context KlasseC
  inv MutliplizitaetOverlap: KlasseD.allInstances()->select(d | d.geom.overlaps(geom))->size() >= 2

context KlasseE
  inv AdverbOverlap: KlasseF.allInstances()->forAll(f | f.geom.overlaps(geom, TopologyAdverbs.
      MOSTLY))
```

Programm 5.7: Einschränkung der möglichen Elemente einer GeometryCollection in der GeoOCL

Aggregationen und Kompositionen lassen ebenso einfach in der GeoOCL abbilden. Dabei werden mehrere Modellierungsvarianten unterstützt, die sich darin unterscheiden wie die Geometrie des Ganzen in Bezug auf die Teile modelliert wird. In Abbildung 5.4 ist das UML-Klassendiagramm mehrerer Varianten zur Aggregation von Stadtteilen zu einer Stadt dargestellt. In der ersten Variante ist in der Klasse StadtV1 keinerlei Geometrie modelliert, sondern lediglich bei den Stadtteilen. In der zweiten Variante ist in der der Klasse StadtV2 die Geometrie als MultiPolygon modelliert, die der Auflistung der Geometrien der einzelnen Stadtteile entspricht. In der dritten Variante ist in der Klasse StadtV3 die Geometrie als einzelnes Polygon modelliert, die der Vereinigung der Geometrien der einzelnen Stadtteile entspricht.

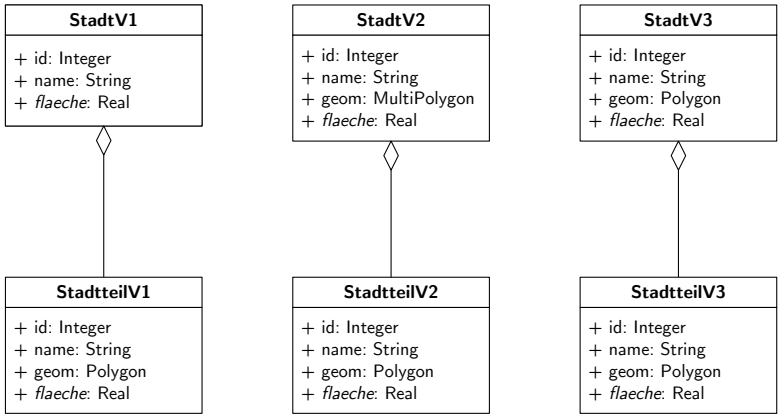


Abbildung 5.4: UML-Klassendiagramm für die Aggregation in der GeoOCL

Die entsprechenden Integritätsbedingungen für die drei Varianten sind im Programm 5.8 definiert. Im Kontext der Stadtteile sind die beiden Invarianten Flaeche und Topologie jeweils identisch. Die erste Invariante fordert, dass der Wert des abgeleiteten Attributs flaeche gleich der Fläche der Geometrie des Stadtteils ist. Die zweite Invariante fordert, dass sich die Geometrien zweier unterschiedlicher Stadtteile nicht überlappen.

In der ersten Modellierungsvariante ist lediglich eine einzelne Integritätsbedingung für den Kontext der Stadt notwendig. Diese verwendet die collect Operation der OCL um eine Auflistung der Werte des Attributs flaeche aller Stadtteile zu erstellen, die dann lediglich zur Gesamtfläche aufsummiert werden müssen. In der zweiten Variante der Modellierung sind sowohl Integritätsbedingungen für den Kontext der Stadt als auch der Stadtteile notwendig. Die erste Bedingung stellt sicher, dass die Anzahl der Geometrien des MultiPolygons mit der Anzahl der Stadtteile übereinstimmt. Die zweite Bedingung stellt sicher, dass jedes Polygon eines Stadtteils im MultiPolygon der Stadt genau einmal vorhanden ist. In der dritten Variante sind wieder zwei Integritätsbedingungen notwendig, die nur zusammen sicherstellen können, dass die Aggregation geometrisch und topologisch korrekt ist. So wird gefordert, dass die Geometrie jedes Stadtteils von der Geometrie der Stadt überdeckt wird und gleichzeitig die Fläche der Geometrie der Stadt gleich der Summe der Flächen der Stadtteile ist. Somit können alle Modellierungsvarianten für die Aggregation mit der GeoOCL abgedeckt werden, wobei die erste Variante aufgrund ihrer Einfachheit bevorzugt werden sollte.

```
context StadtV1
  inv SumFlaeche: flaeche = stadtteile->collect(flaeche)->sum()

context StadtteilV1
  inv Flaeche: flaeche = geom.area()
  inv Topologie: StadtteilV1.allInstances()->forAll(s1, s2 | s1.id <> s2.id implies not s1.geom.
      overlaps(s2.geom))

context StadtV2
  inv SumFlaeche: flaeche = geom.area()
  inv GeometrienAnzahl: geom->size() = stadtteile->size()

context StadtteilV2
  inv Flaeche: flaeche = geom.area()
  inv Topologie: StadtteilV2.allInstances()->forAll(s1, s2 | s1.id <> s2.id implies not s1.geom.
      overlaps(s2.geom))
  inv Geometrie: stadt.geom->geometries()->one(stadtGeom | geom.equals(stadtGeom))

context StadtV3
  inv SumFlaeche: flaeche = geom.area()
  inv SumFlaeche: flaeche = stadtteile->collect(flaeche)->sum()

context StadtteilV3
  inv Topologie: StadtteilV3.allInstances()->forAll(s1, s2 | s1.id <> s2.id implies not s1.geom.
      overlaps(s2.geom))
  inv Geometrie: stadt.geom.covers(geom)
```

Programm 5.8: Aggregation in der GeoOCL

Anforderung 7: Konzeptuelle Generalisierung In Borges u. a. (2002) wird die konzeptuelle Generalisierung so modelliert, dass ein Objekt durch eine Oberklasse ohne expliziten Raumbezug und mehrere Unterklassen abgebildet wird. Als Beispiel wird in in Abbildung 4.2 die Klasse Fluss durch eine Linie (Achse, Ränder), als Fläche und als Netzwerk (Segment) repräsentiert. Sinnvoller ist es jedoch aus Sicht der Modellierung, das Objekt als Klasse mit mehreren Raumbezügen abzubilden. So ist der Bezug zwischen den einzelnen Repräsentation direkt, und muss nicht wie im Ansatz von Borges u. a. (2002) über Beziehungen zwischen einzelnen Objektinstanzen modelliert werden. Eine entsprechende Modellierung der Klasse Fluss mit mehreren Raumbezügen ist in Abbildung 5.5 angegeben. Die Klasse Fluss weist neben nicht-räumlichen Attributen mehrere Geometrien und eine Topologie auf, die als Attribute der Klasse modelliert sind. Mit diesem Modellierungsansatz lassen sich auf einfache Weise Integritätsbedingungen zwischen den einzelnen Repräsentationen aufstellen, wie im Programm 5.9 aufgezeigt ist. Die erste Invariante AchseSegment fordert, dass die Länge der linearen Achse als Gewicht der Kante des Segments gesetzt ist. Die zweite Invariante RandFlaeche fordert, dass der lineare Rand mit dem äußeren Ring der Fläche übereinstimmt. Die dritte Invariante AchseFlaeche fordert schließlich, dass die lineare Achse von der Fläche überdeckt wird. Die in Borges u. a. (2002) zusätzlich angeführten Integritätsbedingungen über mehrere Maßstabsbereiche lassen sich analog aufstellen.

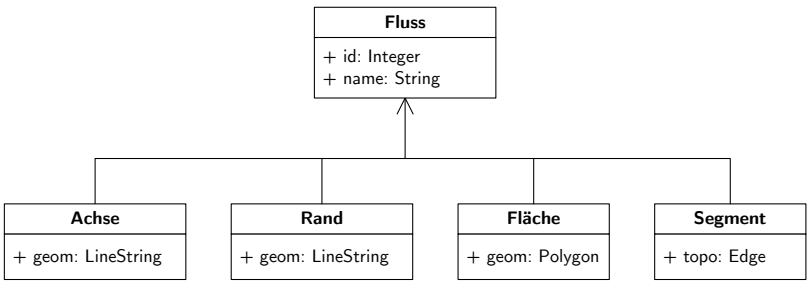


Abbildung 5.5: UML-Klassendiagramm für die konzeptuelle Generalisierung in der GeoOCL

```
context Fluss
  inv AchseSegment: segment.topo.weight() = achse.geom.length()
  inv RandFlaeche: rand.geom = flaeche.geom.exteriorRing()
  inv AchseFlaeche: flaeche.geom.covers(achse.geom)
```

Programm 5.9: Aggregation in der GeoOCL

Anforderung 8: Unscharfe Geometrie Unscharfe Geometrien lassen sich durch eine Erweiterung des Geometriemodells ausdrücken. Eine einfache Modellierung könnte beispielsweise entsprechend der Egg-Yolk Therie anstatt einer Geometrie zwei Geometrien vorsehen. Dazu müsste der GeoOCL einen solcher Geometrietyp sowie Operationen auf diesem Typ hinzugefügt werden. Die in Bejaoui (2009) und Bejaoui u. a. (2010) zusätzlich angegebene Verwendung von Adverbien kann entsprechend des Vorschlags in Anforderung 6 mit der GeoOCL berücksichtigt werden.

Definition von Integritätsbedingungen

Die fünf in Abschnitt 4.2 aufgestellten Anforderungen der Definition an Integritätsbedingungen können wie folgt in Invarianten der GeoOCL beziehungsweise durch eine zusätzliche Erweiterung der GeoOCL ausgedrückt werden.

Anforderung 9: Anzahl der Attributwerte/Attribute/Objekte/Klassen Alle in der Zusammenfassung der Anforderung aufgestellten Kriterien lassen sich durch die GeoOCL abbilden. Mit der GeoOCL lassen sich Anforderungen sowohl an einzelne als auch an mehrere Attributwerte und Attribute stellen. Dies trifft auch für zusammengesetzte Geometrien zu, für die sich beispielsweise einfach die Anzahl der enthaltenen Elemente beschränken lässt. Durch Filter, wie beispielsweise durch das Schlüsselwort implies, lassen sich Integritätsbedingungen auf eine Untermenge aller Objekte einer Klasse begrenzen. Wie im Programm 5.10 exemplarisch für eine Klasse Stadt mit den Attributen stadtteile und rathaeuser, die jeweils einer Auflistung entsprechen, angegeben, kann die Dichte von Objekten sowohl absolut als auch relativ mit Invarianten in der OCL beschränkt werden. Schließlich können durch die in der GeoOCL enthaltenen Erweiterung der OCL auf mehrere Klassen Integritätsbedingungen zwischen allen Klassen eines Modells aufgestellt werden. Zusammenfassend können demnach wie gefordert in der GeoOCL in einer einzelnen Integritätsbedingung eine beliebige Kombination der Anzahlen an Attributwerten, Attribute, Objekte und Klassen enthalten sein.

```
context Stadt
  inv Min3Stadtteile: stadtteile->size() >= 3
  inv Rathaeuser: rathaeuser->size() <= stadtteile->size()
```

Programm 5.10: Beschränkung der relativen und absoluten Dichte in der GeoOCL

Anforderung 10: Flexible Topologie Mit der Erweiterung der Klasse Geometry um die Operation intersectionMatrix in Unterabschnitt 5.1.2 verfügt die GeoOCL bereits um eine Möglichkeit topologische Schnittmatrizen in Integritätsbedingungen zu definieren. Weitere benutzerdefinierte topologischen Relationen lassen sich der GeoOCL einfach durch neue Operationen auf den entsprechenden Geometrietypen hinzufügen.

Anforderung 11: Methoden für die Ableitung von Geometrien Da für die GeoOCL der OGC Simple Feature Standard verwendet wird sind bereits die im Standard enthaltenen Operationen für die Ableitung, Erzeugung und Zerlegung von Objektgeometrien enthalten. Weitere Operationen können der GeoOCL einfach hinzugefügt werden.

Anforderung 12: Funktionale Abhängigkeit In der OCL lassen sich funktionale Abhängigkeiten direkt modellieren. Durch die Erweiterung auf räumliche Attribute in der GeoOCL lassen sich auch funktionale Abhängigkeiten fordern, die geometrische Eigenschaften eines Objekts berücksichtigen. Beispielsweise wird im Programm 5.11 gefordert, dass der Wert des Attributs flaeche gleich dem entsprechenden Rückgabewert der Operation area auf dem geometrischen Attribut der Klasse ist. Die funktionalen Abhängigkeiten einer Aggregation und deren Berücksichtigung in der GeoOCL sind bereits bei der Diskussion der Anforderung 6 berücksichtigt.

```
context Gebaeude
  inv Flaeche: flaeche = geom.area()
```

Programm 5.11: Funktionale Abhängigkeit in der GeoOCL

Die in Reeves u. a. (2006) definierten parametrischen geometrischen Integritätsbedingungen lassen sich auch durch die GeoOCL abbilden. Als Beispiel ist im Programm 5.12 eine Klasse Parkplatz als Rechteck definiert, dessen Geometrie durch Parameter charakterisiert ist und zudem abhängig von der Nachbargeometrie des zugehörigen Objekts der Klasse Supermarkts ist.

Anforderung 13: Generische Bedingungen Generische Bedingungen werden durch die in der GeoOCL übernommene Erweiterung der OCL aus Casanova u. a. (2000) ermöglicht. Im Programm 5.13 ist exemplarisch eine generische Integritätsbedingung aufgestellt, die fordert dass die Geometrie einer Klasse vom Typ LineString

```
context Parkplatz
  inv Geometrie: geom.oclIsTypeOf(Polygon)
  inv Rechteck: let segmente: Sequence(LineString) = geom.exteriorRing().segments() in segmente->at
      (0).length() = segmente->at(2).length() and segmente->at(1).length() = segmente->at(3).
      length() and segmente->at(0).length() = breite and segmente->at(1).length() = laenge --
      Parameter breite und laenge in Meter
  inv Rechtwinklig: geom.exteriorRing().interiorAngles()->forAll(a | a = 90) -- Alle Innenwinkel
      sind 90°
  inv Groeße: geom.area() >= 2 * supermarkt.geom.area() -- Fläche des Parkplatzes ist mindestens
      doppelt so groß wie die Fläche des Supermarkts
```

Programm 5.12: Funktionale Abhängigkeit mit parametrischen Integritätsbedingungen in der GeoOCL

ist und eine Mindestlänge von zwei Metern aufweist. Diese Bedingung kann dann für unterschiedliche Klassen eines Modells verwendet werden, beispielsweise sowohl für Straßen als auch für Flüsse. Zudem kann die generische Bedingung auch in ein anderes Modell übernommen werden, das zwar unterschiedliche Klassennamen aufweist aber ähnliche Anforderungen an die Objekte stellt.

```
minLaengeLinie<@Klasse> where @Klasse.geom.supertypes->includes(Geometry):
  @Klasse.geom.oclIsTypeOf(LineString) and @Klasse.geom.length() > 2 -- [Meter]

context Strasse
  inv: %minLaengeLinie<Strasse>

context Fluss
  inv: %minLaengeLinie<Fluss>
```

Programm 5.13: Generische Integritätsbedingungen in der GeoOCL

Aufstellung einzelner Integritätsbedingungen

Die fünf in Abschnitt 4.3 aufgestellten Anforderungen der Aufstellung einzelner Integritätsbedingungen können wie folgt in Invarianten der GeoOcl beziehungsweise durch eine zusätzliche Erweiterung der GeoOcl ausgedrückt werden.

Anforderung 14: Fordern vs. Einschränken Diese Anforderung ist unabhängig vom für die Aufstellung von Integritätsbedingungen verwendeten Formalismus.

Anforderung 15: Wertebereich Die GeoOcl sieht bereits vielfältige Möglichkeiten zur Einschränkung von Attributwerten vor. Beispiele für Einschränkungen des Wertebereichs von Attributen sind im Programm 5.14 aufgelistet. Komplexere Einschränkungen, wie reguläre Ausdrücke oder Grammatiken, können der GeoOcl als zusätzliche Operationen auf den definierten Typen hinzugefügt werden.

```
context Gebaeude
  inv: anzahlStockwerke > 0 -- Attribut anzahlStockwerke: Integer
  inv: adressen->forAll(a | a.size() >= 8) -- Attribut adressen: Sequence(String)
  inv: geom.exteriorRing().points()->forAll(p | p.X() >= -180 and p.X <= 180 and p.Y() >= -90 and p
      .Y() <= 90) -- Attribut geom: Polygon
```

Programm 5.14: Einschränkung des möglichen Wertebereichs von Attributen in der GeoOCL

Anforderung 16: Genauigkeit Diese Anforderung ist ebenso unabhängig vom für die Aufstellung von Integritätsbedingungen verwendeten Formalismus.

Anforderung 17: Fehlende Werte und Null Die Definition der Ocl enthält den Wert 'null' für alle Typen und berücksichtigt diesen bei Operationen entsprechend. Wird beispielsweise für ein Attribut vom Typ Collection überprüft ob die Auflistung keine Elemente enthält, so wird der Wert 'wahr' zurückgegeben selbst wenn der Attributwert 'null' ist.

Anforderung 18: Schweregrad Der Schweregrad einer Integritätsbedingung könnte der GeoOcl durch ein weiteres Schlüsselwort severity hinzugefügt werden. Im Programm 5.15 ist ein mögliches Beispiel angegeben, bei dem die Verletzung der topologischen Bedingung eine größere Schwere zugemessen wird als die Verletzung der geometrischen Bedingung.

```
context Tankstelle
  inv Topologie severity 10: Tankstelle.allInstances()->forAll(t1, t2 | t1.id <> t2.id implies t1.
      geom.disjoint(t2.geom))
  inv Geometrie severity 2: Tankstelle.allInstances()->forAll(t1, t2 | t1.id <> t2.id implies t1.
      geom.distance(t2.geom) > 200) -- [Meter]
```

Programm 5.15: Schweregrad von Integritätsbedingungen in der GeoOCL

Anforderung 19: Konsistenz Diese Anforderung ist ebenso unabhängig vom für die Aufstellung von Integritätsbedingungen verwendeten Formalismus.

Prüfung von Daten mit Integritätsbedingungen

Die vier in Abschnitt 4.4 aufgestellten Anforderungen der Prüfung von Daten mit Integritätsbedingungen können wie folgt in Invarianten der GeoOcl beziehungsweise durch eine zusätzliche Erweiterung der GeoOcl ausgedrückt werden.

Anforderung 20: Toleranz Diese Anforderung ist unabhängig vom für die Aufstellung von Integritätsbedingungen verwendeten Formalismus.

Anforderung 21: Ausmaß Diese Anforderung ist prinzipiell ebenso unabhängig vom für die Aufstellung von Integritätsbedingungen verwendeten Formalismus. Die konzeptuelle Nachbarschaft topologischer Relationen lässt sich jedoch in der GeoOcl abbilden, wie das Beispiel im Programm 5.16 aufzeigt. Im Beispiel dürfen sich die Geometrien zweier Einzugsbereiche überlappen oder mindestens berühren. Für die Berücksichtigung der Form und der Lage von beispielsweise Schnittflächen könnten in Abhängigkeit von den jeweiligen Anforderungen gegebenenfalls weitere Operationen zur GeoOcl hinzugefügt werden.

```
context Einzugsbereich
  inv: Einzugsbereich.allInstances()->forAll(e1, e2 | e1.id <> e2.id implies e1.geom.overlaps(e2.
      geom) or e1.geom.meets(e2.geom))
```

Programm 5.16: Ausmaß von Integritätsbedingungen in der GeoOCL

Anforderung 22: Globales Ausmaß Diese Anforderung ist prinzipiell unabhängig vom für die Aufstellung von Integritätsbedingungen verwendeten Formalismus. Mit der GeoOcl lassen sich jedoch Bedingungen an die absolute oder relative Anzahl von Objekten einer Klasse stellen, die eine bestimmte Bedingung erfüllen. In Erweiterung des Beispiels im Programm 5.16 wird im Programm 5.17 gefordert, dass sich mindestens 90 % der Einzugsbereiche überlappen und die restlichen Einzugsbereiche sich berühren.

```
context Einzugsbereich
  inv: Einzugsbereich.allInstances()->select(e1, e2 | e1.id <> e2.id implies e1.geom.overlaps(e2.
      geom))->size() >= Einzugsbereich.allInstances()->size() * 0.9
  inv: Einzugsbereich.allInstances()->reject(e1, e2 | e1.id <> e2.id implies e1.geom.overlaps(e2.
      geom))->forAll(e1, e2 | e1.id <> e2.id implies e1.geom.meets(e2.geom))
```

Programm 5.17: Ausmaß von Integritätsbedingungen in der GeoOCL

Anforderung 23: Umfangreiche Datensätze Diese Anforderung ist unabhängig vom für die Aufstellung von Integritätsbedingungen verwendeten Formalismus.

Änderung von Daten und Integritätsbedingungen

Die vier in Abschnitt 4.5 aufgestellten Anforderungen der Änderung von Daten und Integritätsbedingungen können wie folgt in Invarianten der GeoOcl beziehungsweise durch eine zusätzliche Erweiterung der GeoOcl ausgedrückt werden.

Anforderung 24: Korrektur von Fehlern Diese Anforderung ist unabhängig vom für die Aufstellung von Integritätsbedingungen verwendeten Formalismus.

Anforderung 25: Automatische Aktualisierung Diese Anforderung ist ebenso unabhängig vom für die Aufstellung von Integritätsbedingungen verwendeten Formalismus.

Anforderung 26: Ausnahmen Mit der GeoOcl können Ausnahmen detailliert definiert werden. Durch das Schlüsselwort implies sowie durch if-then-else-endif Konstrukte können komplexe Filterkriterien erstellt und überprüft werden. So sind beispielsweise im Programm 5.18 die in Tabelle 4.13 geforderten Ausnahmen formal spezifiziert. Die erste Invariante zeigt auf, wie Objekte basierend auf ihren Attributwerten von Integritätsbedingungen ausgeschlossen werden können. Die zweite Invariante schließt Objekte anhand ihrer Klasse beziehungsweise Unterklasse von der geforderten Bedingung aus.

```
context Strasse
  inv Ids: Gebaude.allInstances()->forAll(g | (id <> 7 and g.id <> 112) and (id <> 8 and g.id <>
      114) implies not g.geom.cross(geom))
  inv Unterklasse: Gebaeude.allInstances()->forAll(g | not oclIsKindOf(Tunnel) implies not g.geom.
      cross(geom))
```

Programm 5.18: Ausnahmen in Integritätsbedingungen in der GeoOCL

Anforderung 27: Sicherheit Diese Anforderung ist nur teilweise abhängig vom für die Aufstellung von Integritätsbedingungen verwendeten Formalismus. Analog zur Umsetzung des Schweregrads könnte der GeoOcl ein weiteres Schlüsselwort rights hinzugefügt werden, dass definiert welche Benutzer oder Benutzergruppen welche Rechte haben. Im Programm 5.19 ist ein mögliches Beispiel angegeben, in dem Administratoren die Integritätsbedingung ändern und löschen dürfen, Experten die Werte der Bedingung ändern dürfen und Benutzer keine Änderungen vornehmen dürfen. Zur Angabe der Zugriffsbeschränkung wird die Terminologie von Datenbanken verwendet, die Rechte zum erstellen (engl. 'create'), lesen (engl. 'read'), ändern (engl. 'update') und löschen (engl. 'delete') vorsieht. Die Rechte eines Benutzers ergeben sich dann aus der Kombination der einzelnen Operationen, wobei die Initialen der englischen Bezeichnungen verwendet werden, d.h. die maximal möglichen Rechte sind 'CRUD'. Die Rechte können dabei global für einen Kontext oder lokal für die einzelnen Integritätsbedingung definiert werden. Mit lokalen Rechten können zudem globale Rechte überschrieben werden.

```
context Strasse rights Administratoren('CRUD')
  inv MinLaenge rights Administratoren('RU'), Experten('RU'), Benutzer('R') : geom.length() >= 2 --
      [Meter]
  inv MinSegmente rights Experten('RUD'), Benutzer('RU') : geom.numSegments() >= 2
```

Programm 5.19: Sicherheit von Integritätsbedingungen in der GeoOCL

5.2 Implementierung

Für die Implementierung der in dieser Arbeit vorgestellten Integritätsbedingungen sind mehrere Aspekte von Interesse. Diese werden kurz theoretisch diskutiert.

Die Implementierung wird aus zwei Gründen in dieser Arbeit nicht tiefgehend behandelt. Erstens setzt diese Arbeit andere Schwerpunkte, wie die auf statistischen Auswertungen basierende Klassenbeschreibung und die Verarbeitung umfangreicher Datenmengen (siehe Kapitel 6 und 7). Zweitens hängt die Auswahl der Werkzeuge für die Implementierung stark von den Rahmenbedingungen ab, in denen im jeweiligen Anwendungsgebiet Modelle und Integritätsbedingungen aufgestellt und überprüft werden. So müssen beispielsweise für unterschiedliche Anwendungen unterschiedliche Softwareprodukte für die Modellierung verwendet werden, die Implementierungen müssen in unterschiedlichen Programmiersprachen und die Datenhaltungen in unterschiedlichen Systemen erfolgen.

In dieser Arbeit sind alle in der GeoOcl definierten Integritätsbedingungen sorgfältig manuell aufgestellt. Die Bedingungen werden somit nicht mit Hilfe eines Werkzeugs erstellt und überprüft, da die Erweiterung eines vorhandenen Werkzeugs aufwändig ist und die anderen Aspekte dieser Arbeit von allgemeinerem Interesse sind.

Da die GeoOcl die Ocl um neue Sprachelemente und Konstrukte erweitert, ist es am sinnvollsten ein Werkzeug zu erweitern, mit dem bereits Bedingungen in der Ocl aufgestellt und überprüft werden können. In Mondzech (2009) werden 15 Werkzeuge zur Modellierung von Integritätsbedingungen in der Ocl vorgestellt, von denen das Dresden Ocl Toolkit des Lehrstuhls für Softwaretechnologie der Technischen Universität Dresden und das Use Toolkit (Uml-based Specification Environment) der Datenbanksystem-Gruppe der Universität Bremen als am geeignetsten für die Erweiterung um neue Sprachelemente und Konstrukte gesehen werden. Die beiden als Open Source verfügbaren Toolkits ermöglichen nicht nur die Modellierung unter Berücksichtigung von Invarianten in der Ocl, sondern auch die Überprüfung von Daten.

Das Dresden OCL Toolkit (DresdenOCL, 2013) besteht aus mehreren Plugins, die für die Entwicklungsumgebung Eclipse bereit gestellt werden. Ein OCL2 Parser, Editor und Interpreter bilden die Kernbestandteile des Toolkits. Diese ermöglichen die Aufstellung von Bedingungen für Modelle in der UML, in Java und in XML Schemata. Aus den Bedingungen können schließlich automatisch entsprechende Klassen in der Programmiersprache Java sowie SQL-Abfragen zur Überprüfung von Daten generiert werden.

Das USE Toolkit (Gogolla u. a., 2007) ist dagegen ein geschlossener Prototyp, der in einer grafischen Oberfläche die Spezifikation von Integritätsbedingungen in der OCL für ein Modell und die Überprüfung von in diesem Modell vorliegenden Daten ermöglicht.

6 Integritätsbedingungen: Flächennutzung in Geobasisdaten

Geobasisdaten sind amtliche Geodaten, welche die Landschaft (Topographie) und die Liegenschaften (Flurstücke und Gebäude) beschreiben. Geobasisdaten haben eine herausragende Bedeutung, da sie nicht nur die Grundlage für das Liegenschaftskataster und die topographische Landesaufnahme bilden, sondern auch als wichtige Basisinformation für die Darstellung und Analyse vielfältiger thematischer Daten benötigt werden. Sie werden in Politik und Verwaltung ebenso eingesetzt wie in Wirtschaft und Forschung. Die vektoriellen Geobasisdaten bestehen aus den Festpunkten, der Automatisierten Liegenschaftskarte (ALK)[1] und dem digitalen Landschaftsmodell des Amtlichen Topographisch-Kartographischen Informationssystems (ATKIS).

In diesem Kapitel werden anhand von drei unterschiedlichen Gebieten Integritätsbedingungen für die Flächennutzung aufgestellt, die sowohl im Datenmodell der ALK als auch im Datenmodell von ATKIS enthalten ist. Die verwendeten Daten sind dabei repräsentativ für andere Datensätze, die vollständig oder nahezu vollständig flächenüberdeckend sind. Einige Beispiele für solche Daten sind Landnutzungsklassifikationen aus Satellitenbeobachtungen, regionale Entwicklungspläne, Verbreitungsgebiete von Zeitschriften und Prospekten, Einzugsgebiete von Geschäften oder Wahlgebiete. Das in diesem Kapitel vorgestellte Vorgehen ist auf diese gleichartigen Datensätze zu einem hohen Prozentsatz übertragbar.

Im Folgenden werden in Abschnitt 6.1 zuerst die drei Gebiete vorgestellt. Anschließend wird in Abschnitt 6.2 die Anreicherung der Objekte um geometrische und topologische Maße erläutert. In Abschnitt 6.3 werden die angereicherten Daten dann durch statistische Methoden untersucht mit dem Ziel die charakteristischen Eigenschaften der einzelnen Objektarten eindeutig durch Integritätsbedingungen beschreiben zu können. Diese Bedingungen entsprechen der Klassenbeschreibung und bilden so die erste Säule des Data Mining für die Geobasisdaten. Die Klassifikation in Abschnitt 6.4 bildet entsprechend die zweite Säule des Data Mining der Geobasisdaten. Durch die Klassifikation wird die Frage beantwortet, ob aus den Daten der ALK automatisch Flächennutzungen in ATKIS abgeleitet werden können. Die bei der Aufstellung der Integritätsbedingungen berücksichtigten Anforderungen aus Kapitel 4 werden in Abschnitt 6.5 aufgelistet. In Abschnitt 6.6 werden einige Aspekte der Implementierung kurz vorgestellt. Schließlich wird in Abschnitt 6.7 nochmals diskutiert, was durch die Analyse der Daten erreicht wurde und wie sich das vorgestellte Vorgehen auf gleichartige Datensätze übertragen lässt.

6.1 Verwendete Daten

Das erste Gebiet für die Untersuchung der Integritätsbedingungen umfasst die Stadt *Hildesheim* sowie deren nähere Umgebung. Eine grobe Übersicht über das Gebiet und dessen Flächennutzung in den beiden Datenmodellen ALK und ATKIS ist in Abbildung 6.1 dargestellt. Der ursprünglich umfangreichere Datensatz von ATKIS wurde dabei so beschnitten, dass dieser nur Objekte enthält, deren Geometrie sich mit mindestens einer Geometrie des ALK-Datensatzes überschneidet.

Der Datensatz der ALK für das Gebiet Hildesheim umfasst insgesamt 8518 Objekte mit 46 unterschiedlichen Objektarten. Die Objektart entspricht dabei einer Objektklasse und hat beispielsweise den Wert 'Ackerland' oder 'Grünland'. Das Gebiet ist so gewählt, dass dieses sowohl typische städtische als auch typisch ländliche Objektarten enthält. Eine Übersicht über alle Objektarten der ALK ist in der Verwaltungsvorschrift zur Führung der Automatisierten Liegenschaftskarte (NVKV, 2003) enthalten.

In einem ersten Schritt der Vorverarbeitung in Abschnitt 6.3 werden 99 Objekte aus dem Datensatz entfernt, da diese eine fehlerhafte Geometrie aufweisen. Für die Aufstellung detaillierter Integritätsbedingungen werden aus den verbleibenden Objekten exemplarisch zehn Objektarten mit insgesamt 3854 zugehörigen Objekten ausgewählt (siehe Tabelle 6.1, mit Abkürzung Gebäude- und Freifläche (GuF)). Die ausgewählten Objektarten weisen alle eine vergleichsweise hohe Anzahl an Objekten auf und entsprechen Nutzungen, die sowohl in der ALK als auch in ATKIS flächenhaft repräsentiert werden. Aus diesem Grund ist beispielsweise die Objektart 'Straße' nicht in der Auswahl enthalten, da diese zwar in der ALK als Fläche repräsentiert wird, in ATKIS jedoch als Linie modelliert wird. Eine vollständige Auflistung der im Gebiet auftretenden Objektarten der ALK gibt die Tabelle A.1 im Anhang.

[1] Der Nachfolger der ALK ist das Automatisierte Liegenschaftskatasterinformationssystem (ALKIS).

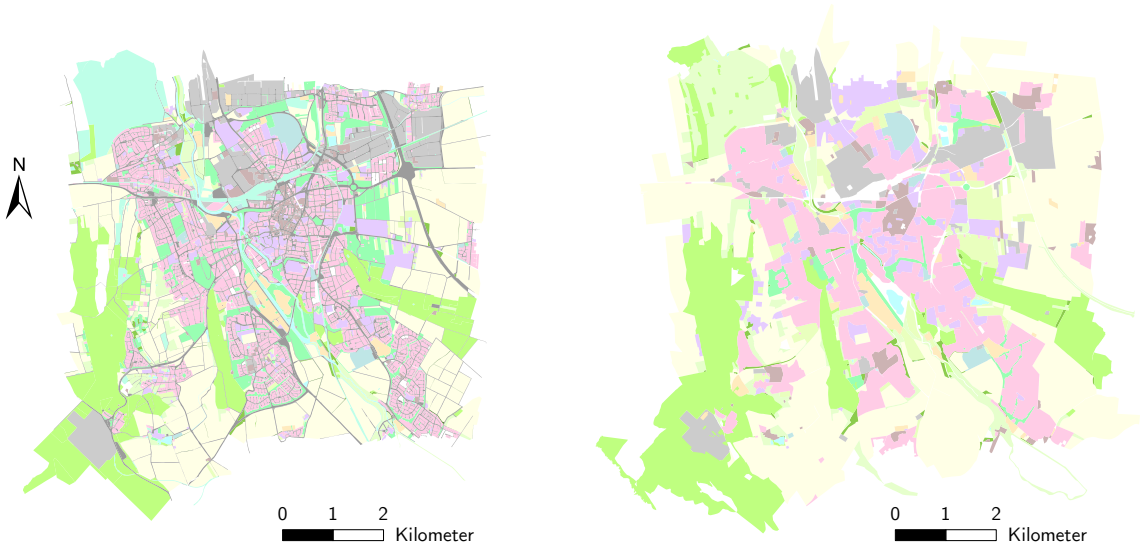


Abbildung 6.1: Flächennutzung in der ALK und in ATKIS im Gebiet Hildesheim

Der Datensatz von ATKIS umfasst ingesamt 2901 Objekte mit 36 unterschiedlichen Objektarten für das Gebiet Hildesheim. Eine entsprechende Übersicht über die Objektarten ist im ATKIS-Objektartenkatalog (AdV, 2004) enthalten. Von diesen werden wiederum exemplarisch neun Objektarten mit insgesamt 2036 Objekten ausgewählt (siehe Tabelle 6.1). Eine vollständige Auflistung der im Gebiet auftretenden Objektarten von ATKIS gibt die Tabelle A.2 im Anhang.

Objektart ALK	Kürzel	Anzahl
Ackerland	LAck	219
Gartenland	LGar	170
Grünanlage	LGan	491
Grünland	LGla	128
GuFGewerbeundIndustrie	LInG	270
GuFHandelDienstleistungen	LHaD	292
GuFMischnutzungMitWohnen	LMis	211
GuFÖffentlicheZwecke	LOef	202
GuFVersorgungsanlagen	LVer	207
GuFWohnen	LWhn	1664

Objektart ATKIS	Kürzel	Anzahl
Ackerland	TAck	190
BesondereFunktionalePrägung	TBFP	138
Gartenland	TGar	97
Grünanlage	TGan	104
Grünland	TGla	189
GemischteNutzung	TMis	105
IndustrieGewerbe	TInG	129
Ortslage	TOrt	30
Wohnbau	TWhn	1054

Tabelle 6.1: Objekte und Objektarten in der ALK und in ATKIS für das Gebiet Hildesheim

Das Gebiet Hildesheim wird ergänzt um zwei Gebiete unterschiedlicher Prägung. Das Gebiet *Hannover* umfasst einen Ausschnitt aus der Innenstadt Hannovers. Die Flächennutzung dieses städtisch geprägten Gebiets ist in Abbildung 6.2 für die beiden Datenmodelle ALK und ATKIS dargestellt. Die städtische Prägung spiegelt sich wider in der Anzahl der Objekte der einzelnen Objektarten der ALK, die in Tabelle 6.2 aufgelistet sind. So enthält das Gebiet keine oder nur wenige Objekte der Objektarten 'Ackerland', 'Gartenland' und 'Grünland'. Insgesamt umfasst das Gebiet Hannover für die zehn ausgewählten Objektarten der ALK 2557 Objekte.

Das zweite Gebiet umfasst das Gebiet der Gemeinde *Ostercappeln*. Die Flächennutzung dieses vor allem ländlich geprägten Gebiets ist in Abbildung 6.3 dargestellt, wobei für das Gebiet lediglich Daten der ALK vorliegen. Die Anzahl der Objekte der einzelnen Objektarten der ALK, die in Tabelle 6.2 aufgelistet sind, zeigen den deutlichen Unterschied der beiden Gebiete Hannover und Ostercappeln. Im Gegensatz zum Gebiet Hannover weist das Gebiet Ostercappeln zahlreiche Objekte der Objektarten 'Ackerland', 'Gartenland' und 'Grünland' auf. Die drei Objektarten machen 64 % der insgesamt 10 217 Objekte der zehn ausgewählten Objektarten der ALK aus.

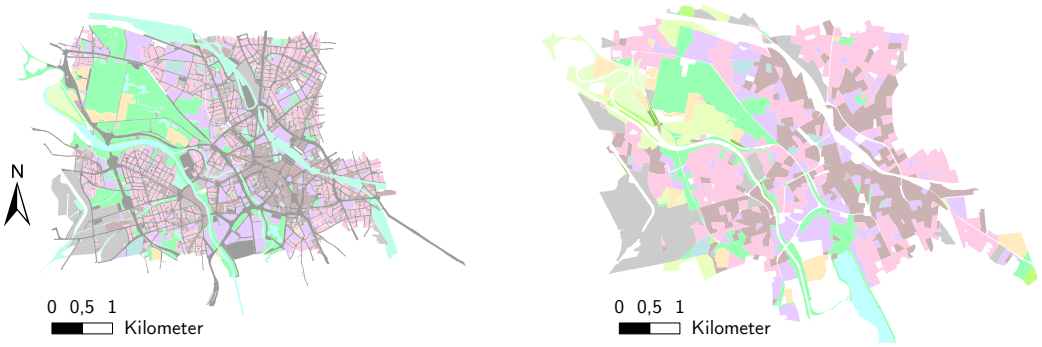


Abbildung 6.2: Flächennutzung in der ALK und in ATKIS für das Gebiet Hannover

Objektart ALK	Anzahl Hannover	Anzahl Ostercappeln
Ackerland	0	3110
Gartenland	9	272
Grünanlage	163	243
Grünland	1	3194
GuFGewerbeundIndustrie	162	322
GuFHandelDienstleistungen	245	251
GuFMischnutzungMitWohnen	878	5
GuFÖffentlicheZwecke	245	120
GuFVersorgungsanlagen	45	139
GuFWohnen	809	2561

Tabelle 6.2: Objekte und Objektarten in der ALK für die Gebiete Hannover und Ostercappeln

6.2 Prüfung der Geometrien und Anreicherung der Daten

Integritätsbedingungen können für alle Attribute eines Objekts aufgestellt werden. Die Daten der drei Gebiete weisen im Original als Objektattribute jedoch nur eine eindeutige Identifikationsnummer, eine Kennziffer für die Aktualität sowie die Objektart auf. Deshalb werden die Objekte um zusätzliche Attribute angereichert, welche die Form und Nachbarschaft der Objekte beschreiben. Mit diesen zusätzlichen Attributen können dann Integritätsbedingungen aufgestellt werden, die sich auf die ermittelte Form und Nachbarschaft der Objekte beziehen.

Die geringe Anzahl an vorhandenen Attributen ist dabei charakteristisch für viele Datensätze. Häufig werden von den Produzenten der Daten nur genau die Attribute modelliert, die für eine bestimmte Anwendung sinnvoll erscheinen. Benötigt ein Nutzer weitere Attribute, dann muss er diese aus den vorhandenen Attributen ableiten, aus der Geometrie und Topologie der Objekte bestimmen, oder durch die Datenintegration mit weiteren Datensätzen hinzufügen. Weist ein Datensatz nur wenige Attribute auf, dann können meist erst durch die Anreicherung der Daten interessante Muster entdeckt werden.

Vor der Anreicherung der Daten wird jedoch zuerst eine Prüfung der Objektgeometrien für beide Datensätze durchgeführt, da für ungültige Geometrien keine geometrischen und topologischen Maße berechnet werden können.

6.2.1 Prüfung der Geometrien

Durch die Prüfung der Objektgeometrien wird sichergestellt, dass diese gültig sind und somit weiter prozessiert werden können. Für Flächen beziehungsweise Polygone werden mehrere Kriterien überprüft, die alle erfüllt sein müssen:

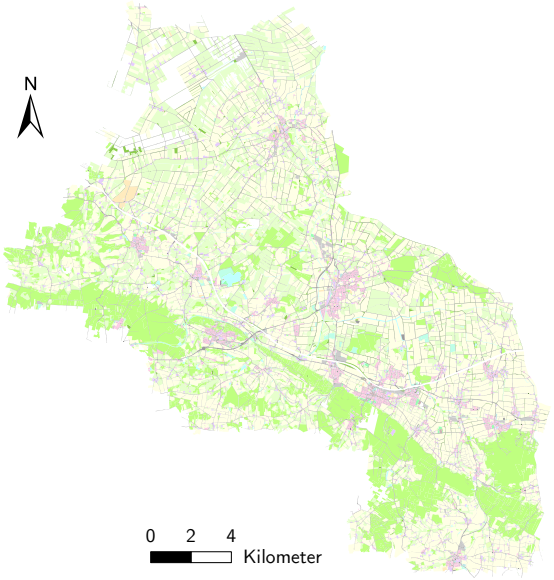


Abbildung 6.3: Flächennutzung in der ALK für das Gebiet Ostercappeln

- *Koordinaten der Stützpunkte*: Die Werte der Koordinaten müssen im gültigen Wertebereich von Fließkommazahlen sein. Ist die räumliche Ausdehnung der Daten bekannt, dann kann die Wertebereich der Koordinaten weiter eingeschränkt werden.
- *Geschlossener Umring*: Der äußere Ring (Umring) und alle inneren Ringe des Polygons müssen geschlossen sein, d.h. die Koordinaten des jeweils ersten Punkts müssen den Koordinaten des jeweils letzten Punkts entsprechen.
- *Anzahl der Stützpunkte*: Jedes Polygon muss mindestens drei voneinander verschiedene Stützpunkte aufweisen.
- *Konsistente Fläche*: Die Anordnung der einzelnen Kanten des Polygons muss eine konsistente Fläche erzeugen. Schneiden sich beispielsweise Kanten, dann ist nicht mehr gewährleistet, dass das innere und äußere eines Polygons eindeutig identifiziert werden können. Ebenso sind Duplikate der einzelnen Ringe nicht erlaubt.
- *Innere Ringe*: Für die inneren Ringe müssen mehrere Kriterien erfüllt sein. Erstens müssen sich diese innerhalb des Umrings des Polygons befinden. Zweitens dürfen innere Ringe nicht ineinander verschachtelt sein. Drittens dürfen innere Ringe nicht miteinander verbunden sein.

Von den 8518 Objekten aller Objektarten der ALK des Gebiets Hildesheim sind die Geometrien von 99 Objekten nicht gültig, was etwa 1,2 % der Daten entspricht. Für Geobasisdaten, die eigentlich vollständig gültig sein sollten, ist dies ein nicht unbeträchtlicher Anteil fehlerhafter Objekte. Alle ungültigen Geometrien verletzen die Anforderung an die Konsistenz der Fläche. Ein Beispiel dafür ist das Objekt mit der eindeutigen Kennnummer 1271, dessen wichtigste Attribute in Tabelle 6.3 aufgelistet sind. Die Koordinaten der beiden letzten Stützpunkte sind nahezu identisch, was auf einen duplizierten Punkt hindeutet. Die betroffenen fehlerhaften Objekte werden in den weiteren Untersuchungen nicht mehr berücksichtigt, da eine Korrektur der Objektgeometrien nicht im Fokus dieser Arbeit steht. In den ATKIS-Daten sind dagegen alle Objektgeometrien gültig.

Attribut	Wert
UID	1271
Objektart	GuFWohnen
Geometrie	(458,51/685,89), (461,12/690,28), . . . , (459,970976736/682,868437638), (458,509999996/685,890000008), (458,51/685,89) Rechtswert um 3 562 000 gekürzt, Hochwert um 5 780 000 gekürzt

Tabelle 6.3: Attribute eines fehlerhaften Objekts

Von den ursprünglich 3985 Objekten aller Objektarten der ALK des Gebiets Hannover sind 32 Objekte nicht gültig, was etwa 0,8 % der Daten entspricht. Und von den urprünglich 1782 Objekten aller Objektarten aus ATKIS des Gebiets Hannover sind 3 Objekte nicht gültig. Für das Gebiet Ostercappeln werden von der Prüfung lediglich 25 der ursprünglich 26 624 Objekte als fehlerhaft identifiziert, was einem sehr geringen Prozentsatz entspricht.

6.2.2 Anreicherung der Daten

Um die Anzahl der Bedingungen überschaubar zu halten, werden in dieser Arbeit nur Integritätsbedingungen für die Objekte der ALK aufgestellt. Selbstverständlich können mit dem gleichen Ansatz wie für die Objekte der ALK auch Integritätsbedingungen für die Objekte aus ATKIS aufgestellt werden. Die Objekte der ALK werden um neun geometrische Maße beziehungsweise Attribute zur Beschreibung der Form angereichert. Die Objekte werden zusätzlich um einige topologische Maße zur Beschreibung der Nachbarschaft der Objekte angereichert. Der wichtige Zusammenhang der Flächennutzung zwischen der ALK und ATKIS wird durch weitere topologische Maße beschrieben. Der Zusammenhang der Flächennutzung soll dabei die Frage klären, ob für ein Objekt der ALK eindeutig und zuverlässig die zugehörige Objektart aus ATKIS prädiziert werden kann. Ist dies der Fall, dann könnte beispielsweise ein ATKIS Datensatz automatisch auf seine Konsistenz zum entsprechenden ALK Datensatz überprüft werden, oder gar der ATKIS Datensatz (teil-) automatisch aus dem ALK Datensatz abgeleitet werden.

Geometrische Maße

Die flächenhaften Objekte der ALK werden mit insgesamt neun geometrischen Maßen angereichert (siehe Tabelle 6.4). Die Form der Geometrie wird durch sechs Maße direkt sowie durch drei abgeleitete Maße indirekt beschrieben. Die abgeleiteten Maße beziehen sich dabei auf das Minimum Enclosing Rectangle (MER) (dt. 'minimal umschließendes Rechteck') der Objektgeometrie. Die einzelnen Maße werden im Folgenden weiter erläutert.

Typ	Attribut	Bemerkung
Direkt	NumPts	Anzahl der Stützpunkte
	VxD{Maß}	Statistik über die Distanz der Stützpunkte
	Perim	Umfang (p)
	Area	Fläche (a)
	Comp	Kompaktheit bezogen auf ein Quadrat
	FracDim	Fraktale Dimension
Abgeleitet	MinDia	Minimaler Durchmesser des minimal umschließenden angepassten Rechtecks
	Elong	Elongation des minimal umschließenden angepassten Rechtecks
	MerDiff	Flächenverhältnis zum minimal umschließenden angepassten Rechteck

Tabelle 6.4: Geometrische Maße zur Anreicherung der Geobasisdaten

Die *Anzahl der Stützpunkte* gibt einen groben Überblick über die Komplexität einer Geometrie. Da flächenhafte Objekte angereichert werden, ist die einfachste mögliche Geometrie ein Dreieck und damit weist das Minimum der Anzahl der Stützpunkte den Wert drei auf. Bei der Ermittlung der Anzahl wird berücksichtigt, dass eine Fläche als Umring eines Polygons modelliert wird, bei dem der Anfangs- und Endpunkt identisch sind. Dieser wird folglich nur einmal gezählt. Die *Statistik über die Distanz der Stützpunkte* fasst Lage- und Streuungsmaße der Distanzen aufeinanderfolgender Stützpunkte des Umrings zusammen. So bezeichnet beispielsweise das Attribut *VxDMin* den minimalen Abstand zweier aufeinanderfolgender Stützpunkte einer Geometrie. Der *Umfang* p und die *Fläche* a der Geometrie sind wichtige direkte Maße, die zusätzlich für die Berechnung weiterer Maße verwendet werden. Die *Kompaktheit* drückt aus, wie ähnlich eine Geometrie in Bezug auf eine Referenzgeometrie ist. Ist die Referenzgeometrie ein Quadrat oder ein Kreis, dann berechnet sich die Kompaktheit nach Gleichung 6.1. Von Menschen geschaffene Geometrien, wozu auch Flächen beziehungsweise Flurstücke zählen, sind in der Regel mehr quadratisch beziehungsweise rechteckig denn kreisrund, weshalb das Maß $\text{Comp}_{\square}$ besser zur Beschreibung der Form geeignet ist. Die *fraktale Dimension* quantifiziert die Komplexität einer Geometrie und kann nach Gleichung 6.2 bestimmt werden. Für einige beispielhafte Geometrien der ALK sind die entsprechenden Werte für die Kompaktheit und die fraktale Dimension in Abbildung 6.4 dargestellt.

$$\text{Comp}_{\square} = \frac{p^2}{16 \cdot a}, \;\; \text{Comp}_{\bigcirc} = \frac{p^2}{4 \cdot \pi \cdot a} \tag{6.1}$$

$$\text{FracDim} = \frac{2 \cdot |\ln(p)|}{\ln(a)} \tag{6.2}$$

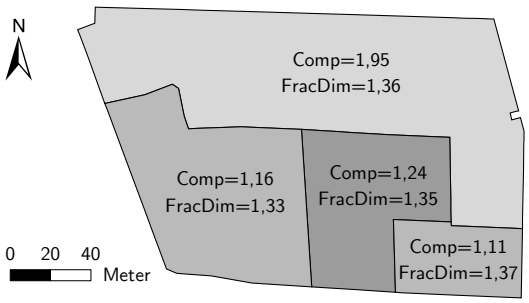


Abbildung 6.4: Beispiel für die Kompaktheit und die fraktale Dimension

Das minimal umschließende angepasste Rechteck bildet die Grundlage für die abgeleiteten Maße. Die Kenngrößen des MER sind dessen Breite b, dessen Länge $l \geq b$ und dessen Fläche a_{MER}. Der *minimale Durchmesser* des MER entspricht der Breite b und ist ein Indikator für die minimale Breite einer Geometrie. Die *Elongation* charakterisiert die Dimensionen des Rechtecks und kann nach Gleichung 6.3 ermittelt werden. Das *Flächenverhältnis* zum MER lässt sich nach Gleichung 6.4 berechnen und drückt aus um wie viel größer die Fläche des MER im Vergleich zur Fläche der Objektgeometrie ist.

$$\text{Elong} = \frac{l}{b}, \;\; \text{Elong} \geq 1 \tag{6.3}$$

$$\text{MerDiff} = \frac{a_{\text{MER}} - a}{a}, \;\; \text{MerDiff} \geq 1 \tag{6.4}$$

Topologische Maße

Die topologischen Maße beschreiben die Nachbarschaft der Objekte in der ALK und den Zusammenhang der Flächennutzung zwischen der ALK und ATKIS. Die Objekte der ALK werden um insgesamt vier Typen von Maßen angereichert, deren tatsächliche Anzahl abhängig von den zu berücksichtigenden Objektarten ist (siehe Tabelle 6.5). Die Maße basieren dabei nicht auf dem absoluten Zählen von topologisch verbundenen Objekten, sondern beziehen die verbundenen Objekte auf die Objektgeometrie des zu untersuchenden Objekts. Die einzelnen Maße werden im Folgenden näher erläutert.

Typ	Attribut	Bemerkung
Direkt	IL{Objektart}LRel	Anteil der Objektart an der Länge des Umrings
	ILDomLRel	Dominante Objektart des Umrings
Indirekt	L{Distanz}{Objektart}	Anteil der Objektart an der um die Distanz erweiterten Fläche
	L{Distanz}LDom	Dominante Objektart der um die Distanz erweiterten Fläche
Blöcke	BL{Objektart}ARel	Anteil der Objektart an der Fläche des Blocks
	BLDomARel	Dominante Objektart des Blocks
Zusammenhang	IT{Objektart}ARel	Anteil der Objektart aus ATKIS an der Fläche
	ITDomARel	Dominante Objektart aus ATKIS der Fläche

Tabelle 6.5: Topologische Maße zur Anreicherung der Objekte

Durch die Bestimmung der Objektarten der direkten Nachbarn eines Objekts kann untersucht werden, ob bestimmte Objektarten häufiger benachbart sind als andere, oder ob bestimmte Objektarten nie miteinander benachbart sind. Weist ein Objekt zudem nur Objekte einer einzigen Objektart als direkte Nachbarn auf, dann wird dieses vollständig von der anderen Objektart umschlossen. In dieser Arbeit werden die Objektarten

der direkten Nachbarn eines Objektes durch deren prozentuale Anteile an der Gesamtlänge des Umrings des Objekts ausgedrückt. Dieses Maß wird für jede Objektart der ALK getrennt berechnet. Zusätzlich wird diejenige Objektart, die den größten Anteil am Umring des Objekts aufweist, als dominante Objektart abgespeichert. Dabei muss die dominante Objektart mindestens 20 % des Umrings ausmachen, ansonsten wird die dominante Objektart auf den Wert 'None' gesetzt.

Ein Beispiel für die so berechneten Attribute ist in Abbildung 6.5 für einen Ausschnitt der ALK dargestellt. In einem dreieckigen Block, der komplett von Straßen umgeben ist, liegen je zwei Objekte der Objektarten 'GUFHandelDienstleistungen' ('LHaD') und 'GUFWohnen' ('LWhn') sowie ein Objekt der Objektart 'GUF-ÖffentlicheZwecke' ('LOef'). Die Werte der Attribute 'IL{Objektart}LRel' werden im Folgenden anhand des in der Abbildung gestrichelten Umrings des Objekts mit der Objektart 'LHaD' erläutert. Das Objekt hat keine direkten Nachbarn der Objektart 'LHaD', deshalb weist das Attribut 'ILHaDLRel' den Wert 0 % auf. Das Objekt ist im Osten durch ein Objekt der Objektart 'LOef' begrenzt, welches einen Anteil 'ILOefLRel' von 36 % des Umrings mit dem Objekt der Objektart 'LHaD' gemeinsam hat. Im Westen ist das Objekt durch ein Objekt der Objektart 'LWhn' begrenzt, das einen Anteil 'ILWhnLRel' von 42 % Anteil am Umfang aufweist. Der verbleibende Anteil des Umfangs in Höhe von 22 % ist den angrenzenden Straßen zuzuordnen. Alle weiteren Attribute 'IL{Objektart}LRel' weisen demnach einen Anteil von 0 % auf. Die dominante Objektart am Umring des Objekts ist 'LWhn', da diese den höchsten prozentualen Anteil aufweist und dieser größer als die minimal geforderten 20 % ist.

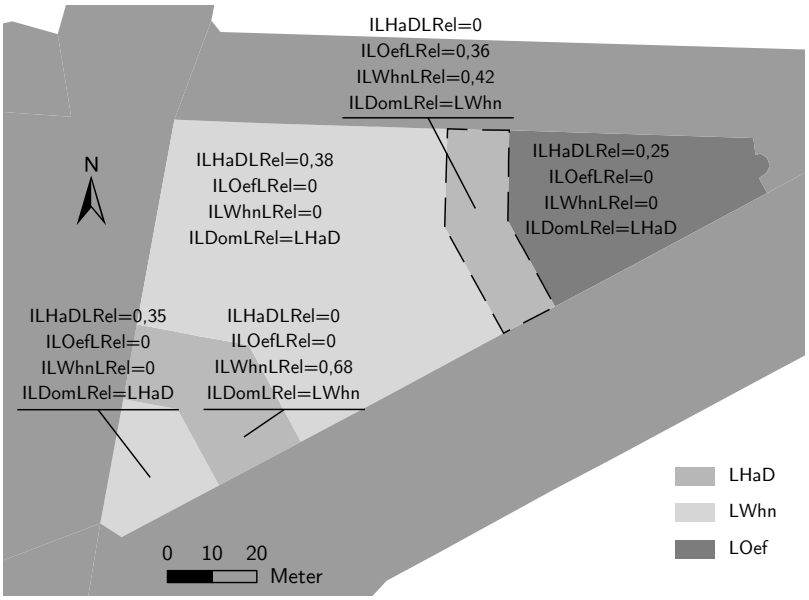


Abbildung 6.5: Beispiel für die Bestimmung der direkten Nachbarn der Objekte

Durch die Bestimmung der Objektarten der indirekten Nachbarn eines Objekts kann untersucht werden, welche Objektarten in der weiteren Umgebung um ein Objekt auftreten. So können Aussagen über die Nachbarn in einem größeren Bereich um ein Objekt getroffen werden. Zudem eignet sich dieses Maß, um Nachbarn über Korridore hinweg bestimmen zu können, beispielsweise auf der gegenüberliegenden Straßenseite eines Objekts. In dieser Arbeit werden die Objektarten der indirekten Nachbarn eines Objekts durch prozentuale Anteile an der Fläche der um eine bestimmte Distanz erweiterten Geometrie ausgedrückt. Dieses Maß wird wiederum für alle Objektarten der ALK getrennt berechnet und zusätzlich wird die dominante Objektart abgespeichert. Auch hier muss die dominante Objektart mindestens 20 % der erweiterten Geometrie ausmachen, ansonsten wird der Wert auf 'None' gesetzt.

Für die so berechneten Attribute wird für den gleichen Ausschnitt aus der ALK und dasselbe Objekt wieder ein Beispiel beschrieben, welches in Abbildung 6.6 dargestellt ist. Um die gestrichelte Geometrie des Objekts ist der Puffer mit einer Distanz von 50 m gestrichelt dargestellt. Als Geometrie wird für die folgende Statistik die symmetrische Differenz der um den Puffer erweiterten Geometrie und der Objektgeometrie selbst verwendet, d.h. es wird nur das 'Band' um die Objektgeometrie für die Statistik verwendet und nicht die Objektgeometrie selbst. Die so ermittelte Puffergeometrie wird mit den Geometrien aller benachbarten Objekte geschnitten und so deren Flächenanteile an der Puffergeometrie in Abhängigkeit von der Objektart bestimmt. Für das Objekt der Objektart 'LHaD' gehören die indirekten Nachbarn zu 10 % der Objektart 'LHaD', zu 29 % der Objektart 'LOef' und zu 22 % der Objektart 'LWhn' an. Der verbleibende Anteil der Fläche in Höhe von 39 % ist den

angrenzenden Straßen zuzuordnen. Alle weiteren Attribute 'L50L{Objektart}' weisen demnach einen Anteil von 0 % an der Puffergeometrie auf. Die dominante Objektart ist 'LOef', da diese den höchsten prozentualen Anteil aufweist und dieser größer als die minimal geforderten 20 % ist.

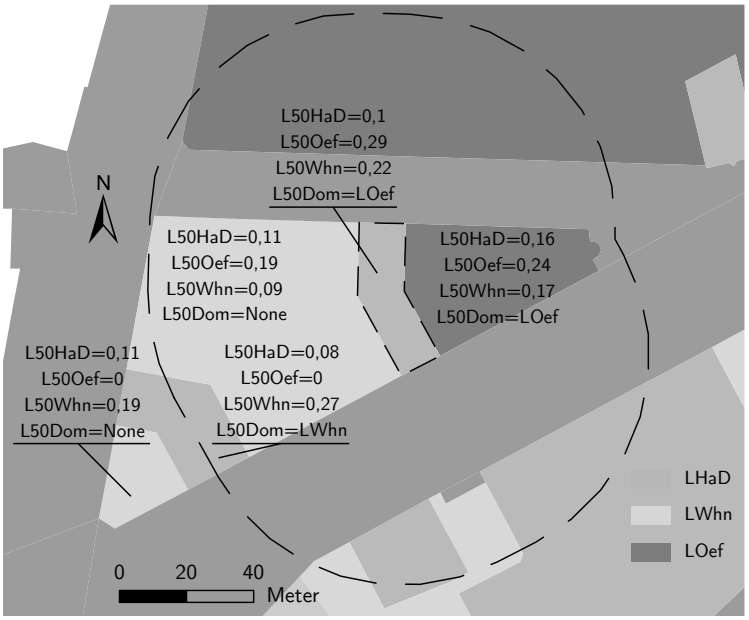


Abbildung 6.6: Beispiel für die Bestimmung der indirekten Nachbarn der Objekte

Im Hinblick auf die Generalisierung ist eine wichtige Eigenschaft eines Objekts der ALK die Zugehörigkeit zu einem Block. Ein Block enthält alle benachbarten Objekte, die nicht durch flächenhafte Infrastruktur oder Gewässer getrennt sind. Zur flächenhaften Infrastruktur zählen alle Straßen, Wege, Verkehrsflächen und Bahnanlagen. Zu Gewässern zählen alle Seen, Teiche, Flüsse und Bäche sowie Gräben. Bei der Generalisierung werden bei kleineren Blöcken häufig mehrere Objekte zu einem einzigen Objekt mit nur einer Objektart verschmolzen, die repräsentativ für den gesamten Block ist. Zur Bestimmung der Blöcke werden alle Objekte der entsprechenden Objektarten aus den Daten entfernt und anschließend benachbarte Geometrien verschmolzen. Für jeden Block werden die prozentualen Anteile der Objektarten der ALK an der Gesamtfläche des Blocks berechnet. Zusätzlich wird die dominante Objektart abgespeichert, die mindestens 40 % der Geometrie des Blocks ausmachen muss, da ansonsten der Wert auf 'None' gesetzt wird. Diese Informationen über einen Block werden für alle Objekte des Blocks als zusätzliche Attribute gespeichert.

Für den gleichen Ausschnitt aus der ALK ist in Abbildung 6.7 der entsprechende Block dargestellt, der mit einer gestrichelten Linie umrandet ist. Die zwei Objekte der Objektart 'LHaD' machen 19 % der Fläche des Blocks aus. Die zwei Objekte der Objektart 'LWhn' machen analog 60 % und das Objekt der Objektart 'LOef' 21 % der Gesamtfläche des Blocks aus. Die dominante Objektart des Blocks ist 'LWhn', da diese deutlich den höchsten Prozentsatz an der Fläche des Blocks aufweist und dieser über den minimal geforderten 40 % liegt.

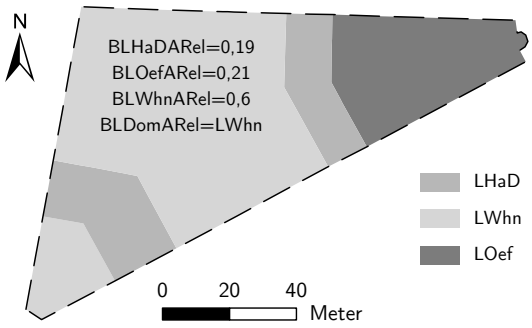


Abbildung 6.7: Beispiel für die Bestimmung der Blöcke

Auch der Zusammenhang zwischen Objekten der ALK und ATKIS wird durch Flächenverhältnisse bezogen auf die Objektarten ausgedrückt. Dazu wird die Geometrie eines Objekts der ALK mit den entsprechenden Objektgeometrien der ATKIS-Objekte verschnitten. Die Flächenanteile der ATKIS-Objektarten an der ALK-Geometrie

werden als topologische Maße abgespeichert. Zudem wird analog zu den anderen Typen von topologischen Maßen zusätzlich die dominante Objektart aus ATKIS für jedes ALK-Objekt bestimmt. Dabei muss die Objektart aus ATKIS jedoch mindestens 50 % der Fläche des ALK-Objekts überdecken, um als dominante Objektart zu gelten. Ansonsten wird diese analog auf den Wert 'None' gesetzt.

6.3 Statistik und Klassenbeschreibung

Wenn die Werte für die geometrischen und topologischen Maße bestimmt sind, dann können diese durch Methoden der deskriptiven und inferentiellen Statistik sowie der explorativen Datenanalyse untersucht werden (siehe Abschnitt 2.5). Dabei ist nicht nur von Interesse wie die Maße für alle ausgewählten Objektarten ausfallen, sondern auch wie sich diese zwischen den Objektarten unterscheiden. Das Ziel der Untersuchung sind Integritätsbedingungen, mit denen die charakteristischen Eigenschaften der einzelnen Objektarten eindeutig beschrieben werden können. Diese entsprechen der Klassenbeschreibung in Unterabschnitt 2.6.3 und bilden so die erste Säule des Data Mining für die Geobasisdaten.

Dieser Abschnitt zeigt nicht nur die statistischen Ergebnisse auf, sondern demonstriert allgemein, wie die Werkzeuge der Statistik verwendet werden können um analog zur Arbeit eines Detektivs die Daten zu analysieren und damit verborgene Aspekte der Daten offen zu legen (siehe auch die Definition der explorativen Datenanalyse nach Tukey (1977) in Unterabschnitt 2.5.3).

Die geometrischen Integritätsbedingungen werden in Unterabschnitt 6.3.1 zuerst anhand der Objekte des Gebiets Hildesheim aufgestellt. In Unterabschnitt 6.3.2 wird dann die Übertragbarkeit der Integritätsbedingungen auf die beiden Gebiete Hannover und Ostercappeln untersucht. Dabei werden im Rahmen der Untersuchung einige Bedingungen geringfügig angepasst. Die geometrischen Bedingungen werden zuerst anhand eines einzelnen repräsentativen Gebiets aufgestellt, da dadurch erstens die Analyse übersichtlicher wird und zweitens davon auszugehen ist dass geometrische Bedingungen nur in seltenen Fällen zwischen unterschiedlichen Gebieten abweichen. Die Bereinigung der Datensätze erfolgt in Unterabschnitt 6.3.3, da diese die vollständigen und endgültigen Integritätsbedingungen benötigt. In Unterabschnitt 6.3.4 werden schließlich die topologischen Integritätsbedingungen für die drei Gebiete zusammen aufgestellt. Die Bedingungen werden für die Gebiete zusammen aufgestellt, da erstens die Topologie als korrekt eingestuft wird und zweitens Unterschiede zwischen den Gebieten in der Topologie als zufällige Variationen der möglichen topologischen Konstellationen gesehen werden. Bei einer größeren Anzahl an Datensätzen kann die Aufstellung wie bei den geometrischen Bedingungen zuerst für einige wenige Datensätze durchgeführt werden und anschließend die Übertragbarkeit überprüft werden.

6.3.1 Geometrische Maße

Die Werte und Verteilungen der geometrischen Maße der Objekte des Gebiets Hildesheim werden im Folgenden basierend auf ihrer Definition in Unterabschnitt 6.2.2 analysiert.

Anzahl der Stützpunkte und Statistik über die Distanz der Stützpunkte

Die Lage- und Streuungsmaße für die Anzahl der Stützpunkte sind im Pentagramm und Box-Plot in Abbildung 6.8 dargestellt. Bei der Prüfung der Geometrien in Unterabschnitt 6.2.2 wird bereits sichergestellt, dass jede Fläche aus mindestens drei Stützpunkten besteht. Etwa die Hälfte der Objektgeometrien besteht aus weniger als 19 Stützpunkten, wovon insgesamt 13 der Geometrien Dreiecke und 114 der Geometrien Vierecke sind. Auffällig ist vor allem das hohe Maximum der Anzahl der Stützpunkte im Pentagramm und die hohe Anzahl von über 300 Ausreißern[2] im Box-Plot. Diese lassen sich prinzipiell durch zwei Interpretation erklären. Erstens können die Objektgeometrien einen entsprechend großen Umfang haben. So kann beispielsweise der Umring einer großen und komplex geformten Waldfläche aus vergleichsweise vielen Stützpunkten bestehen. Zweitens können die Geometrien viele nahe aneinanderliegende Stützpunkte aufweisen. Die beiden Interpretationen schließen sich dabei nicht gegenseitig aus.

Die zweite Interpretation lässt sich durch die Bestimmung der minimalen Distanz aufeinanderfolgender Stützpunkte des Umrings prüfen, deren Ergebnis in Abbildung 6.9 dargestellt ist. Das Minimum im Pentagramm

[2] Der obere innere Zaun liegt bei einem Wert von 87 Stützpunkten pro Objektgeometrie, wodurch sich 303 Ausreißer ergeben.

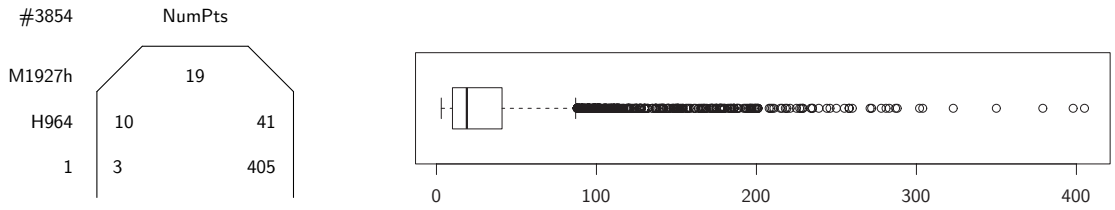


Abbildung 6.8: Pentagramm und Box-Plot der Anzahl der Stützpunkte der Objekte

ist auf Millimeter gerundet gleich Null, d.h. zwei Stützpunkte haben nahezu identische Koordinaten. Der Ausschnitt aus dem Histogramm für Distanzen $< 0{,}10\,\text{m}$ in Abbildung 6.9 zeigt einen deutlichen Anstieg der absoluten Häufigkeit bei minimalen Distanzen der Stützpunkte von weniger als 5 mm. Insgesamt weisen 208 Objekte minimale Distanzen der Stützpunkte $< 0{,}01\,\text{m}$ und 293 Objekte Distanzen $< 0{,}03\,\text{m}$ auf. Eben jene Schwellwerte entsprechen der geforderten Genauigkeit von Punkten in der ALK im innerstädtischen beziehungsweise ländlichen Bereich, d.h. die Koordinaten eines Grenzpunkts dürfen im innerstädtischen Bereich bis zu 1 cm von den Sollkoordinaten abweichen. Weisen aufeinanderfolgende Stützpunkte einer Objektgeometrie also eine entsprechend geringe Distanz auf, dann orientiert sich diese nicht an der Erfassungsgenauigkeit der ALK.

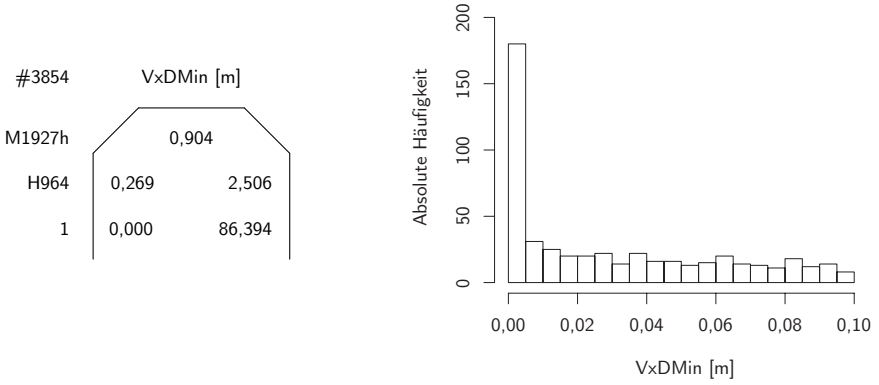


Abbildung 6.9: Pentagramm und Histogrammausschnitt der minimalen Distanz der Stützpunkte der Objekte

An diesem Punkt der Untersuchung ist ein Blick auf die Daten unerlässlich, um die Ursache für die ungünstige Verteilung der beiden vorgestellten Maße ermitteln zu können. In Abbildung 6.10 sind zwei Ursachen für die geringen Distanzen aufeinanderfolgender Stützpunkte dargestellt. Die erste Ursache sind unnötige Stützpunkte, die nicht zur Form der Geometrie beitragen und damit entfernt werden können. Die zweite Ursache ist die Modellierung von Kreisbögen. Da die Objektgeometrien linienhafte Umringe aufweisen, müssen Kreisbögen durch kurze aufeinanderfolgende Liniensegmente modelliert werden. Die Liniensegmente sind in den Daten jedoch häufig sehr kurz und führen dadurch zu vielen zusätzlichen Stützpunkten, deren Detailgrad nicht der Erfassungsgenauigkeit der ALK entspricht.

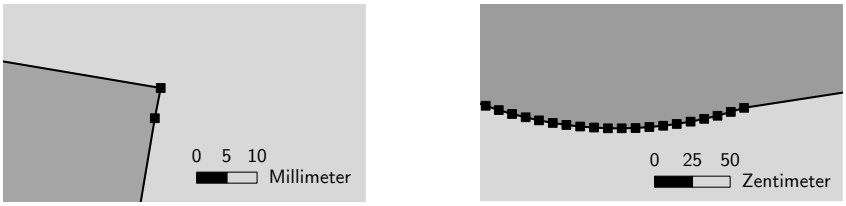


Abbildung 6.10: Minimale Abstände zwischen aufeinanderfolgenden Stützpunkten von Objekten

Integritätsbedingungen

Die Integritätsbedingungen für die Anzahl der Stützpunkte und deren minimale Distanz werden in der Geo Object Constraint Language (OCL) im Programm 6.1 aufgestellt. Die Anzahl der Stützpunkte muss für Flächen mindestens den Wert 4 aufweisen, da der erste und letzte Stützpunkt des geschlossenen Umrings identisch sind. Das Maximum der Stützpunkte kann prinzipiell auch beschränkt werden, jedoch ist es sinnvoller stattdessen die minimale Distanz aufeinanderfolgender Stützpunkte beziehungsweise die minimale Länge der Liniensegmente festzulegen. Diese kann einheitlich festgelegt werden, beispielsweise auf 5 cm. Die erforderliche minimale Länge der Liniensegmente kann jedoch auch unterschiedlich für innerstädtische und ländliche Gebiete festgelegt werden.

Dabei wird ein Objekt in der ALK als innerstädtisch angesehen, wenn dessen Geometrie mindestens ein Objekt der Objektart 'Ortslage' aus ATKIS schneidet.

```
context ALKNutzung
  inv NumPts: geom.exteriorRing().numPoints() >= 4
  inv VxDMin: geom.exteriorRing().segments()->forAll(s | s.length() >= 0.05) -- [Meter]

  inv VxDMinTOrt: if ALKNutzung.allInstances()->select(objektart = 'Ortslage')->any(o | o.geom.
      intersects(geom))->isEmpty() then geom.exteriorRing().segments()->forAll(s | s.length() >=
      0.15) else geom.exteriorRing().segments()->forAll(s | s.length() >= 0.05) endif
```

Programm 6.1: Integritätsbedingungen für die Anzahl der Stützpunkte und Statistik über die Distanz der Stützpunkte

Umfang und Fläche

Der Umfang und die Fläche sind wichtige Maße zur Beschreibung der Objektgeometrien. Sie bilden zudem die Grundlage für weitere Maße, die auf dem Verhältnis der beiden Größen basieren. Das Pentagramm des Umfangs ist in Abbildung 6.11 dargestellt. Aus diesem lässt sich ablesen, dass wenige Objekte einen überdurchschnittlich hohen Umfang aufweisen, dessen Maximum weit über dem oberen hinge liegt. Die Abbildung enthält zudem den Scatter-Plot des Umfangs und der Fläche, der deutlich den Zusammenhang der beiden Maße aufzeigt. Der Zusammenhang drückt sich auch durch einen hohen Wert des Korrelationskoeffizienten von $\tau = 0{,}88$ aus. Damit ist es ausreichend, wenn nur eines der beiden Maße genauer untersucht wird. Da die Flächennutzung untersucht wird, ist in diesem Fall die Fläche das sinnvollere der beiden Maße.

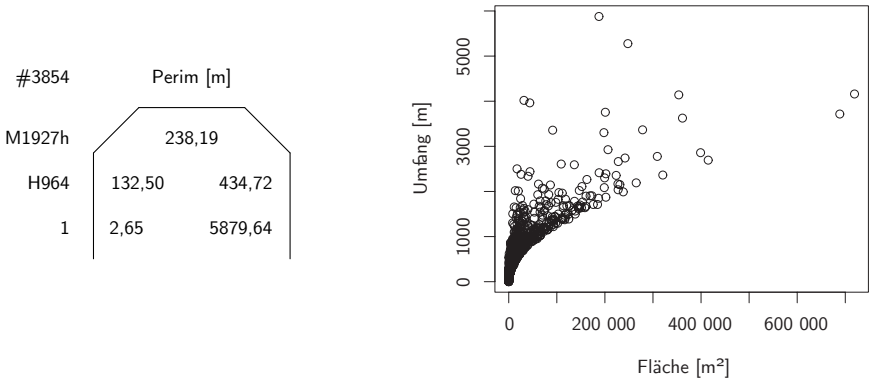


Abbildung 6.11: Pentagramm des Umfangs und Scatter-Plot des Umfangs und der Fläche der Objekte

Die Fläche weist für die Objekte in der ALK eine hohe Streuung auf, die sich in Abbildung 6.12 an der hohen Anzahl an Ausreißern im Box-Plot erkennen lässt. Der Median der Fläche $\tilde{x}_{0,5} = m = 2519\,\text{m}$ weicht zudem erheblich vom Mittelwert $\overline{x} = 9991\,\text{m}$ ab. Außerdem ist die Standardabweichung mit $s = 31\,597$ sehr hoch. Deshalb werden zuerst die extremsten Ausreißer untersucht und erst danach die verbleibenden Objekte.

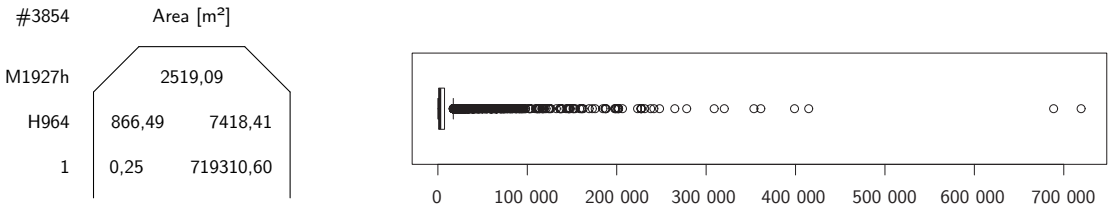


Abbildung 6.12: Pentagramm und Box-Plot der Fläche der Objekte

Lediglich ein Prozent der Daten liegt oberhalb des 99-ten Perzentils mit einer Fläche von mehr als $x_{0,99} = 149\,604{,}9\,\text{m}^2$. Von diesen 39 Objekten sind die meisten, nämlich 30, der Objektart 'Ackerland' zugeordnet. Die übrigen Objekte verteilen sich auf die drei Objektarten 'Grünanlage' (1), 'GuFGewerbeundIndustrie' (5) und 'GuFÖffentlicheZwecke' (3). Zusammenfassend lässt sich feststellen, dass diese vier Objektarten im Gebiet Hildesheim offenbar einige sehr große Flächen aufweisen. Auf der gegenüberliegenden Seite des Wertebereichs liegen unterhalb des ersten Perzentils 39 Objekte mit den kleinsten Flächen von weniger als $x_{0,01} = 14{,}4\,\text{m}^2$. Von diesen sind wiederum die meisten einer einzigen Objektart zugeordnet, nämlich 22 Objekte der Objektart 'GuFVersorgungsanlagen'. Die verbleibenden Objekte verteilen sich auf die Objektarten 'Ackerland' (2), 'Grünanlage' (3),

'Gartenland' (2), 'GuFHandelDienstleistungen' (1) und 'GuFWohnen' (9). Zusammenfassend lässt sich feststellen, dass einige der Objektarten sehr kleine Flächen aufweisen. Inwieweit die kleinen beziehungsweise großen Flächen der Objekte die Datenintegrität verletzen, wird im weiteren noch untersucht.

Die Verteilungen der Flächen der verbleibenden Objekte sind aufgeschlüsselt nach Objektart in den Box-Plots in Abbildung 6.13 dargestellt. Deutlich tritt der Box-Plot der Fläche der Objektart 'Ackerland' hervor, da sich dieser signifikant von den Box-Plots der anderen Objektarten unterscheidet. Die Flächen von Objekten der Objektart 'Ackerland' sind demnach deutlich größer als die der anderen Objektarten. Die Box-Plots der Fläche einiger Objektarten, wie beispielsweise der Objektarten 'Grünland' und 'GuFÖffentlicheZwecke' ähneln sich hingegen. Diese Ähnlichkeit bedeutet jedoch nicht, dass die Verteilungen identisch sind, wie die voneinander abweichenden Histogramme der beiden Objektarten in Abbildung 6.14 belegen. Diese zeigen die relative Häufigkeit der Flächen, die zwischen zwei Objektarten mit einer unterschiedlichen Anzahl an Objekten verglichen werden darf. Zudem zeigen die Histogramme den gleichen Wertebereich der Flächen. Dieser umfasst für die Objektarten jeweils mehr als 80 % der Objekte. Für die Definition von Integritätsbedingungen ist es deshalb sinnvoll die Flächen der einzelnen Objektarten getrennt voneinander zu untersuchen. Im Folgenden wird diese Untersuchung exemplarisch für die beiden Objektarten 'GuFVersorgungsanlagen' und 'GuFWohnen' durchgeführt.

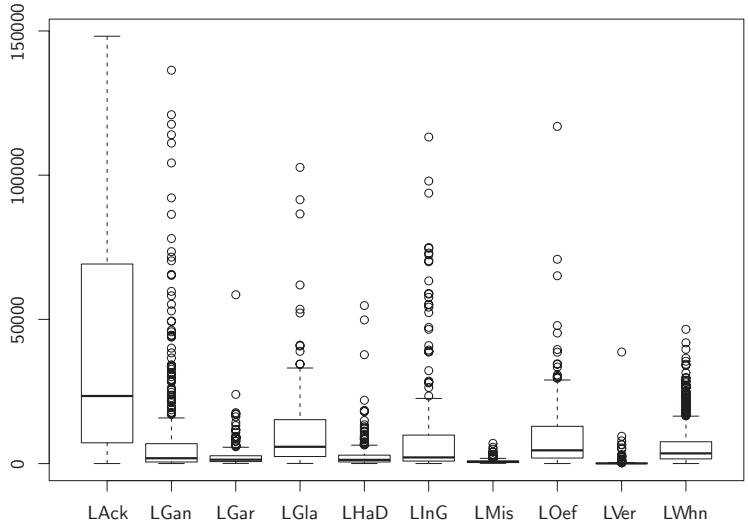


Abbildung 6.13: Box-Plots der Fläche der Objekte zwischen dem ersten und 99-ten Perzentil

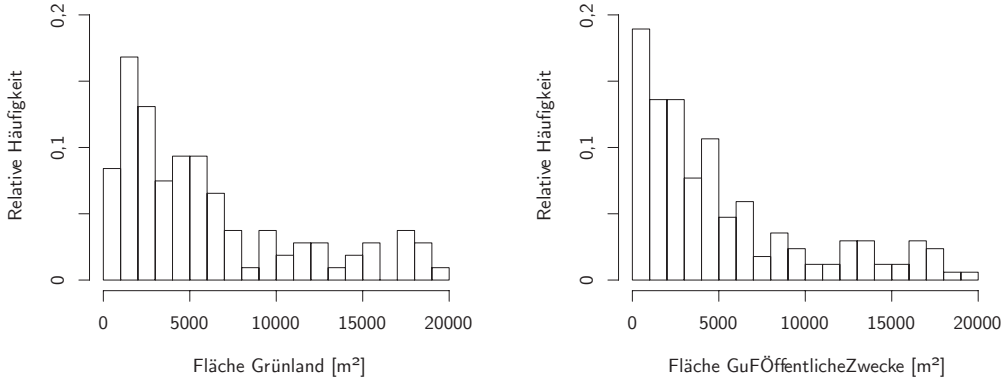


Abbildung 6.14: Histogrammausschnitte der Fläche der Objektarten 'Grünland' und 'GuFÖffentlicheZwecke'

Für die Objektart 'GuFVersorgungsanlagen' ist die Verteilung der Flächen in den Histogrammen in Abbildung 6.15 dargestellt. Das zweite Histogramm stellt dabei die Flächen bis zum achten Dezil $x_{0,80} = 143{,}2\,\mathrm{m}^2$ dar, also die Flächen von 80 % der Objekte. Aus den Histogrammen wird deutlich, dass die meisten Versorgungsanlagen kleine Flächen aufweisen. Demnach ist es sinnvoll die maximale Fläche für die Objektart zu beschränken, beispielsweise auf $10\,000\,\mathrm{m}^2$. Der eine Ausreißer mit etwa $38\,700\,\mathrm{m}^2$ wird dann als Fehler identifiziert. Ein Vergleich der Daten der ALK mit Luftbildern zeigt, dass dieser Ausreißer dem Betriebsgelände der Stadtwerke Hildesheim entspricht, welches dementsprechend groß ist. Die Beschränkung der minimalen Fläche für die Objektart basiert wiederum auf der Analyse des Histogramms. Im zweiten Histogramm in Abbildung 6.15 ist ein deutlicher Anstieg der absoluten Häufigkeit ab einer Fläche von $10\,\mathrm{m}^2$ zu erkennen. Lediglich fünf Ob-

jekte liegen unter diesem Schwellwert mit Flächen von 4,19 m², 7,96 m², 9,00 m², 9,50 m² und 9,71 m². Wird die Mindestfläche in den Integritätsbedingungen also auf 9 m² festgelegt, dann ist erstens die Anzahl der ungültigen Objekte niedriger und zweitens wird die Auswirkung der Klassengrenzen auf das Histogramm berücksichtigt. Ein Vergleich mit Luftbildern zeigt, dass die Flächen für Versorgungsanlagen durchaus so gering sein können, beispielsweise für kleine Umspannstationen.

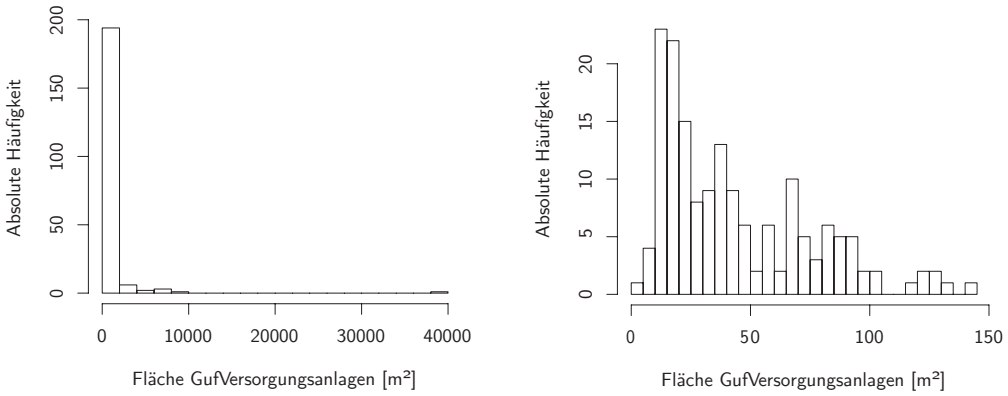


Abbildung 6.15: Histogramme der Fläche der Objektart 'GuFVersorgungsanlagen'

Für die Objektart 'GuFWohnen' werden die Histogramme in Abbildung 6.16 entsprechend analysiert. Das zweite Histogramm stellt wiederum die Flächen bis zum achten Dezil $x_{0,80} = 8539{,}3\,\mathrm{m}^2$ dar. Aus den Histogrammen wird deutlich, dass die Anzahl der Objekte für größere Flächen deutlich abnimmt. Es ist deshalb auch für die Wohnflächen sinnvoll eine maximale Fläche festzulegen, die von keinem Objekt überschritten werden sollte, beispielsweise 50 000 m². Im Gegensatz zur Objektart 'GuFVersorgungsanlagen' ist bei den Wohnflächen kein sinnvolles Minimum der Flächen aus dem zweiten Histogramm in Abbildung 6.16 ersichtlich. Erst das dritte Histogramm zeigt zwei Auffälligkeiten. Erstens haben 14 Objekte eine Fläche von weniger als 20 m². Die kleinste Fläche beträgt sogar nur 0,25 m². Zweitens gibt es eine Lücke zwischen den Klassengrenzen 40 m² und 60 m², wobei der erste nachfolgende Wert 70,14 m² ist. Die Mindestfläche für die Objektart 'GuFWohnen' wird deshalb auf 70 m² festgelegt. Die Nutzung von Flächen $< 40\,\mathrm{m}^2$ für Wohnen in den Daten ist nicht nachvollziehbar, da auf den Flächen errichtete Gebäude dann eine entsprechend geringe Anzahl an Quadratmetern aufweisen müssten.

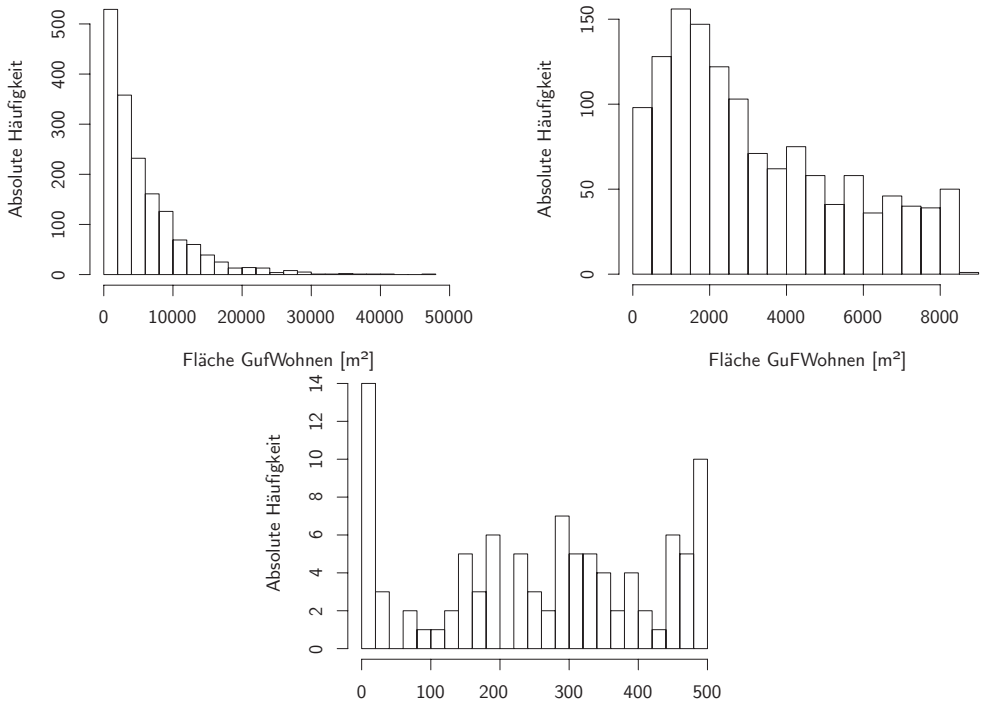


Abbildung 6.16: Histogramme der Fläche der Objektart 'GuFWohnen'

Für das Maß der Fläche ist nun noch die Frage zu klären, ob dieses einer bestimmten theoretischen Verteilung unterliegt. Das passende Werkzeug der explorativen Datenanalyse sind Quantile-Quantile-Plots, die in

Abbildung 6.17 dargestellt sind. Der erste Quantile-Quantile-Plot zeigt die Verteilung der Fläche gegen die entsprechende Normalverteilung. Dabei weicht die Verteilung der Fläche der Objekte der ALK deutlich von der Normalverteilung ab. Der zweite Quantile-Quantile-Plot zeigt die Verteilung des natürlichen Logarithmus der Fläche gegen die entsprechende Normalverteilung. Diese Datentransformation passt die Daten besser an die Normalverteilung an, jedoch ist immer noch eine deutliche Abweichung der minimalen und maximalen Werte zu erkennen. In Tabelle 6.6 belegen die Größen der Jarque-Bera-Tests, dass die Nullhypothese H_0 einer Normalverteilung sowohl für die Fläche als auch für den natürlichen Logarithmus der Fläche auf dem Signifikanzniveau 0,5 verworfen werden müssen.

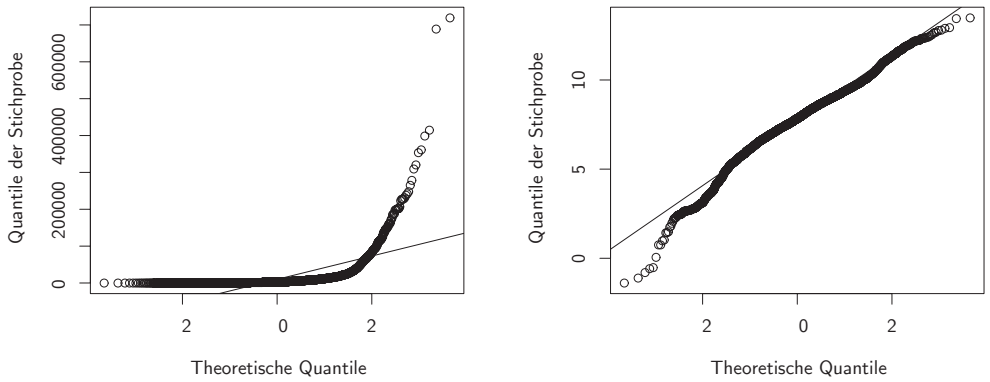


Abbildung 6.17: Quantile-Quantile-Plots der Fläche und des Logarithmus der Fläche

Attribut	Schiefe	Wölbung	Prüfgröße T	Kritischer Wert $\chi^2_{2;0,95}$	Testentscheidung
Area	10,16	160,57	4 053 428	5,99	H_0 verwerfen
ln(Area)	−0,54	4,28	452		

Tabelle 6.6: Jarque-Bera-Tests für die Fläche und den natürlichen Logarithmus der Fläche

Integritätsbedingungen

Aufgrund der hohen Korrelation der beiden Maße Umfang und Fläche werden im Programm 6.2 nur Integritätsbedingungen für die Fläche aufgestellt. Da sich die Verteilungen der Flächen zwischen den Objektarten deutlich unterscheiden (siehe Abbildung 6.13), werden für jede Objektart eigene Integritätsbedingungen aufgestellt. Diese Bedingungen beschränken die minimale und maximale Fläche. Da die Fläche der Objekte der ALK keiner bestimmten theoretischen Verteilung unterliegt, kann diesbezüglich keine zusätzliche Integritätsbedingung aufgestellt werden.

```
context ALKNutzung
  def: areaIsValid(objart: String, minInkl: Double, maxExkl: Double): Boolean = ALKNutzung.
      allInstances()->forAll(o | o.objektart = objart implies o.geom.area() >= minInkl and o.geom.
      area() < maxExkl) -- [Quadratmeter]

  inv AreaLVer: areaIsValid('GuFVersorgungsanlagen', 9, 10000)
  inv AreaLWhn: areaIsValid('GuFWohnen', 70, 50000)

  inv AreaLAck: areaIsValid('Ackerland', 500, 800000)
  inv AreaLGar: areaIsValid('Gartenland', 60, 600000)
  inv AreaLGan: areaIsValid('Grünanlage', 5, 250000)
  inv AreaLGla: areaIsValid('Grünland', 300, 150000)
  inv AreaLInG: areaIsValid('GuFGewerbeUndIndustrie', 130, 700000)
  inv AreaLHaD: areaIsValid('GuFHandelUndDienstleistungen', 20, 60000)
  inv AreaLMis: areaIsValid('GuFMischnutzungMitWohnen', 100, 7500)
  inv AreaLOef: areaIsValid('GuFÖffentlicheZwecke', 60, 300000)
```

Programm 6.2: Integritätsbedingungen für die Fläche

Kompaktheit und fraktale Dimension

Die Kompaktheit und die fraktale Dimension drücken die Komplexität einer Objektgeometrie durch das Verhältnis von Umfang zu Fläche aus. Das Pentagramm und die Box-Plots der Kompaktheit sind in Abbildung 6.18

dargestellt. Aus dem Pentagramm der Kompaktheit lässt sich der minimale Wert von 0,79 ablesen, der unter der Kompaktheit eines Quadrats mit einem Wert von 1 liegt. Insgesamt weisen etwa 4 % der Objekte eine Kompaktheit < 1 auf. Diese Objekte enthalten Kreisbögen als Teil ihrer Geometrie. Da das Vorhandensein von Kreisbögen prinzipiell nicht als Fehler zu sehen ist, wird für den minimalen Wert der Kompakheit keine Integritätsbedingung aufgestellt. Die maximalen Werte der Kompaktheit unterscheiden sich deutlich zwischen den Objektarten, wie aus den Box-Plots in Abbildung 6.18 ersichtlich ist. Deshalb wird für jede Objektart die maximal gültige Kompaktheit festgelegt. Exemplarisch wird dies im Folgenden für zwei Objektarten durchgeführt.

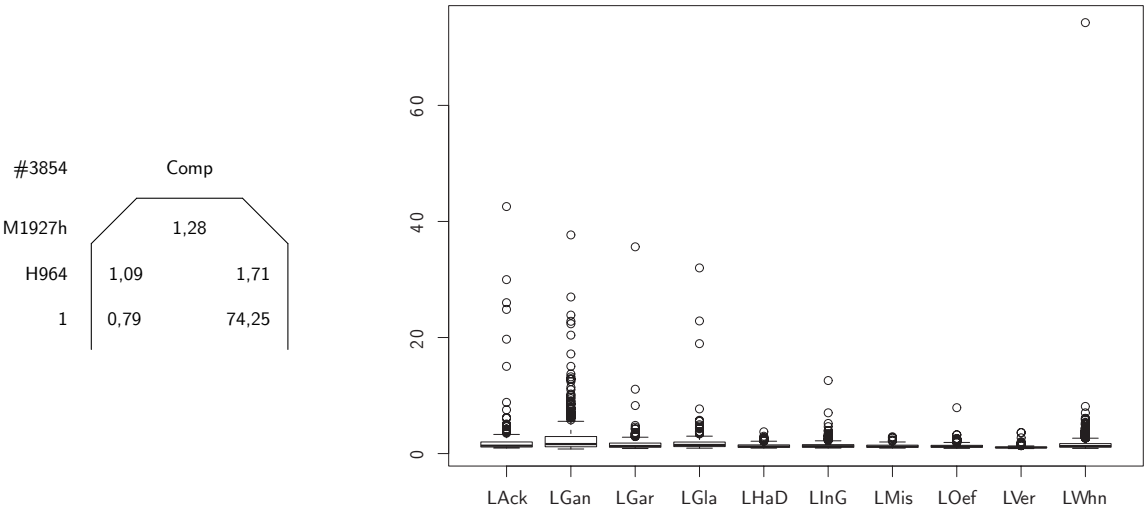


Abbildung 6.18: Pentagramm und Box-Plots der Kompaktheit der Objekte

Die maximale Kompaktheit der Objektart 'GuFHandelDienstleistungen' beträgt 3,77, weshalb der maximal gültige Wert auf 4 aufgerundet wird. Für die Objektart 'GuFWohnen' betragen die drei maximalen Werte der Kompaktheit 7,02, 8,13 und 74,25. Der letzte Wert ist ein deutlicher Ausreißer[3], weshalb die maximale Kompaktheit für die Objektart 'GuFWohnen' auf 9 aufgerundet wird. Für die anderen Objektarten werden die maximalen Werte für die Kompaktheit analog bestimmt.

Die fraktale Dimension ist das zweite Maß zu Bestimmung der Komplexität einer Geometrie. Das Pentagramm und die Box-Plots der fraktalen Dimension sind in Abbildung 6.19 dargestellt. Die Box-Plots enthalten nur fraktale Dimensionen mit Werten < 5, die lediglich von den Objektarten 'Gartenland' (9,5) und 'GuFWohnen' (5,07, 5,54 und 60,16) überschritten werden. Die Grenzwerte für die fraktale Dimension werden vergleichbar festgelegt wie bei der Kompaktheit. Jedoch wird für die fraktale Dimension ein minimaler Wert für alle Objektarten von 1,2 festgelegt, der das ermittelte Minimum nach unten geringfügig abrundet. Die maximal gültige fraktale Dimension wird wiederum für jede Objektart getrennt festgelegt.

Für die Objektart 'GuFHandelDienstleistungen' betragen die drei maximalen Werte 1,91, 1,93 und 4,75, wobei der letzte Wert wieder ein Ausreißer ist. Deshalb wird für die Objektart 'GuFHandelDienstleistungen' die maximale fraktale Dimension auf den Wert 2 aufgerundet. Die Festlegung der maximalen fraktalen Dimension für die Objektart 'GuFWohnen' ist etwas aufwändiger. Die Abbildung 6.20 zeigt mehrere deutliche Ausreißer im Box-Plot, wobei dieser nur Werte < 6 enthält. Die maximale fraktale Dimension wird mit dem Wert 1,9 so gewählt, dass nur Ausreißer im Box-Plot einbezogen werden, die nahezu lückenlos aneinander grenzen.

Für die Bestimmung des Zusammenhangs der geometrischen Maße werden die Korrelationen zwischen jeweils zwei Maßen ermittelt. Die Werte für die Korrelation mit Kendall's τ sind in Tabelle 6.7 aufgelistet. Aus der Tabelle wird deutlich, dass die fraktale Dimension negativ korreliert ist mit der Fläche sowie dem minimalen Durchmesser des Mer, d.h. je höher die fraktale Dimension ist, desto geringer ist die Fläche beziehungsweise der minimale Durchmesser des Mer. Dagegen ist die Kompaktheit mit keinem der anderen Maße deutlich linear korreliert.

Abschließend ist für die beiden Maße Kompaktheit und fraktale Dimension zu klären, ob diese einer Normalverteilung oder einer anderen bekannten theoretischen Verteilung unterliegen. Durch die Jarque-Bera-Tests, deren Ergebnisse für die verbleibenden geometrischen Maße in Tabelle 6.8 aufgelistet sind, wird belegt, dass die Nullhypothese H_0 einer Normalverteilung sowohl für die Kompaktheit als auch für die fraktale Dimensi-

[3] Das Objekt mit der Kompaktheit von 74,25 hat eine Fläche von 7,12 m^2 bei einem Umfang von 91,97 m.

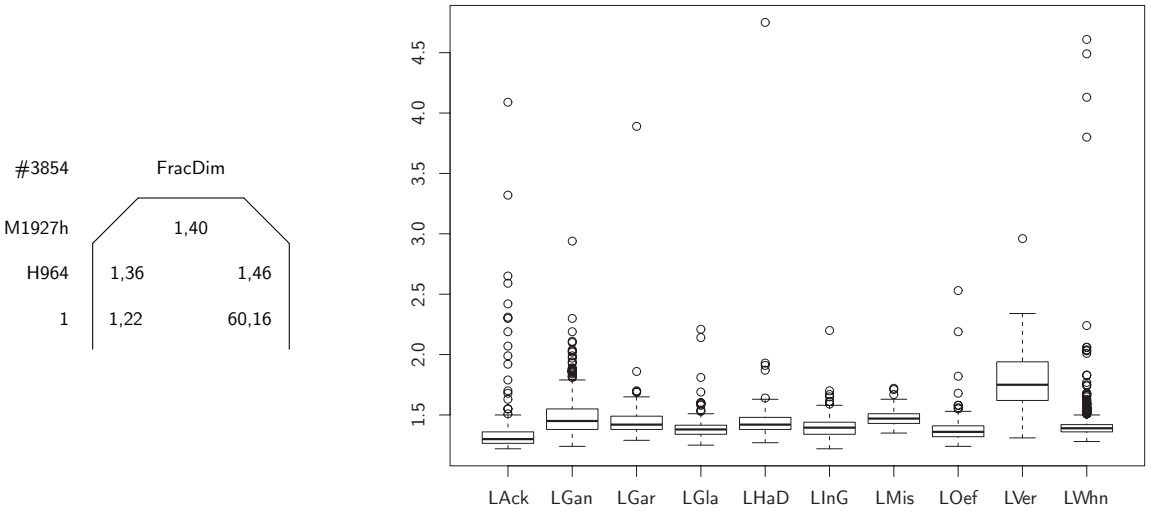


Abbildung 6.19: Pentagramm und Box-Plot-Ausschnitte der fraktalen Dimension der Objekte

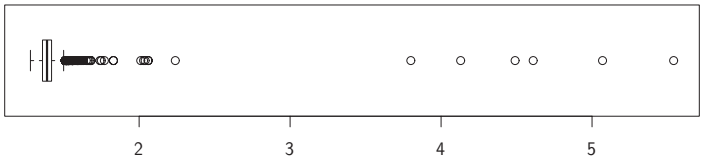


Abbildung 6.20: Box-Plot-Ausschnitt der fraktalen Dimension der Objektart 'GuFWohnen'

on auf dem Signifikanzniveau 0,5 verworfen werden müssen. Zudem führt für beide Maße auch keine der in Unterabschnitt 2.5.3 aufgelisteten Datentransformationen zu einer Normalverteilung.

Integritätsbedingungen

Für die Kompaktheit und die fraktale Dimension werden für jede Objektart eigene Integritätsbedingungen aufgestellt. Diese sind im Programm 6.3 zusammengefasst. Da die beiden Maße keiner bestimmten theoretischen Verteilung unterliegen, können diesbezüglich keine zusätzlichen Integritätsbedingungen aufgestellt werden.

Minimaler Durchmesser des umschließenden angepassten Rechtecks

Der minimale Durchmesser des umschließenden angepassten Rechtecks ist das erste der abgeleiteten Maße. Das Pentagramm und der Box-Plot des minimalen Durchmessers sind in Abbildung 6.21 dargestellt. Das Minimum des Maßes weist einen Wert von lediglich 0,14 m auf. Das Objekt mit dieser Breite des MERs hat jedoch auch nur eine Fläche von 2,11 m^2. Tatsächlich besteht ein signifikanter linearer Zusammenhang zwischen der Fläche und dem minimalen Durchmesser des MER, der sich in einem hohen Korrelationskoeffizienten von $\tau = 0{,}81$ ausdrückt (siehe Tabelle 6.7).

Dennoch beschreiben die beiden Maße unterschiedliche Eigenschaften einer Geometrie. Eines der Objekte der ALK hat einen minimalen Durchmesser von 1,31 m bei einer Fläche von 129,8 m^2. Für ein Objekt der Objektart 'Grünanlage' ist diese Kombination der Werte nicht auffällig, da das Objekt beispielsweise ein längerer Grünstrei-

Attribut	Area	Comp	FracDim	MinDia	Elong	MerDiff
Area	—	0,22	−0,66	0,81	0,01	0,24
Comp		—	0,14	0,18	0,38	0,42
FracDim			—	−0,64	0,12	−0,03
MinDia				—	−0,15	0,35
Elong					—	−0,03
MerDiff						—

Tabelle 6.7: Korrelationen der geometrischen Maße

Attribut	Schiefe	Wölbung	Prüfgröße T	Kritischer Wert $\chi^2_{2;0,95}$	Testentscheidung
Comp	14,39	320,06	16 275 870	5,99	H_0 verwerfen
FracDim	56,30	3365,69	1 817 862 494		
MinDia	3,91	26,80	100 774		
Elong	23,33	381,70	75 191 006		
MerDiff	15,27	381,70	23 179 237		

Tabelle 6.8: Jarque-Bera-Tests für die verbleibenden geometrischen Maße

```
context ALKNutzung
  def: compIsValid(objart: String, maxInkl: Double): Boolean = ALKNutzung.allInstances()->forAll(o
      | o.objektart = objart implies o.geom.compactnessSquare() <= maxInkl) -- []

  inv CompLHaD: compIsValid('GuFHandelUndDienstleistungen', 4)
  inv CompLWhn: compIsValid('GuFWohnen', 9)

  inv CompLAck: compIsValid('Ackerland', 7)
  inv CompLGar: compIsValid('Gartenland', 9)
  inv CompLGan: compIsValid('Grünanlage', 30)
  inv CompLGla: compIsValid('Grünland', 8)
  inv CompLInG: compIsValid('GuFGewerbeUndIndustrie', 13)
  inv CompLMis: compIsValid('GuFMischnutzungMitWohnen', 3)
  inv CompLOef: compIsValid('GuFÖffentlicheZwecke', 4)
  inv CompLVer: compIsValid('GuFVersorgungsanlagen', 4)

  def: fracDimIsValid(objart: String, maxInkl: Double): Boolean = ALKNutzung.allInstances()->forAll
      (o | o.objektart = objart implies o.geom.fractalDimension() <= maxInkl) -- []

  inv FracDimMin: ALKNutzung.allInstances()->forAll(o | o.geom.fractalDimension() >= 1.2) -- []
  inv FracDimLHaD: fracDimIsValid('GuFHandelUndDienstleistungen', 2)
  inv FracDimLWhn: fracDimIsValid('GuFWohnen', 1.9)

  inv FracDimLAck: fracDimIsValid('Ackerland', 1.8)
  inv FracDimLGar: fracDimIsValid('Gartenland', 1.9)
  inv FracDimLGan: fracDimIsValid('Grünanlage', 2.4)
  inv FracDimLGla: fracDimIsValid('Grünland', 1.7)
  inv FracDimLInG: fracDimIsValid('GuFGewerbeUndIndustrie', 1.8)
  inv FracDimLMis: fracDimIsValid('GuFMischnutzungMitWohnen', 1.8)
  inv FracDimLOef: fracDimIsValid('GuFÖffentlicheZwecke', 1.7)
  inv FracDimLVer: fracDimIsValid('GuFVersorgungsanlagen', 3)
```

Programm 6.3: Integritätsbedingungen für die Kompaktheit und die fraktale Dimension

fen sein könnte. Dagegen ist die Kombination der Werte für ein Objekt der Objektart 'GuFWohnen' auffällig, da zwar die Fläche ausreichen würde um ein Gebäude zu errichten, jedoch nicht der minimale Durchmesser des Mer. Analog zur Fläche ist demnach auch der minimale Durchmesser abhängig von der Objektart. Im Folgenden wird diese Untersuchung exemplarisch für die beiden Objektarten 'Grünanlage' und 'GuFWohnen' durchgeführt.

Für die beiden Objektarten ist die Verteilung des minimalen Durchmessers bis zum unteren hinge in den Histogrammen in Abbildung 6.22 dargestellt. Lediglich zwei Objekte der Objektart 'Grünanlage' weisen einen minimalen Durchmesser von weniger als zwei Metern auf. Die absolute Häufigkeit der Objekte steigt darüber deutlich an. Deshalb wird das Minimum von zwei Metern in die Integritätsbedingungen übernommen. Bei der Objektart 'GuFWohnen' ist die Identifikation eines unteren Schwellwerts für die minimale Breite des Mer schwieriger. Logischerweise gibt es eine untere Grenze für die minimale Breite, da sonst auf der Fläche errichtete Gebäude zu schmal wären. In den Bedingungen wird dieses Minimum nach einem Vergleich mit Luftbildern auf 5 m festgelegt, welches lediglich durch 15 Objekte unterschritten wird.

Integritätsbedingungen

Für den minimalen Durchmesser des Mer wird im Programm 6.4 wieder für jede Objektart eine eigene Integritätsbedingung aufgestellt. Die Schwellwerte für die beiden Objektarten 'Grünanlage' und 'GuFWohnen' sind bereits exemplarisch bestimmt. Die weiteren Schwellwerte werden analog ermittelt. Da die beiden Maße nach Tabelle 6.8 keiner Normalverteilung unterliegen und auch nicht durch Datentransformationen in eine Normalver-

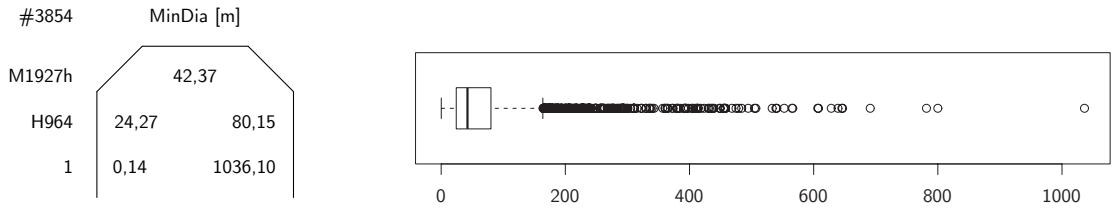


Abbildung 6.21: Pentagramm und Box-Plot des minimalen Durchmessers des MERs der Objekte

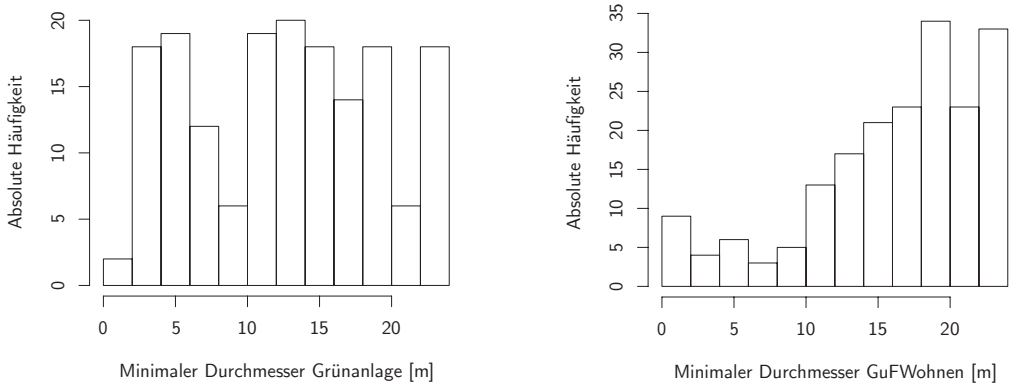


Abbildung 6.22: Histogrammausschnitte des minimalen Durchmessers des MERs der Objektarten 'Grünanlage' und 'GuF-Wohnen'

teilung überführt werden können, werden keine entsprechenden Bedingungen an die statistischen Verteilungen der Maße formuliert.

```
context ALKNutzung
  def: minDiaIsValid(objart: String, minInkl: Double): Boolean = ALKNutzung.allInstances()->forAll(
      o | o.objektart = objart implies o.geom.minimumEnclosingAdaptedRectangle().width() >=
      minInkl) -- [Meter]

  inv MinDiaLGan: minDiaIsValid('Grünanlage', 2)
  inv MinDiaLWhn: minDiaIsValid('GuFWohnen', 5)

  inv MinDiaLAck: minDiaIsValid('Ackerland', 20)
  inv MinDiaLGar: minDiaIsValid('Gartenland', 3)
  inv MinDiaLGla: minDiaIsValid('Grünland', 8)
  inv MinDiaLInG: minDiaIsValid('GuFGewerbeUndIndustrie', 5)
  inv MinDiaLHaD: minDiaIsValid('GuFHandelUndDienstleistungen', 3)
  inv MinDiaLMis: minDiaIsValid('GuFMischnutzungMitWohnen', 5)
  inv MinDiaLOef: minDiaIsValid('GuFÖffentlicheZwecke', 4)
  inv MinDiaLVer: minDiaIsValid('GuFVersorgungsanlagen', 1.5)
```

Programm 6.4: Integritätsbedingungen für den minimalen Durchmesser des MERs

Elongation des und Flächenverhältnis zum minimal umschließenden angepassten Rechteck

Neben dem minimalen Durchmesser des MER sind dessen Elongation sowie das Flächenverhältnis zur Objektgeometrie weitere wichtige Maße zur Beschreibung der Form eines Objekts. Die Pentagramme und Box-Plots der beiden Maße in den Abbildungen 6.23 und 6.24 weisen einige Gemeinsamkeiten auf, weshalb diese zusammen behandelt werden. Beide Maße haben per Definition einen minimalen Wert von 1. Bei der Elongation entspricht das MER dann einem Quadrat, wohingegen beim Flächenverhältnis die Objektgeometrie dann dem MER entspricht. Beide Minima sind für alle Objektarten der ALK realistisch, weshalb der minimale Wert der beiden Maße nicht durch Integritätsbedingungen beschränkt wird. Anders ist dies bei den maximalen Werten für die Objektarten, die bei beiden Maßen durch deutliche Ausreißer gekennzeichnet sind. Die Bestimmung der Bedingungen wird wieder exemplarisch für zwei Objektarten durchgeführt.

Für die Objektart 'Grünanlage' hat die Elongation einen maximalen Wert von 47,01, der nach Überprüfung mit Luftbildern keinen Fehler darstellt. Der maximal gültige Wert der Elongation für die Objektart 'Grünanlage' wird deshalb großzügig aufgerundet auf 50. Das maximale Flächenverhältnis weist einen Wert von 15,64 auf,

der wiederum einer gültigen Geometrie zugeordnet werden kann. Entsprechend wird der maximal gültige Wert für die Flächendifferenz zum MER der Objektart 'Grünanlage' auf den Wert 16 festgelegt. Die drei maximalen Werte der Elongation für die Objektart 'GUFWohnen' betragen 10,09, 12,18 und 169,38, wobei der letzte Wert ein deutlicher Ausreißer ist. Deshalb wird die maximal gültige Elongation auf den Wert 15 aufgerundet. Das Flächenverhältnis zum MER der Objektart 'GUFWohnen' ist genauso wie die Elongation um einige Größen kleiner als das der Objektart 'Grünanlage'. Der maximale Wert beträgt 3,80, d.h. das MER hat eine knapp viermal größere Fläche als die Objektgeometrie. Der maximal gültige Wert für das Flächenverhältnis zum MER der Objektart 'GUFWohnen' wird deshalb auf den Wert 4 aufgerundet.

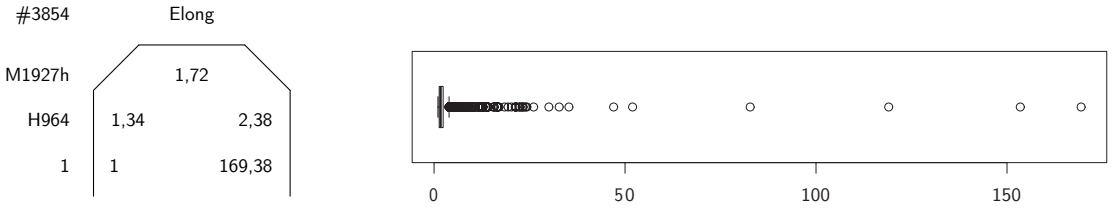


Abbildung 6.23: Pentagramm und Box-Plot der Elongation des MERs der Objekte

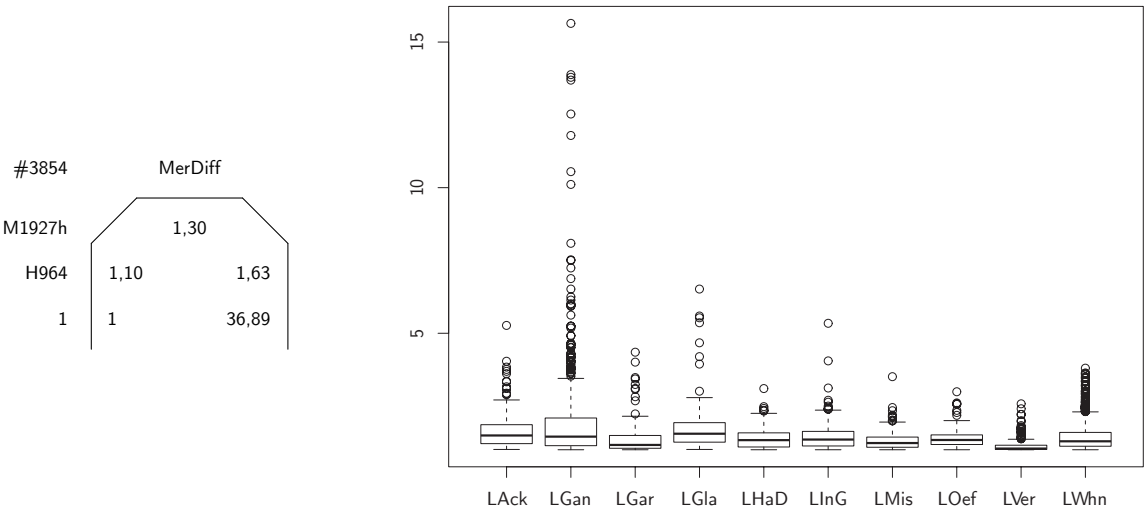


Abbildung 6.24: Pentagramm und Box-Plot-Ausschnitte des Flächenverhältnisses zum MER der Objekte

Aus Tabelle 6.8 ist wiederum ersichtlich, dass sowohl die Elongation als auch das Flächenverhältnis zum MER keiner Normalverteilung unterliegen. Auch können diese durch keine der Datentransformationen in eine Normalverteilung überführt werden. Die beiden Maße haben zudem bei einem Korrelationskoeffizienten von $\tau = -0{,}03$ keinen deutlichen linearen Zusammenhang (siehe Tabelle 6.7).

Integritätsbedingungen

Im Programm 6.5 sind die Integritätsbedingungen für die Elongation des MER und die Flächendifferenz der Objektgeometrie zum MER aufgelistet. Wie bereits erwähnt, werden dabei nur die maximalen Werte durch Integritätsbedingungen eingeschränkt, da die minimalen Werte per Definition den Wert 1 aufweisen.

6.3.2 Übertragbarkeit der geometrischen Bedingungen

Die für das Gebiet Hildesheim Hildesheim in Unterabschnitt 6.3.1 aufgestellten geometrischen Integritätsbedingungen werden nun hinsichtlich ihrer Übertragbarkeit auf die beiden Gebiete Hannover und Ostercappeln untersucht und falls notwendig entsprechend angepasst.

Gebiet Hannover

Die für das Gebiet Hildesheim aufgestellten Integritätsbedingungen erzielen vergleichbare Ergebnisse für das Gebiet Hannover mit Ausnahme von zwei Objektarten.

Für die Objektart 'Grünanlage' wird die maximale Fläche im Programm 6.2 auf 250 000 m^2 festgelegt. Dieses Maximum wird von keinem Objekt des Gebiets Hildesheim überschritten. In der Innenstadt von Hannover

```
context ALKNutzung
  def: elongIsValid(objart: String, maxInkl: Double): Boolean = ALKNutzung.allInstances()->forAll(o
       | o.objektart = objart implies o.geom.minimumEnclosingAdaptedRectangle().elongation() <=
      maxInkl) -- []

  inv ElongLGan: elongIsValid('Grünanlage', 50)
  inv ElongLWhn: elongIsValid('GuFWohnen', 15)

  inv ElongLAck: elongIsValid('Ackerland', 13)
  inv ElongLGar: elongIsValid('Gartenland', 14)
  inv ElongLGla: elongIsValid('Grünland', 13)
  inv ElongLInG: elongIsValid('GuFGewerbeUndIndustrie', 6)
  inv ElongLHaD: elongIsValid('GuFHandelUndDienstleistungen', 6)
  inv ElongLMis: elongIsValid('GuFMischnutzungMitWohnen', 6)
  inv ElongLOef: elongIsValid('GuFÖffentlicheZwecke', 6)
  inv ElongLVer: elongIsValid('GuFVersorgungsanlagen', 5)

  def: merDiffIsValid(objart: String, maxInkl: Double): Boolean = ALKNutzung.allInstances()->forAll
      (o | o.objektart = objart implies (o.geom.minimumEnclosingAdaptedRectangle().area() - o.geom
      .area()) / o.geom.area() <= maxInkl) -- []

  inv MerDiffLGan: merDiffIsValid('Grünanlage', 16)
  inv MerDiffLWhn: merDiffIsValid('GuFWohnen', 4)

  inv MerDiffLAck: merDiffIsValid('Ackerland', 6)
  inv MerDiffLGar: merDiffIsValid('Gartenland', 5)
  inv MerDiffLGla: merDiffIsValid('Grünland', 7)
  inv MerDiffLInG: merDiffIsValid('GuFGewerbeUndIndustrie', 6)
  inv MerDiffLHaD: merDiffIsValid('GuFHandelUndDienstleistungen', 4)
  inv MerDiffLMis: merDiffIsValid('GuFMischnutzungMitWohnen', 4)
  inv MerDiffLOef: merDiffIsValid('GuFÖffentlicheZwecke', 4)
  inv MerDiffLVer: merDiffIsValid('GuFVersorgungsanlagen', 3)
```

Programm 6.5: Integritätsbedingungen für die Elongation des MERs und für das Flächenverhältnis zum MER

weisen jedoch zwei Objekte der Objektart 'Grünland' Flächen von 393 313 m^2 und 479 108 m^2 auf, die nach Vergleich mit Luftbildern beide gültig sind. Deshalb wird im Programm 6.6 das Maximum der Fläche für die Objektart 'Grünland' auf 500 000 m^2 erhöht.

```
context ALKNutzung
  inv AreaLGan: areaIsValid('Grünanlage', 5, 500000) -- Maximum 500.000 anstatt 250.000
      Quadratmeter
  inv AreaLMis: areaIsValid('GuFMischnutzungMitWohnen', 100, 60000) -- Maximum 60.000 anstatt 7.500
       Quadratmeter

  inv CompLMis: compIsValid('GuFMischnutzungMitWohnen', 4) -- [], Maximum 4 anstatt 3
```

Programm 6.6: Geänderte Integritätsbedingungen für die Fläche aufgrund des Gebiets Hannover

Die maximale Fläche wird für die Objektart 'GuFMischnutzungMitWohnen' analog angepasst. Das im Programm 6.2 definierte, und von den Objekten im Gebiet Hildesheim nicht überschrittene, Maximum von 7500 m^2 wird auf 60 000 m^2 im Programm 6.6 erhöht. Damit werden die flächenmäßig deutlich größeren Objektgeometrien der Klasse 'GuFMischnutzungMitWohnen' im Gebiet Hannover entsprechend berücksichtigt. Einige Objektgeometrien im Gebiet Hannover sind zudem komplexer als die des Gebiets Hildesheim, wie das Beispiel in Abbildung 6.25 zeigt. Deshalb wird der Wert für die maximale Kompaktheit der Objekte von 3 auf 4 erhöht.

Gebiet Ostercappeln

Auch für das Gebiet Ostercappeln erzielen die für das Gebiet Hildesheim aufgestellten Integritätsbedingungen vergleichare Ergebnisse, wobei jedoch für drei Objektarten einzelne Bedingungen angepasst werden müssen.

Für die beiden Objektarten 'Grünland' und 'GuFWohnen' werden die maximalen Flächen erhöht, da diese im Gebiet Ostercappeln über den Maxima der Gebiete Hildesheim und Hannover liegen. Das Maximum der Fläche für die Objektart 'Grünland' wird im Programm 6.7 auf 900 000 m^2 und das Maximum der Fläche für die Objektart 'GuFWohnen' auf 400 000 m^2 erhöht. Außerdem wird für die Objektart 'Gartenland' der maximale Wert für die Elongation des Mer geringfügig von 14 auf den Wert 15 erhöht.

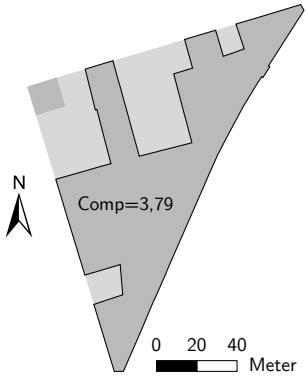


Abbildung 6.25: Beispiel für ein Objekt der Objektart 'GuFMischnutzungMitWohnen' mit einer Kompaktheit größer 3

```
context ALKNutzung
  inv AreaLGla: areaIsValid('Grünland', 300, 900000) -- Maximum 900000 anstatt 150000 Quadratmeter
  inv AreaLWhn: areaIsValid('GuFWohnen', 70, 400000) -- Maximum 400000 anstatt 50000 Quadratmeter

  inv ElongLGar: compIsValid('Gartenland', 15) -- [], Maximum 15 anstatt 14
```

Programm 6.7: Geänderte Integritätsbedingungen für die Fläche aufgrund des Gebiets Ostercappeln

6.3.3 Datenbereinigung

Da die geometrischen Integritätsbedingungen anhand der Gebiete nun vollständig und endgültig aufgestellt sind, können die entsprechenden Datensätze der ALK bereinigt werden. Objekte die eine oder mehrere der aufgestellten Integritätsbedingungen verletzen, werden dabei in dieser Arbeit als fehlerhaft bezeichnet. Diese sind im Sinne der Bedingung fehlerhaft, müssen jedoch nach Überprüfung durch einen Experten nicht unbedingt auch einem Fehler in der Realität entsprechen, da sie beispielweise eine Ausnahme darstellen. Solche Objekte sind demnach zuerst einmal nur auffällig, da sie vom durch die Integritätsbedingungen beschriebenen Sollzustand abweichen.

Die im Folgenden vorgestellten prozentualen Fehler in den Daten sowie die exemplarisch dargestellten fehlerhaften Objekte zeigen auf, dass mit Hilfe der aufgestellten Integritätsbedingungen erfolgreich fehlerhafte Objekte identifizieren werden können. Da die fehlerhaften Objekte nicht die Ergebnisse der Klassifikation in Abschnitt 6.4 negativ beeinträchtigen sollen, werden diese aus den Daten entfernt und somit ein bereinigter Datensatz bereitgestellt. Alternativ können einzelne Objekte auch als Ausnahmen deklariert oder korrigiert werden.

Gebiet Hildesheim

Im Gebiet Hildesheim werden durch die Integritätsbedingungen insgesamt 59 der 3854 Objekte aus dem Datensatz entfernt, was einem prozentualen Fehler von 1,5 % entspricht. Auch wenn die Anzahl der Fehler nur gering ist, so besteht die wichtige Aufgabe gerade darin, mit Hilfe der aufgestellten Integritätsbedingungen genau die wenigen fehlerhaften Objekte in einer großen Anzahl an Objekten zu identifizieren. In Tabelle 6.9 wird die Anzahl der Objekte für jede Objektart vor und nach der Datenbereinigung gegenübergestellt. Die meisten Fehler weist dabei die Objektart 'Ackerland' auf, gefolgt von der Objektart 'GuFWohnen'. Dagegen weist die Objektart 'GuFMischnutzungMitWohnen' keine fehlerhaften Objekte auf.

Am Beispiel der Objektart 'Ackerland' werden im Folgenden einige Eigenschaften der fehlerhaften Objekte diskutiert. Von den 59 fehlerhaften Objekten verletzen 12 die Integritätsbedingungen an die Fläche, 8 die der Kompaktheit, 11 die der fraktalen Dimension, 16 die des minimalen Durchmessers, 9 die der Elongation des MER und 1 Objekt die Integritätsbedingung der Flächendifferenz zum MER. Insgesamt identifiziert die Prüfung der Datenintegrität also 57 Fehler bei 19 Objekten. Eine Analyse der Verteilung der Anzahl der Fehler ergibt, dass 7 Objekte jeweils 1 Fehler aufweisen, 5 Objekte jeweils 3 Fehler und 7 Objekte jeweils 5 Fehler.

Objektart ALK	Anzahl Original	Anzahl Bereinigt	Fehler Total	Fehler [%]
Ackerland	219	200	19	8,7
Gartenland	170	168	2	1,2
Grünanlage	491	488	3	0,6
Grünland	128	122	6	4,7
GuFGewerbeundIndustrie	270	267	3	1,1
GuFHandelDienstleistungen	292	291	1	0,3
GuFMischnutzungMitWohnen	211	211	0	0,0
GuFÖffentlicheZwecke	202	199	3	1,5
GuFVersorgungsanlagen	207	203	4	1,9
GuFWohnen	1664	1646	18	1,1

Tabelle 6.9: Bereinigte Objekte und Objektarten in der ALK für das Gebiet Hildesheim

Die 19 als Fehler identifizierten Objekte lassen sich vier Gruppen zuordnen, für die exemplarische Beispiele in den Abbildungen 6.26 und 6.27 dargestellt sind[4]. Die erste Gruppe umfasst 12 Objekte und ist charakterisiert durch kleine Flächen, die als direkte Nachbarn größere Objekte der Objektart 'Ackerland' aufweisen (siehe Beispiel 1). Die Geometrien der fehlerhaften Objekte weisen zudem häufig eine hohe Kompaktheit, fraktale Dimension und Elongation des MER auf. Die zweite Gruppe umfasst 2 Objekte und ist charakterisiert durch Objekte mit vergleichsweise kleinen Flächen, die zwischen Objekten anderer Objektarten liegen (siehe Beispiel 2). Die dritte Gruppe, der 2 Objekte angehören, ist charakterisiert durch eine hohe Kompaktheit und direkte Nachbarn der Objektart 'Ackerland' (siehe Beispiel 3). Eine mögliche Hypothese für die Objekte der dritten Gruppe wäre, dass ehemalige Feldwege wieder in Ackerland umgewandelt wurden. Die vierte und letzte Gruppe besteht aus 3 Objekten, deren Fläche zwar größer als die minimal geforderten 500 m^2 ist, die jedoch einen minimalen Durchmesser von weniger als 20 m aufweisen und damit für Ackerflächen auffällig schmal sind.

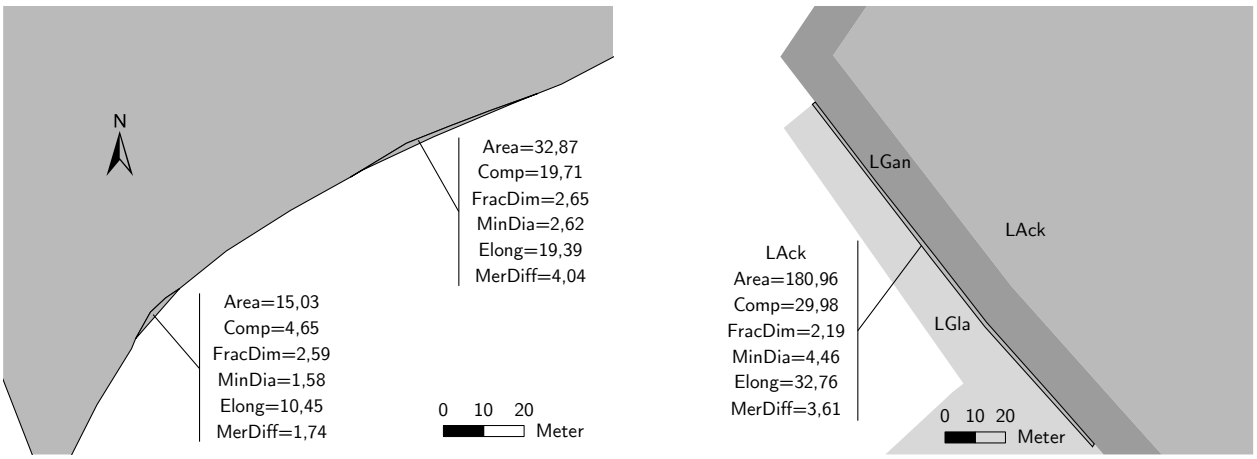


Abbildung 6.26: Beispiele 1 und 2 für fehlerhafte Objekte der Objektart 'Ackerland' im Gebiet Hildesheim

Für die 18 fehlerhaften Objekte der Objektart 'GUFWohnen' sind in Abbildung 6.28 einige exemplarische Beispiele dargestellt[5]. Alle Objektgeometrien in den Beispielen weisen eine zu kleine Fläche und eine auffällige Form auf. Es bleibt fraglich, welcher Prozess zur Entstehung dieser Artefakte geführt hat.

[4] Die verletzten Integritätsbedingungen der Beispiele in den Abbildungen 6.26 und 6.27 sind für die Objektart 'Ackerland' Area >= 500, Comp <= 7, FracDim <= 1.8, MinDia >= 20, Elong <= 13 und MerDiff <= 6.

[5] Die verletzten Integritätsbedingungen der Beispiele in Abbildung 6.28 sind für die Objektart 'GuFWohnen' Area >= 70, FracDim <= 1.9 und MinDia >= 5.

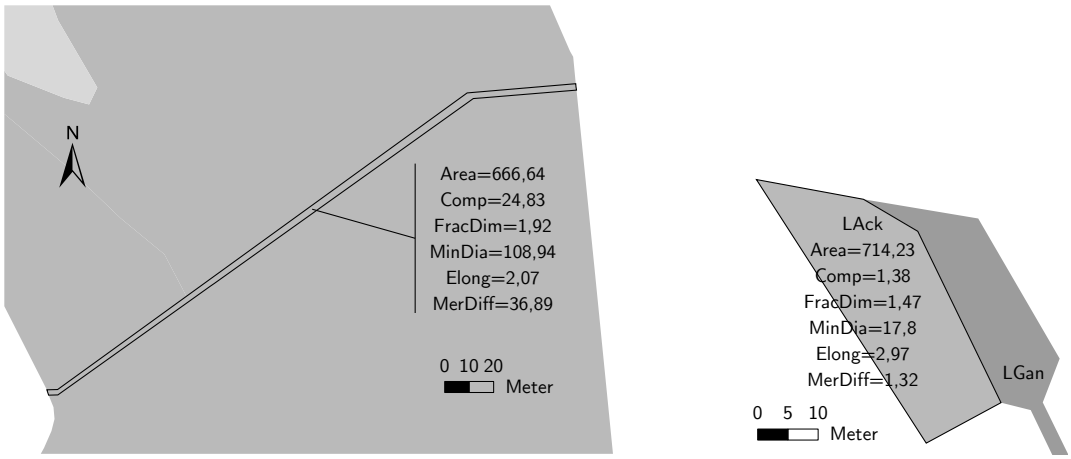


Abbildung 6.27: Beispiele 3 und 4 für fehlerhafte Objekte der Objektart 'Ackerland' im Gebiet Hildesheim

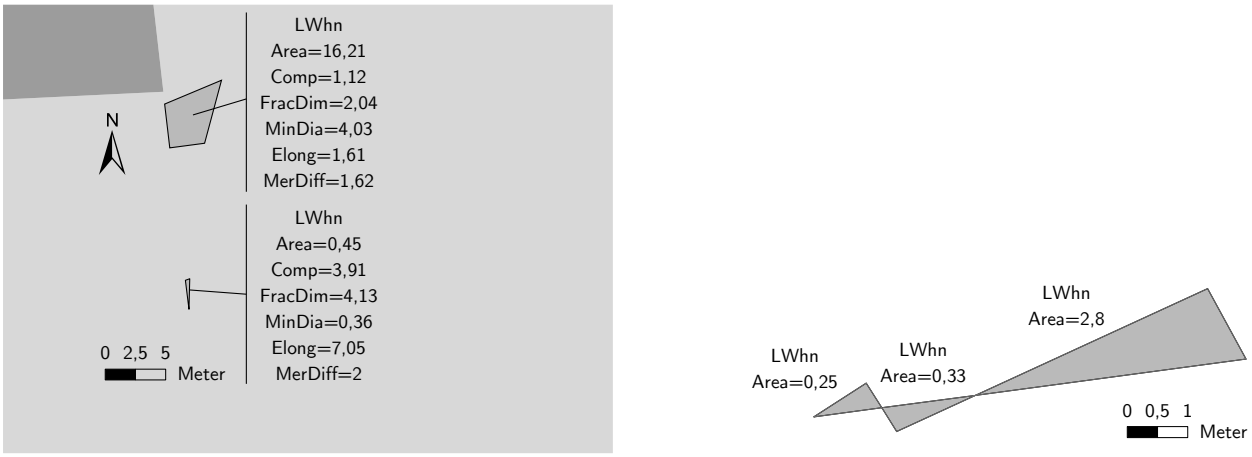


Abbildung 6.28: Beispiele 1 und 2 für fehlerhafte Objekte der Objektart 'GuFWohnen' im Gebiet Hildesheim

Gebiet Hannover

Im Gebiet Hannover werden durch die Integritätsbedingungen insgesamt 32 der 2557 Objekte als fehlerhaft identifiziert, was einem prozentualen Fehler von 1,3 % entspricht. Die Verteilung der fehlerhaften Objekte auf die Objektarten ist in Tabelle 6.10 aufgelistet. Die Objektart 'GuFMischnutzungMitWohnen' weist dabei die meisten fehlerhaften Objekte auf.

Am Beispiel der Objektart 'GuFGewerbeundIndustrie' werden im Folgenden wieder einige Eigenschaften der als fehlerhaft identifizierten Objekte diskutiert[6]. Das erste Beispiel in Abbildung 6.29 zeigt eine Objektgeometrie, deren Elongation größer als der maximal zugelassene Wert von 6 ist. Im gesamten Datensatz ist dies das einzige Objekt, das die maximale Elongation überschreitet. Das Objekt ist ein gutes Beispiel für auffällige Objekte, die zwar als fehlerhaft identifiziert werden, aber nach Begutachtung durch einen Experten als Ausnahme gekennzeichnet werden können. Das zweite Beispiel in Abbildung 6.29 zeigt ein Objekt, das die minimale Fläche von 130 m^2 unterschreitet. Da die Objektgeometrie ein Kreis ist, zeigt sie charakteristische Werte für die Kompaktheit (0,79), die Elongation des MER (1) und die Flächendifferenz zum MER (1,27). Das dritte Beispiel für die Objektart 'GuFGewerbeundIndustrie' in Abbildung 6.30 demonstriert die gleichzeitige Verletzung von insgesamt drei Integritätsbedingungen durch das hervorgehobene Objekt. Die Geometrie verletzt die Anforderungen an die Fläche, die fraktale Dimension und den minimalen Durchmesser des MER. Die zwei Geometrien im vierten Beispiel in Abbildung 6.30 zeigen wiederum auffällige Geometrien beziehungsweise Artefakte, die in diesem Fall sogar vier Bedingungen gleichzeitig verletzen.

[6] Die verletzten Integritätsbedingungen der Beispiele in den Abbildungen 6.29 und 6.30 sind für die Objektart 'GuFGewerbeundIndustrie' Area >= 130, FracDim <= 1.8, MinDia >= 5 und Elong <= 6.

Objektart ALK	Anzahl Original	Anzahl Bereinigt	Fehler Total	Fehler [%]
Ackerland	0	0	0	0,0
Gartenland	9	9	0	0,0
Grünanlage	163	163	0	0,0
Grünland	1	1	0	0,0
GuFGewerbeundIndustrie	162	155	7	4,3
GuFHandelDienstleistungen	245	241	4	1,6
GuFMischnutzungMitWohnen	878	868	10	1,1
GuFÖffentlicheZwecke	245	241	4	1,6
GuFVersorgungsanlagen	45	44	1	2,2
GuFWohnen	809	803	6	0,7

Tabelle 6.10: Bereinigte Objekte und Objektarten in der ALK für das Gebiet Hannover

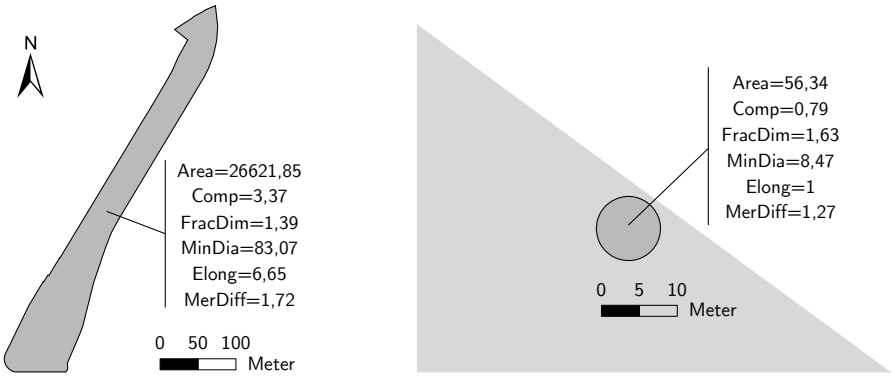


Abbildung 6.29: Beispiele 1 und 2 für fehlerhafte Objekte der Objektart 'GuFGewerbeundIndustrie' im Gebiet Hannover

Gebiet Ostercappeln

Im Gebiet Ostercappeln werden durch die Integritätsbedingungen von den 10 217 Objekten insgesamt 196 als fehlerhaft identifiziert, was einem prozentualen Fehler von 1,9 % entspricht. Mit einem prozentualen Fehler von 1,5 % im Gebiet Hildesheim und einem prozentualen Fehler von 1,3 % im Gebiet Hannover bewegt sich der Anteil der fehlerhaften Objekte der ALK ungefähr auf gleichem Niveau. Die Verteilung der Fehler auf die Objektarten ist in Tabelle 6.11 aufgelistet.

Für das Gebiet Ostercappeln werden im Folgenden wieder einige Eigenschaften der als fehlerhaft identifizierten Objekte diskutiert. Die folgenden vier Beispiele zeigen Objektgeometrien der Objektart 'Grünland', welche die größte Anzahl an Fehlern aufweist[7]. Im ersten Beispiel in Abbildung 6.31 verletzt das Objekt die Integritätsbedingungen an die minimale Fläche, an die maximale fraktale Dimension und an den minimalen Durchmesser des MER. Das zweite Beispiel in der Abbildung zeigt ein Objekt mit einer auffällig hohen Kompaktheit, die den Grenzwert von 8 weit überschreitet. Das Objekt ist wiederum ein gutes Beispiel für ein zwar auffälliges, aber nicht unbedingt fehlerhaftes Objekt, das von einem Experten als Ausnahme gekennzeichnet werden kann. Die beiden Objekte im dritten Beispiel in Abbildung 6.32 zeigen auffällige Geometrien beziehungsweise Artefakte, die durch geringe Flächen und eine hohe fraktale Dimension gekennzeichnet sind. Das vierte Beispiel zeigt wiederum ein auffälliges aber nicht unbedingt fehlerhaftes Objekt mit einer vergleichsweise hohen Kompaktheit.

6.3.4 Topologische Maße

Die Werte und Verteilungen der topologischen Maße werden im Folgenden basierend auf ihrer Definition in Unterabschnitt 6.2.2 analysiert. Die insgesamt 16 628 Objekte der drei Gebiete werden zusammen analysiert,

[7] Die verletzten Integritätsbedingungen der Beispiele in den Abbildungen 6.31 und 6.32 sind für die Objektart 'Grünland' Area >= 300, Comp <= 8, FracDim <= 1.7 und MinDia >= 8.

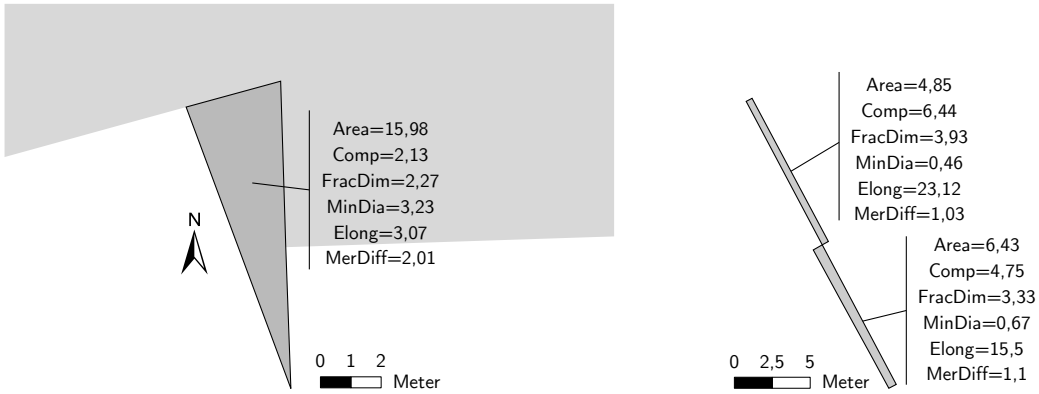


Abbildung 6.30: Beispiele 3 und 4 für fehlerhafte Objekte der Objektart 'GuFGewerbeundIndustrie' im Gebiet Hannover

	Anzahl		Fehler	
Objektart ALK	Original	Bereinigt	Total	[%]
Ackerland	3110	3052	58	1,9
Gartenland	272	268	4	1,5
Grünanlage	243	241	2	0,8
Grünland	3194	3126	68	2,1
GuFGewerbeundIndustrie	322	296	26	8,1
GuFHandelDienstleistungen	251	248	3	1,2
GuFMischnutzungMitWohnen	5	4	1	20,0
GuFÖffentlicheZwecke	120	115	5	4,2
GuFVersorgungsanlagen	139	130	9	6,5
GuFWohnen	2561	2541	20	0,8

Tabelle 6.11: Bereinigte Objekte und Objektarten in der ALK für das Gebiet Ostercappeln

da die Nachbarschaft der Objekte als prinzipiell korrekt eingestuft wird. Das Ziel der Analyse besteht darin, in den Daten seltene und häufige topologische Zusammenhänge der Objekte aufzudecken, um so möglichst allgemeingültige Aussagen für die zehn ausgewählten Objektarten der ALK aufstellen zu können.

Für die Analyse der Daten werden, wie beispielsweise in Abbildung 6.33, Box-Plots verwendet, die sich jedoch von den bisherigen Box-Plots in zwei Merkmalen unterscheiden. Im Unterschied zu den bisherigen Box-Plots, die jeweils nur die Verteilung eines einzelnen Attributs zeigen, zeigt dieser Box-Plot die Verteilung von insgesamt zehn Attributen, entsprechend der Gesamtanzahl der untersuchten Objektarten. Zudem sind die Verteilungen für die drei Datensätze durch verschiedene Farben gekennzeichnet. Die Daten des Gebiets Hildesheim sind in schwarz, die des Gebiets Hannover in blau und die des Gebiets Ostercappeln in orange gekennzeichnet.

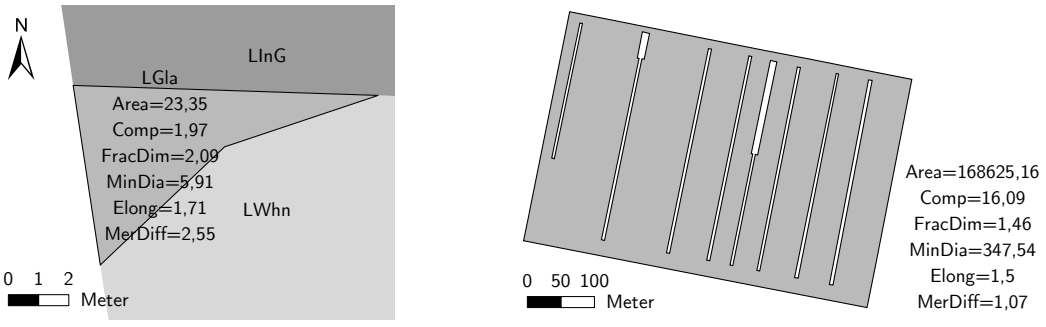


Abbildung 6.31: Beispiele 1 und 2 für fehlerhafte Objekte der Objektart 'Grünland' im Gebiet Ostercappeln

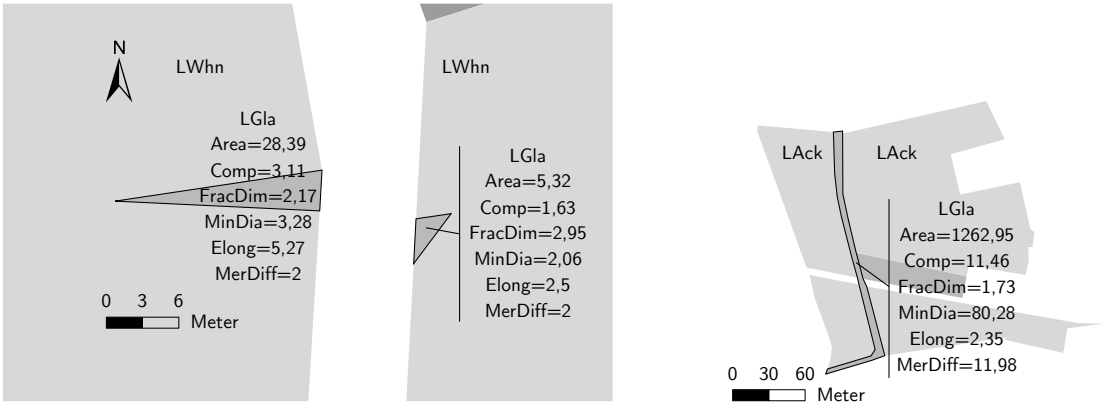


Abbildung 6.32: Beispiele 3 und 4 für fehlerhafte Objekte der Objektart 'Grünland' im Gebiet Ostercappeln

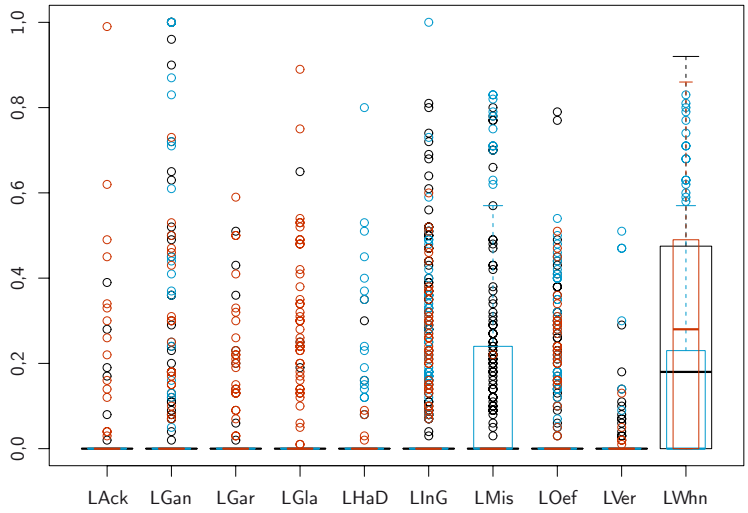


Abbildung 6.33: Box-Plots der Anteile der Objektarten am Umring der Objekte mit der Objektart 'GuFHandelDienstleistung'

Direkte Nachbarschaft

Aus der direkten Nachbarschaft der Objekte kann abgeleitet werden, ob bestimmte Objektarten bevorzugt benachbart sind oder niemals miteinander benachbart sind. Zudem kann mit der direkten Nachbarschaft ermittelt werden, ob ein Objekt nur von Objekten einer einzigen Objektart umschlossen ist. Die direkte Nachbarschaft eines Objekts wird durch die prozentualen Anteile der anderen Objektarten an der Gesamtlänge des Umrings des Objekts ausgedrückt. Zusätzlich wird die dominante Objektart bestimmt, wobei diese mindestens 20 % des Umrings ausmachen muss. Die Integritätsbedingungen an die direkte Nachbarschaft werden anhand zweier Objektarten exemplarisch untersucht. Die Objektart 'GuFHandelDienstleistung' umfasst für die drei Gebiete jeweils ähnlich viele Objekte und erleichtert damit die Interpretation der Diagramme der explorativen Datenanalyse. Die Objektart 'GuFWohnen' umfasst in allen Datensätze eine hohe Anzahl an Objekten und eignet sich damit insbesondere für die Interpretation der Übertragbarkeit der Ergebnisse.

Die Anteile der anderen Objektarten am Umring der Objekte mit der Objektart 'GuFHandelDienstleistung' sind in Abbildung 6.33 dargestellt. Aus der Abbildung ist ersichtlich, dass in allen Gebieten die Objekte der Objektart 'GuFWohnen' den größten Anteil am Umring der Objekte der 'GuFHandelDienstleistung' aufweisen. Außerdem lässt sich keine der Objektarten von der direkten Nachbarschaft ausschließen, da alle prozentualen Anteile am Umring der Objekte größer Null sind. Für die drei Objektarten 'Ackerland', 'Grünanlage' und 'GuFGewerbeundIndustrie' sind die prozentualen Anteile am Umring der Objekte maximal mit Werten bis zu 100 %. In Abbildung 6.34 ist diejenige Objektgeometrie der Objektart 'GuFHandelDienstleistung' dargestellt, für welche die Objektart 'Ackerland' den größten prozentualen Anteil am Umring aufweist, nämlich 99 %. Die Geometrie ist somit nahezu vollständig von Objekten der Objektart 'Ackerland' umschlossen. Die zweite Geometrie in Abbildung 6.34 zeigt ein Objekt, das auf drei Seiten von Objekten der Objektart 'GuFWohnen' flankiert wird, die zusammen 80 % des Umrings ausmachen.

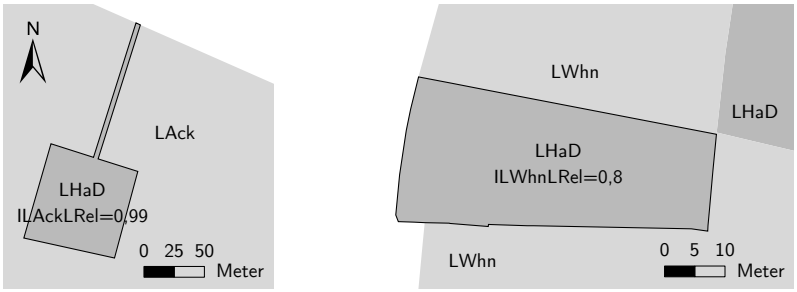


Abbildung 6.34: Beispiele für direkte Nachbarn der Objektart 'GuFHandelDienstleistung' im Gebiet Ostercappeln

Basierend auf den Box-Plots können als Integritätsbedingungen Elemente des 5-Zahlenmaßes festgelegt werden, beispielsweise der maximal zulässige Wert oder ein minimaler Wert für den Median. Dadurch können bestimmte Objektarten von der direkten Nachbarschaft ausgeschlossen oder deren Einfluss beschränkt werden. In den Integritätsbedingungen werden die prozentualen Anteile der Einfachheit halber auf Vielfache von 10 % festgelegt.

Für die Objektart 'GuFHandelDienstleistung' werden basierend auf den Box-Plots in Abbildung 6.33 die maximalen prozentualen Anteile an der Gesamtlänge des Umrings für die Objektarten 'Gartenland' und 'GuF-Versorgungsanlagen' auf 60 %, für die Objektart 'GuFÖffentlicheZwecke' auf 80 % und für die Objektarten 'Grünland', 'GuFHandelDienstleistung' und 'GuFMischnutzungMitWohnen' auf 90 % festgelegt. Außerdem wird für die Objektart 'GuFWohnen' das Minimum des oberen hinge auf 20 % festgelegt.

Für die Objekte der Objektart 'GuFWohnen' sind die Anteile der anderen Objektarten am Umring in Abbildung 6.35 dargestellt. Im Gegensatz zu den Box-Plots der Objektart 'GuFHandelDienstleistung' sind deutlich mehr Werte in der Abbildung enthalten, was auf die hohe Anzahl der Objekte mit der Objektart 'GuFWohnen' zurückzuführen ist. Zudem sind wiederum die prozentualen Anteile am Umring der Objekte für mehrere Objektarten maximal mit Werten bis zu 100 %. Der Box-Plot zeigt auch deutlich die Unterschiede zwischen den Gebieten. Während für das Gebiet Hildesheim die Objektart 'Grünanlage' der wichtigste direkte Nachbar der Objektart 'GuFWohnen' ist, so ist für das Gebiet Hannover der wichtigste direkte Nachbar die Objektart 'GuF-MischnutzungMitWohnen'. Für das Gebiet Ostercappeln sind dagegen die beiden Objektarten 'Ackerland' und 'Grünland' die wichtigsten direkten Nachbarn, was die ländliche Prägung dieses Gebiets unterstreicht. Basierend auf den Box-Plots werden die maximalen prozentualen Anteile für die Objektart 'GuFMischnutzungMitWohnen' auf 90 % und für die Objektart 'GuFVersorgungsanlagen' auf 60 % festgelegt

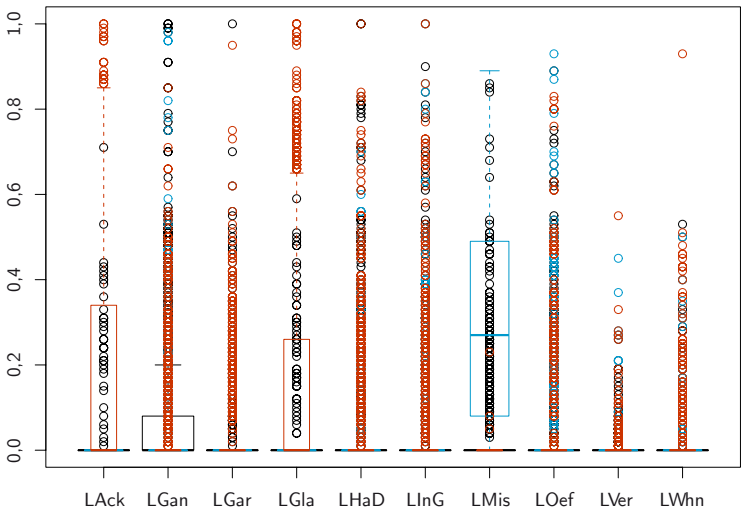


Abbildung 6.35: Box-Plots der Anteile der Objektarten am Umring der Objekte mit der Objektart 'GuFWohnen'

Aus den erstellten Box-Plots lässt sich nicht nur direkt ablesen, welche Objektarten mit welchen anderen Objektarten direkt benachbart sind, sondern auch ob Objekte derselben Objektart direkt benachbart sind. Soll dieser Wert auf Null reduziert werden, dann müssen lediglich die Geometrien aller direkt benachbarten Objekte derselben Objektart miteinander verschmolzen werden.

Integritätsbedingungen

Die Integritätsbedingungen für die direkte Nachbarschaft sind im Programm 6.8 zusammengefasst. Diese folgen aus der vorgestellten Analyse der Box-Plots für jede Objektart. Dabei wird angenommen, dass die Werte für die topologischen Beziehungen als zusätzliche Attribute der Klassen vorhanden sind. Alternativ könnten die Integritätsbedingungen auch vollständig in der GeoOcl definiert werden, jedoch sind die Bedingungen dann aufgrund ihrer Komplexität nur schwer nachvollziehbar. In Programm 6.9 ist exemplarisch die Integritätsbedingung ILRelWhn aus dem Programm 6.8 in der alternativen Variante definiert.

Die Objektarten 'Grünland' und 'GuFMischnutzungMitWohnen' sind dabei niemals direkt benachbart, wobei in diesem Fall die Beziehung symmetrisch ist. Zudem sind Objekte der Objektart 'GuFVersorgungsanlagen' niemals mit einem Objekt derselben Objektart direkt benachbart.

```
context ALKNutzung
  def: fiveNumLWhn(objart: String): FiveNumberSummary = ALKNutzung.allInstances()->select(objektart
       = objart)->collect(LWhn)->fiveNumberSummary()

  inv ILRelLHaD: objektart = 'GuFHandelUndDienstleistungen' implies LGar <= 0.6 and LGla <= 0.9 and
       LHaD <= 0.9 and LMis <= 0.9 and LOef <= 0.8 and LVer <= 0.6 and fiveNumLWhn('
      GuFHandelUndDienstleistungen').upperHinge() >= 0.2
  inv ILRelWhn: objektart = 'GuFWohnen' implies LMis <= 0.9 and LVer <= 0.6

  inv ILRelLAck: objektart = 'Ackerland' implies LGan <= 0.7 and LGar <= 0.6 LHaD <= 0.4 and LInG
      <= 0.7 and LMis <= 0.2 and LOef <= 0.5 and LVer <= 0.4
  inv ILRelLGan: objektart = 'Grünanlage' implies LGan <= 0.7 and LGar <= 0.8 and LGla <= 0.8 and
      LHaD <= 0.6 and LInG <= 0.9 and LMis <= 0.5 and LOef <= 0.8 and LVer <= 0.6
  inv ILRelLGar: objektart = 'Gartenland' implies LAck <= 0.9 and LGla <= 0.9 and LHaD <= 0.7 and
      LInG <= 0.6 and LMis <= 0.6 and LOef <= 0.7 and LVer <= 0.3 and fiveNumLWhn('Gartenland').
      upperHinge() >= 0.3 and fiveNumLWhn('Gartenland').median() >= 0.1
  inv ILRelGla: objektart = 'Grünland' implies LGan <= 0.9 and LGar <= 0.7 and LGla <= 0.9 and LHaD
       <= 0.6 and LInG <= 0.8 and LMis = 0 and LOef <= 0.5 and LVer <= 0.5 -- []
  inv ILRelLInG: objektart = 'GuFGewerbeUndIndustrie' implies LAck <= 0.9 and LGar <= 0.8 and LGla
      <= 0.7 and LInG <= 0.8 and LMis <= 0.9 and LVer <= 0.4 and fiveNumLWhn('
      GuFGewerbeUndIndustrie').upperHinge() >= 0.3
  inv ILRelLMis: objektart = 'GuFMischnutzungMitWohnen' implies LAck <= 0.8 and LGan <= 0.7 and
      LGar <= 0.5 and LGla = 0 and LHaD <= 0.9 and LMis <= 0.7 and LOef <= 0.9 and LVer <= 0.2 and
       fiveNumLWhn('GuFMischnutzungMitWohnen').upperHinge() >= 0.1
  inv ILRelLOef: objektart = 'GuFÖffentlicheZwecke' implies LAck <= 0.8 and LGar <= 0.5 and LGla <=
       0.9 and LHaD <= 0.9 and LInG <= 0.9 and LMis <= 0.9 and LOef <= 0.5 and LVer <= 0.3  and
      fiveNumLWhn('GuFÖffentlicheZwecke').upperHinge() >= 0.2 and fiveNumLWhn('
      GuFÖffentlicheZwecke').median() >= 0.1
  inv ILRelLVer: objektart = 'GuFVersorgungsanlagen' implies LGar <= 0.3 and LMis <= 0.8 and LOef
      <= 0.9 and LVer = 0 and fiveNumLWhn('GuFVersorgungsanlagen').upperHinge() >= 0.2
```

Programm 6.8: Integritätsbedingungen für die direkte Nachbarschaft der Objekte

```
context ALKNutzung
  inv ILRelWhn: objektart = 'GuFWohnen' implies ALKNutzung.allInstances()->select(o | o.objektart =
       'GuFMischnutzungMitWohnen' and o.geom.intersects(geom) and o.geom.intersection(geom).
      oclIsTypeOf(LineString))->collect(oc | oc.geom.intersection(geom).length())->sum() <= 0.9 *
      geom.exteriorRing().length() and ALKNutzung.allInstances()->select(o | o.objektart = '
      GuFVersorgungsanlagen' and o.geom.intersects(geom) and o.geom.intersection(geom).oclIsTypeOf
      (LineString))->collect(oc | oc.geom.intersection(geom).length())->sum() <= 0.6 * geom.
      exteriorRing().length()
```

Programm 6.9: Integritätsbedingungen für die direkte Nachbarschaft der Objekte

Dominante Objektart

Durch die Bestimmung der dominanten Objektart müssen für jedes einzelne Objekt nicht insgesamt zehn Attribute, entsprechend der Anzahl der untersuchten Objektarten der Alk, sondern nur ein einzelnes Attribut analysiert werden. Die Vereinfachung führt jedoch zwangsläufig zu einem Verlust an Information, weshalb das Maß an dieser Stelle nicht für die Aufstellung von Integritätsbedingungen verwendet wird. Jedoch ist das Maß eine wichtige Größe für die Klassifikation in Abschnitt 6.4.

Für jede Objektart lässt sich die Verteilung der dominanten Objektart bestimmen, die beispielsweise als Stabdiagramm dargestellt werden kann. Die Ergebnisse können jedoch auch in einer einzigen Tabelle für alle Objektarten

zusammengefasst werden. Die Tabelle 6.12 listet die dominanten Objektarten der ALK für das Gebiet Hildesheim auf, wobei nur Werte $> 5\,\%$ aufgelistet sind. Eine vollständige Auflistung ist in Tabelle A.3 zu finden. Am Beispiel der Objektart 'Grünland' wird die Interpretation der Werte der Tabelle im Folgenden diskutiert. Für 27 % der Objekte ist die dominante Objektart 'Ackerland', d.h. diese Objektart hat den größten Anteil am Umring der Objekte. Danach folgen die Objektarten 'GuFWohnen' mit 16 % und gleichauf mit 8 % die beiden Objektarten 'Grünanlage' und 'Gartenland'. Damit sind die häufigsten direkten Nachbarn eindeutig bestimmt. Die hohen Werte in der Spalte 'None' sind darauf zurückzuführen, dass viele Objekte zu einem hohen Prozentsatz oder sogar vollständig von Straßen umgeben sind. Zudem sind die Werte in der Spalte 'LWhn' im Vergleich zu den anderen Spalten deutlich höher, was auf die hohe Anzahl der Objekte mit der Objektart 'GuFWohnen' zurückzuführen ist. An dieser Stelle sei auch explizit darauf hingewiesen, dass die Tabelle 6.12 nicht als Matrix verstanden werden darf, d.h. sie darf nur zeilenweise und nicht spaltenweise analysiert werden. Insbesondere sind die Werte nicht symmetrisch zwischen zwei Objektarten.

Objektart	LAck	LGan	LGar	LGla	LHaD	LInG	LMis	LOef	LVer	LWhn	None
LAck	12	11		8							59
LGan	5							5		42	33
LGar	9	15		6						44	12
LGla	27	8	8							16	37
LHaD		6				14	9	5		44	20
LInG		8			12					43	23
LMis					17	6				70	
LOef		16			5	5				39	31
LVer		14				14		7		33	23
LWhn		15			6	6		5			58

Tabelle 6.12: Dominante Objektart der direkten Nachbarschaft der Objekte für das Gebiet Hildesheim

Indirekte Nachbarschaft

Aus der indirekten Nachbarschaft der Objekte kann abgeleitet werden, welche Objektarten in der weiteren Umgebung um welche anderen Objektarten auftreten. So kann im Gegensatz zur direkten Nachbarschaft nicht nur eine Aussage über direkt anliegende Nachbarn getroffen werden, sondern in einer größeren räumlichen Nachbarschaft. Insbesondere eignet sich dieses Maß um Korridore, wie beispielsweise Straßen oder Gewässer, überbrücken zu können. Dadurch werden die Nachbarn jenseits der Korridore in der Analyse berücksichtigt. Die indirekte Nachbarschaft eines Objekts wird durch die prozentualen Anteile der anderen Objektarten an der um eine bestimmte Distanz erweiterten Geometrie ausgedrückt. Zusätzlich wird die dominante Objektart bestimmt, wobei diese mindestens 20 % der erweiterten Geometrie ausmachen muss. Die Integritätsbedingungen an die indirekte Nachbarschaft werden wiederum anhand von zwei Objektarten exemplarisch untersucht.

Die Objektarten der indirekten Nachbarn im 50 Meter Puffer um die Objekte der Objektart 'GuFHandelDienstleistung' sind in den Box-Plots in Abbildung 6.36 dargestellt. Im Vergleich zur entsprechenden Abbildung 6.33 für die direkten Nachbarn werden mehrere Unterschiede deutlich. Erstens weisen der obere hinge und der Median der Verteilungen für viele Objektarten deutlich höhere Werte auf. Zweitens sind die maximalen Werte meist kleiner als die der direkten Nachbarn. Die Integritätsbedingungen werden analog zu den Bedingungen der direkten Nachbarn in Vielfachen von 10 % ausgedrückt.

Für die Objektart 'GuFWohnen' sind die Objektarten der indirekten Nachbarn im 50 Meter Puffer um die Objekte in Abbildung 6.37 abgebildet. Auch hier werden die Abweichungen von der entsprechenden Abbildung 6.35 für die direkten Nachbarn deutlich. Neben den bereits genannten Unterschieden, die analog zur Objektart 'GuFHandelDienstleistung' sind, fällt der deutlich höhere Anteil der eigenen Objektart auf.

Integritätsbedingungen

Die Integritätsbedingungen für die indirekte Nachbarschaft sind im Programm 6.10 zusammengefasst. Diese folgen aus der vorgestellten Analyse der Box-Plots jeder Objektart. Dabei wird wiederum angenommen, dass die Werte für die topologischen Beziehungen als zusätzliche Attribute der Klassen vorhanden sind.

```
context ALKNutzung
  def: fiveNumL50LAck(objart: String): FiveNumberSummary = ALKNutzung.allInstances()->select(
      objektart = objart)->collect(L50LAck)->fiveNumberSummary()
  def: fiveNumL50LHaD(objart: String): FiveNumberSummary = ALKNutzung.allInstances()->select(
      objektart = objart)->collect(L50LHaD)->fiveNumberSummary()
  def: fiveNumL50LInG(objart: String): FiveNumberSummary = ALKNutzung.allInstances()->select(
      objektart = objart)->collect(L50LInG)->fiveNumberSummary()
  def: fiveNumL50LWhn(objart: String): FiveNumberSummary = ALKNutzung.allInstances()->select(
      objektart = objart)->collect(L50LWhn)->fiveNumberSummary()

  inv L50LHaD: objektart = 'GuFHandelUndDienstleistungen' implies LAck <= 0.9 and L50LGar <= 0.3
      and L50LGla <= 0.7 and L50LHaD <= 0.7 and L50LMis <= 0.6 and L50LOef <= 0.7 and L50LVer <=
      0.4 and fiveNumL50LHaD('GuFHandelUndDienstleistungen').upperHinge() >= 0.1 and
      fiveNumL50LWhn('GuFHandelUndDienstleistungen').upperHinge() >= 0.2
  inv L50Whn: objektart = 'GuFWohnen' implies L50LGar <= 0.7 and L50LHaD <= 0.8 and L50LMis <= 0.6
      and L50LOef <= 0.9 and L50LVer <= 0.5 and fiveNumL50LWhn('GuFWohnen').upperHinge() >= 0.3
      and fiveNumL50LWhn('GuFWohnen').median() >= 0.1

  inv L50LAck: objektart = 'Ackerland' implies L50LGan <= 0.8 and L50LGar <= 0.4 and L50LHaD <= 0.3
       and L50LInG <= 0.7 and L50LMis <= 0.2 and L50LOef <= 0.5 and L50LVer <= 0.5 and L50LWhn <=
      0.9 and fiveNumL50LAck('Ackerland').upperHinge() >= 0.4 and fiveNumL50LAck('Ackerland').
      median() >= 0.2
  inv L50LGan: objektart = 'Grünanlage' implies L50LAck <= 0.9 and L50LGan <= 0.7 and L50LGar <=
      0.6 and L50LGla <= 0.7 and L50LHaD <= 0.5 and L50LInG <= 0.9 and L50LMis <= 0.5 and L50LOef
      <= 0.7 and L50LVer <= 0.3
  inv L50LGar: objektart = 'Gartenland' implies L50LGar <= 0.5 and L50LGla <= 0.9 and L50LHaD <=
      0.4 and L50LInG <= 0.8 and L50LMis <= 0.4 and L50LOef <= 0.7 and L50LVer <= 0.1 and
      fiveNumL50LWhn('Gartenland').upperHinge() >= 0.3 and fiveNumL50LWhn('Gartenland').median()
      >= 0.1
  inv L50Gla: objektart = 'Grünland' implies L50LGan <= 0.6 and L50LGar <= 0.5 and L50LHaD <= 0.5
      and L50LInG <= 0.7 and L50LMis = 0 and L50LOef <= 0.6 and L50LVer <= 0.5 and L50LWhn <= 0.9
  inv L50LInG: objektart = 'GuFGewerbeUndIndustrie' implies L50LAck <= 0.9 and L50LGan <= 0.7 and
      L50LGar <= 0.3 and L50LGla <= 0.8 and L50LHaD <= 0.9 and L50LInG <= 0.8 and L50LMis <= 0.5
      and L50LOef <= 0.7 and L50LVer <= 0.4  and fiveNumL50LInG('GuFGewerbeUndIndustrie').
      upperHinge() >= 0.1 and fiveNumL50LWhn('GuFGewerbeUndIndustrie').upperHinge() >= 0.2 and
      fiveNumL50LWhn('GuFGewerbeUndIndustrie').median() >= 0.1
  inv L50LMis: objektart = 'GuFMischnutzungMitWohnen' implies L50LAck <= 0.7 and L50LGan <= 0.4 and
       L50LGar <= 0.2 and L50LGla <= 0.1 and L50LHaD <= 0.7 and L50LInG <= 0.6 and L50LMis <= 0.5
      and L50LOef <= 0.8 and L50LVer <= 0.2 and fiveNumL50LWhn('GuFMischnutzungMitWohnen').
      upperHinge() >= 0.4 and fiveNumL50LWhn('GuFMischnutzungMitWohnen').median() >= 0.1
  inv L50LOef: objektart = 'GuFÖffentlicheZwecke' implies L50LGar <= 0.4 and L50LGla <= 0.7 and
      L50LHaD <= 0.8 and L50LInG <= 0.8 and L50LMis <= 0.6 and L50LOef <= 0.7 and L50LVer <= 0.2
      and L50LWhn <= 0.9 and fiveNumL50LWhn('GuFÖffentlicheZwecke').upperHinge() >= 0.2 and
      fiveNumL50LWhn('GuFÖffentlicheZwecke').median() >= 0.1
  inv L50LVer: objektart = 'GuFVersorgungsanlagen' implies L50LGan <= 0.7 and L50LGar <= 0.3 and
      L50LGla <= 0.9 and L50LHaD <= 0.5 and L50LMis <= 0.5 and L50LOef <= 0.7 and L50LVer <= 0.2
      and fiveNumL50LWhn('GuFVersorgungsanlagen').upperHinge() >= 0.2
```

Programm 6.10: Integritätsbedingungen für die indirekte Nachbarschaft der Objekte

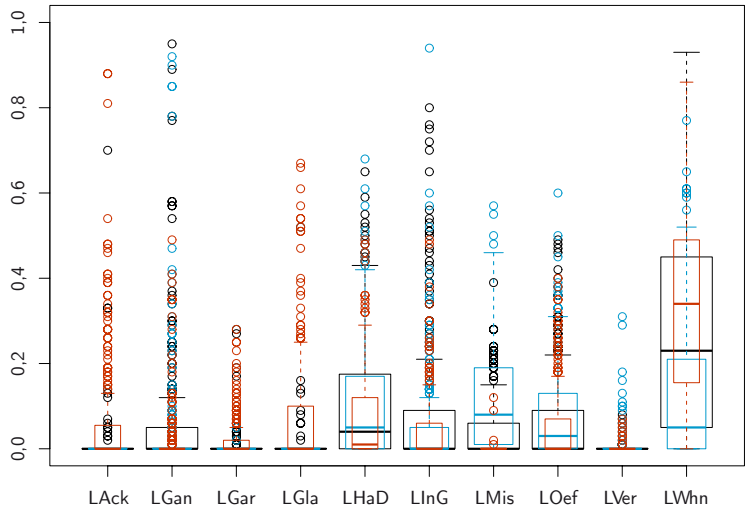


Abbildung 6.36: Box-Plots der Anteile der Objektarten im 50 Meter Puffer um die Objekte mit der Objektart 'GuFHandelDienstleistung'

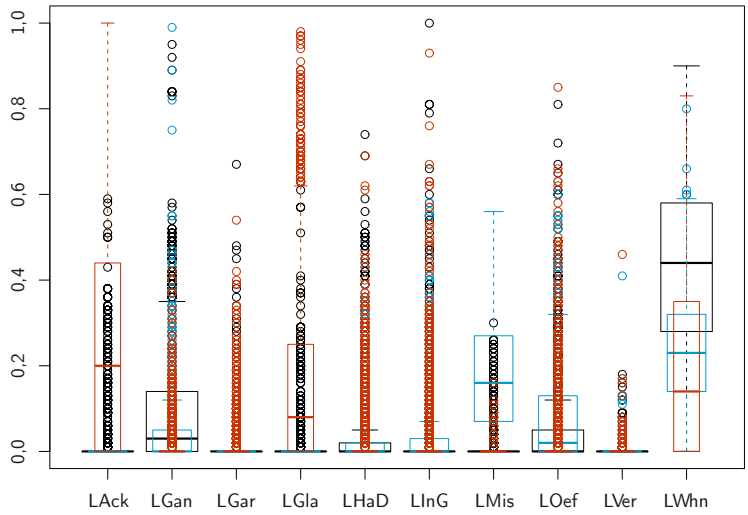


Abbildung 6.37: Box-Plots der Anteile der Objektarten im 50 Meter Puffer um die Objekte mit der Objektart 'GuFWohnen'

Dominante Objektart

Auch für die indirekten Nachbarn wird die dominante Objektart an dieser Stelle nicht für die Aufstellung von Integritätsbedingungen verwendet, sondern erst für die Klassifikation in Abschnitt 6.4.

Für eine Distanz beziehungsweise einen Puffer von 50 m sind die dominanten Objektarten für das Gebiet Hildesheim in Tabelle 6.13 aufgelistet, wobei nur Werte $> 5\,\%$ in der Tabelle enthalten sind. Eine vollständige Auflistung ist in Tabelle A.4 zu finden. Im Umkreis von 50 m um die Objekte der Objektart 'Grünland' ist mit 38 % deutlich die Objektart 'Ackerland' die häufigste dominante Objektart. Darauf folgen mit jeweils 18 % die beiden Objektarten 'Grünland' und 'GuFWohnen'. Im Gegensatz zur direkten Nachbarschaft ist nun der Anteil der restlichen Objektarten, die unter 'None' zusammengefasst sind, mit lediglich 14 % statt 37 % (siehe Tabelle 6.12) deutlich geringer. Dies ist unter anderem darauf zurückzuführen, dass nun Straßen 'übersprungen' werden, d.h. die Objekte auf der gegenüberliegenden Straßenseite eines Objekts werden bei der indirekten Nachbarschaft mit berücksichtigt.

Blöcke

Die Zugehörigkeit zu einem Block ist die interessanteste Nachbarschaft, da diese, im Gegensatz zur direkten und indirekten Nachbarschaft, auf einem abstrakteren Konzept der Zusammengehörigkeit basiert. Insbesondere ist die Zugehörigkeit eines Objekts zu einem Block auch für die Generalisierung von Interesse, da häufig bei kleineren Blöcken alle Objekte eines Blocks zu einem einzelnen Objekt mit nur einer Objektart verschmolzen werden. Wie

Objektart	LAck	LGan	LGar	LGla	LHaD	LInG	LMis	LOef	LVer	LWhn	None
LAck	67	9									11
LGan	10	10						6		56	10
LGar	15	16						5		49	
LGla	38	7		18						18	14
LHaD		6			17	11		7		49	8
LInG					14	23				46	8
LMis					11					76	
LOef		11			6			8		58	9
LVer		6				17		7		54	8
LWhn		6								76	

Tabelle 6.13: Dominante Objektart im 50 Meter Puffer um die Objekte für das Gebiet Hildesheim

bei der indirekten Nachbarschaft werden wieder die Flächenanteile der einzelnen Objektarten berechnet sowie die dominante Objektart bestimmt, wobei diese mindestens 40 % der Fläche des Blocks ausmachen muss. Die Integritätsbedingungen an die Flächenanteile der Objektarten an den Blöcken werden wieder exemplarisch anhand von zwei Objektarten untersucht.

Für die Objektart 'GuFHandelDienstleistung' sind die Box-Plots der Anteile der Objektarten an der Fläche der Blöcke in Abbildung 6.38 dargestellt. Deutlich ist die Dominanz der Objektarten 'GuFHandelDienstleistung' und 'GuFWohnen' erkennbar. Die maximalen Werte betragen für einige Objektarten an die 100 %, d.h. ein Objekt der Klasse 'GuFHandelDienstleistung' macht in einem Block nur einen minimalen Anteil der Fläche aus, wohingegen der Rest der Fläche durch eine andere Objektart belegt wird. Aus Abbildung 6.38 ist auch ersichtlich, dass die Objektart 'Gartenland' nur in geringen Flächenanteilen im selben Block enthalten ist wie Objekte der Objektart 'GuFHandelDienstleistung'. In den Integritätsbedingungen werden diese Anteile wieder durch Werte in Vielfachen von 10 % beschränkt.

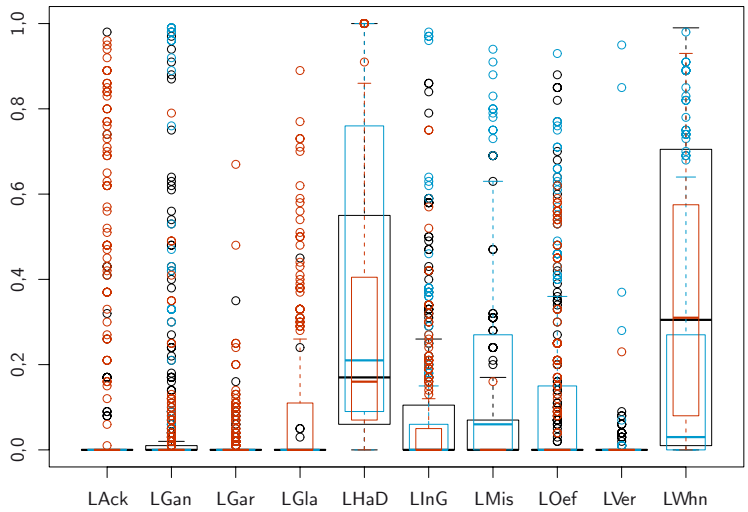


Abbildung 6.38: Box-Plots der Anteile der Objektarten an der Fläche der Blöcke der Objekte mit der Objektart 'GuFHandelDienstleistung'

Die hohe Anzahl der Objekte mit der Objektart 'GuFWohnen' zeigt sich wiederum in den Box-Plots in Abbildung 6.39. Hier ist die eigene Objektart deutlich mit den höchsten Flächenanteilen an den Blöcken der Objekte vertreten. Aus den Box-Plots kann auch abgelesen werden, dass Objekte der Objektart 'GuFWohnen' in verschieden zusammengesetzten Blöcken vertreten sind, die keine der anderen Objektarten ausschließen oder deutlich einschränken.

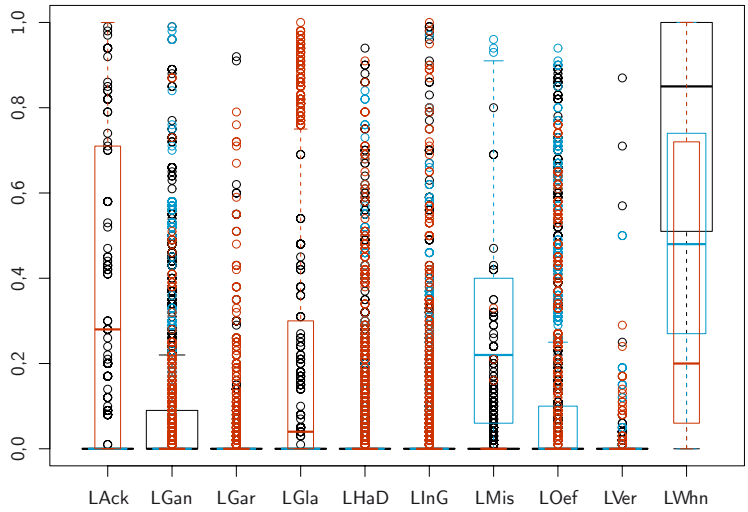


Abbildung 6.39: Box-Plots der Anteile der Objektarten an der Fläche der Blöcke der Objekte mit der Objektart 'GuF-Wohnen'

Integritätsbedingungen

Die Integritätsbedingungen für die Blöcke sind im Programm 6.11 zusammengefasst. Für jede Objektart werden dafür die Box-Plots analysiert. Dabei wird wiederum angenommen, dass die Werte für die topologischen Beziehungen als zusätzliche Attribute der Klassen vorhanden sind.

Dominante Objektart

Auch für die Blöcke wird die dominante Objektart an dieser Stelle nicht für die Aufstellung von Integritätsbedingungen verwendet, sondern stattdessen erst für die Klassifikation in Abschnitt 6.4.

Die vereinfachte Repräsentation der Flächenanteile der Blöcke durch die dominante Objektart sind für das Gebiet Hildesheim in Tabelle 6.14 für Werte $> 5\,\%$ sowie vollständig in Tabelle A.5 aufgelistet. Die Tabellen zeigen eine deutliche Diagonalstruktur, d.h. Objekte derselben Objektart sind häufig dominant für den gesamten Block.

Objektart	LAck	LGan	LGar	LGla	LHaD	LInG	LMis	LOef	LVer	LWhn	None
LAck	84										6
LGan	7	36						5		39	5
LGar	16	5	13	5						41	14
LGla	27	6		40						12	12
LHaD		5			31	6		5		41	8
LInG					8	39				40	7
LMis					11		5	5		68	6
LOef		9						45		29	7
LVer		7				16		6	20	42	
LWhn										80	

Tabelle 6.14: Dominante Objektart in den Blöcken der Objekte für das Gebiet Hildesheim

Zusammenhang zwischen den Objekten der ALK und ATKIS

Durch die Verschneidung der Objektgeometrien der ALK mit den Geometrien der ATKIS-Objekte kann der Zusammenhang zwischen den beiden Datensätzen bestimmt werden. Um eine eindeutige Zuordnung zu gewährleisten, wird für jedes Objekt der ALK jeweils nur die dominante Objektart aus ATKIS untersucht. Diese ist für das Gebiet Hildesheim in Tabelle 6.15 für Werte $> 5\,\%$ und vollständig in Tabelle A.6 aufgelistet. Für das Gebiet Hannover ist der Zusammenhang der Objekte entsprechend in Tabelle 6.16 für Werte $> 5\,\%$ und vollständig in Tabelle A.7 aufgelistet. Die Ergebnisse für die beiden Gebiete werden im Folgenden für alle Objektarten der

```
context ALKNutzung
  def: fiveNumBLARelLAck(objart: String): FiveNumberSummary = ALKNutzung.allInstances()->select(
      objektart = objart)->collect(BLARelLAck)->fiveNumberSummary()
  def: fiveNumBLARelLGan(objart: String): FiveNumberSummary = ALKNutzung.allInstances()->select(
      objektart = objart)->collect(BLARelLGan)->fiveNumberSummary()
  def: fiveNumBLARelLGar(objart: String): FiveNumberSummary = ALKNutzung.allInstances()->select(
      objektart = objart)->collect(BLARelLGar)->fiveNumberSummary()
  def: fiveNumBLARelLGla(objart: String): FiveNumberSummary = ALKNutzung.allInstances()->select(
      objektart = objart)->collect(BLARelLGla)->fiveNumberSummary()
  def: fiveNumBLARelLHaD(objart: String): FiveNumberSummary = ALKNutzung.allInstances()->select(
      objektart = objart)->collect(BLARelLHaD)->fiveNumberSummary()
  def: fiveNumBLARelLInG(objart: String): FiveNumberSummary = ALKNutzung.allInstances()->select(
      objektart = objart)->collect(BLARelLInG)->fiveNumberSummary()
  def: fiveNumBLARelLMis(objart: String): FiveNumberSummary = ALKNutzung.allInstances()->select(
      objektart = objart)->collect(BLARelLMis)->fiveNumberSummary()
  def: fiveNumBLARelLWhn(objart: String): FiveNumberSummary = ALKNutzung.allInstances()->select(
      objektart = objart)->collect(BLARelLWhn)->fiveNumberSummary()

  inv BLARelLHaD: objektart = 'GuFHandelUndDienstleistungen' implies BLARelLGar <= 0.7 and
      fiveNumBLARelLHaD('GuFHandelUndDienstleistungen').upperHinge() >= 0.4 and fiveNumBLARelLHaD(
      'GuFHandelUndDienstleistungen').median() >= 0.1 and  fiveNumBLARelLWhn('
      GuFHandelUndDienstleistungen').upperHinge() >= 0.2
  inv BLARelLWhn: objektart = 'GuFWohnen' implies BLARelLVer <= 0.9 and fiveNumBLARelLWhn('
      GuFWohnen').upperHinge() >= 0.7 and fiveNumBLARelLWhn('GuFWohnen').median() >= 0.1

  inv BLARelLAck: objektart = 'Ackerland' implies BLARelLHaD <= 0.3 and BLARelLMis <= 0.4 and
      BLARelLOef <= 0.7 and BLARelLVer <= 0.4  and fiveNumBLARelLAck('Ackerland').upperHinge() >=
      0.9 and fiveNumBLARelLAck('Ackerland').median() >= 0.7
  inv BLARelLGan: objektart = 'Grünanlage' implies BLARelLGar <= 0.7 BLARelLMis <= 0.6 and
      BLARelLVer <= 0.6 and fiveNumBLARelLGan('Grünanlage').upperHinge() >= 0.2 and
      fiveNumBLARelLWhn('Grünanlage').upperHinge() >= 0.4
  inv BLARelLGar: objektart = 'Gartenland' implies BLARelLHaD <= 0.8 and BLARelLInG <= 0.9 and
      BLARelLMis <= 0.4 and BLARelLOef <= 0.9 and BLARelLVer <= 0.2 and fiveNumBLARelLGar('
      Gartenland').upperHinge() >= 0.1 and fiveNumBLARelLWhn('Gartenland').upperHinge() >= 0.3
  inv BLARelGla: objektart = 'Grünland' implies BLARelLGan <= 0.7 and BLARelLGar <= 0.8 and
      BLARelLHaD <= 0.7 and BLARelLIng <= 0.8 and BLARelLMis = 0 and BLARelLOef <= 0.7 and
      BLARelLVer <= 0.4 and fiveNumBLARelLGla('Grünland').upperHinge() >= 0.8 and
      fiveNumBLARelLGla('Grünland').median() >= 0.2
  inv BLARelLInG: objektart = 'GuFGewerbeUndIndustrie' implies BLARelLGar <= 0.6 and BLARelLVer <=
      0.9 and fiveNumBLARelLInG('GuFGewerbeUndIndustrie').upperHinge() >= 0.4 and
      fiveNumBLARelLInG('GuFGewerbeUndIndustrie').median() >= 0.1 and fiveNumBLARelLWhn('
      GuFGewerbeUndIndustrie').upperHinge() >= 0.3 and fiveNumBLARelLWhn('GuFGewerbeUndIndustrie')
      .median() >= 0.1
  inv BLARelLMis: objektart = 'GuFMischnutzungMitWohnen' implies BLARelLGar <= 0.3 and BLARelLGla
      <= 0.7 and BLARelLVer <= 0.6  and fiveNumBLARelLWhn('GuFMischnutzungMitWohnen').upperHinge()
       >= 0.4 and fiveNumBLARelLWhn('GuFMischnutzungMitWohnen').median() >= 0.1
  inv BLARelLOef: objektart = 'GuFÖffentlicheZwecke' implies BLARelLGar <= 0.4 and BLARelLGla <=
      0.9 and BLARelLInG <= 0.9 and BLARelLMis <= 0.9 and BLARelLVer <= 0.2 and fiveNumBLARelLMis(
      'GuFÖffentlicheZwecke').upperHinge() >= 0.5 and fiveNumBLARelLMis('GuFÖffentlicheZwecke').
      median() >= 0.2 and fiveNumBLARelLWhn('GuFÖffentlicheZwecke').upperHinge() >= 0.2
  inv BLARelLVer: objektart = 'GuFVersorgungsanlagen' implies BLARelLGar <= 0.3 and BLARelLMis <=
      0.9 and fiveNumBLARelLWhn('GuFVersorgungsanlagen').upperHinge() >= 0.2
```

Programm 6.11: Integritätsbedingungen für die Blöcke der Objekte

ALK diskutiert. Dabei werden die Definitionen der jeweiligen Objektarten in der ALK und ATKIS berücksichtigt, die in Tabelle 6.17 angegeben sind.

Objektart	TAck	TBFP	TGan	TGar	TGla	TInG	TMis	TWhn	None
LAck	87				5				
LGan		9	10	13	7			43	10
LGar	5	8		13		10	10	45	6
LGla	13			9	47		12	6	7
LHaD						12	26	52	
LInG		8				36	14	39	
LMis							14	79	
LOef		41					9	43	
LVer		9				17		50	12
LWhn							7	86	

Tabelle 6.15: Zusammenhang zwischen den Objekten der ALK und ATKIS für das Gebiet Hildesheim

Objektart	TBFP	TGan	TGar	TGla	TInG	TMis	TWhn	None
LGan	9	25	8			24	23	8
LGar					33	44	22	
LGla				100				
LHaD	6					63	23	
LInG					18	56	21	
LMis						43	55	
LOef	44					30	22	
LVer	9	7			13	33	27	11
LWhn						42	55	

Tabelle 6.16: Zusammenhang zwischen den Objekten der ALK und ATKIS für das Gebiet Hannover

Ausgehend von der Definition der Objektarten in Tabelle 6.17 wäre zu erwarten, dass sich identisch oder ähnlich definierte Objektarten in der ALK und ATKIS entsprechen und sich dieser Zusammenhang auch in den Ergebnissen für die Objekte der Gebiete zeigt. Dies betrifft die identisch benannten Objektarten 'Ackerland', 'Gartenland', 'Grünanlage' und 'Grünland', die ähnlich benannten Objektarten 'GuFGewerbeundIndustrie' / 'IndustrieGewerbe' und 'GuFWohnen' / 'Wohnbau' sowie die die beiden Objektarten 'GuFÖffentlicheZwecke' / 'BesondereFunktionalePrägung'. Für die Objektart 'Ackerland' trifft die Annahme für das Gebiet Hildesheim zu. Für die Objektart 'GuFWohnen' trifft die Annahme jedoch nur für das Gebiet Hildesheim, aber nicht für das Gebiet Hannover zu. Dort werden 42 % der Objekte der abweichenden Objektart 'GemischteNutzung' zugeordnet, die jedoch in ihrer Definition auch Wohngebäude umfasst. Die Objektart 'Grünanlage' wird in den beiden Gebieten mehrheitlich den Objektarten 'Wohnbau' und 'GemischteNutzung' zugeordnet und nur zu einem geringen Anteil von 10 % beziehungsweise 25 % der zu erwartenden Objektart 'Grünanlage'. Ähnliches gilt für die Objektart 'Gartenland' die im Gebiet Hildesheim wiederum mehrheitlich der Objektart 'Wohnbau' zugeordnet wird und lediglich zu 13 % der zu erwartenden Objektart. Die Werte des Gebiets Hannover sind für die Objektart 'Gartenland' nicht repräsentativ, da dort lediglich neun Objekte vorhanden sind. Aus dem gleichen Grund sind die Werte der Objektart 'Grünland' für das urban geprägte Gebiet Hannover ebenso wenig repräsentativ. Im Gebiet Hildesheim hingegen werden die Objekte der Objektart 'Grünland' zu 47 % der entsprechenden Objektart aus ATKIS zugeordnet. Die Objektart 'GuFGewerbeundIndustrie' wird im Gebiet Hildesheim zu gleichen Teilen der zu erwartenden Objektart 'IndustrieGewerbe' sowie der Objektart 'Wohnbau' zugeordnet. Im Gebiet Hannover entspricht das Ergebnis der Zuordnung noch weniger den Erwartungen, da dort lediglich 18 % der Objekte der Objektart 'GuFGewerbeundIndustrie' der entsprechenden Objektart 'IndustrieGewerbe' aus ATKIS zugeordnet werden. Die Objektart 'GuFÖffentlicheZwecke' hat dagegen in beiden Gebieten mit jeweils mehr als 40 % eine deutliche Entsprechung in der Objektart 'BesondereFunktionalePrägung' aus ATKIS, wenn auch die verbleibenden Objekte hauptsächlich den beiden Objektarten 'GemischteNutzung' und 'Wohnbau' zu-

Objektarten der ALK (VKVRP, 2010)	Objektarten aus ATKIS (AdV, 2004)
Ackerland: Flächen, die dem feldmäßigen Anbau von Pflanzen dienen.	Ackerland: Fläche für den Anbau von Feldfrüchten [...] und Beerenfrüchten [...].
Gartenland: Flächen, die dem Gartenbau dienen.	Gartenland: Fläche für den Anbau von Gemüse, Obst und Blumen sowie die Aufzucht von Kulturpflanzen.
Grünanlage: Unbebaute Flächen, die vorherrschend der Erholung dienen.	Grünanlage: Größere Anlage [...], die vor allem der Erholung und Verschönerung des Stadtbildes dient.
Grünland: Grasflächen, die gemäht oder geweidet werden.	Grünland: Gras- und Rasenflächen, die gemäht oder beweidet werden.
GuFGewerbeundIndustrie: Gebäude- und Freiflächen, die vorherrschend gewerblichen und industriellen Zwecken dienen.	IndustrieGewerbe: Baulich geprägte Fläche, die ausschließlich oder vorwiegend der Unterbringung von Gewerbe- und Industriebetrieben dient. [...]
GuFHandelDienstleistung: Gebäude- und Freiflächen, die vorherrschend Einrichtungen von Handel und Dienstleistung dienen.	BesondereFunktionalePrägung: Baulich geprägte Fläche, auf der Gebäude und/oder Anlagen bestimmter Funktion vorherrschen. Hierzu gehören u.a. die Funktionen 'Verwaltung', 'Gesundheit und Soziales' [...], 'Bildung', 'Forschung' [...], 'Kultur' [...], 'Sicherheit und Ordnung' [...], 'Wochenend- und Ferienhausbebauung' und 'Landesverteidigung'.
GuFMischnutzungMitWohnen: Gebäude- und Freiflächen, die Wohn- und anderen Nutzungen zugleich dienen und bei denen die Wohn- oder andere Nutzung nicht von ganz untergeordneter Bedeutung ist.	GemischteNutzung: Baulich geprägte Fläche, auf der keine Art der baulichen Nutzung vorherrscht. Solche Flächen sind insbesondere ländlich – dörflich geprägte Flächen mit land- und forstwirtschaftlichen Betrieben, Wohngebäuden u.a. sowie städtisch geprägte Kerngebiete mit Handelsbetrieben und zentralen Einrichtungen für die Wirtschaft und die Verwaltung.
GuFÖffentlicheZwecke: Gebäude- und Freiflächen, die vorherrschend der Erfüllung öffentlicher Aufgaben und der Allgemeinheit dienen.	Wohnbau: Baulich geprägte Fläche, die ausschließlich oder vorwiegend dem Wohnen dient. [...]
GuFVersorgungsanlagen: Gebäude- und Freiflächen, die vorherrschend der Versorgung dienen.	Ortslage: Im Zusammenhang bebaute Fläche mit einer Ausdehnung von mindestens etwa 10 ha oder 10 Anwesen. [...]
GuFWohnen: Gebäude- und Freiflächen, die vorherrschend Wohnzwecken dienen.	

Tabelle 6.17: Definition der Objektarten der ALK und ATKIS

geordnet werden. Obwohl die genannten Objektarten ähnlich in der ALK und ATKIS definiert sind, lässt sich die zu erwartende Übereinstimmung für die Objektarten in beiden Gebieten nur für wenige Objektarten deutlich erkennen. Stattdessen werden die Objekte häufig anderen Objektarten zugeordnet, wobei die beiden Objektarten 'GemischteNutzung' und 'Wohnbau' den größten Einfluss aufweisen.

Für die verbleibenden Objektarten 'GuFHandelDienstleistung', 'GuFMischnutzungMitWohnen' und 'GuFVersorgungsanlagen' der ALK existieren keine direkt entsprechenden Objektarten aus ATKIS. Für diese Objektarten wäre es demnach zu erwarten, dass die Objekte jeweils einer Objektart der ALK auf mehrere Objektarten aus ATKIS aufgeteilt werden. Die Objekte der Objektart 'GuFHandelDienstleistung' teilen sich in beiden Gebieten vor allem auf die beiden Objektarten 'GemischteNutzung' und 'Wohnbau' aus ATKIS auf. Gleiches gilt für die Objektarten 'GuFMischnutzungMitWohnen' und 'GuFVersorgungsanlagen' der ALK. Demnach werden für die verbleibenden Objektarten der ALK die Erwartung erfüllt, wobei die zwei Objektarten 'GemischteNutzung' und 'Wohnbau' aus ATKIS wiederum die meisten Objekte aufnehmen.

Trotz der Analyse des Zusammenhangs zwischen den Objekten der ALK und ATKIS bleiben zwei entscheidende Fragen offen, die in Abschnitt 6.4 beantwortet werden:

1. *Lässt sich der Zusammenhang der Objekte zuverlässig durch ein Modell beschreiben?* Dabei entspricht das gesuchte Modell einer Klassifikation, mit der für ein Objekt der ALK eindeutig die zugehörige Objektart aus ATKIS prädiziert werden kann.

2. *Kann der Zusammenhang der Objekte im Gebiet Hildesheim und im Gebiet Hannover mit demselben Modell erklärt werden?* Oder sind die deutlichen Unterschiede zwischen den beiden Gebieten (siehe Tabellen 6.15 und 6.16) nur durch verschiedene Entstehungsprozesse zu erklären?

Eindeutiger ist dagegen der Zusammenhang zwischen den Objekten der ALK und der Objektart 'Ortslage' aus ATKIS, deren Definition auch in Tabelle 6.17 zu finden ist. Die Objektart 'Ortslage' wird dabei getrennt von den

anderen Objektarten aus Atkis betrachtet, da diese als zusätzliche Ebene über den anderen Objektarten liegt. So kann eine Fläche in Atkis sowohl der Objektart 'Wohnbau' als auch der Objektart 'Ortslage' zugeordnet sein. Für die Analyse des Zusammenhangs der Objekte der Alk mit den Objekten der Objektart 'Ortslage' aus Atkis wird wiederum eine Verschneidung der entsprechenden Objektgeometrien durchgeführt. Die Abbildung 6.40 zeigt die dominante Objektart für die unterschiedlichen Objektarten der Alk für das Gebiet Hildesheim, wobei diese entweder den Wert 'TOrt' oder 'None' annimmt, d.h. ein Objekt aus der Alk wird entweder der Objektart 'Ortslage' aus Atkis zugeordnet oder nicht. Auch hier muss die Objektart aus Atkis mindestens 50 % der Fläche des Alk-Objekts überdecken. Die exakten prozentualen Werte sind in Tabelle A.6 angegeben. Deutlich wird, dass die beiden Objektarten 'Ackerland' und 'Grünland' der Alk nur geringe Überschneidungen mit der Objektart 'Ortslage' aus Atkis aufweisen. Die anderen Objektarten der Alk haben dagegen eine deutliche Überschneidung mit der Objektart 'Ortslage', d.h. dass aus Sicht von Atkis Objekte dieser Objektarten sich meist innerhalb von im Zusammenhang bebauten Flächen befinden.

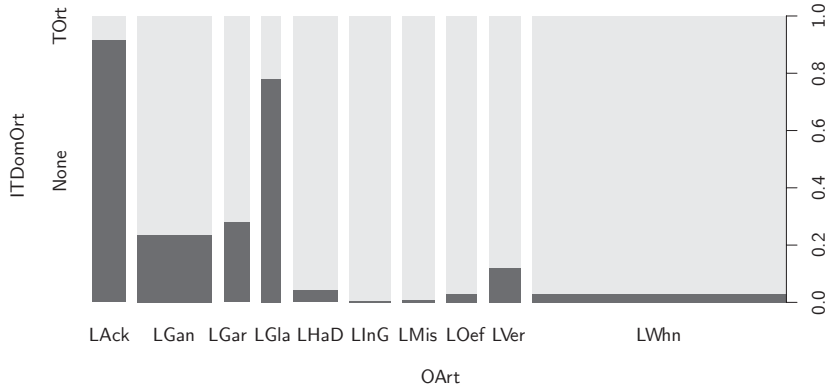


Abbildung 6.40: Dominante Objektart der Objekte der ALK in Bezug auf die Objektart 'Ortslage' aus ATKIS für das Gebiet Hildesheim

6.4 Klassifikation

Die Klassifikation bildet die zweite Säule des Data Mining der Geobasisdaten. Mit Hilfe der Klassifikation werden in diesem Abschnitt Modelle aufgestellt, die den Zusammenhang der Flächennutzung der Objekte der Alk und Atkis beschreiben. Durch die Klassifikation werden die beiden in Abschnitt 6.3 aufgestellten Fragen nach dem Zusammenhang der Objekte beantwortet, nämlich ob der Zusammenhang der Objekte der Alk und von Atkis zuverlässig durch ein Modell beschrieben werden kann und ob dieses Modell den Zusammenhang in den beiden Gebieten Hildesheim und Hannover gleichermaßen erklären kann.

Im Gegensatz zur Statistik und Klassenbeschreibung in Abschnitt 6.3 werden für die Klassifikation alle Schritte des Knowledge Discovery from Data (Kdd)-Prozesses abgearbeitet, bei dem die Klassifikation nur einer von sieben Schritten ist (siehe Abschnitt 2.6). Vor der Klassifikation werden in Unterabschnitt 6.4.1 die vier Schritte der Datenvorverarbeitung durchgeführt. Anschließend werden in Unterabschnitt 6.4.2 die interessanten Muster durch das Data Mining mit mehreren Klassifikatoren identifiziert und ausgewertet. Schließlich wird in Unterabschnitt 6.4.3 die entsprechende Wissensrepräsentation vorgestellt.

6.4.1 Datenvorverarbeitung

Durch die Datenvorverarbeitung werden die Daten so aufbereitet, dass diese als Eingabe für das Data Mining geeignet sind.

Datenbereinigung

Durch die Datenbereinigung werden unvollständige, verrauschte und inkonsistente Daten korrigiert. Da die Daten vollständig und nicht verrauscht sind, ist lediglich die Korrektur inkonsistenter Objekte durchzuführen. Für die Objekte der Alk bedeutet dies, dass alle Objekte, welche die in Abschnitt 6.3 definierten Integritätsbedingungen verletzen, aus den Eingabedaten für das Data Mining entfernt werden. Dieser Schritt des Kdd-Prozesses wurde im Rahmen der statistischen Analyse der Daten bereits in Unterabschnitt 6.3.3 durchgeführt.

Datenintegration

Eine Datenintegration ist für die Geobasisdaten nicht notwendig. Die Bestimmung des Zusammenhangs der Objekte der ALK und ATKIS kann jedoch allgemein als eine Form der Datenintegration angesehen werden.

Datenreduktion

Der Datenumfang lässt sich durch eine Reduktion der Anzahl der Objekte, Attribute oder möglichen Werte eines Attributs verringern. Die wenigen tausend Objekte der ALK stellen kein Problem für das Data Mining dar, weshalb diese nicht aggregiert oder Stichproben erstellt werden müssen. Ebenso wird die Anzahl der möglichen Werte der Attribute nicht durch Diskretisierung oder durch die Bildung von Konzepthierarchien reduziert, da die verwendete Klassifikatoren dies nicht voraussetzen und gerade die exakten Größen der Werte von Interesse sind.

Die Reduktion der Anzahl der Attribute kann durch zwei Ansätze erreicht werden, die beide berücksichtigt werden. Der erste Ansatz ist die Zusammenfassung von Attributen, bei der aus mehreren Attributen ein neues Attribut erzeugt wird, welches die in den Ausgangsattributen enthaltenen Informationen effektiver abbildet. Einige der in Unterabschnitt 6.2.2 aufgestellten geometrischen und topologischen Maße sind genau durch eine solche Zusammenfassung definiert. Bei den geometrischen Maßen ist dies die Statistik über die Distanz der Stützpunkte, welche die einzelnen Distanzen zwischen aufeinanderfolgenden Stützpunkten geeignet durch statistische Maße zusammenfasst, beispielsweise durch das Minimum der Distanzen. Aber auch die Kompaktheit und die fraktale Dimension sind zusammengefasste Größen, die jeweils die beiden Attribute Umfang und Fläche in einem einzigen Attribut vereinen. Die Elongation des MER als Verhältnis der Länge zur Breite ist ebenso ein zusammengefasstes geometrisches Maß. Gleiches gilt für das Flächenverhältnis, durch welches die beiden Größen Fläche des MER und Fläche der Objektgeometrie zu einem einzelnen Attribut zusammengefasst werden. Bei den topologischen Maßen stellt die dominante Objektart jeweils eine zusammengefasste Größe dar, da diese durch ein einzelnes Attribut die Informationen aller Objektarten geeignet zusammenfasst.

Der zweite Ansatz zur Reduktion der Anzahl der Attribute ist die Selektion relevanter Attribute. Da als Methode für das Data Mining Entscheidungsbäume und dessen Vereinfachungen eingesetzt werden, wird diese Auswahl automatisch bei der Erstellung der Entscheidungsbäume durchgeführt, d.h. alle nicht in einem Entscheidungsbaum verwendeten Attribute sind für die Problemstellung nicht signifikant relevant.

Datentransformation

Eine Normalisierung der Attributwerte ist für die Objekte der ALK nicht erforderlich, da diese keine Voraussetzung für die Verwendung der Klassifikatoren ist. Zudem sind die Größen der originären Werte von Interesse und nicht die transformierten Werte.

6.4.2 Data Mining mit Klassifikation und Auswertung der Muster

Durch die Klassifikation wird der Zusammenhang der Flächennutzung der ALK und ATKIS untersucht. Als Klassifikatoren werden in diesem Abschnitt Entscheidungsbäume sowie der OneRule- und ZeroRule-Algorithmus eingesetzt (siehe Unterabschnitt 2.6.4). Zudem werden die entdeckten Muster direkt nach der Klassifikation ausgewertet (siehe Unterabschnitt 2.6.7).

Um die beiden Fragen aus Unterabschnitt 6.3.4 beantworten zu können, wird die Klassifikation der beiden Gebiete Hildesheim und Hannover in mehreren Schritten durchgeführt:

1. Als Vorbereitung werden die Objekte der ALK im Gebiet Hildesheim in einen westlichen und einen östlichen Bereich unterteilt, siehe Abbildung 6.41. Von den insgesamt 3795 Objekten befinden sich 2023 Objekte im Gebiet Hildesheim-West und 1772 Objekte im Gebiet Hildesheim-Ost.
2. Zuerst werden nur die Objekte des Gebiets Hildesheim-West mit mehreren Modellen klassifiziert. Zur zufälligen Unterteilung der Objekte in zwei Teilmengen für das Training und den Test wird eine 10-fache Kreuzvalidierung verwendet. Das bedeutet, dass in jedem der zehn Schritte jeweils neun Zehntel der Objekte als Trainingsmenge und ein zehntel der Objekte als Testmenge für die Klassifikation verwendet werden. Das Ergebnis sind Modelle für den Zusammenhang der Flächennutzung der ALK und ATKIS sowie deren Klassifikationsgüte. Von den Modellen werden anhand der Klassifikationsgüte und der Komplexität die geeignetsten Modelle selektiert.
3. Anschließend werden im zweiten Schritt die selektierten Modelle vorgestellt und für die Objekte des Gebiets Hildesheim-Ost getestet. Durch die räumliche Unterteilung sind die Objekte vollständig unabhängig von

den Objekten des Gebiets Hildesheim-West und unterliegen dennoch demselben Entstehungsprozess. Mit dem Test wird die erste Frage beantwortet, ob der Zusammenhang der Objekte zuverlässig durch ein Modell beschrieben werden kann.

4. Schließlich werden die im zweiten Schritt erstellten Modelle auch für die Objekte des Gebiets Hannover getestet. Mit dem Test wird die zweite Frage beantwortet, ob der Zusammenhang der Objekte im Gebiet Hildesheim und im Gebiet Hannover mit demselben Modell erklärt werden kann.

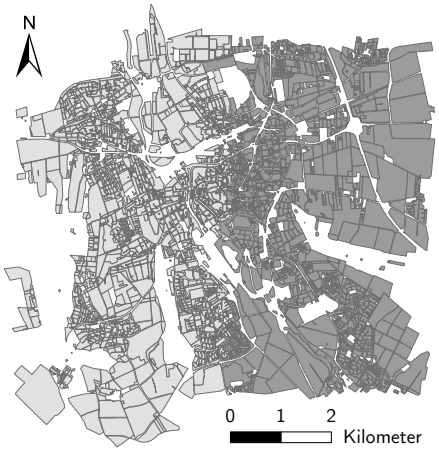


Abbildung 6.41: Unterteilung der Objekte der ALK im Gebiet Hildesheim in West und Ost

Modellselektion für die Objekte des Gebiets Hildesheim-West

Für die Modellselektion werden die Objekte des Gebiets Hildesheim-West mit Entscheidungsbäumen sowie dem OneRule- und ZeroRule-Algorithmus klassifiziert. Durch die Verwendung der drei Klassifikatoren wird es ermöglicht die Komplexität des Modells gegen die Klassifikationsgüte abzuwägen. Dort wo es sinnvoll ist werden zudem mehrere Varianten der Algorithmen durch eine entsprechende Wahl der Parameter berechnet.

Die Ergebnisse der Klassifikation der Objekte des Gebiets Hildesheim-West mit den drei genannten Klassifikatoren und unterschiedlichen Parametern sind in Tabelle 6.18 aufgelistet, wobei die Erkennungsrate mit 'ER' abgekürzt ist. Für alle Objektarten der ALK erzielt der ZeroRule-Algorithmus eine Erkennungsrate von 60,1 %, indem alle Objekte der ALK unabhängig von ihren Attributen der Objektart 'Wohnbau' aus ATKIS zugeordnet werden. Das Ergebnis zeigt die deutliche Dominanz der Objektart 'Wohnbau' in ATKIS bezogen auf die Anzahl der Objekte der ALK und damit die städtische Prägung des Gebiets.

ZeroRule	
ATKIS	ER
TWhn	60,1

OneRule			
Attribut	Gain	Num	ER
BLDomARel	0,71	11	68,2
Objektart	0,57	10	66,5
L50LDom	0,55	11	67,9
BLWhnARel	0,52	20	64,3
L50LWhn	0,40	7	63,0

Entscheidungsbaum		
Min	Blätter	ER
2	191	81,0
5	83	80,3
10	60	78,6
20	42	75,4
40	25	70,3

Tabelle 6.18: Klassifikation für alle Objektarten der ALK des Gebiets Hildesheim-West

Wird stattdessen der OneRule-Algorithmus verwendet, dann erhöht sich die Erkennungsrate auf 68,2 % bei Verwendung des Attributs 'BLDomARel'. Für ein Objekt der ALK wird dabei die Objektart aus ATKIS auf Basis der dominanten Objektart des Blocks prädiziert, in dem sich das Objekt befindet. Die Anzahl der Testentscheidungen des OneRule-Algorithmus ist erwartungsgemäß gering und beläuft sich auf 11 Entscheidungen. Während das erste Attribut 'BLDomARel' für den OneRule-Algorithmus automatisch von diesem selektiert wird, folgen die weiteren angegebenen Attribute aus einer Auswertung der Rangliste der Attribute. Diese Liste ordnet die Attribute nach dem in Unterabschnitt 2.6.4 vorgestellten Informationsgewinn (engl. 'Gain'). Für die fünf besten Attribute sind die Ergebnisse des OneRule-Algorithmus in Tabelle 6.18 aufgelistet. Aus diesen ist ersichtlich, dass die beiden Attribute 'BLDomARel' und 'L50LDom' die mit Abstand besten Erkennungsraten

aufweisen. Mit dem Attribut 'L50LDom' weist der OneRule-Algorithmus nur eine geringfügig reduzierte Erkennungsrate von 67,9 % auf. Wird die Objektart aus ATKIS für ein Objekt der ALK also auf Basis der dominanten Objektart an der um 50 m erweiterten Objektgeometrie prädiziert, dann ergeben die entsprechenden Regeln eine gleichwertige Klassifikationsgüte.

Neben dem ZeroRule- und OneRule-Algorithmus wird der Zusammenhang der Objektarten der ALK und ATKIS auch mit Entscheidungsbäumen analysiert. Wird die minimale Anzahl der Objekte pro Blattknoten (abgekürzt als 'Min') auf den Wert 2 gesetzt, dann erstellt der Klassifikator einen Baum mit 191 Blättern und einer Erkennungsrate von 81,0 %. Die in dieser Arbeit genannten Ergebnisse für Entscheidungsbäume beziehen sich immer auf zurückgeschnittene Bäume. Der vergleichbare nicht zurückgeschnittene Entscheidungsbaum enthält statt 191 Blätter insgesamt 425 Blätter bei einer Erkennungsrate von lediglich 79,5 %. Indem die minimale Anzahl an Objekten pro Blattknoten erhöht wird, kann die Anzahl der Blätter und damit die Komplexität des Entscheidungsbaums reduziert werden. Für ein Minimum von 5 Objekten wird vom Klassifikator ein Entscheidungsbaum mit 83 Blattknoten und einer Erkennungsrate von 80,3 % erstellt. Ein einfacheres Modell führt also zu einer marginalen Verringerung der Erkennungsrate. Wird die minimale Anzahl der Objekte pro Blattknoten weiter erhöht, dann weisen die entsprechenden Entscheidungsbäume eine jeweils geringere Erkennungsrate bei einer ebenso geringeren Anzahl an Blattknoten auf.

Die vorgestellten Klassifikationen mit unterschiedlichen Modellen und Parametern ermöglichen nun eine Modellselektion, um die geeignetsten Klassifikatoren auszuwählen. Aufgrund ihrer geringen Komplexität werden der Zero-Rule Algorithmus und die beiden besten OneRule-Algorithmen ausgewählt, um im weiteren untersuchen zu können wie gut die sehr einfachen Modelle gegenüber den komplexeren Entscheidungsbäumen abschneiden. Von den Entscheidungsbäumen wird das Modell mit der höchsten Erkennungsrate mit 191 Blättern sowie zum Vergleich das einfachere Modell mit lediglich 42 Blättern ausgewählt.

Selektierte Modelle und Test für die Objekte des Gebiets Hildesheim-Ost

Die selektierten Modelle werden im Folgenden kurz vorgestellt und anschließend für die Objekte des Gebiets Hildesheim-Ost getestet. Die Modelle werden dabei auf Basis aller Objekte des Gebiets Hildesheim-West aufgestellt, d.h. es werden keine Objekte für den Test in diesem Gebiet zurückgehalten.

Beim ZeroRule-Algorithmus wird jedem Objekt aus der ALK die Klasse 'Wohnen' aus ATKIS zugewiesen. Dagegen wird beim OneRule-Algorithmus für jedes Objekt der ALK der Wert eines einzelnen Attributs ausgewertet, um anschließend die entsprechende Klasse in ATKIS prädizieren zu können. In Tabelle 6.19 sind die Regeln der OneRule-Algorithmen für die beiden Attribute 'BLDomARel' und 'L50LDom' aufgelistet. Für jede der zehn ausgewählten Objektarten der ALK, sowie für den Fall dass keine der Objektarten dominant im Block beziehungsweise dominant in der um 50 m erweiterten Objektgeometrie ist (Wert 'None'), wird eine eigene Testentscheidung erzeugt. Die meisten der Regeln entsprechen dem in Unterabschnitt 6.3.4 diskutierten Zusammenhang der Objektarten, wie beispielsweise die Paare 'Ackerland' oder 'GuFHandelDienstleistung' / 'GemischteNutzung' belegen. Jedoch enthalten die Regeln auch überraschende Kombinationen, wie beispielsweise das Paar 'LGar' / 'TBFP'. Diese Kombination bedeutet, dass Objekten mit der Objektart 'Gartenland' in der ALK die Objektart 'BesondereFunktionalePrägung' in ATKIS zugeordnet wird.

Attribut	Bedingung	Objektart ATKIS
BLDomARel	LAck	TAck
	LGar	TBFP
	LGan	TWhn
	LGla	TGla
	LInG	TInG
	LHaD	TMis
	LMis	TMis
	LOef	TBFP
	LVer	None
	LWhn	TWhn
	None	TWhn

Attribut	Bedingung	Objektart ATKIS
L50LDom	LAck	TAck
	LGar	TBFP
	LGan	TWhn
	LGla	TAck
	LInG	TInG
	LHaD	TMis
	LMis	TWhn
	LOef	TWhn
	LVer	TWhn
	LWhn	TWhn
	None	TWhn

Tabelle 6.19: OneRule-Klassifikationregeln für alle Objektarten der ALK des Gebiets Hildesheim-West

Im Folgenden wird der Entscheidungsbaum mit 191 Blättern genauer vorgestellt. Ein Ausschnitt aus dem erstellten Entscheidungsbaum ist im Programm 6.12 als Text dargestellt. Der Wurzelknoten verwendet das Attribut

'BLAckARel', also den Flächenanteil der Objektart 'Ackerland' an der Fläche des Blocks, in dem das zu klassifizierende Objekt der ALK liegt. Darauf folgende innere Knoten sind gemäß ihrer Tiefe im Programm 6.12 durch einen Querstrich gekennzeichnet. Die sechs Blattknoten enthalten die prädizierte Klasse aus ATKIS. Bei jedem Blattknoten sind in Klammern die Anzahl der Objekte des Knotens sowie falls vorhanden die Anzahl der falsch klassifizierten Objekte im Knoten enthalten. Beispielsweise sind im dritten Blattknoten insgesamt 89 Objekte enthalten. Davon ist ein Objekt falsch klassifiziert, d.h. dessen Objektart aus ATKIS ist nicht 'Ackerland'. Als Ergänzung ist im Programm A.1 der vollständige Entscheidungsbaum mit 42 Blättern aufgelistet.

```
BLAckARel > 0.17
|   Objektart = Ackerland
|   |   L50LVer <= 0.05
|   |   |   ILWhnLRel <= 0.15
|   |   |   |   BLAckARel <= 0.24: TGla (3.0/1.0) // Blattknoten 1
|   |   |   |   BLAckARel > 0.24
|   |   |   |   |   BMinDia <= 47.34: TGla (3.0/1.0) // Blattknoten 2
|   |   |   |   |   BMinDia > 47.34: TAck (89.0/1.0) // Blattknoten 3
|   |   |   ILWhnLRel > 0.15
|   |   |   |   L50LGar <= 0.08: TAck (6.0) // Blattknoten 4
|   |   |   |   L50LGar > 0.08: TBFP (2.0) // Blattknoten 5
|   |   L50LVer > 0.05: TGla (2.0/1.0) // Blattknoten 6
|   Objektart = Gartenland
```

Programm 6.12: Ausschnitt aus dem Entscheidungsbaum für alle Objektarten der ALK des Gebiets Hildesheim-West

Die für das Gebiet Hildesheim-West erstellten Modelle werden nun für die Objekte des Gebiets Hildesheim-Ost getestet. Werden die erstellten Modelle auf die Objekte des Gebiets Hildesheim-Ost angewendet, dann erzielen diese die in Tabelle 6.20 aufgelisteten Ergebnisse. Der ZeroRule-Algorithmus, der alle Objekte der Objektart 'Wohnbau' aus ATKIS zuweist, erreicht eine Erkennungsrate von 62,2 %. Der Wert belegt die Dominanz der Objektart 'Wohnbau' in ATKIS für das gesamte Gebiet Hildesheim. Die Erkennungsraten des OneRule-Algorithmus für die beiden Attribute 'BLDomARel' und 'L50LDom' sind für das Gebiet Hildesheim-Ost gleichwertig und mit über 70 % für die Einfachheit des Modells gut. Die beiden Alternativen beim OneRule-Algorithmus zeigen, dass der Zusammenhang der Objekte der ALK und ATKIS nicht nur durch ein Modell, sondern zuverlässig durch zwei verschiedene Modelle und damit Hypothesen erklärt werden kann. Dagegen liegt die Erkennungsrate beim Entscheidungsbaum mit 191 Blättern sogar unter der des OneRule-Algorithmus. Der einfachere Entscheidungsbaum mit 42 Blättern erzielt schließlich die beste Erkennungsrate der Algorithmen von 74,4 %. Die Erkennungsraten von OneRule-Algorithmus und Entscheidungsbaum liegen demnach nahe zusammen, trotz ihrer unterschiedlichen Komplexität von 11 Entscheidungen gegenüber 42 Entscheidungen. Zusammenfassend lässt sich feststellen, dass der Zusammenhang der Objekte der ALK und ATKIS zuverlässig durch ein Modell beschrieben werden kann, wenn der zugrundeliegende Entstehungsprozess vergleichbar ist.

ZeroRule	
ATKIS	ER
TWhn	62,2

OneRule		
Attribut	Num	ER
BLDomARel	11	71,6
L50LDom	11	71,6

Entscheidungsbaum		
Min	Blätter	ER
2	191	70,6
20	42	74,4

Tabelle 6.20: Klassifikation für alle Objektarten der ALK des Gebiets Hildesheim-Ost basierend auf Hildesheim-West

Die Auswertung der Muster umfasst neben der bereits verwendeten Erkennungsrate die Bestimmung weiterer Maße, die eine differenziertere Analyse der Klassifikationsergebnisse ermöglichen (siehe Unterabschnitt 2.6.7). Dazu ist in Tabelle 6.21 die Konfusionsmatrix des Entscheidungsbaums mit der höchsten Erkennungsrate von 74,4 % angegeben. Im Gegensatz zum Beispiel der binären Klassifikation in Unterabschnitt 2.6.7 enthält die Konfusionsmatrix nun die acht ausgewählten Objektarten aus ATKIS sowie die Objektart 'None'. Die Elemente auf der Hauptdiagonalen bezeichnen dabei die korrekt klassifizierten Objekte, während die verbleibenden Elemente die falsch klassifizierten Objekte bezeichnen. Beispielsweise ist in der ersten Zeile der Matrix ablesbar, dass bei 76 Objekten die tatsächliche Objektart gleich der prädizierten Objektart 'Ackerland' ist. Dagegen ordnet der Klassifikator fälschlicherweise ein Objekt der Objektart 'Grünland' zu. Aus der Konfusionsmatrix kann abgelesen werden, dass manche Objektarten gut (z.B. 'Ackerland') und manche Objektarten weniger gut (z.B. 'Gartenland') vom Klassifikator prädiziert werden. Noch deutlicher wird dieser Unterschied, wenn statt der absoluten Anzahl der Objekte relative Maße bei der Auswertung der Muster berücksichtigt werden.

Objektart	TAck	TBFP	TGan	TGar	TGla	TInG	TMis	TWhn	None
TAck	76	7	1	0	1	1	1	10	0
TBFP	2	31	5	0	0	3	6	30	2
TGan	0	4	5	0	0	1	0	21	3
TGar	4	21	14	0	1	0	3	24	4
TGla	0	4	1	0	15	1	7	7	3
TInG	2	21	0	5	1	68	12	14	3
TMis	1	14	0	0	4	0	86	62	2
TWhn	5	22	1	0	1	16	26	1019	12
None	1	10	6	0	3	0	4	14	18

Tabelle 6.21: Konfusionsmatrix des Entscheidungsbaums für alle Objektarten der ALK des Gebiets Hildesheim-Ost

In Tabelle 6.22 sind für die beiden Klassifikatoren Entscheidungsbaum und OneRule-Algorithmus die Werte der Maße für die Klassifikationsgüte angegeben, wobei alle Werte die Einheit Prozent aufweisen. Zuerst werden die Maße für den Entscheidungsbaum mit 42 Blättern diskutiert. Die Werte für die Genauigkeit, d.h. die Exaktheit der Klassifikation, und die Sensitivität, d.h. die Vollständigkeit der Klassifikation, weisen für fast alle Objektarten jeweils vergleichbare Werte auf. Eine Ausnahme bilden die drei Objektarten 'BesondereFunktionalePrägung', 'Grünland' und 'IndustrieGewerbe' aus ATKIS, bei denen die Werte für die Genauigkeit und die Sensitivität einen deutlichen Abstand voneinander aufweisen. Der Unterschied zwischen den Objektarten lässt sich jedoch am besten am F-Maß ablesen, das dem harmonischen Mittel der beiden Maße Genauigkeit und Sensitivität entspricht. Mit mehr als 80 % werden die Objektarten 'Wohnbau' und 'Ackerland' sehr gut vom Modell abgebildet. Die Objektart 'IndustrieGewerbe' wird mit über 60 % gut bestimmt, wohingegen die Objektarten 'GemischteNutzung' und 'Grünland' mit etwa 50 % nur befriedigend vom Modell abgebildet werden. Die verbleibenden Objektarten 'None', 'BesondereFunktionalePrägung', 'Grünanlage' und 'Gartenland' werden lediglich mangelhaft prädiziert. Der Zusammenhang zwischen den Flächennutzungen in der ALK und ATKIS lässt sich also nicht für alle Objektarten gleich gut mit dem Entscheidungsbaum bestimmen.

	Entscheidungsbaum (42 Blätter)					OneRule (BLDomARel)				
Objektart	RP	FP	Genauigkeit	Sensitivität	F-Maß	RP	FP	Gen.	Sens.	F-Maß
TAck	78	1	84	78	81	79	5	51	79	62
TBFP	39	6	23	39	29	62	4	42	62	50
TGan	15	2	15	15	15	0	0	0	0	0
TGar	0	0	0	0	0	0	0	0	0	0
TGla	40	1	58	40	47	32	1	60	32	41
TInG	54	1	76	54	63	59	2	71	59	64
TMis	51	4	59	51	55	43	2	69	43	53
TWhn	93	27	85	93	89	89	41	78	89	83
None	32	2	38	32	35	13	1	37	13	19
Gewichtetes Mittel	74	18	72	74	73	72	27	67	72	68

Tabelle 6.22: Prozentuale Maße für die Klassifikationsgüte des Entscheidungsbaums und OneRule-Algorithmus für alle Objektarten der ALK des Gebiets Hildesheim-Ost

Die Maße für die Klassifikationsgüte in Tabelle 6.22 sind erwartungsgemäß für den OneRule-Algorithmus mit dem Attribut 'BLDomARel' schlechter als beim Entscheidungsbaum. Da durch die Regeln in Tabelle 6.19 die beiden Objektarten 'Grünanlage' und 'Gartenland' aus ATKIS niemals prädiziert werden, sind die Maße für diese Objektarten gleich null Prozent. Für die verbleibenden Objektarten ähnelt die Reihenfolge der Werte für das F-Maß denen des Entscheidungsbaums. Deutliche Unterschiede zeigen sich lediglich bei der Objektart 'BesondereFunktionalePrägung', die mit dem OneRule-Algorithmus besser prädiziert wird, sowie bei den Objektarten 'Ackerland' und 'None', die mit dem OneRule-Algorithmus schlechter prädiziert werden.

Gebiet Hannover

Die für das Gebiet Hildesheim-West erstellten Modelle werden zusätzlich für die Objekte des Gebiets Hannover getestet. Ist die Klassifikationsgüte vergleichbar zum Gebiet Hildesheim-Ost, dann können die Zusammenhänge der Flächennutzung zwischen der ALK und ATKIS für beide Gebiete mit denselben Modellen erklärt werden.

Werden die Modelle auf die Objekte des Gebiets Hannover angewendet, dann werden die in Tabelle 6.23 aufgelisteten Erkennungsraten erzielt. Da die Objektart 'Wohnbau' aus ATKIS im Gebiet Hannover eine geringere Dominanz aufweist, beträgt die Erkennungsrate des OneRule-Algorithmus lediglich 44,4 %. Die maximalen Erkennungsraten des OneRule-Algorithmus liegen mit 51,8 % mehr als 19 % unter den entsprechenden Erkennungsraten für das Gebiet Hildesheim-West. Der Abstand der Erkennungsraten ist noch deutlicher für die Entscheidungsbäume. Hier liegen die Erkennungsraten des Gebiets Hannover etwa 23 % unter denen des Gebiets Hildesheim-Ost.

ZeroRule	
ATKIS	ER
TWhn	44,4

OneRule		
Attribut	Num	ER
BLDomARel	11	51,8
L50LDom	11	47,6

Entscheidungsbaum		
Min	Blätter	ER
2	191	50,4
20	42	51,2

Tabelle 6.23: Klassifikation für alle Objektarten der ALK des Gebiets Hannover basierend auf Hildesheim-West

Zusammenfassend lässt sich feststellen, dass aufgrund der hohen Differenz der Erkennungsraten der Zusammenhang der Objekte der ALK und ATKIS im Gebiet Hannover nicht zufriedenstellend mit denselben Modellen erklärt werden kann wie der entsprechende Zusammenhang im Gebiet Hildesheim. Die Modelle des Gebiets Hildesheim-West können also nur bedingt auf das Gebiet Hannover übertragen werden. Dies lässt sich vor allem auf die deutlich unterschiedlichen Charakteristika der beiden Gebiete Hildesheim und Hannover zurückführen. Während das Gebiet Hildesheim neben städtischen Bereichen auch ländliche Bereiche in der näheren Umgebung umfasst, sind im Gebiet Hannover nur innerstädtische Flächennutzungen enthalten. Es ist demnach zu erwarten, dass die erstellten Modelle bessere Erkennungsraten aufweisen, je ähnlicher die Charakteristika der Gebiete sind, wie beispielsweise die guten Erkennungsraten für das Gebiet Hildesheim aufzeigen. Leider konnten solche Daten für diese Arbeit nicht bereitgestellt werden.

6.4.3 Wissensrepräsentation

Ist das Ergebnis der Klassifikation ein Ereignisbaum, dann kann dieser entweder direkt als Baum wie im Programm 6.12, als Wenn-Dann-Regeln wie in Tabelle 2.16 oder in Integritätsbedingungen der GeoOCL repräsentiert werden. Dieselbe Repräsentation kann auch bei Verwendung des OneRule- und ZeroRule-Algorithmus verwendet werden. Als Beispiel sind im Programm 6.13 die Klassifikationsregeln des OneRule-Algorithmus aus Tabelle 6.19 für das Attribut 'BLDomARel' als Integritätsbedingungen in der GeoOCL aufgestellt.

```
context ALKNutzung
  inv: BLDomARel = 'Ackerland' implies objektartATKIS = 'Ackerland'
  inv: BLDomARel = 'Gartenland' implies objektartATKIS = 'BesondereFunktionalePrägung'
  inv: BLDomARel = 'Grünanlage' implies objektartATKIS = 'Wohnbau'
  inv: BLDomARel = 'Grünland' implies objektartATKIS = 'Grünland'
  inv: BLDomARel = 'GuFGewerbeundIndustrie' implies objektartATKIS = 'IndustrieGewerbe'
  inv: BLDomARel = 'GuFHandelDienstleistung' implies objektartATKIS = 'GemischteNutzung'
  inv: BLDomARel = 'GuFMischnutzungMitWohnen' implies objektartATKIS = 'GemischteNutzung'
  inv: BLDomARel = 'GuFÖffentlicheZwecke' implies objektartATKIS = 'BesondereFunktionalePrägung'
  inv: BLDomARel = 'GuFVersorgungsanlagen' implies objektartATKIS = 'None'
  inv: BLDomARel = 'GuFWohnen' implies objektartATKIS = 'Wohnbau'
  inv: BLDomARel = 'None' implies objektartATKIS = 'Wohnbau'
```

Programm 6.13: OneRule-Klassifikationsregeln für das Attribut BLDomARel in der GeoOCL

6.5 Anforderungen

Die für die Flächennutzung in Geobasisdaten berücksichtigten Anforderungen aus dem Anforderungskatalog in Kapitel 4 sind in Tabelle 6.24 aufgelistet. Die nicht in der Tabelle enthaltenen Anforderungen sind entweder

nicht berücksichtigt, da sie auf die verwendeten Daten nicht zutreffen, oder weil sie bewusst nicht ausgewählt werden.

Nr.	Anforderung	Diskussion
1	Diskrete und kontinuierliche Objekte	Prinzipiell entsprechen die Flächennutzungen in den Geobasisdaten kontinuierlichen Objekten, da für jede Fläche eine Nutzung angegeben sein muss. Durch die Selektion der zehn Objektarten entstehen jedoch Lücken, wodurch die Daten schließlich diskreten Objekten entsprechen.
9	Anzahl der Attributwerte/Attribute/Objekte/Klassen	Die Attribute weisen jeweils nur einen Wert auf. Dagegen verwenden nahezu alle aufgestellten Integritätsbedingungen mehrere Attribute, da sie sich jeweils auf eine Objektart und ein weiteres Attribut beziehen. An die Anzahl der Objekte werden keine Bedingungen gestellt. Da nur eine Klasse verwendet wird, werden auch keine klassenübergreifenden Bedingungen benötigt.
11	Methoden für die Ableitung von Geometrien	Es werden einige Methoden für die Ableitung von Geometrien verwendet, wie beispielsweise die Ableitung der Stützpunkte und der Liniensegmente aus den Umringen der Flächen sowie die Ableitung von umschließenden Rechtecken.
12	Funktionale Abhängigkeit	Die funktionale Abhängigkeit wird in der Anwendung implizit berücksichtigt. Keines der im Originaldatensatz vorhandenen Attribute weist eine funktionale Abhängigkeit auf. Dagegen werden alle angereicherten Attribute aus der Geometrie und Topologie der Objekte bestimmt und sind damit funktional von diesen abhängig. Dieser Zusammenhang muss jedoch nicht als Integritätsbedingungen ausgedrückt werden, da die angereicherten Attribute automatisch und damit korrekt erzeugt werden.
14	Fordern vs. Einschränken	Alle Integritätsbedingungen sind in der Anwendung als fordernd definiert, da diese jeweils weniger Vergleichswerte benötigen.
15	Wertebereich	Nahezu alle aufgestellten Integritätsbedingungen schränken den gültigen Wertebereich der Attribute ein.
16	Genauigkeit	Die Genauigkeit der Attributwerte wird durch eine geeignete Wahl von Nachkommastellen berücksichtigt. So sind beispielsweise die Bedingungen an die Flächen mit Metergenauigkeit angegeben, wohingegen die Schwellwerte für die fraktale Dimension mit einer Nachkommastelle angegeben sind.

Tabelle 6.24: Berücksichtigte Anforderungen für die Flächennutzung in Geobasisdaten

Einige der Anforderungen des Anforderungskatalogs treffen auf die verwendeten Daten nicht zu. So sind die Daten lediglich statisch (Anf. 2) und zweidimensional (Anf. 3). Die Daten weisen außerdem keine temporalen Informationen (Anf. 4), topologischen Netzwerke (Anf. 5), zusammengesetzte Geometrien und Aggregation (Anf. 6), konzeptuelle Generalisierung (Anf. 7), unscharfe Geometrien (Anf. 8) sowie fehlende Werte (Anf. 17) auf. Da die Daten nur eine Klasse enthalten, sind generische Bedingungen (Anf. 13) nicht einsetzbar. Zudem sind die verwendeten Datensätze nicht umfangreich (Anf. 23). Werden die aufgestellten Integritätsbedingungen jedoch zur Prüfung der Objekte der ALK der gesamten Fläche Deutschlands verwendet, dann sind die in Kapitel 7 vorgestellten Aspekte der Parallelisierung durchaus von Relevanz.

Die verbleibenden Anforderungen werden bewusst nicht berücksichtigt. Es wird keine flexible Topologie (10) benötigt, da die binären topologischen Relationen der 9 Intersection Method (9IM) für die Anwendung ausreichen. In der Anwendung werden zudem aus Gründen der Einfachheit alle Integritätsbedingungen mit demselben Schweregrad (Anf. 18) behandelt. Eine automatische Konsistenzprüfung der Bedingungen (Anf. 19) findet nicht statt, da wie in Abschnitt 5.2 beschrieben kein automatisches Werkzeug dafür zur Verfügung steht. Die Toleranz (Anf. 20) wird bei der Prüfung der Objekte aus Gründen einer einfacheren Implementierung in der entsprechenden Software nicht berücksichtigt. Das Ausmaß (Anf. 21) und das globale Ausmaß (Anf. 22) der Verletzung der Integritätsbedingungen wird nicht untersucht, da es für die Darstellung in dieser Arbeit ausreichend ist zu wissen ob ein Objekt gültig ist oder nicht. Fehler werden nicht korrigiert (Anf. 24), da dies für Geobasisdaten nur durch die entsprechenden amtlichen Stellen erfolgen darf. Aus diesem Grund ist auch die automatische Aktualisierung (Anf. 25) und die Erstellung von Ausnahmen (Anf. 26) nicht von Interesse. Schließlich ist die Berücksichtigung der Sicherheit (Anf. 27) für diese Arbeit auch nicht notwendig.

6.6 Implementierung

Die in diesem Kapitel beschriebene Aufstellung der Integritätsbedingungen für Geobasisdaten wird durch mehrere Softwareprodukte ermöglicht beziehungsweise unterstützt. Diese werden im Folgenden kurz vorgestellt.

Anreicherung der Daten

Die Anreicherung der Daten in Unterabschnitt 6.2.2 wird durch selbst entwickelte Programme durchgeführt, da nur wenige der benötigten geometrischen und topologischen Maße in kommerziellen Geoinformationssystem (Gis)-Programmen enthalten sind. Dazu zählen insbesondere die abgeleiteten geometrischen Maße (siehe Tabelle 6.4) sowie alle topologischen Maße (siehe Tabelle 6.5). Die Anreicherung erfolgt durch Programme, die in der Programmiersprache Java entwickelt sind. Diese verwenden zur Verarbeitung der Geodaten die beiden quelloffenen Bibliotheken GeoTools (2013) und Java Topology Suite (Jts) (Davis, 2013). Als Datenformat zur Ein- und Ausgabe werden von den Programmen Dateien im Environmental Systems Research Institute (Esri) Shapefile-Format verwendet (siehe Unterabschnitt 7.3.1).

Statistik und Klassenbeschreibung

Die statistischen Auswertungen und Klassenbeschreibungen in Abschnitt 6.3 werden mit dem quelloffenen Statistikprogramm R (2013) durchgeführt. Der Standardumfang wird dabei um frei verfügbare Module sowie einige selbst entwickelte Methoden erweitert. Insbesondere sind alle Box-Plots, Histogramme und Scatter-Plots mit R erstellt. Die Eingabedaten für das Statistikprogramm sind formatierte Textdateien, die aus den alphanumerischen Fachdaten der Objekte in den Shapefiles abgeleitet sind.

Klassifikation

Die Klassifikation in Abschnitt 6.4, und damit neben der Klassenbeschreibung die zweite Säule des Datenmining, wird mit dem quelloffenen Data Mining Programm Weka (2013) durchgeführt, das in Hall u. a. (2009) detaillierter beschrieben ist. Dieses bietet neben den verwendeten Algorithmen ZeroRule, OneRule und Entscheidungsbaum viele weitere Algorithmen und Werkzeuge zur Datenanalyse und zum Data Mining. Als Eingabedateien werden wiederum formatierte Textdateien verwendet, die direkt aus den alphanumerischen Fachdaten der Shapefiles abgeleitet sind.

6.7 Übertragbarkeit

Als Abschluss des Kapitels wird nochmals diskutiert, was durch die Analyse der Daten erreicht wurde und wie sich das vorgestellte Vorgehen auf gleichartige Datensätze übertragen lässt. Als gleichartig sind dabei alle Datensätze anzusehen, die vollständig oder nahezu vollständig flächenüberdeckend sind.

Die Analyse der Daten verfolgt in diesem Kapitel zwei Ziele. Das erste Ziel ist die Klassenbeschreibung, mit der die charakteristischen Eigenschaften der Objekte der Datensätze beschrieben werden. Wenn für die einzelnen Klassen jeweils Integritätsbedingungen aufgestellt sind, dann können beliebige Datensätze derselben Art auf ihre Einhaltung der Bedingungen überprüft werden und so fehlerhafte beziehungsweise auffällige Objekte identifiziert und anschließend im Detail untersucht werden. Für die verwendeten Daten der Alk bedeutet dies, dass die Integritätsbedingungen, die anhand der drei Gebiete aufgestellt werden, auf Teile oder die Gesamtheit der Objekte der Alk Deutschlands übertragbar sind. Dabei ist es wichtig die Gebiete entsprechend repräsentativ zu wählen, damit die Übertragbarkeit gesichert ist. Das Gebiet Hildesheim weist sowohl urbane als auch ländliche Bereiche auf, die sich in dem urban geprägten Datensatz Hannover und dem ländlich geprägten Datensatz Ostercappeln wiederfinden. Das zweite Ziel der Analyse der Daten ist die Klassifikation, mit der Modelle für den Zusammenhang von Attributwerten innerhalb eines Datensatzes ermittelt werden können. Werden entsprechend Attribute erzeugt, die den Zusammenhang von Attributwerten über mehrere Datensätze beschreiben, so kann auch der Zusammenhang zwischen mehreren Datensätzen ermittelt werden. Für die verwendeten Daten der Alk und von Atkis wird der Zusammenhang durch die Verschneidung der Objektgeometrien bestimmt und so ermittelt, welche Objektarten der Alk welchen Objektarten in Atkis entsprechen. Allgemein lässt sich der Zusammenhang zwischen verschiedenen Datensätzen für beliebige Geometrietypen ermitteln. In Abbildung 6.42 sind als Beispiel zwei punktuelle Datensätze dargestellt. Der erste Datensatz ist in schwarz mit alphanumerischer Bezeichnung für die Objekte und der zweite Datensatz in grau mit numerischer Bezeichnung für die Objekte visualisiert. Wird der Zusammenhang beispielsweise durch den jeweils nächstgelegenen Punkt beschrieben, so können die Attribute des Objekts A um die Attribute des Objekts 1 erweitert werden. Zusätzlich kann

der Zusammenhang durch weitere Attribute beschrieben werden, beispielsweise um die Entfernung der jeweils zugeordneten Punkte.

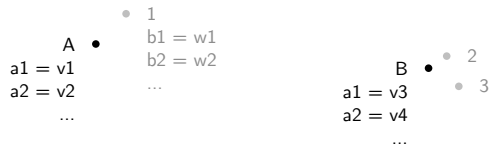


Abbildung 6.42: Beispiel für den Zusammenhang zwischen zwei Datensätzen

Das methodische Vorgehen ist auf beliebige Datensätze übertragbar, wie auch die Anwendung des gleichen Vorgehens auf Gebäudegrundrisse in frei verfügbaren Datensätzen in Kapitel 7 aufzeigt. Um die Daten in den weiteren Schritten korrekt prozessieren zu können, müssen diese zuerst auf ihre Gültigkeit überprüft werden. Die Kriterien die dabei überprüft werden müssen hängen vom Geometrietyp ab. Die Kriterien für Flächen werden in Unterabschnitt 6.2.1 vorgestellt. In den Daten der ALK wird die Forderung an die Konsistenz der Fläche von einer beträchtlichen Anzahl an Objekten verletzt. Um die Daten dennoch analysieren zu können, werden die identifizierten Objekte aus den Daten entfernt. Alternativ können diese auch korrigiert werden, um so einen vollständig gültigen Datensatz zu erhalten. Der zweite Schritt ist die Anreicherung der Daten um zusätzliche geometrische und topologische Maße. Die Motivation für diesen Schritt begründet sich in der Annahme, dass die Form und die Nachbarschaft von Objekten erheblichen Einfluss auf deren Charakteristika haben und mit diesen die einzelnen Klassen beschrieben und voneinander unterschieden werden können. Da solche Attribute selten Bestandteil der Daten sind, müssen diese entsprechend angereichert werden. Für die Daten der ALK sind die Maße in Unterabschnitt 6.2.2 beschrieben. Für andere vollständig oder nahezu vollständig flächenüberdeckende Datensätze können die gleichen oder teilweise gleichen Maße verwendet werden um die Form und Nachbarschaft der Objekte geeignet zu beschreiben und zu quantifizieren.

Die eigentliche Klassenbeschreibung bildet dann den dritten Schritt. Mit Methoden der deskriptiven und inferentiellen Statistik sowie der explorativen Datenanalyse werden die Maße analysiert um eine geeignete Beschreibung der Klassen ableiten zu können. In Abschnitt 6.3 wird aufgezeigt, wie Histogramm, Pentagramm, Box-Plot, Scatter-Plot sowie Quantile-Quantile-Plot eingesetzt werden können um schließlich Integritätsbedingungen für die einzelnen Klassen in der GeoOCL zu erhalten. Von besonderem Interesse sind dabei zwei Aspekte. Durch die Analyse der Korrelation werden wichtige Zusammenhänge aufgedeckt und zudem die Anzahl der auszuwertenden Maße reduziert. Durch die Überprüfung auf das Vorliegen einer bestimmten theoretischen Verteilung wird entweder eine elegante Modellierung von Bedingungen ermöglicht oder bestätigt dass solche Verteilungen auf bestimmte Maße nicht zutreffen. Für die Daten der ALK unterscheiden sich die Maße und damit die Integritätsbedingungen zwischen den einzelnen Klassen deutlich und liefern so eine aussagekräftige Klassenbeschreibung. Für die zehn ausgewählten Objektarten der ALK ergeben sich insgesamt 92 Integritätsbedingungen basierend auf acht geometrischen und 33 topologischen Maßen. Alle Maße unterliegen keiner der geprüften theoretischen Verteilungen, was dazu führt, dass bestimmte Instrumente der Statistik und des Data Mining, für die beispielsweise eine Normalverteilung gefordert wird, nicht anwendbar sind. Jedoch können unabhängig von der theoretischen Verteilung geeignete Bedingungen für die Lage- und Streuungsmaße der explorativen Statistik aufgestellt und formalisiert werden. Insgesamt weisen die verwendeten Datensätze der ALK einen Anteil an fehlerhaften beziehungsweise auffälligen Objekten zwischen 1,3 % und 1,9 % auf. Für andere Datensätze und andere Maße ist das Vorgehen analog, d.h. es können die gleichen Werkzeuge und Überlegungen verwendet werden um entsprechende Integritätsbedingungen in der GeoOCL abzuleiten.

Der vierte Schritt dient der Bestimmung des Zusammenhangs zweier Datensätze. Bei flächenhaften Datensätzen kann der Zusammenhang durch die Verschneidung der Objektgeometrien ermittelt werden. Wie der Zusammenhang für andere Datensätze ermittelt wird, hängt nicht nur von den jeweiligen Geometrietypen, sondern auch von der Logik für die Zuordnung der Objekte ab. Im Beispiel in Abbildung 6.42 werden Punkte anhand der minimalen Distanz zugeordnet. Wird die Zuordnung zu einem anderen Datensatz als Attribut eines Datensatzes modelliert, dann kann überprüft werden ob der Zusammenhang einer gewissen Regelmäßigkeit unterliegt. Der Zusammenhang zweier Attribute kann dann wie in Unterabschnitt 6.3.4 durch eine Konfusionsmatrix veranschaulicht werden beziehungsweise für numerische Attribute in einem Scatter-Plot. Komplexere Modelle für den Zusammenhang, die mehrere Attribute einbeziehen, können durch das Data Mining Verfahren der Klassifikation bestimmt werden. Dieses wird in Abschnitt 6.4 für die Bestimmung des Zusammenhangs zwischen den Objekten der ALK und ATKIS verwendet. Ausgehend von der Objektart in der ALK sowie den geometrischen und topologischen Maßen eines Objekts wird ein Modell aufgestellt um die entsprechende Objektart des Objekts in ATKIS zu bestimmen. Die auf den Daten des Gebiets Hildesheim-West erlernten Modelle erreichen gute Erkennungsraten für das Testgebiet Hildesheim-Ost. Der OneRule-Algorithmus erreicht mit elf Regeln bereits eine Erkennungsra-

te von 71,6 % und mit einer Steigerung der Komplexität durch den Entscheidungsbaum-Algorithmus kann eine Erkennungsrate von 74,4 % bei 42 Regeln erreicht werden. Sind die Gebiete also vergleichbar, dann liefert die Klassifikation mit den genannten Algorithmen gute Erkennungsraten. Sind die Gebiete jedoch nicht vergleichbar, wie bei dem Testgebiet Hannover, dann ist die Erkennungsrate mit maximal 51,8 % nicht zufriedenstellend. Es ist demnach wichtig entweder einen entsprechend homogenen Gesamtdatensatz als Ausgangsbasis für die Klassifikation zu haben, oder mehrere Modelle für mehrere charakteristische Typen von Gebieten aufzustellen. Leider konnten solche Daten für diese Arbeit nicht bereitgestellt werden. Mit den verwendeten Algorithmen lassen sich aus dem Ergebnis der Klassifikation ebenso Integritätsbedingungen in der GeoOcl ableiten.

7 Integritätsbedingungen: Gebäude in Open Data und Parallelisierung

In diesem Kapitel werden anhand von drei unterschiedlichen Gebieten Integritätsbedingungen für Gebäudegrundrisse aufgestellt. Die verwendeten Daten sind dabei repräsentativ für andere Datensätze, die punkt-, linien- oder flächenhaft und im Gegensatz zu Kapitel 6 nicht flächenüberdeckend sind. Einige Beispiele für solche Daten sind Geschäfte, Versorgungspunkte, Straßen, Gewässer oder Siedlungen. Das in diesem Kapitel vorgestellte Vorgehen ist auf diese gleichartigen Datensätze zu einem hohen Prozentsatz übertragbar.

In diesem Kapitel wird zudem die Parallelisierung, deren allgemeine Aspekte in Abschnitt 2.7 vorgestellt werden, auf die Verarbeitung von Geodaten erweitert. Durch den Einsatz der Parallelisierung können Integritätsbedingungen selbst für Datensätze aufgestellt werden, deren serielle Prozessierung aufgrund der Komplexität der Bedingungen oder des Umfangs der Daten zu viel Zeit oder zu viele Ressourcen in Anspruch nehmen würde. Dies trifft auch auf zeitkritische Anwendungen zu bei denen das Ergebnis der Prüfung der Bedingungen innerhalb eines gewissen Zeitfensters vorliegen muss, beispielsweise aufgrund einer häufigen Änderung der Daten.

Zuerst wird in Abschnitt 7.1 die Einteilung räumlicher Objekte in Partitionen vorgestellt. Dabei wird allgemein diskutiert, wie diese Einteilung durchgeführt werden kann und welche Auswirkungen die Einteilung auf die Verarbeitung der Daten hat. Die Parallelisierung wird in diesem Kapitel anhand von Gebäudegrundrissen in frei verfügbaren Datensätzen untersucht, welche in Abschnitt 7.2 vorgestellt werden. Die Datensätze sind Beispiele für Open Data, d.h. für Daten die von Institutionen oder von Freiwilligen erhoben wurden und kostenlos zur Verfügung gestellt werden. Um die Daten effizient mit Hadoop parallel verarbeiten zu können, müssen diese zuerst in ein geeignetes Format konvertiert werden. In Abschnitt 7.3 wird das verwendete Datenformat GeoAvro und die Konvertierung von Dateien im Shapefile-Format des Environmental Systems Research Institute (Esri) vorgestellt.

In den darauf folgenden Abschnitten wird die parallele Verarbeitung der Daten mit MapReduce und dem Hadoop-Framework vorgestellt. Zuerst werden die Gebäudeobjekte in Abschnitt 7.4 mit MapReduce um geometrische und topologische Maße angereichert. Danach wird die Häufigkeit der berechneten Attributwerte in Abschnitt 7.5 mit einem MapReduce-Workflow bestimmt. Schließlich werden die Häufigkeiten in Abschnitt 7.6 statistisch untersucht, um Klassenbeschreibungen und damit erstmals Integritätsbedingungen für die Daten abzuleiten. Die bei der Aufstellung der Integritätsbedingungen berücksichtigten Anforderungen aus Kapitel 4 werden in Abschnitt 7.7 aufgelistet. In Abschnitt 7.8 werden einige Aspekte der Implementierung kurz vorgestellt. Schließlich wird in Abschnitt 7.9 nochmals diskutiert, was durch die Analyse der Daten erreicht wurde und wie sich das vorgestellte Vorgehen auf gleichartige Datensätze übertragen lässt.

7.1 Parallelisierung und Partitionierung

Um Geodaten parallel verarbeiten zu können, müssen diese geeignet in räumliche und nicht-räumliche Partitionen unterteilt werden. Für die Parallelisierung eignet sich das Konzept der Datenparallelität, das Konzept der Task-Parallelität sowie die Kombination der beiden Konzepte. Zudem werden im Folgenden einige Aspekte der räumlichen Partitionierung detaillierter vorgestellt.

Daten- und Task-Parallelität

Datenparallelität lässt sich durch die Einteilung der Objekte eines Datensatzes nach einem oder mehreren der folgenden Kriterien erreichen:

- *Lage*: räumliche Partitionierung anhand der Lage, beispielsweise anhand der Flächen administrativer Gebiete
- *Geometrietyp*: nicht-räumliche Partitionierung anhand des Typs der Geometrie, beispielsweise Partitionierung in die Geometrietypen Punkt, Linie und Fläche
- *Attributwert(e)*: nicht-räumliche Partitionierung anhand der Werte eines oder mehrerer alphanumerischer Attribute, beispielsweise Partitionierung nach Objektart
- *Anzahl*: nicht-räumliche Partitionierung mit einer festen maximalen Anzahl von Objekten pro Partition

– *Level of Detail*: nicht-räumliche Partitionierung nach diskreten Skalenbereichen beziehungsweise Detailgraden, sogenannten Level of Detail (LoD), beispielsweise in einfache Klötzchenmodelle von Gebäuden und detaillierte Gebäudemodelle

Task-Parallelität lässt sich durch die Einteilung der auf den Daten auszuführenden *Funktionen* beziehungsweise *Operationen* erreichen.

Vor der detaillierten Diskussion der räumlichen Partitionierung wird im Folgenden zuerst ein Beispiel für die Parallelisierung der Integritätsprüfung eines Datensatzes vorgestellt. In Abbildung 7.1 ist links der Ausgangsdatensatz bestehend aus rechteckigen Gebäuden und linienhaften Straßen dargestellt. Die Partitionsgrenzen sind durch gestrichelte Linien markiert. Der Ausgangsdatensatz wird in einem ersten Schritt nach dem Attributwert des Attributs Objektart partitioniert. Das Ergebnis ist ein Datensatz der nur Gebäude enthält und ein Datensatz der nur Straßen enthält. Im nächsten Schritt wird der Datensatz mit den Gebäuden in räumliche Partitionen anhand der Lage der Objekte unterteilt. Die bis zu dieser Stelle durchgeführten Schritte basieren auf dem Konzept der Datenparallelität. Der letzte Schritt basiert auf dem Konzept der Task-Parallelität und führt die beiden Funktionen zur Überprüfung der Rechtwinkligkeit und von Mindestflächen parallel aus. Da jede der Funktionen auf mehreren räumlichen Partitionen parallel ausgeführt werden kann, wird eine Kombination aus Daten- und Task-Parallelität erreicht.

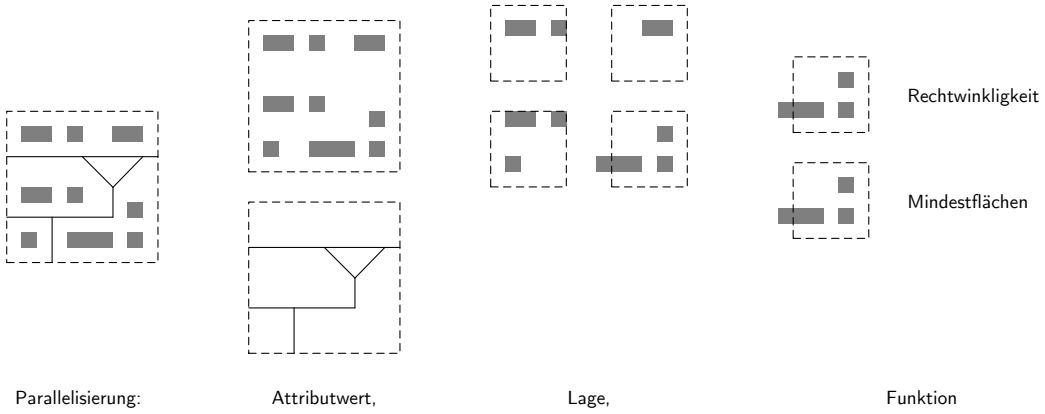


Abbildung 7.1: Parallelisierung der Integritätsprüfung eines Datensatzes

Räumliche Partitionierung

Die räumliche Partitionierung umfasst mehrere Aspekte, die im Folgenden diskutiert werden. Eine zweidimensionale *räumliche Partition* kann einer beliebig geformten Fläche entsprechen. Diese Definition wird in Abgrenzung zum Begriff der Kachel (engl. 'tile') getroffen, da dieser Begriff mit quadratischen oder rechteckigen Flächen als Teil eines regelmäßigen Gitters assoziiert ist. Eine räumliche Partition kann jedoch beispielsweise auch irregulär geformten administrativen Flächen oder Gebäudeblöcken entsprechen.

Berechnungen die alle Objekte in der Nachbarschaft eines Objekts berücksichtigen, wie beispielsweise die Ermittlung der minimalen Distanz zu benachbarten Objekten, können durch die Erweiterung einer Partition um einen *Randbereich* parallelisiert werden. Die Größe des Randbereichs hängt von den verwendeten Algorithmen und deren Schwellwerten ab. In der Regel reicht eine Vergrößerung der Partition um einen geeigneten Pufferwert aus. Wird der Pufferwert zu klein gewählt, dann kann dies zu Berechnungsfehlern führen. Wird dagegen der Pufferwert zu groß gewählt, dann müssen zu viele Daten analysiert und über das Kommunikationsnetz transportiert werden. Wie in Werder und Krüger (2009) ausgeführt ist, kann der Datenumfang des Randbereichs auch den Datenumfang der Partition weit übersteigen. Beispielsweise wird für die Simulation der Lärmausbreitung für eine $1\,km^2$ große quadratische Partition ein Puffer von 3 km benötigt, der zu einer Gesamtfläche des Randbereichs von $49\,km^2$ führt. Alle in diesem Randbereich enthaltenen Lärmquellen, wie Straßen- oder Industrielärm, tragen zum absoluten Lärmpegel der zentralen Partition bei.

Strategien für die räumliche Partitionierung

Berührt oder schneidet eine Objektgeometrie die Partitionsgrenze, die durch den linearen Rand der Partitionsfläche definiert ist, oder deren Randbereich, dann können verschiedene *Strategien* für die räumliche Partitionierung des Objekts angewendet werden. Schneiden Linien oder Flächen eine Partitionsgrenze, dann ist die Zuordnung des entsprechenden Objekts zu einer Partition nicht eindeutig. Beispielsweise schneidet in Abbildung 7.1 das Gebäude, das in der Partitionierung nach der Lage in der rechten unteren Partition liegt, die

beiden unteren Partitionen. Als Strategie für die Partitionierung wird in Abbildung 7.1 jedes Gebäude derjenigen Partition zugeordnet, in welcher der Schwerpunkt der Gebäudefläche liegt.

Einige Strategien für die Partitionierung sind in Tabelle 7.1 zusammengefasst. Diese Strategien lassen sich anhand von zwei Kriterien unterscheiden. Erstens kann eine Strategie dazu führen, dass Duplikate von Objekten erzeugt werden, d.h. ein Objekt ist mehrfach in aneinandergrenzenden Partitionen enthalten. Zweitens kann die Anwendung einer Strategie zu einer Änderung von Objektgeometrien führen. Im Folgenden werden die in Tabelle 7.1 aufgelisteten Strategien vorgestellt und diskutiert. Mit der Strategie *Geometrie* wird ein Objekt zu allen Partitionen hinzugefügt, deren Flächen die Objektgeometrie schneiden. Mit der Strategie *umschließendes Rechteck* (engl. 'bounding box') wird ein Objekt zu allen Partitionen hinzugefügt, deren Flächen das umschließende Rechteck der Objektgeometrie schneiden. Für diese Strategie wird also nicht die Geometrie selbst, sondern eine Ableitung davon genutzt. Durch die Verwendung des umschließenden Rechtecks kann der Fall auftreten, dass eine Partition ein Objekt enthält, aber dennoch dessen Fläche nicht die Objektgeometrie schneidet. Dieser Fall ist in Abbildung 7.2 dargestellt, in der die Objektgeometrie als graue Fläche und das umschließende Rechteck als graue Linie dargestellt sind. Das Objekt ist auch in der Partition mit der Nummer 3 enthalten, da das umschließende Rechteck in der Partition liegt. Mit der Strategie *Schwerpunkt* wird ein Objekt derjenigen Partition zugeordnet, deren Fläche den Schwerpunkt der Objektgeometrie enthält. Diese Strategie ist für Flächen besser geeignet als für Linien. Mit den Strategien *erster Punkt* und *letzter Punkt* wird ein Objekt derjenigen Partition zugeordnet, deren Fläche den ersten beziehungsweise letzten Punkt der Objektgeometrie enthält. Diese beiden Strategien sind für Linien besser geeignet als für Flächen. Mit der Strategie *Schnitt* wird schließlich die Objektgeometrie an den Partitionsgrenzen abgeschnitten und die jeweiligen Schnitte den entsprechenden Partitionen zugeordnet. Dieser Schnitt ist in Abbildung 7.2 anhand des bereits beschriebenen Beispiels dargestellt. Bei der Prozessierung der einzelnen Partitionen muss die durch den Schnitt veränderte Geometrie gegebenenfalls berücksichtigt werden. Beispielsweise verändert sich durch den Schnitt die Fläche der Objektgeometrie in den jeweiligen Partitionen. Hingegen sind beispielsweise keine Änderungen bei der Bestimmung minimaler Abstände zu anderen Objekten notwendig.

Strategie	Erzeugt Duplikate	Verändert Geometrie
Geometrie	■	□
Umschließendes Rechteck	■	□
Schwerpunkt	□	□
Erster Punkt	□	□
Letzter Punkt	□	□
Schnitt	■	■

Tabelle 7.1: Strategien zur räumlichen Partitionierung

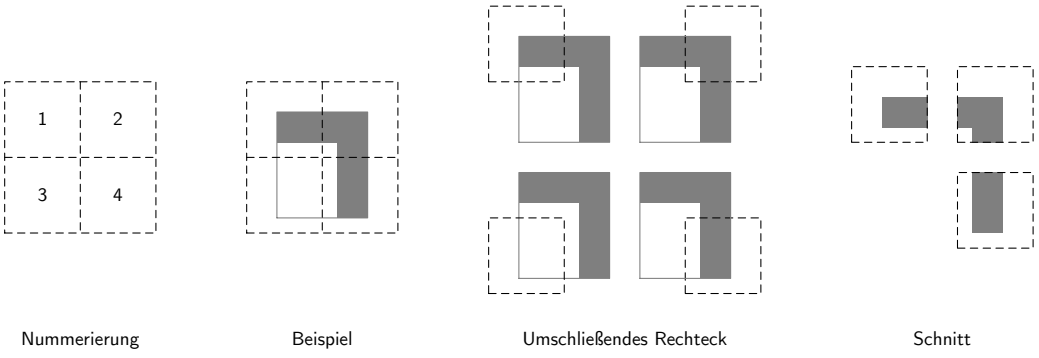


Abbildung 7.2: Beispiel für Strategien zur räumlichen Partitionierung

Aufteilen und Zusammenfügen

Jede Partitionierung besteht aus zwei Schritten. Im ersten Schritt, der *Aufteilung* (engl. 'split'), werden die Daten in Partitionen unterteilt. Dieser Schritt ist meist nicht anwendungsspezifisch und kann durch die Einteilung der Objekte bezüglich der vorgestellten Kriterien der Daten- oder Task-Parallelität durchgeführt werden. Sollen die Daten räumlich partitioniert werden, so kann eine der vorgestellten Strategien verwendet werden und gegebenenfalls ein Randbereich berücksichtigt werden. Im Anschluss an die Aufteilung können die Daten

dann parallel verarbeitet werden. Im zweiten Schritt, dem *Zusammenfügen* (engl. 'merge') werden die Daten der einzelnen Partitionen wieder zu einem einzelnen Datensatz vereint. Dieser Schritt ist bei Verwendung einer räumlichen Partitionierung anwendungsspezifisch, da der erzeugte Randbereich, veränderte Geometrien sowie Duplikate berücksichtigt werden müssen. Für nicht-räumliche Partitionierungen ist das Zusammenfügen meist nicht anwendungsspezifisch, da einfach die Objekte der einzelnen Partitionen aneinandergereiht werden müssen.

7.2 Verwendete Daten

Für eine aussagekräftige Untersuchung der Parallelisierung sind umfangreiche raumbezogene Datensätze notwendig. Für die Untersuchung werden drei verschiedene Datensätze mit flächenhaften Gebäudegrundrissen verwendet (siehe Tabelle 7.2). Die Datensätze sind dabei Beispiele für Open Data, d.h. für Daten die von Institutionen oder von Freiwilligen erhoben wurden und kostenlos zur Verfügung gestellt werden. Um mit entsprechend umfangreichen Datensätzen arbeiten zu können, werden unter anderem Daten des OpenStreetMap (Osm)-Projekts verwendet.

Datensatz	Anzahl Objekte	Anzahl Attribute
Chicago	820 154	10
Osm-Cz	2 792 601	4
Osm-Fr	31 782 920	

Tabelle 7.2: Objekte der verwendeten Daten

Der erste Datensatz enthält etwa 0,8 Millionen Gebäudegrundrisse der Stadt *Chicago* (USA), die von der Stadtverwaltung kostenlos zur Verfügung gestellt werden (Chicago, 2013). Der Originaldatensatz liegt im Shapefile-Format vor und verfügt über 40 alphanumerische Attribute, von denen die zehn in Tabelle 7.3 aufgelisteten Attribute ausgewählt werden. Ein grober Überblick über die Daten Chicagos ist in Abbildung 7.3 dargestellt. Der zweite Datensatz umfasst etwa 2,8 Millionen Gebäudegrundrisse *Tschechiens* (Osm-Cz) aus den Daten des OpenStreetMap-Projekts. Die Gebäudegrundrisse aus Osm enthalten lediglich vier alphanumerische Attribute, die abgesehen von den eindeutigen Kennziffern den Namen und den Typ der Gebäude umfassen. Eine vollständige Übersicht über den gesamten Objektartenkatalog des Osm-Projekts ist in OpenStreetMap (2013a) enthalten. Der dritte Datensatz enthält etwa 31,8 Millionen Gebäudegrundrisse *Frankreichs* (Osm-Fr), die wiederum aus den Daten des OpenStreetMap-Projekts entnommen sind. Dabei entstammen viele der Gebäudegrundrisse einem Import aus dem amtlichen französischen Kataster (OpenStreetMap, 2013d). Die Gebäudegeometrien Frankreichs weisen damit eine entsprechend hohe Qualität auf.

7.3 GeoAvro als Dateiformat für Hadoop

Räumliche Daten müssen für die Verarbeitung mit Hadoop und für die Speicherung im Hadoop Distributed File System (Hdfs) in der Regel aufgearbeitet werden. Häufig liegen räumliche Daten im Shapefile-Format der Firma Esri vor, welches in Unterabschnitt 7.3.1 beschrieben wird. Dieses Format weist jedoch einige Nachteile auf, die bei der Verwendung im Hadoop-Framework relevant sind. Deshalb wurde im Rahmen dieser Arbeit das Avro-Format (siehe Unterabschnitt 2.8.4) um die Speicherung räumlicher Daten erweitert. Die Erweiterung sowie die Konvertierung vom und ins Shapefile-Format werden detailliert in Unterabschnitt 7.3.2 vorgestellt.

7.3.1 ESRI Shapefile-Format

Das Shapefile-Format ist ein von Esri spezifiziertes Format zur Speicherung vektorieller Geodaten. Aufgrund seiner Einfachheit und der weitreichenden Unterstützung durch Geoinformationssysteme (Gis) und Programmbibliotheken wird das Shapefile-Format häufig zur Speicherung und zum interoperablen Austausch von Geodaten verwendet. Die Spezifikation des Shapefile-Formats ist in ESRI (1998) beschrieben. Im Folgenden werden einige wichtige Aspekte des Formats vorgestellt.

Das Shapefile-Format besteht aus mehreren einzelnen Dateien, die den gleichen Dateinamen aufweisen und sich nur durch ihre Dateiendungen unterscheiden. Drei Dateien müssen immer vorhanden sein, damit die Anforderungen des Formats erfüllt sind. In der Hauptdatei mit der Dateiendung .shp sind die Geometrien aufeinanderfolgend abgespeichert. Die Geometrien sind dahingehend beschränkt, dass in einer Hauptdatei alle Geometrien

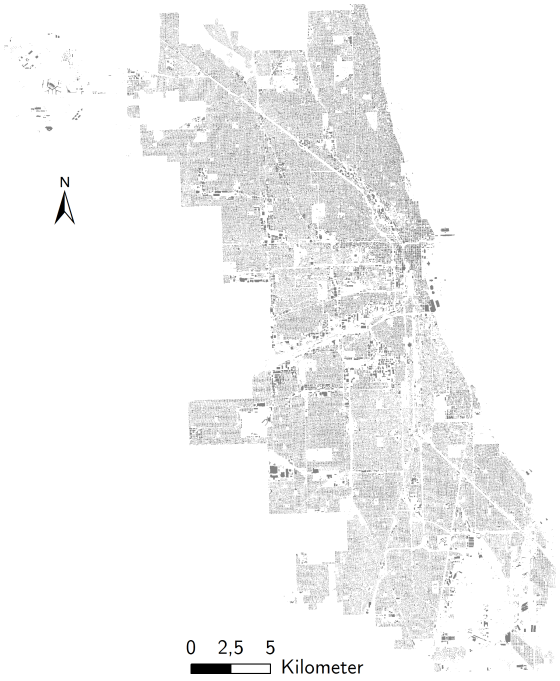


Abbildung 7.3: Gebäudegrundrisse der Stadt Chicago

denselben Typ aufweisen müssen. So können beispielsweise Linien und Flächen nicht in derselben Hauptdatei gespeichert werden. In der Indexdatei mit der Dateiendung .shx sind die Offsets der einzelnen Geometrien in der Hauptdatei abgespeichert, damit direkt auf eine beliebige Geometrie in der Hauptdatei zugriffen werden kann. In der dBase-Tabelle mit der Dateiendung .dbf sind die Fachdaten der einzelnen Geometrien aufeinanderfolgend abgespeichert. Das dBase-Format ist ein einfaches Dateiformat, das die Speicherung einer Datenbank in einer einzelnen Datei ermöglicht (dBase, 2012). Von den weiteren optionalen Dateien des Shapefile-Formats ist für diese Arbeit lediglich die Projektionsdatei mit der Dateiendung .prj von Interesse. In dieser wird das einheitliche Koordinatensystem der Geometrien im Well Known Text (WKT)-Format festgelegt (siehe Unterabschnitt 2.1.2).

Als Beispiel für die in diesem Abschnitt vorgestellten räumlichen Dateiformate werden die Gebäudegrundrisse der Stadt Chicago verwendet, die original im Shapefile-Format vorliegen. Das Schema des Shapefile-Formats besteht aus einer Auflistung der Namen und Typen der enthaltenen Attribute. Für das Beispiel sind in Tabelle 7.3 die Attribute, deren Bedeutung und die Dateiendung, in der die Attributwerte gespeichert sind, aufgelistet. Die Geometrien werden in der Hauptdatei abgespeichert und sind vom Typ Polygon, d.h. die Gebäudegrundrisse werden durch Flächen repräsentiert. Die verbleibenden Attribute sind in der dBase-Tabelle gespeichert und nehmen die Datentypen Long und Integer (Ganzzahl) sowie String (Text) an. Das Koordinatensystem der Daten ist in der Projektionsdatei gespeichert, deren Inhalt im Programm 7.1 aufgelistet ist.

```
PROJCS["NAD_1983_StatePlane_Illinois_East_FIPS_1201_Feet",GEOGCS["GCS_North_American_1983",DATUM["D
    _North_American_1983",SPHEROID["GRS_1980",6378137.0,298.257222101]],PRIMEM["Greenwich",0.0],
    UNIT["Degree",0.0174532925199433]],PROJECTION["Transverse_Mercator"],PARAMETER["False_Easting"
    ,984250.0],PARAMETER["False_Northing",0.0],PARAMETER["Central_Meridian",-88.33333333333333],
    PARAMETER["Scale_Factor",0.999975],PARAMETER["Latitude_Of_Origin",36.66666666666666],UNIT["
    Foot_US",0.3048006096012192]]
```

Programm 7.1: Projektion der Gebäudegrundrisse der Stadt Chicago im Well Known Text-Format

Das Shapefile-Format weist jedoch einige Nachteile auf, die im Kontext der Parallelisierung mit Hadoop relevant sind:

- Die zusammengehörenden räumlichen Daten sind in insgesamt vier Dateien enthalten. Im HDFS werden diese jedoch automatisch verteilt gespeichert (siehe Unterabschnitt 2.8.3), wodurch nicht mehr gewährleistet werden kann, dass die vier Dateien auf derselben DataNode gespeichert werden. Da das Prinzip der Lokalität dadurch nicht mehr vollständig berücksichtigt wird, müssen für die Verarbeitung der Daten gegebenenfalls umfangreiche Daten übertragen werden.

Attributname	Bedeutung	Attributtyp	Datei
—	Geometrie	Polygon	.shp
BLDG_ID	Eindeutige Kennziffer	Long	.dbf
BLDG_STATU	Status (Aktiv, geplant, zerstört)	String	.dbf
PRE_DIR1	Richtung der Straße für die Adresse	String	.dbf
ST_NAME1	Name der Straße für die Adresse	String	.dbf
ST_TYPE1	Typ der Straße für die Adresse	String	.dbf
NON_STANDA	Besondere Strukturen (Garage, Monument, ...)	String	.dbf
STORIES	Anzahl der Stockwerke	Long	.dbf
NO_OF_UNIT	Anzahl der Wohneinheiten	Long	.dbf
NO_STORIES	Anzahl der unterirdischen Stockwerke	Long	.dbf
YEAR_BUILT	Jahr der Errichtung	Integer	.dbf

Tabelle 7.3: Attribute des Shapefile-Schemas der Gebäudegrundrisse der Stadt Chicago

- Eine einzelne Datei des Shapefile-Formats kann nicht in mehreren Blöcken im HDFS gespeichert werden, da eine Unterteilung der Datei an Blockgrenzen von beispielsweise 64 MiB dazu führt, dass die komplette Datei nicht mehr gelesen werden kann. Dies ist darauf zurückzuführen, dass die Unterteilung nicht nativ vom Shapefile-Format unterstützt beziehungsweise von Hadoop bereitgestellt wird. So enthält der erste HDFS-Block einer unterteilten Datei den Header und eine entsprechende Anzahl an Bytes bis zur Füllung der maximalen Kapazität des Blocks. Die nachfolgenden HDFS-Blöcke enthalten die jeweils folgenden 64 MiB. Die Trennung erfolgt dabei rein anhand der Dateigröße ohne Rücksicht auf die Grenzen aufeinanderfolgender Objekte. Somit muss beim Kopieren der Dateien im Shapefile-Format die Blockgröße im HDFS gegebenenfalls so vergrößert werden, dass jede Datei komplett in einen einzelnen Block passt.
- Im Shapefile-Format kann jeweils nur ein Geometrietyp abgespeichert werden, d.h. es können beispielsweise nicht Punkte, Linien und Flächen in einer gemeinsamen Datei abgelegt werden.
- Nach ESRI (2009) ist die Dateigröße jeder Datei des Shapefile-Formats auf 2 GiB beschränkt. Dies entspricht etwa 70 Millionen Punkten beziehungsweise Stützpunkten für die Geometrietypen Linie und Fläche (ESRI, 2009). In der Praxis ist jedoch häufig die Größe der dBase-Tabelle der limitierende Faktor, da die Dateigröße bei einer hohen Anzahl an Attributen pro Objekt deutlich schneller an Umfang zunimmt als die Hauptdatei mit den zugehörigen Geometrien.
- Numerische Attribute werden nach ESRI (2009) im Textformat in der dBase-Tabelle gespeichert, was zu Rundungsfehlern bei der Konvertierung führen kann. Zudem sind die Attributnamen in der dBase-Tabelle auf zehn Zeichen und die Werte des Attributtyps String auf 254 Zeichen beschränkt.

7.3.2 GeoAvro

Zur Speicherung räumlicher Daten wird für diese Arbeit das Dateiformat GeoAvro verwendet, das auf dem Datenformat von Avro (siehe Unterabschnitt 2.8.4) basiert. Die räumlichen Daten können somit nicht nur durch das Hadoop MapReduce-Framework verarbeitet werden, sondern unabhängig von Hadoop auch mit allen von Avro unterstützten Programmiersprachen.

GeoAvro erweitert das Datenformat von Avro um die Möglichkeit Geometrien sowie deren Koordinatensystem zu speichern. Am Beispiel der Gebäudegrundrisse der Stadt Chicago aus Unterabschnitt 7.3.1 wird die Erweiterung im Folgenden vorgestellt. Das Schema des Beispiels ist im Programm 7.2 in JavaScript Object Notation (JSON)-Notation definiert. Die Auflistung von Feldern umfasst ein Attribut mit dem unveränderlichen Namen 'geometry' und dem unveränderlichen Typ bytes. Die Geometrie wird in einer Bytesequenz gespeichert, die entsprechend dem Well Known Binary (WKB)-Format (siehe Unterabschnitt 2.1.2) abgelegt ist. Dadurch kann die Geometrie in jeder Programmiersprache verarbeitet werden, die der Simple Feature Spezifikation (OGC Simple Feature, 2011) folgt. In der Dokumentation des Attributs (doc) ist schließlich das Koordinatensystem der Geometrien im WKT-Format (siehe Unterabschnitt 2.1.2) angegeben. Alle alphanumerischen Attribute aus Tabelle 7.3 werden entsprechend in das GeoAvro-Schema übertragen.

```
{
  "type": "record",
  "name": "chicago_buildings",
  "doc": "Autogenerated by Shp2GeoAvro",
  "fields": [
    {"name": "geometry", "type": "bytes", "doc": "PROJCS['NAD_1983_StatePlane_Illinois_East_FIPS_
        1201_Feet', GEOGCS['GCS_North_American_1983', DATUM['D_North_American_1983', SPHEROID['GRS
        _1980', 6378137.0, 298.257222101]], PRIMEM['Greenwich', 0.0], UNIT['degree', 0.
        0174532925199433295], AXIS['Longitude', EAST], AXIS['Latitude', NORTH]], PROJECTION['
        Transverse_Mercator'], PARAMETER['central_meridian', -88.33333333333333], PARAMETER['
        latitude_of_origin', 36.666666666666664], PARAMETER['scale_factor', 0.999975], PARAMETER['
        false_easting', 984250.0], PARAMETER['false_northing', 0.0], UNIT['foot_survey_us', 0.
        3048006096012192], AXIS['X', EAST], AXIS['Y', NORTH]]"},
    {"name": "BLDG_ID", "type": "long"},
    {"name": "BLDG_STATU", "type": "string"},
    {"name": "PRE_DIR1", "type": "string"},
    {"name": "ST_NAME1", "type": "string"},
    {"name": "ST_TYPE1", "type": "string"},
    {"name": "NON_STANDA", "type": "string"},
    {"name": "STORIES", "type": "long"},
    {"name": "NO_OF_UNIT", "type": "long"},
    {"name": "NO_STORIES", "type": "long"},
    {"name": "YEAR_BUILT", "type": "int"}
  ]
}
```

Programm 7.2: GeoAvro-Schema der Gebäudegrundrisse der Stadt Chicago

Da GeoAvro eine Erweiterung von Avro ist, zeichnet es sich durch dieselben Eigenschaften aus, die im Kontext von Hadoop essentiell sind. Im Container-Format können beliebig viele räumliche Objekte als einzelne Datei abgespeichert werden. Das Format ist weiterhin selbstbeschreibend, was insbesondere durch die Angabe der Projektion im GeoAvro-Schema gewährleistet wird. GeoAvro-Dateien sind teilbar und können durch Kompressionsverfahren im Dateiumfang reduziert werden. Zudem können räumliche Objekte aus GeoAvro-Dateien als Ein- und Ausgabe für die Map- und Reduce-Funktionen von Hadoop verwendet werden. Schließlich ist auch für GeoAvro die Erzeugung von spezifischen Datentypen optional, d.h. die räumlichen Objekte können auch mit einem generischen Ansatz verarbeitet werden.

Prinzipiell können GeoAvro-Dateien aus beliebigen Datenquellen erzeugt werden. Beispielsweise können Objekte aus einer räumlichen Datenbank ausgelesen und als GeoAvro-Datei gespeichert werden. In dieser Arbeit werden GeoAvro-Dateien aus Dateien im Shapefile-Format erstellt, beziehungsweise wiederum in diesem Format abgespeichert. In Abbildung 7.4 sind die beiden Richtungen der Konvertierung von GeoAvro- und Shapefile-Format dargestellt. Jeweils vier Dateien im Shapefile-Format im lokalen Dateisystem entsprechen dabei jeweils einer Datei im GeoAvro-Format im HDFS. Für die Konvertierung werden die beiden Programme 'Shp2GeoAvro' und 'GeoAvro2Shp' bereitgestellt, die auf der Kommandozeile ausgeführt und mit entsprechenden Parametern konfiguriert werden können. Insbesondere kann bei der Konvertierung der Dateien im Shapefile-Format in das GeoAvro-Format die Größe der Blöcke im HDFS für die Zieldatei angegeben werden. Da die Größe der Blöcke direkt der Größe der Splits für die Eingabe in die Map-Funktion entspricht, kann so die Anzahl der zu verwendenden Mapper beeinflusst werden.

Abbildung 7.4: Wechselseitige Konvertierung der Dateiformate Shapefile und GeoAvro

Durch die Verwendung des GeoAvro-Formats für die räumlichen Daten werden alle in Unterabschnitt 7.3.1 aufgelisteten Nachteile des Shapefile-Formats vermieden:

- Da die zusammengehörenden Daten im GeoAvro-Format in einer einzelnen Datei enthalten sind, wird das Prinzip der Lokalität vollständig berücksichtigt.
- Eine GeoAvro-Datei kann von Hadoop automatisch an den Blockgrenzen des HDFS geteilt werden.

– Das Schema von GeoAvro unterstützt beliebig viele Geometrietypen in einer einzelnen Datei, beispielsweise eine Mischung von Linien und Flächen.
– Die Dateigröße einer GeoAvro-Datei wird lediglich durch den verfügbaren Speicherplatz im HDFS beschränkt.
– Numerische Attribute werden im jeweiligen Datentyp abgespeichert, wodurch Rundungsfehler vermieden werden. Zudem sind weder Attributnamen noch Werte des Attributtyps String signifikant bezüglich der Anzahl der Zeichen beschränkt.

Ein weiterer positiver Effekt der Konvertierung der Dateien vom Shapefile-Format in das GeoAvro-Format ist die deutliche Verringerung des benötigten Speicherplatzes. In Tabelle 7.4 sind für die drei verwendeten Datensätze die Dateigrößen der beiden Formate gegenübergestellt. Die Dateigröße Shapefile bezieht sich dabei auf die Summe der Dateigrößen der vier zugehörigen Dateien. Bei einer geringen Anzahl an Attributen beträgt der Kompressionsfaktor etwa 20 %. Bei einer hohen Anzahl an Attributen fällt der Kompressionsfaktor geringer aus, da die Fachdaten dann einen erheblichen Anteil am Datenumfang aufweisen.

Daten	Ca. Anzahl Objekte	Anzahl Attribute	Dateigröße Shapefile [MiB]	Dateigröße GeoAvro [MiB]	Kompressionsfaktor [%]
Chicago	820 000	10	223	122	55
OSM-CZ	2 790 000	4	1911	380	20
OSM-FR	31 780 000	4	21 952	4524	21

Tabelle 7.4: Vergleich der Dateigrößen von Shapefile- und GeoAvro-Dateien

7.4 Anreicherung der Daten mit MapReduce

Um Integritätsbedingungen für die Gebäude der drei Datensätze aufstellen zu können, werden die Objekte zuerst um mehrere geometrische und topologische Maße angereichert. Die insgesamt 13 verwendeten Maße werden in Unterabschnitt 7.4.2 vorgestellt. Der Anreicherung geht eine Überprüfung der Geometrien in Unterabschnitt 7.4.1 voraus, um ungültige Geometrien vorab aus den Daten zu entfernen. Danach wird in Unterabschnitt 7.4.3 die Implementierung der Anreicherung als MapReduce-Workflow erläutert. Schließlich wird in Unterabschnitt 7.4.4 der Effekt der Parallelisierung anhand von Laufzeituntersuchungen analysiert.

7.4.1 Prüfung der Geometrien

Vor der Anreicherung wird erst eine Prüfung der Objektgeometrien durchgeführt, da für ungültige Geometrien keine Maße berechnet werden können.

Die Objektgeometrie der Gebäude ist flächenhaft und damit vergleichbar zur Flächennutzung in den Geobasisdaten. Demnach können die in Unterabschnitt 6.2.1 aufgestellten Kriterien auch für die Überprüfung der Gebäudegeometrien verwendet werden. Die Anzahl der Objekte sowie der fehlerhaften Objekte sind für die drei Datensätze in Tabelle 7.5 aufgelistet. Obwohl die Anzahl der fehlerhaften Objekte für die tschechischen OpenStreetMap-Daten am höchsten ist, so ist der prozentuale Anteil der Fehler vernachlässigbar.

Daten	Anzahl Objekte	Anzahl gültige Objekte	Anzahl fehlerhafte Objekte
Chicago	820 154	820 146	8
OSM-CZ	2 792 601	2 791 447	1154
OSM-FR	31 782 920	31 782 785	135

Tabelle 7.5: Objekte der drei Datensätze mit Gebäudegrundrissen

7.4.2 Geometrische und topologische Maße

Die einzelnen Objekte der drei Datensätze werden jeweils um elf geometrische und zwei topologische Maße angereichert.

Geometrische Maße

Von den elf verwendeten geometrischen Maßen (siehe Tabelle 7.6) zur Anreicherung der Objekte sind acht Maße bereits detailliert in Unterabschnitt 6.2.2 ausgeführt. Die drei bisher nicht verwendeten Maße werden im Folgenden weiter erläutert.

Typ	Attribut	Bemerkung
Direkt	NumVx	Anzahl der Stützpunkte
	MinVxDist	Minimale Distanz aufeinanderfolgender Stützpunkte
	MinVxAngle	Minimaler Winkel zwischen zwei aufeinanderfolgenden Liniensegmenten
	Area	Fläche
	Comp	Kompaktheit bezogen auf ein Quadrat
	FracDim	Fraktale Dimension
	Rect	Rechtwinkligkeit
Abgeleitet	MMinWidth	Minimaler Durchmesser des umschließenden angepassten Rechtecks
	MAzim	Azimut der längeren Seite des umschließenden angepassten Rechtecks
	MElong	Elongation des umschließenden angepassten Rechtecks
	MAreaDiff	Flächenverhältnis zum umschließenden angepassten Rechteck

Tabelle 7.6: Geometrische Maße zur Anreicherung der Objekte

Der *minimale Winkel zwischen zwei aufeinanderfolgenden Liniensegmenten* wird berechnet, um spitze Winkel in den Stützpunkten der Gebäudegeometrien zu ermitteln. Drei Beispiele für den minimalen Winkel sind in Abbildung 7.5 dargestellt. Der minimale Winkel zwischen zwei aufeinanderfolgenden Liniensegmenten wird dabei so normiert, dass dieser im Intervall $[0°, 180°]$ liegt.

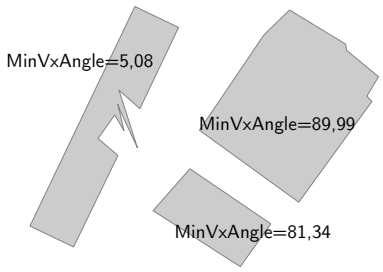


Abbildung 7.5: Minimaler Winkel zwischen zwei aufeinanderfolgenden Liniensegmenten

Ein weiteres Maß zur Überprüfung der Form ist die *Rechtwinkligkeit*. Die Form einer Gebäudegeometrie ist häufig durch gerade Liniensegmente charakterisiert, die durch rechte Winkel miteinander verbunden sind. Vergleichsweise seltene Formelemente sind dagegen Kreisbögen. In Abbildung 7.6 sind einige beispielhafte Objektgeometrien und deren Rechtwinkligkeit dargestellt. Für die Bestimmung der Rechtwinkligkeit wird dabei in jedem Stützpunkt der Objektgeometrie der Winkel zwischen den zwei anliegenden Liniensegmenten überprüft. Entspricht der Winkel innerhalb einer vorgegebenen Toleranz von 10° einem Vielfachen des Winkels 90°, dann wird der Winkel als rechtwinklig angesehen und die Differenz zu einem rechten Winkel auf den Wert 0 gesetzt. Ein Winkel von 180° wird dabei als Sonderfall eines rechten Winkels behandelt, da dieser zu einer Geraden durch den Stützpunkt führt. Anschließend wird die Winkeldifferenz vom Intervall $[0°, 45°]$ auf das Intervall $[1, 0]$ normiert. Die normierte Winkeldifferenz hat damit den Wert 1, falls im Stützpunkt der Winkel zwischen den zwei anliegenden Liniensegmenten innerhalb der Toleranz liegt. Dagegen ist die normierte Winkeldifferenz mit dem Wert von 0 minimal, wenn der Winkel zwischen den Liniensegmenten um 45° von einem Vielfachen des Winkels 90° abweicht. Die normierte Winkeldifferenz wird anschließend gewichtet, wobei das Gewicht aus dem Verhältnis der halbierten Summe der Länge der beiden anliegenden Liniensegmente zur Gesamtlänge des Umrings der Objektgeometrie ermittelt wird. Durch die Gewichtung werden lange Liniensegmente der Objektgeometrie höher gewichtet als kurze Liniensegmente und damit dem innenliegenden Winkel eine entsprechende Bedeutung zugemessen. Schließlich ergibt sich der Wert für die Rechtwinkligkeit aus der Summe der normalisierten und gewichteten Winkeldifferenzen im Intervall $[1, 0]$. Ein Wert von 1 entspricht dabei, innerhalb der vorgegebenen Winkeltoleranz, einer vollständig rechtwinkligen Objektgeometrie.

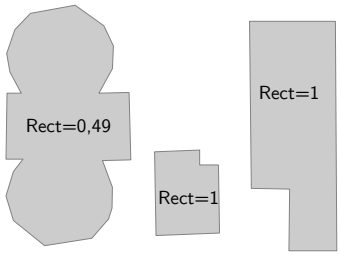


Abbildung 7.6: Rechtwinkligkeit

Um ermitteln zu können, ob die Ausrichtung von Gebäuden einer statistischen Regelmäßigkeit unterliegt, wird das *Azimut der längeren Seite des minimal umschließenden angepassten Rechtecks* für jedes Objekt bestimmt. Einige Beispiele für die so ermittelten Azimute sind in Abbildung 7.7 dargestellt. Die Azimute werden dabei auf das Intervall $[0°, 180°]$ normiert. Eine mögliche Hypothese für das Azimut ist die Annahme, dass Gebäude häufig in Nord-Süd- beziehungsweise Ost-West-Ausrichtung erbaut werden, da so der Lichteinfall für unterschiedliche Räume des Gebäudes optimiert werden kann[1].

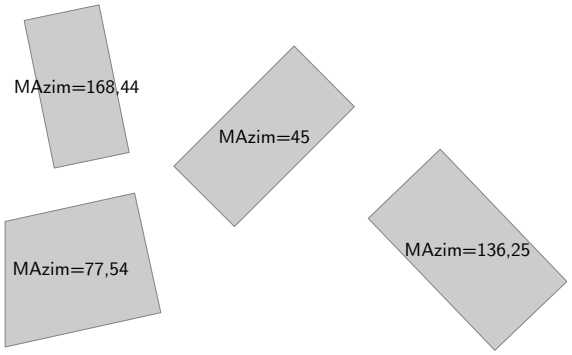


Abbildung 7.7: Azimut der längeren Seite des umschließenden Rechtecks

Topologische Maße

Für die Anreicherung der Objekte werden jeweils zwei topologischen Maße ermittelt (siehe Tabelle 7.7). Diese werden im Folgenden vorgestellt.

Typ	Attribut	Bemerkung
Direkt	IntArea	Größe der Schnittfläche mit benachbarten Objekten
Indirekt	MinDist	Minimale Distanz zu benachbarten Objekten

Tabelle 7.7: Topologische Maße zur Anreicherung der Objekte

Durch die Bestimmung der *Größe der Schnittfläche mit benachbarten Objekten* kann für jedes Objekt überprüft werden, ob dessen Objektgeometrie durch die Geometrien benachbarter Objekte überlappt wird. An die Gebäudeflächen wird die Forderung aufgestellt, dass diese überlappungsfrei sind, da ansonsten dieselbe Grundfläche durch mehrere Gebäude beansprucht würde, was im allgemeinen nicht möglich ist[2]. Aus den Werten für das Maß der *minimalen Distanz zu benachbarten Objekten* können mehrere Aussagen abgeleitet werden. Ist die Distanz gleich 0 m, dann weist das Objekt einen oder mehrere direkte Nachbarn auf. Geringe Werte für die Distanz, beispielsweise < 1 m, sind auffällig, da Gebäude meist entweder direkt benachbart sind oder einen gewissen funktional bedingten Mindestabstand aufweisen, beispielsweise aufgrund von Brandschutzbedingungen oder für einen Durchgang zwischen benachbarten Gebäuden. Aus größeren Werten für die Distanz lässt sich ableiten, ob ein Gebäude alleinstehend ist.

[1] Ein Beispiel für eine mögliche Anordnung der Räume in einem Gebäude ist Nord für Räume mit geringem Lichtbedarf (Schlafzimmer, Abstellraum), Ost für Räume mit Lichtbedarf am Morgen (Küche) sowie Süd und West für Räume mit Lichtbedarf am Nachmittag und Abend (Wohnzimmer, Esszimmer).

[2] Zu dieser Anforderung gibt es berechtigte Ausnahmen, die jedoch aus den Attributen der Objekte nicht ersichtlich sind und deshalb ausgeschlossen werden. Beispiele für Ausnahmen sind unterirdische Gebäude, wie beispielsweise Tiefgaragen, sowie Überstände, die erst ab einer gewissen Gebäudehöhe auftreten und dreidimensional betrachtet andere Gebäudekörper nicht schneiden.

7.4.3 Implementierung als MapReduce-Workflow

Um die Anreicherung der Daten als MapReduce-Workflow umsetzen zu können, ist eine Analyse der damit verbundenen Prozesse beziehungsweise Algorithmen notwendig. Die Anreicherung lässt sich prinzipiell in die beiden Schritte geometrische und topologische Anreicherung unterteilen. Für die Ermittlung der Werte der geometrischen Maße eines Objekts sind dabei keine Kenntnisse über andere Objekte des Datensatzes erforderlich, d.h. sie können für ein Objekt isoliert ermittelt werden. Beispielsweise kann die Fläche eines Objekts direkt aus dessen Geometrie bestimmt werden. Dagegen werden für die Ermittlung der Werte der topologischen Maße eines Objekts dessen räumliche Nachbarn benötigt, beispielsweise um den minimalen Abstand zum nächsten Nachbarn zu ermitteln.

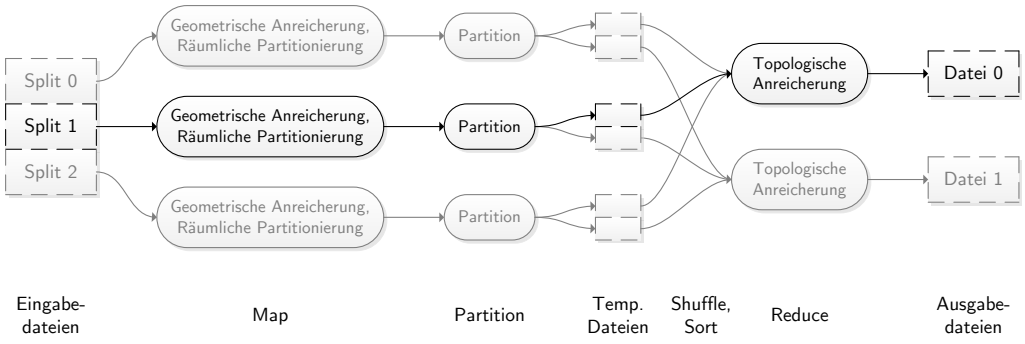


Abbildung 7.8: MapReduce-Workflow für die Anreicherung der Daten

Aus diesen Anforderungen lässt sich ableiten, dass eine räumliche Partitionierung lediglich für die topologische Anreicherung der Daten notwendig ist. Somit können die Daten effizient mit dem in Abbildung 7.8 abgebildeten MapReduce-Workflow angereichert werden. Dieser besteht aus fünf aufeinanderfolgenden Schritten, die dem allgemeinen Workflow von Hadoop MapReduce (siehe Unterabschnitt 2.8.3) entsprechen:

1. *Unterteilung der Eingabedateien*: Die Eingabedateien für den MapReduce-Workflow bilden Dateien im GeoAvro-Format (siehe Unterabschnitt 7.3.2). Die Unterteilung einer einzelnen oder mehrerer GeoAvro-Dateien in Splits basiert auf der Blockgröße im HDFS. In einem einzelnen Split müssen die einzelnen Objekte keiner räumlichen Sortierung unterliegen, sondern können beliebig angeordnet sein.

2. *Map*: Für jeden der insgesamt M Splits wird ein eigener Map-Task erzeugt, der die Objekte des Splits um die elf geometrischen Maße anreichert. Als Eingabe erhält die Map-Funktion jeweils ein Objekt im GeoAvro-Format als Schlüssel, sowie einen vernachlässigbaren Wert (siehe Tabelle 7.8).

Funktion	Eingabe	Ausgabe
map	$(AvroKey<GR>, NullWritable)$	$list(AvroKey<Utf8>, AvroValue<GR>)$
reduce	$(AvroKey<Utf8>, list(AvroValue<GR>))$	$list(AvroKey<GR>, NullWritable)$

Tabelle 7.8: Schlüssel und Werte der Map- und Reduce-Funktionen für die Anreicherung der Daten

   Die Anforderungen an das Avro-Objekt sind dabei so minimal wie möglich gehalten. Das Schema des Objekts muss den Anforderungen des GeoAvro-Formats genügen, d.h. ein Attribut muss das definierte Format der Geometrie erfüllen. Die weiteren Attribute des Schemas des Eingabeobjekts unterliegen keinerlei Einschränkungen, weder hinsichtlich der Anzahl der Attribute noch deren Typen. Dies wird ermöglicht durch die Verwendung des Datentyps Avro GenericRecord (GR) als Ein- und Ausgabe der Map-Funktion.

   Dazu wird bei der Initialisierung des MapReduce-Jobs das Schema der Eingabeobjekte um entsprechende Attribute für die Werte der geometrischen Maße erweitert. Beispielsweise wird das im Programm 7.2 aufgelistete Schema der Gebäudegrundrisse der Stadt Chicago automatisch um die elf in Tabelle 7.6 angegebenen geometrischen Maße erweitert. Die Ausgabeobjekte der Map-Funktion entsprechen dann dem Schema im Programm 7.3. Durch diesen generischen Ansatz ist der spezifizierte MapReduce-Workflow auf beliebige Geodaten übertragbar.

3. *Partition*: Bevor die Partitionierung im Detail vorgestellt wird, werden im Folgenden zuerst einige Rahmenbedingungen diskutiert. Wie in Unterabschnitt 2.8.5 erläutert, kann für räumliche Daten die Gesamtlaufzeit eines MapReduce-Jobs durch zwei Ansätze optimiert werden. Der erste Ansatz zielt darauf ab, die

```
{
  "type": "record",
  "name": "chicago_buildings",
  "doc": "Autogenerated by Shp2GeoAvro",
  "fields": [
    {"name": "geometry", "type": "bytes", "doc": "PROJCS['NAD_1983_StatePlane_Illinois_East_FIPS_
        1201_Feet', GEOGCS['GCS_North_American_1983', DATUM['D_North_American_1983', SPHEROID['GRS
        _1980', 6378137.0, 298.257222101]], PRIMEM['Greenwich', 0.0], UNIT['degree', 0.
        0174532925199433295], AXIS['Longitude', EAST], AXIS['Latitude', NORTH]], PROJECTION['
        Transverse_Mercator'], PARAMETER['central_meridian', -88.33333333333333], PARAMETER['
        latitude_of_origin', 36.666666666666664], PARAMETER['scale_factor', 0.999975], PARAMETER['
        false_easting', 984250.0], PARAMETER['false_northing', 0.0], UNIT['foot_survey_us', 0.
        3048006096012192], AXIS['X', EAST], AXIS['Y', NORTH]]"},
    {"name": "BLDG_ID", "type": "long"},
    {"name": "BLDG_STATU", "type": "string"},
    {"name": "PRE_DIR1", "type": "string"},
    {"name": "ST_NAME1", "type": "string"},
    {"name": "ST_TYPE1", "type": "string"},
    {"name": "NON_STANDA", "type": "string"},
    {"name": "STORIES", "type": "long"},
    {"name": "NO_OF_UNIT", "type": "long"},
    {"name": "NO_STORIES", "type": "long"},
    {"name": "YEAR_BUILT", "type": "int"},
    {"name": "NumVx", "type": "int", "doc": "Number of vertices"},
    {"name": "Area", "type": "double", "doc": "Area"},
    {"name": "Comp", "type": "double", "doc": "Compactness with a square as reference shape"},
    {"name": "FracDim", "type": "double", "doc": "Fractal dimension"},
    {"name": "Rect", "type": "double", "doc": "Weighted rectangularity"},
    {"name": "MinVxDist", "type": "double", "doc": "Minimum vertex distance"},
    {"name": "MinVxAngle", "type": "double", "doc": "Minimum angle between consecutive line
        segments at vertices"},
    {"name": "MMinWidth", "type": "double", "doc": "Minimum diameter of MER"},
    {"name": "MAzim", "type": "double", "doc": "Azimut of longer line segment of MER"},
    {"name": "MElong", "type": "double", "doc": "Elongation of MER"},
    {"name": "MAreaDiff", "type": "double", "doc": "Symmetric area difference [\%] between geometry
         and MER"},
    {"name": "IntArea", "type": "double", "doc": "Absolute intersection area with the geometries of
         other records"},
    {"name": "MinDist", "type": "double", "doc": "Minimum distance to the geometries of other
        records"}
  ]
}
```

Programm 7.3: GeoAvro-Schema der geometrisch angereicherten Gebäudegrundrisse der Stadt Chicago

Eingabedaten durch räumliche Partitionierungsverfahren möglichst gleichmäßig auf die einzelnen Tasks beziehungsweise MapReduce-Partitionen aufzuteilen. Alle in Unterabschnitt 2.8.5 vorgestellten Verfahren benötigen jedoch eine Vorprozessierung um die Partitionierung zu optimieren. Der zweite Ansatz unterscheidet, wann und wo räumliche Indexstrukturen zur Beschleunigung des räumlichen Zugriffs erstellt werden. Die Indexstrukturen können bei Bedarf auf den Knoten erzeugt werden, für den gesamten Datensatz verteilt erstellt werden oder gar nicht eingesetzt werden, indem alle Objekte in den verfügbaren Arbeitsspeicher geladen werden.

Für die Anreicherung der Daten wird eine von den vorgestellten Arbeiten abweichende Strategie zur Partitionierung eingesetzt. Es wird eine einfache Partitionierung verwendet, bei der die Daten durch ein gleichmäßiges räumliches Gitter unterteilt werden (siehe Unterabschnitt 2.8.5). Diese Unterteilung benötigt keine Vorprozessierung und führt somit nicht zu einer Verlängerung der Gesamtlaufzeit. Die Größe der Gitterzellen wird dabei so gewählt, dass die unterschiedlichen Laufzeiten einzelner MapReduce-Tasks nicht erheblich voneinander abweichen. In Abbildung 7.9 ist ein Beispiel angegeben, das die Motivation für die einfache Partitionierung verdeutlichen soll. Im Beispiel können drei Reduce-Tasks gleichzeitig ausgeführt werden, d.h. *R1*, *R2* und *R3* prozessieren simultan die ihnen zugeordneten Daten. Die Laufzeit der einzelnen Reduce-Tasks unterscheidet sich deutlich, jedoch nicht erheblich. Die Aufgabe des MapReduce-Frameworks besteht gerade darin, solche Laufzeitunterschiede durch die automatische Zuordnung der einzelnen MapReduce-Tasks auf die zur Verfügung stehenden DataNodes beziehungsweise Worker auszugleichen. Die Aufteilung der Daten sollte demnach nicht nur in Bezug auf die einzelnen MapReduce-Partitionen, sondern auch in Bezug auf die DataNodes betrachtet werden. Zusammenfas-

send kann also festgestellt werden, dass keine zusätzliche Vorprozessierung notwendig ist, da die vom MapReduce-Framework bereitgestellte Funktionalität hinreichend ist.

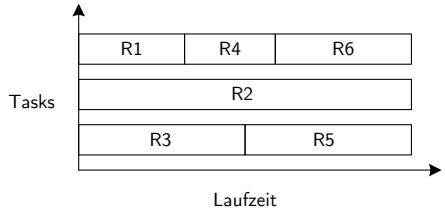


Abbildung 7.9: Laufzeit von Reduce-Tasks als Motivation für die Partitionierung

Die Größe der Gitterzellen wird zudem so gewählt, dass alle Objekte einer Gitterzelle in den verfügbaren Arbeitsspeicher eines Reduce-Tasks passen. Zusätzlich wird für alle Objekte einer Gitterzelle eine räumliche Indexstruktur für den schnellen Zugriff erstellt. Bezüglich der Geschwindigkeit werden also die in Unterabschnitt 2.8.5 vorgestellten Ansätze kombiniert.

Für die topologische Anreicherung der Objekte wird neben der Größe der Schnittfläche mit benachbarten Objekten auch die minimale Distanz zu benachbarten Objekten ermittelt. Da die Werte auch für Objekte an den Partitionsrändern korrekt bestimmt werden sollen, müssen die einzelnen Partitionen jeweils um einen Randbereich erweitert werden (siehe Abschnitt 7.1). Für die Anreicherung der Daten werden quadratische Partitionen mit einer Seitenlänge von 10 km verwendet. Da die minimale Distanz zu benachbarten Objekten bis zu einer Distanz von 1 km bestimmt werden soll, wird der Pufferwert beziehungsweise die Größe des Rands auf 2 km festgesetzt. So kann die Berechnung selbst für Gebäude die bis zu 1 km über den Partitionsrand hinaus ragen korrekt bestimmt werden (siehe Abbildung 7.10). Die so erstellten Partitionen haben eine Seitenlänge von 14 km und enthalten für alle Datensätze eine ausreichend geringe Anzahl an Objekten, die komplett in den Arbeitsspeicher der zugeordneten Reduce-Tasks geladen werden können.

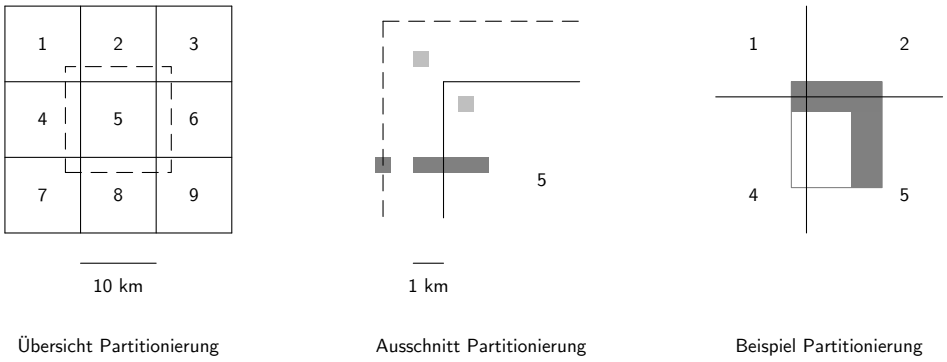


Abbildung 7.10: Partitionierung für die Anreicherung der Daten

Als geringfügige Modifikation des MapReduce-Workflows wird die Partitionierung anstatt in einem Schritt in zwei Schritten durchgeführt:

– Map: Die räumliche Partitionierung der Objekte ist Teil der Map-Funktion. Dies ist notwendig, da als Strategie für die räumliche Partitionierung das umschließende Rechteck der Objekte verwendet wird, wodurch Duplikate erzeugt werden (siehe Abschnitt 7.1). Dabei werden Objektduplikate entsprechend gekennzeichnet, damit sie beim Zusammenfügen der einzelnen Partitionen wieder aus den Daten entfernt werden können.

  Die Logik der räumlichen Partitionierung wird im Folgenden anhand des Beispiels in Abbildung 7.10 beschrieben. Die Grundlage für die Partitionierung bildet das Minimum Enclosing Rectangle (MER) der Objektgeometrie, welches die vier Partitionen 1, 2, 4 und 5 schneidet. Für jede der vier Partitionen wird ein Schlüssel-Wert-Paar von der Map-Funktion ausgegeben. Der Schlüssel ist jeweils ein String, der im UTF-8 Format kodiert ist (siehe Tabelle 7.8) und die Koordinaten der jeweiligen Partition in der Form *minX-minY-maxX-maxY* enthält. Der Wert ist jeweils das Objekt als GR im GeoAvro-Format. Dem Objekt wird jedoch ein zusätzliches boolesches Attribut TmpRecord hinzugefügt, das nur für die Partition falsch ist, in dem der Mittelpunkt des MER der Objektgeometrie liegt. Der

Mittelpunkt des MER in Abbildung 7.10 liegt in Partition 5, d.h. dass nur für diese Partition das Objekt als nicht temporär gekennzeichnet wird. Die Kennzeichnung wird schließlich in der Reduce-Funktion verwendet um die entstandenen Duplikate wieder aus den Daten entfernen zu können.

– Partition: Für die MapReduce-Partitionierung wird die Standardimplementierung verwendet. Diese unterteilt die Schlüssel-Wert-Paare der Map-Funktion anhand der Hashwerte der Schlüssel in R Partitionen. Die so entstehende Partitionierung ist charakterisiert durch ein diagonales Streifenmuster, das exemplarisch in Abbildung 7.11 dargestellt ist. In der Abbildung sind dabei gleichfarbige Objekte derselben räumlichen Partition zugeordnet.

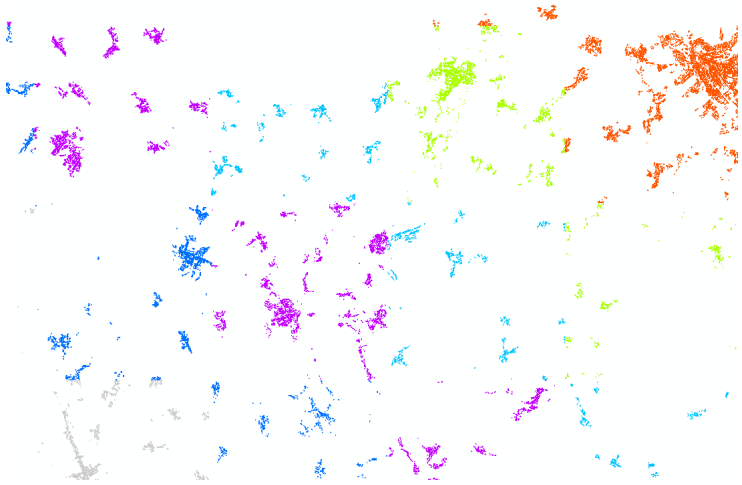


Abbildung 7.11: Räumliche Parititionierung basierend auf den Hashwerten der Schlüssel

4. *Shuffle und Sort*: Die beiden Schritte Shuffle und Sort entsprechen der Standardimplementierung von Hadoop MapReduce.

5. *Reduce*: Für jede der R MapReduce-Partitionen wird ein eigener Reduce-Task erzeugt, der die Objekte der Partition um die zwei topologischen Maße anreichert. Als Eingabe erhält die Reduce-Funktion den Schlüssel mit den Koordinaten der Partition und als Wert die Liste aller Objekte der Partition im GeoAvro-Format.

   In einem Reduce-Task werden als erstes die Objekte anhand des Attributs TmpRecord in zwei Listen unterteilt. Um benachbarte Objekte bestimmen zu können, wird für alle Objekte ein räumlicher Index basierend auf deren MER erstellt. Schließlich werden nur diejenigen Objekte um die topologischen Maße angereichert, die als nicht temporär gekennzeichnet sind. Die Ausgabe der Reduce-Funktion sind die nicht temporären Objekte im GeoAvro-Format als Schlüssel sowie vernachlässigbare Werte (siehe Tabelle 7.8). Hadoop erzeugt aus den Objekten automatisch eine Datei im Container-Format von Avro (siehe Unterabschnitt 2.8.4), die dann mit allen von Avro unterstützen Programmiersprachen weiterverarbeitet werden kann.

   Um die Laufzeit der topologischen Anreicherung der Objekte zu verringern, wird für die Bestimmung der minimalen Distanz zu benachbarten Objekten ein optimierter Algorithmus verwendet. Für die Anreicherung der Gebäude wird maximal eine Distanz von 1000 m zwischen benachbarten Objekten untersucht. Weist ein Objekt keine Nachbarn in diesem Umkreis auf, dann wird der Attributwert für die minimale Distanz auf einen vorgegebenen Wert gesetzt[3], beispielsweise auf -10 km. Die Standardimplementierung der Distanzbestimmung würde im räumlichen Index alle Objekte selektieren, die sich innerhalb der maximalen Distanz befinden. Für diese Objekte würde dann paarweise die Distanz zum Referenzobjekt bestimmt, um so das absolute Minimum der Distanz zu ermitteln. In einem dicht bebauten Gebiet befinden sich jedoch im Umkreis von 1 km um ein Gebäude eine Vielzahl an benachbarten Gebäuden. Deshalb wird zur Verringerung der Laufzeit die maximale Distanz iterativ erhöht. Zuerst werden die benachbarten Objekte in einer Distanz von 10 m ermittelt. Liefert diese Suche keine Ergebnisse, dann wird die Distanz schrittweise auf 40 m, 160 m, 640 m und schließlich auf 1000 m erweitert. Dadurch verringert sich die Anzahl der paarweise zu berechnenden Distanzen erheblich.

[3] Der vorgegebene Wert sollte einen deutlichen Abstand von den erwarteten Werten aufweisen, da die statistische Untersuchung der Ergebnisse in Abschnitt 7.5 ansonsten zu fehlerhaften Interpretationen aufgrund der Diskretisierung führen kann.

7.4.4 Analyse der Parallelisierung

Das Ziel des MapReduce-Workflows ist die effiziente Anreicherung umfangreicher Datenmengen. Im Folgenden wird zuerst das verwendete Hadoop-Cluster vorgestellt. Anschließend werden für die beiden Datensätze Untersuchungen der Laufzeit und damit der Leistungsfähigkeit der Parallelisierung durchgeführt.

Hadoop-Cluster am Institut für Kartographie und Geoinformatik

Für die Analyse der Parallelisierung wird das Hadoop-Cluster des Instituts für Kartographie und Geoinformatik (ikg) der Leibniz Universität Hannover verwendet. Dieses besteht aus fünf Servern und einem Switch (siehe Abbildung 7.12). Entsprechend der in Unterabschnitt 2.8.3 beschriebenen Architektur von Hadoop ist einem der Server die Rolle des *Masters* zugeordnet, d.h. auf diesem laufen die Prozesse für die NameNode sowie für den JobTracker ('hadoop-master'). Aufgrund der geringen Größe des Clusters wird auf einen eigenen Server für die Secondary NameNode verzichtet. Den restlichen vier Servern ist die Rolle der Slaves beziehungsweise *Worker* zugeordnet, d.h. auf diesen laufen jeweils die Prozesse für die DataNode sowie für den TaskTracker ('hadoop-worker1', ..., 'hadoop-worker4'). Die eigentliche Speicherung der Daten sowie deren Prozessierung wird demnach nur auf vier der fünf Server des Clusters durchgeführt. Ein Switch verbindet schließlich die Server des Hadoop-Clusters untereinander.

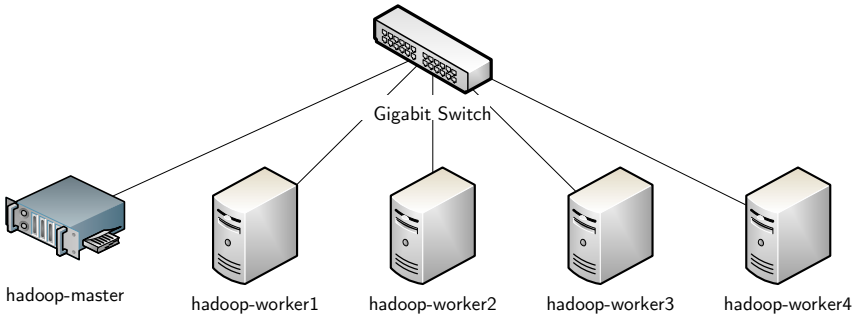


Abbildung 7.12: Aufbau des Hadoop-Clusters am ikg

Die Hardware der Server des Hadoop-Clusters des ikg ist in Tabelle 7.9 aufgelistet. Da der Master nur Verwaltungsaufgaben übernimmt, muss dieser nicht über besonders leistungsfähige Hardware verfügen. Jedoch wird für den Master redundant ausgelegte Hardware verwendet, wie beispielsweise doppelt vorhandene Netzteile, da dieser die entscheidende Schwachstelle (engl. 'single point of failure') des Gesamtsystems ist. Für die Worker wird dagegen auf einfache und günstige Massenprodukte (engl. 'commodity') gesetzt. Für diese ist die Ausfallwahrscheinlichkeit der Hardware bedeutend höher, jedoch können solche Ausfälle vom Master erkannt und entsprechend automatisch berücksichtigt werden. Die Worker verfügen über einen leistungsfähigen Vierkernprozessor sowie über einen entsprechend umfangreichen Arbeitsspeicher. Die Server verwenden dabei sogenannten Error-Correcting Code (Ecc) Arbeitsspeicher, durch den Fehler in der Hardware des Arbeitsspeichers erkannt werden können. Außerdem verfügen die Worker über jeweils drei Festplatten. Nach Abzug des Speicherplatzes für das Betriebssystem sowie einer entsprechenden Reserve folgt eine Gesamtspeicherkapazität von etwa 10,8 TiB. Bei einem Replikationsfaktor von drei beträgt der effektiv nutzbare Speicherplatz jedoch nur noch 3,6 TiB. Die Server sind schließlich über einen 24 Port Gigabit-Switch untereinander verbunden.

Typ	Prozessor	Arbeitsspeicher	Festplatten
Master	Intel Xeon Quad-Core 2,0 GHz	2 GiB Ecc	1 × 2 TiB
Worker	Intel Xeon Quad-Core 3,4 GHz	16 GiB Ecc	3 × 1 TiB

Tabelle 7.9: Hardware des Hadoop-Clusters am ikg

Die Konfiguration eines Hadoop-Clusters erfolgt im allgemeinen über mehrere Konfigurationsdateien. Für das Cluster des ikg ist in den Konfigurationsdateien ein Replikationsfaktor von drei für das Hdfs festgelegt, der jedoch individuell vom Benutzer für einzelne Dateien überschrieben werden kann. Die Anzahl der Map- und Reduce-Tasks pro TaskTracker ist auf jeweils drei festgelegt, d.h. das Cluster stellt insgesamt 12 Map- sowie 12 Reduce-Slots bereit. Auf einem einzelnen Worker können also gleichzeitig drei Map- und drei Reduce-Tasks ausgeführt werden. Jedem Task wird auf den Workern jeweils 1 GiB Arbeitsspeicher bereitgestellt. Dieser Wert kann wiederum bei Bedarf durch den Benutzer überschrieben werden.

Auf dem Hadoop-Cluster des ikg wird die stabile Hadoop-Version 1.0.3 eingesetzt. Diese Version wird für alle in dieser Arbeit durchgeführten Laufzeituntersuchungen verwendet.

Laufzeituntersuchung für die OpenStreetMap-Daten Tschechiens

Für die Anreicherung der Daten werden nun die Laufzeiten der entsprechenden MapReduce-Jobs ermittelt, wobei zwei Anforderungen berücksichtigt werden. Erstens wird auf dem Hadoop-Cluster des ikg während der Untersuchung jeweils nur ein einzelner MapReduce-Job ausgeführt, so dass die Laufzeit nicht durch andere Jobs verfälscht wird. Zweitens wird jede Laufzeit als arithmetisches Mittel von drei unabhängig ausgeführten Testläufen ermittelt. Für die Beurteilung des Effekts der Parallelisierung werden die in Abschnitt 2.7 vorgestellten Maße verwendet.

Die Laufzeiten der Anreicherung der OpenStreetMap-Daten Tschechiens (Osm-Cz) sind in Tabelle 7.10 für verschiedene Blockgrößen und für eine verschiedene Anzahl von Map- und Reduce-Tasks aufgelistet. Die Laufzeit der sequentiellen Ausführung des MapReduce-Jobs beträgt 7 Minuten und 38 Sekunden. Zur Bestimmung der sequentiellen Laufzeit müssen einige Bedingungen erfüllt sein. Erstens muss die gesamte Eingabedatei in einem einzigen Hdfs-Block enthalten sein. Nach Tabelle 7.4 haben die Daten eine Größe von 380 MiB, weshalb bei einer Blockgröße von 512 MiB die gesamten Datei in einem Block abgelegt werden kann. Der Replikationsfaktor wird entsprechend auf 1 gesetzt, da durch eine verteilte Speicherung im Dateisystem keine Verringerung der sequentiellen Laufzeit erreicht werden kann[4]. Durch die Speicherung in einem einzigen Block folgt automatisch, dass nur ein Map-Task ausgeführt wird. Die zweite Bedingung zur Bestimmung der sequentiellen Laufzeit ist, dass wiederum nur ein Reduce-Task ausgeführt wird, was durch den Benutzer manuell festgelegt werden kann.

Blockgröße [MiB]	Map-Tasks	Reduce-Tasks	Laufzeit [min:sec]	Speedup
512	1	1	7:38	—
64	6	1	3:46	2,0
		4	2:32	3,0
		16	1:59	3,8
		32	2:14	3,4
64	6	16	1:59	3,8
32	12		1:41	4,5
16	24		1:51	4,1

Tabelle 7.10: Laufzeit der Anreicherung der Daten für OSM-CZ

Zuerst wird im Folgenden der Einfluss der Anzahl der Reduce-Tasks auf die Gesamtlaufzeit untersucht. Dazu wird die Standardgröße eines Hdfs-Blocks von 64 MiB verwendet. Bei einer Dateigröße von 380 MiB entstehen entsprechend sechs Splits, was wiederum sechs parallel ausgeführten Map-Tasks entspricht. Wird nur ein Reduce-Task ausgeführt, dann wird ein Speedup von 2,0 erreicht, d.h. der Job wird doppelt so schnell abgearbeitet. Eine weitere Erhöhung der Anzahl der Reduce-Tasks führt zu einer weiteren Erhöhung des Speedups auf bis zu 3,8 bei 16 Reduce-Tasks. Wird die Anzahl der Reduce-Tasks jedoch darüber hinaus auf 32 erhöht, so verringert sich der erreichte Speedup wieder. Dies ist auf einen erhöhten Aufwand für die Verteilung der Daten und für die Synchronisation der Tasks zurückzuführen.

Außerdem wird der Einfluss der Anzahl der Map-Tasks auf die Gesamtlaufzeit untersucht. Dazu werden die Eingabedaten mit unterschiedlichen Blockgrößen im Hdfs abgelegt. Die Anzahl der Reduce-Tasks wird auf 16 festgelegt, da für diese Anzahl der Speedup in der vorhergehenden Untersuchung maximal ist. Bei einer Blockgröße von 32 MiB werden 12 Map-Tasks ausgeführt, wodurch der Speedup auf 4,5 erhöht werden kann. Eine weitere Verringerung der Blockgröße auf 16 MiB führt wiederum zu einem erhöhten Aufwand und dadurch zu einem geringeren Speedup.

Bei der Laufzeituntersuchung für die Anreicherung der OpenStreetMap-Daten Tschechiens (Osm-Cz) wird die optimale Kombination demnach bei 12 Map- und 16 Reduce-Tasks erreicht. Mit dem Amdahlschen Gesetz (siehe Gleichung 2.36) kann dann der sequentielle Bruchteil des Programms nach Gleichung 7.1 abgeschätzt

[4] Dies gilt unter der Annahme, dass alle Worker gleich leistungsfähig sind und das Cluster keine weiteren MapReduce-Jobs prozessieren muss. Diese Annahme ist bei den vorgenommenen Laufzeituntersuchungen erfüllt.

werden. Dazu wird die Anzahl der Prozessoren p auf den Mittelwert der gleichzeitig ausgeführten Map- und Reduce-Tasks gesetzt. Der sequentielle Bruchteil entspricht demnach 16 %.

$$\text{Mit } p = 14: \; 4,5 = \frac{1}{f + \frac{1-f}{14}} \;\Rightarrow\; f = 0,16 \tag{7.1}$$

Wird dagegen eine Kombination von 24 Map- und 16 Reduce-Tasks gewählt, dann erhöht sich der sequentielle Bruchteil des Programms nach Gleichung 7.2 auf 20 %, was auf den erhöhten Aufwand für die 12 zusätzlichen Map-Tasks zurückzuführen ist.

$$\text{Mit } p = 20: \; 4,1 = \frac{1}{f + \frac{1-f}{20}} \;\Rightarrow\; f = 0,20 \tag{7.2}$$

Zusammenfassend lässt sich also feststellen, dass das Ahmdalsche Gesetz die Laufzeiten beziehungsweise den Speedup nicht optimal bestimmen kann, da der sequentielle Bruchteil nicht konstant ist. Die Bestimmung der optimalen Anzahl an Map- und Reduce-Tasks ist demnach nicht mit einer einfachen Formel möglich, sondern hängt von den zu verarbeitenden Daten sowie den Eigenschaften der Prozessierung ab. Eine weitere wichtige Rolle spielt die momentane Auslastung des Clusters.

Laufzeituntersuchung für die OpenStreetMap-Daten Frankreichs

Die Laufzeiten der Anreicherung der OpenStreetMap-Daten Frankreichs (Osm-Fr) werden analog bestimmt und sind in Tabelle 7.11 für eine verschiedene Anzahl an Reduce-Tasks aufgelistet. Auf die Bestimmung der Laufzeit der sequentiellen Ausführung wird dabei verzichtet, da die Eingabedaten mit einer Größe von 4,5 GiB bedeutend umfangreicher sind (siehe Tabelle 7.4). Als Referenz für die Bestimmung des Speedups wird deshalb die Gesamtlaufzeit mit 1 Reduce-Task bestimmt. Diese beträgt 45 Minuten und 35 Sekunden. Durch eine Erhöhung der Anzahl der Reduce-Tasks auf 128 kann der Speedup relativ um den Faktor 2,8 erhöht werden. Eine weitere Erhöhung der Anzahl der Reduce-Tasks führt zu einer Verringerung des relativen Speedups, was wiederum auf einen erhöhten Aufwand für die Verteilung der Daten und für die Synchronisation der Tasks zurückzuführen ist.

Blockgröße [MiB]	Map-Tasks	Reduce-Tasks	Laufzeit [min:sec]	Relativer Speedup
64	81	1	45:35	—
		64	21:01	2,2
		128	16:31	2,8
		256	16:51	2,7

Tabelle 7.11: Laufzeit der Anreicherung der Daten für OSM-FR

Anhand der beiden Datensätze kann die Frage beantwortet werden, wie gut die Anreicherung der Daten skaliert. Dazu sind in Tabelle 7.12 die Laufzeiten sowie einige Eigenschaften der beiden Datensätze aufgelistet. Grundsätzlich kommen zwei Eigenschaften für den Vergleich in Frage. Die erste Eigenschaft ist die Anzahl der gültigen Objekte, d.h. bei der Anreicherung der Daten Frankreichs müssen etwa 11 mal mehr Objekte prozessiert werden. Die zweite Eigenschaft ist die Fläche die theoretisch prozessiert werden muss, da diese ein Hinweis für die Größenordnung der Anzahl der räumlichen Partitionen liefert. Im Vergleich zur Fläche Tschechiens ist die Fläche Frankreichs um etwa 9 mal größer. Interessanterweise ist die minimale Laufzeit der Anreicherung bei den Daten Frankreichs etwa um den Faktor 10 länger als bei den Daten Tschechiens, und liegt damit zwischen den Werten der beiden genannten Eigenschaften. Die Anreicherung skaliert damit annähernd linear, d.h. bei einer Verzehnfachung der Daten verzehnfacht sich auch die minimale Laufzeit. Deutlich unterschiedlich ist jedoch die Anzahl der Map- und Reduce-Tasks die für diese Skalierung benötigt werden, d.h. dass für größere Datenmengen auch eine deutlich höhere Anzahl an MapReduce-Tasks eingeplant werden sollte.

7.5 Bestimmung der Häufigkeit von Attributwerten mit MapReduce

Die Häufigkeit von Attributwerten ist ein wichtiges Element der deskriptiven Statistik. Sie bildet beispielsweise die Grundlage für Stabdiagramme und Histogramme (siehe Unterabschnitt 2.5.1). Durch die Bestimmung der Häufigkeit der Werte eines alphanumerischen Attributs kann insbesondere bei umfangreichen Daten ein guter

Kriterium	Osm-Cz	Osm-Fr	Faktor
Anzahl der gültigen Objekte	2 791 447	31 782 785	11,4
Fläche des Staats [km^2]	78 864	668 763	8,5
Minimale Laufzeit [min:sec]	1:41	16:31	9,8
Anzahl Map-Tasks	12	81	6,8
Anzahl Reduce-Tasks	16	128	8,0

Tabelle 7.12: Vergleich der Eigenschaften der Daten für OSM-CZ und OSM-FR

Überblick über die Verteilung der Werte gewonnen werden. MapReduce ist hervorragend für die verteilte Bestimmung der Häufigkeit von Attributwerten geeignet, wie der in Unterabschnitt 7.5.1 vorgestellte MapReduce-Workflow aufzeigt. In Unterabschnitt 7.5.2 wird analog zur Anreicherung der Daten auch für diesen Workflow eine Analyse der Parallelisierung durchgeführt. Die Auswertung der Ergebnisse wird schließlich in Abschnitt 7.6 bis hin zur Aufstellung von Integritätsbedingungen vorgestellt.

7.5.1 Implementierung als MapReduce-Workflow

Die Bestimmung der Häufigkeit von Attributwerten benötigt keinen räumlichen Bezug, da lediglich alphanumerische Attribute ausgewertet werden. Der für die Bestimmung der Häufigkeit in Abbildung 7.13 abgebildete MapReduce-Workflow basiert deshalb auf einfachen Erweiterungen der Schritte des in Unterabschnitt 2.8.3 vorgestellten allgemeinen MapReduce-Workflows von Hadoop.

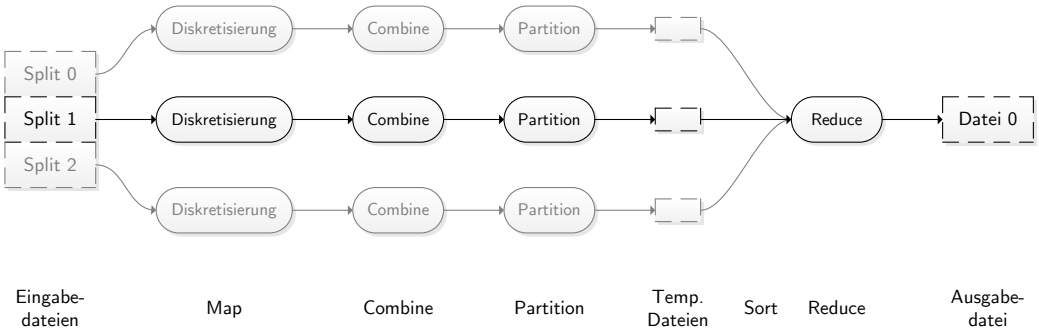


Abbildung 7.13: MapReduce-Workflow für die Bestimmung der Häufigkeit von Attributwerten

Der MapReduce-Workflow für die Bestimmung der Häufigkeit von Attributwerten besteht aus den folgenden sechs Schritten, wobei bereits in Unterabschnitt 7.4.3 detailliert ausgeführte Aspekte entsprechend verkürzt wiedergegeben werden:

1. *Unterteilung der Eingabedateien*: Die Eingabedateien für den MapReduce-Workflow bilden Dateien im GeoAvro-Format (siehe Unterabschnitt 7.3.2).
2. *Map*: Für jeden der insgesamt M Splits wird ein eigener Map-Task erzeugt, der die Werte eines Attributs diskretisiert beziehungsweise die Klasseneinteilung in die einzelnen Bereiche durchführt. Als Eingabe erhält die Map-Funktion jeweils ein Objekt im Avro-Format als Schlüssel, sowie einen vernachlässigbaren Wert (siehe Tabelle 7.13).

Funktion	Eingabe	Ausgabe
map	$(AvroKey<GR>, NullWritable)$	$list(DoubleWritable, LongWritable)$
reduce	$(DoubleWritable, list(LongWritable))$	$list(DoubleWritable, LongWritable)$

Tabelle 7.13: Schlüssel und Werte der Map- und Reduce-Funktionen für die Bestimmung der Häufigkeit von Attributwerten

Die Anforderungen an das Avro-Objekt sind wiederum so minimal wie möglich gehalten. Entscheidend ist lediglich, dass das zu diskretisierende alphanumerische Attribut mit dem korrekten Namen und Da-

tentyp im Schema enthalten ist, d.h. es können sowohl Objekte im allgemeinen Avro-Format als auch im GeoAvro-Format verarbeitet werden. Der generische Ansatz basiert auf zwei Design-Entscheidungen. Erstens wird der Datentyp GR als Eingabe der Map-Funktion verwendet, der keine Anforderungen an das Schema stellt. Zweitens wird das Eingabeschema durch eine Projektion auf das zu diskretisierende Attribut reduziert, d.h. das Schema enthält lediglich ein Attribut. Die Projektion eines Avro-Schemas wird in Unterabschnitt 2.8.4 erläutert und an dieser Stelle nochmals anhand eines Beispiels vorgestellt. Als Beispiel soll für das Attribut Area der Gebäudegrundrisse der Stadt Chicago die Häufigkeit der Attributwerte bestimmt werden. Die angereicherten Daten sind im GeoAvro-Format und weisen insgesamt 24 Attribute auf (siehe Programm 7.3). Für die Diskretisierung wird jedoch nur ein Attribut benötigt, weshalb zum lesen der Daten das Schema im Programm 7.4 verwendet wird, bei dem alle nicht benötigten Attribute entsprechend nicht im Schema enthalten sind.

```
{
  "type": "record",
  "name": "chicago_buildings",
  "doc": "Autogenerated by Shp2GeoAvro",
  "fields": [
    {"name": "Area", "type": "double", "doc": "Area"},
  ]
}
```

Programm 7.4: GeoAvro-Schema des Attributs Fläche der Gebäudegrundrisse der Stadt Chicago

Je nach Datentyp des zu diskretisierenden Attributs kann die Map-Funktion mit verschiedenen Parametern konfiguriert werden (siehe Tabelle 7.14). Der Datentyp des Attributs entscheidet zudem über den Datentyp des Schlüssels der Ausgabe der Map-Funktion sowie über den Schlüssel der Ein- und Ausgabe der Reduce-Funktion. Soll die Häufigkeit der Attributwerte eines boolesches Attributs bestimmt werden, so sind keine Parameter notwendig. Die Map-Funktion erstellt für jeden Wert der 'wahr' ist ein Schlüssel-Wert-Paar vom Datentyp (*BooleanWritable*, *LongWritable*) mit den Werten (*true*, *1*). Entsprechend erstellt die Map-Funktion für Werte die 'falsch' sind ein Paar (*false*, *1*). Die Werte von Attributen mit dem Datentyp String können in der Map-Funktion optional in Klein- beziehungsweise Großschreibung konvertiert werden, um unabhängig von unterschiedlichen Schreibweisen zu sein. Die Map-Funktion erstellt dann Schlüssel-Wert-Paare vom Datentyp (*Text*, *LongWritable*) mit den Werten (*konvertierter String*, *1*).

Datentyp	Parameter	Datentyp Schlüssel
Boolean	—	BooleanWritable
String	— / Klein- / Großschreibung	Text
Integer	Bereichsgröße	DoubleWritable
Long	Bereichsgröße	DoubleWritable
Double	Bereichsgröße, Genauigkeit	DoubleWritable

Tabelle 7.14: Datentypen für die Diskretisierung

Bei den drei numerischen Datentypen kann als Parameter die Bereichsgröße (engl. 'bin size') angegeben werden. Die einzelnen Attributwerte werden dann entsprechend Gleichung 7.3 diskretisiert, wobei der Operator % dem Modulo-Operator entspricht. Das Ergebnis wird als Schlüssel-Wert-Paar vom Datentyp (*IntWritable / LongWritable / DoubleWritable*, *LongWritable*) mit den Werten (*diskretisierter Wert*, *1*) ausgegeben. Der diskretisierte Wert liegt demnach immer in der Mitte eines Bereichs, beispielsweise wird der numerische Wert 35 bei einer Bereichsgröße von 50 in den diskretisierten Wert 25 konvertiert. Beim Datentyp Double kann zusätzlich die Genauigkeit (engl. 'precision') als Parameter angegeben werden, mit dem die Anzahl der signifikanten Nachkommastellen gesetzt werden kann[5].

$$\text{Diskretisierter Wert} = \text{Wert} - (\text{Wert} \,\%\, \text{Bereichsgröße}) + (\text{Bereichsgröße} \,/\, 2) \tag{7.3}$$

[5] Die Genauigkeit wird automatisch aus der der Bereichsgröße abgeleitet, die als Text übergeben wird. Hat diese beispielsweise den Wert 0,10, dann werden die diskretisierten Werte automatisch mit zwei signifikanten Nachkommastellen ausgegeben.

3. *Combine*: Der Combine-Schritt entspricht einem lokalen Reduce, bei dem die Ergebnisdaten eines einzelnen Map-Tasks zusammengefasst werden.
4. *Partition*: Für die MapReduce-Partitionierung wird die Standardimplementierung verwendet. Diese unterteilt die Schlüssel-Wert-Paare der Map-Funktion anhand der Hashwerte der Schlüssel in R Partitionen. Da die Anzahl der Partitionen fest auf den Wert 1 gesetzt ist, führt die Partitionierung zu keiner Veränderung der Daten.
5. *Shuffle und Sort*: Die beiden Schritte Shuffle und Sort entsprechen der Standardimplementierung von Hadoop MapReduce.
6. *Reduce*: Da die Häufigkeit der Attributwerte global für alle Objekte eines Datensatzes bestimmt werden soll, wird lediglich ein einziger Reduce-Task erzeugt. Dieser summiert die Werte für jeden Schlüssel auf und bestimmt damit die absolute Häufigkeit jedes (diskretisierten) Attributwerts.

7.5.2 Analyse der Parallelisierung

Für die Bestimmung der Häufigkeit von Attributwerten werden wiederum die Laufzeiten der entsprechenden MapReduce-Jobs ermittelt. Dabei werden die in Unterabschnitt 7.4.4 vorgestellten Anforderungen an die Laufzeitbestimmung eingehalten.

Die Laufzeiten für die Bestimmung der Häufigkeit von Attributwerten für die OpenStreetMap-Daten Tschechiens (Osm-Cz) und Frankreichs (Osm-Fr) sind in Tabelle 7.15 für eine variable Anzahl an Eingabedateien aufgelistet. Die Anzahl der Eingabedateien folgt direkt aus der Angabe der Anzahl der Reduce-Tasks, die bei der Anreicherung der Daten verwendet werden. Bei der Verwendung eines einzelnen Reduce-Tasks für die Anreicherung ist die Laufzeit für die Bestimmung der Häufigkeit von Attributwerten minimal. Dies ist auf die geringere Anzahl an Map-Tasks und somit einen geringeren Mehraufwand für die Verteilung der Daten und für die Synchronisation der Tasks zurückzuführen.

Daten	Anzahl Dateien	Blockgröße [MiB]	Map-Tasks	Reduce-Tasks	Laufzeit [min:sec]
Osm-Cz	1	64	10	1	0:34
	16		18		0:37
Osm-Fr	1	64	117	1	1:21
	128		212		2:12

Tabelle 7.15: Laufzeit der Bestimmung der Häufigkeit von Attributwerten der Daten für OSM-CZ und OSM-FR

Wichtiger ist es jedoch an dieser Stelle die Gesamtlaufzeit der Anreicherung und der Bestimmung der Häufigkeit der Attributwerte zu betrachten. Werden die Laufzeiten der beiden MapReduce-Jobs kombiniert, dann ist diese minimal, wenn die Laufzeit der Anreicherung minimal ist. Werden demnach mehrere MapReduce-Jobs nacheinander ausgeführt, dann ist eine Abwägung der Anzahl der Map- und Reduce-Tasks aller Jobs wichtiger als die Reduzierung der Laufzeit eines einzelnen Jobs.

	Anreicherung			Bestimmung Häufigkeit			
Daten	Map	Reduce	Laufzeit	Map	Reduce	Laufzeit	Kombinierte Laufzeit [min:sec]
Osm-Cz	6	1	3:46	10	1	0:34	4:20
	12	16	1:41	18		0:37	2:18
Osm-Fr	81	1	45:35	117	1	1:21	46:56
	81	128	16:31	212		2:12	18:43

Tabelle 7.16: Kombinierte Laufzeit der Daten für OSM-CZ und OSM-FR

7.6 Statistik und Klassenbeschreibung

Die in Abschnitt 7.5 bestimmten Häufigkeiten für die geometrischen und topologischen Maße der Gebäude können durch Methoden der deskriptiven Statistik und explorativen Datenanalyse untersucht werden (siehe

Abschnitt 2.5). Das Ziel der Untersuchung sind einerseits Integritätsbedingungen, mit denen die charakteristischen Eigenschaften der Objekte eindeutig beschrieben werden können. Andererseits zielt die Untersuchung auch auf die klassische Definition des Data Mining ab, nämlich die Entdeckung von interessanten Mustern und Wissen in umfangreichen Daten (siehe Abschnitt 2.6).

Zuerst werden die Werte für die geometrischen Maße in Unterabschnitt 7.6.1 für die drei Datensätze zusammen untersucht. Anschließend werden in Unterabschnitt 7.6.2 die topologischen Maße untersucht. In Unterabschnitt 7.6.3 wird zudem exemplarisch die Bestimmung von Häufigkeiten für alphanumerische Attributwerte vorgestellt. Aus der Untersuchung der geometrischen und topologischen Maße folgen Integritätsbedingungen, die in Unterabschnitt 7.6.4 zur Bereinigung der drei Datensätze genutzt werden.

7.6.1 Geometrische Maße

Die Werte und Verteilungen der elf verwendeten geometrischen Maße werden im Folgenden basierend auf ihrer Definition in Unterabschnitt 7.4.2 analysiert. Dabei werden die drei Datensätze Chicago, Osm-Cz und Osm-Fr gemeinsam untersucht, um Unterschiede aufdecken beziehungsweise Gemeinsamkeiten identifizieren zu können.

Die Analyse der Werte und Verteilungen berücksichtigt zwei Randbedingungen, die durch den Umfang der Daten sowie deren Prozessierung entstehen. Erstens führen die umfangreichen Daten dazu, dass einige Visualisierungen nicht mehr sinnvoll einsetzbar sind, da die entsprechenden Abbildungen durch die Vielzahl der Daten unübersichtlich werden oder deren Erstellung zu komplex wird. So werden beispielsweise im Folgenden keine Scatter-Plots verwendet, da für deren Erstellung zu viele Datenpunkte gezeichnet werden müssten. Außerdem werden bei den erstellten Pentagrammen die Tiefe des Medians, der hinges und der Extremwerte weggelassen, die diese sehr hohe Werte aufweisen. Beispielsweise liegt für die Daten Osm-Fr und die Anzahl der Stützpunkte die Tiefe des Medians bei einem Wert von 15 891 393. Die zweite Randbedingung leitet sich aus der in Abschnitt 7.5 vorgestellten Prozessierung der Daten ab. Da die einzelnen Attributwerte der Objekte diskretisiert werden, beziehen sich die Ergebnisse der Auswertung auch auf diskretisierte Werte. Die Auswertung ist demnach zu einem gewissen Grad von den gewählten Parametern der Diskretisierung abhängig. Deshalb wird für alle Untersuchungen die jeweils gewählte Diskretisierung explizit angegeben und gegebenenfalls die Auswirkung unterschiedlicher Parameter für die Diskretisierung vorgestellt.

Anzahl der Stützpunkte und minimale Distanz aufeinanderfolgender Stützpunkte

Bei der Bestimmung der Häufigkeit der Attributwerte wird für die Anzahl der Stützpunkte eine Bereichsgröße von 1 gewählt, die auch als Zusatz zum Attributnamen NumVx in den Abbildungen angegeben ist. Für die drei Datensätze sind die Lage- und Streuungsmaße in den Pentagrammen in Abbildung 7.14 dargestellt. Durch die Prüfung der Geometrien ist bereits sichergestellt, dass alle Objektgeometrien mindestens drei Stützpunkte aufweisen, weshalb das mögliche Minimum bei drei Stützpunkten liegt. Der Median der Anzahl der Stützpunkte ist für die drei Datensätze vergleichbar, wobei die beiden OpenStreetMap-Datensätze jeweils einen Median von 5 aufweisen. Zudem lässt sich aus den hinges ablesen, dass etwa 75 % der Objekte maximal acht Stützpunkte aufweisen und damit geometrisch vergleichsweise einfach sind. Die drei Datensätze weisen jedoch auch Objektgeometrien mit auffällig vielen Stützpunkten auf. Werte von mehreren hundert Stützpunkten für eine einzelne Gebäudegeometrie entsprechen nicht der Erwartung an einfache Geometrien. Wie in Unterabschnitt 6.3.1 ausgeführt ist, können jedoch auch geringe Abstände zwischen aufeinanderfolgen Stützpunkten Ursache für eine hohe Anzahl an Stützpunkten sein, was im Anschluss an die folgende Betrachtung untersucht wird.

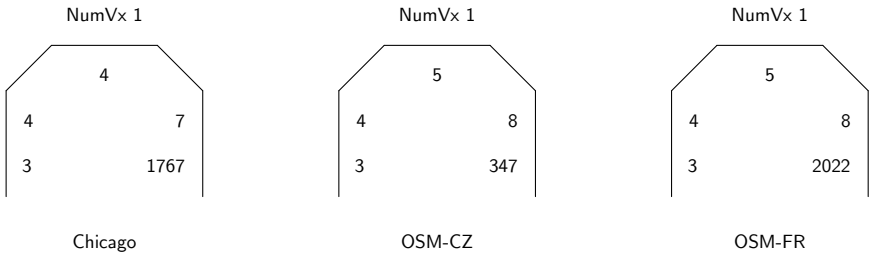


Abbildung 7.14: Pentagramme der Anzahl der Stützpunkte

Die Verteilung der Anzahl der Stützpunkte zeigt ein interessantes Muster auf. In Abbildung 7.15 sind für die drei Datensätze die relativen Häufigkeiten der Anzahl der Stützpunkte für den Bereich von 3 bis 12 Stützpunkten dargestellt. Zuerst wird deutlich, dass die relative Häufigkeit für eine Anzahl von 4 Stützpunkten am höchsten

ist, was auch für den gesamten Wertebereich verifiziert werden kann. Im Datensatz Chicago weisen 62 % der Objekte 4 Stützpunkte auf. Für die Datensätze Osm-Cz und Osm-Fr bestehen immerhin noch 48 % und 44 % der Gebäudegeometrien aus 4 Stützpunkten. Auffällig ist jedoch auch die ungleiche Verteilung der relativen Häufigkeiten für eine gerade beziehungsweise eine ungerade Anzahl an Stützpunkten. Diese betrifft nicht nur den in Abbildung 7.15 dargestellten Ausschnitt, sondern den kompletten Wertebereich. Die Auswertung der relativen Häufigkeiten zeigt, dass im Datensatz Chicago 93 % und in den Datensätzen Osm-Cz 82 % und Osm-Fr 73 % der Gebäudegeometrien eine gerade Anzahl an Stützpunkten aufweisen.

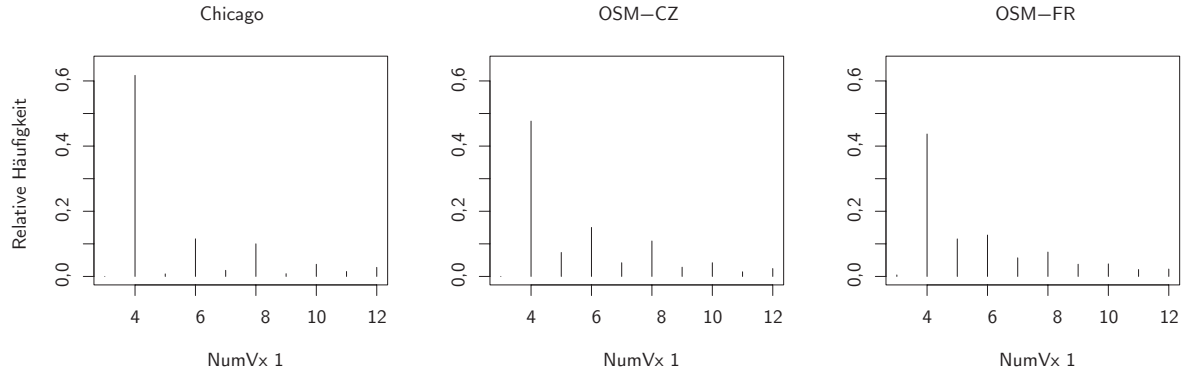


Abbildung 7.15: Stabdiagrammausschnitte der Anzahl der Stützpunkte

Für die Attributwerte der minimalen Distanz aufeinanderfolgender Stützpunkte sind die entsprechenden Pentagramme in den Abbildungen 7.16 und 7.17 dargestellt. Die Pentagramme in den beiden Abbildungen unterscheiden sich in ihrer Bereichsgröße. Für den Datensatz Chicago liegt beispielsweise der Median bei einer Bereichsgröße von 0,1 bei einem Wert von 19,05, also zwischen den beiden Werten 19,00 und 19,10. Wird die Bereichsgröße auf 1 vergrößert, so liegt der Median entsprechend bei 19,5, also zwischen den beiden Werten 19,0 und 20,0.

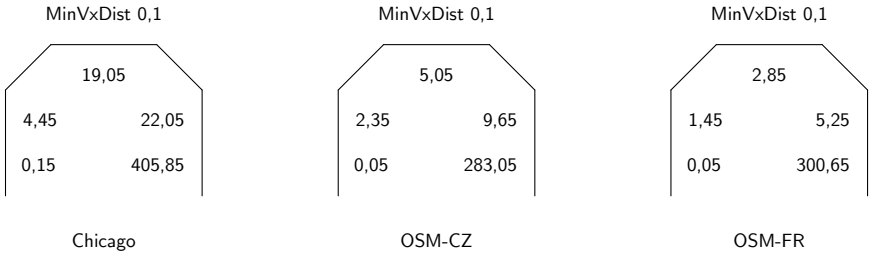


Abbildung 7.16: Pentagramme der minimalen Distanz aufeinanderfolgender Stützpunkte

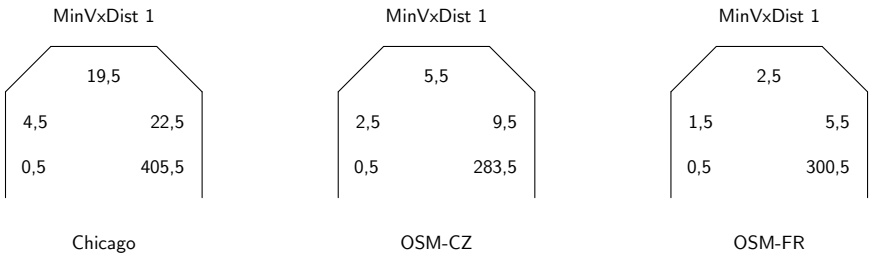


Abbildung 7.17: Pentagramme der minimalen Distanz aufeinanderfolgender Stützpunkte

Die statistischen Lagemaße für die minimale Distanz aufeinanderfolgender Stützpunkte, wie Median und hinges, unterscheiden sich deutlich zwischen den drei Datensätzen. Deutlich ist vor allem der Unterschied zwischen dem Datensatz Chicago und den beiden OpenStreetMap-Datensätzen, beispielsweise beim Median. Wird die Verteilung der Attributwerte für den Wertebereich von 0 bis 40 in einem Stabdiagramm abgetragen, dann ergeben sich die drei Stabdiagrammausschnitte in Abbildung 7.18. Aus diesen ist ersichtlich, dass die Attributwerte für den Datensatz Chicago zwei lokale Maxima aufweisen, wohingegen die Attributwerte für die beiden OpenStreetMap-Datensätze jeweils nur ein Maximum aufweisen.

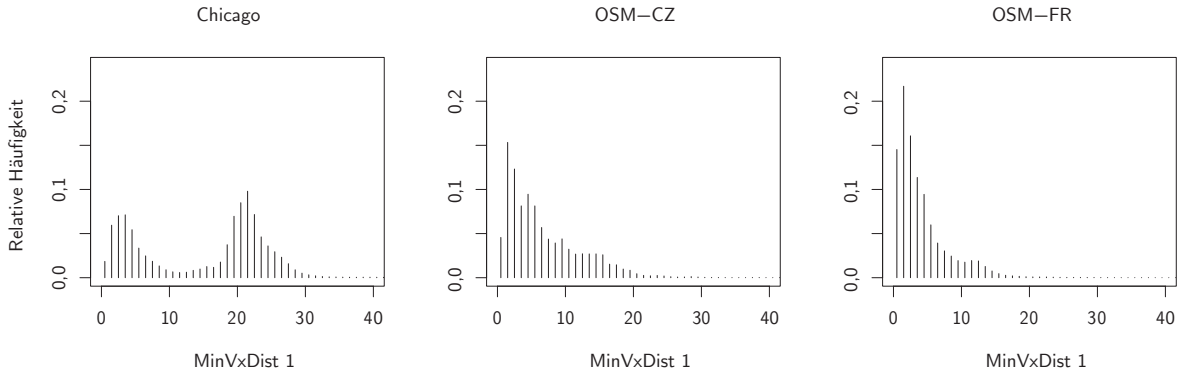


Abbildung 7.18: Stabdiagrammausschnitte der minimalen Distanz aufeinanderfolgender Stützpunkte

Integritätsbedingungen

Für die Anzahl der Stützpunkte und die minimale Distanz aufeinanderfolgender Stützpunkte sind die entsprechenden Integritätsbedingungen in der Geo Object Constraint Language (OCL) im Programm 7.5 aufgestellt. Wie in Unterabschnitt 6.3.1 werden nur Bedingungen für die minimale Anzahl, aber nicht die maximale Anzahl, an Stützpunkten festgelegt. Da dreieckige Gebäudegrundrisse selten beziehungsweise auffällig sind, wird in den Integritätsbedingung eine minimale Anzahl an Stützpunkten von 4 gefordert.

```
context Gebaeude
  inv NumVx: geom.exteriorRing().numPoints() >= 4
  inv MinVxDist: geom.exteriorRing().segments()->forAll(s | s.length() >= 0.1) -- [Meter]
```

Programm 7.5: Integritätsbedingungen für die Anzahl der Stützpunkte und Statistik über die Distanz der Stützpunkte

Minimaler Winkel zwischen zwei aufeinanderfolgenden Liniensegmenten

Durch die Bestimmung des minimalen Winkels zwischen zwei aufeinanderfolgenden Liniensegmenten kann unter anderem bestimmt werden, ob die Form von Gebäudegeometrien erwartungsgemäß durch einen hohen Anteil rechter Winkel charakterisiert ist. Für die Bestimmung der Häufigkeiten der Attributwerte wird eine Bereichsgröße von 1° gewählt um eine ausreichend genaue Auflösung zu erhalten. In Abbildung 7.19 sind die Pentagramme für die drei Datensätze dargestellt.

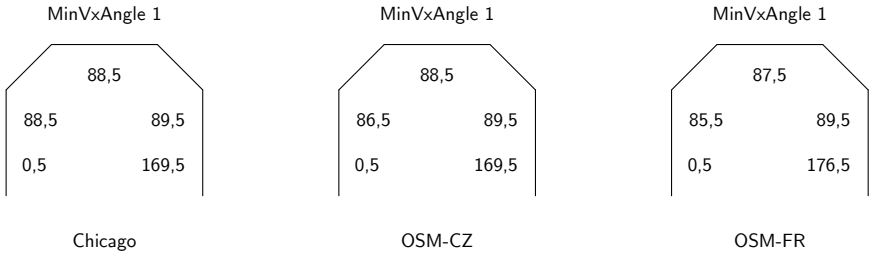


Abbildung 7.19: Pentagramme des minimalen Winkels zwischen zwei aufeinanderfolgenden Liniensegmenten

Um die Verteilung der Winkel genauer analysieren zu können sind in Tabelle 7.17 die Häufigkeiten für ausgewählte Bereiche aufgelistet. Für den Datensatz Chicago liegen 98,5 % der Winkel zwischen 80° und 100° und sind damit annähernd orthogonal. Die entsprechenden Prozentsätze betragen für OSM-CZ 93,3 % sowie für OSM-FR 90,0 %. Die Annahme, dass die Form von Gebäudegeometrien durch eine hohe Anzahl an rechten Winkel charakterisiert sind, kann somit bestätigt werden. Kritisch für die Form sind vor allem kleine Winkel, da diese zu geometrischen Spitzen in Gebäudegeometrien führen. In Tabelle 7.17 sind deshalb auch die prozentualen Anteile der Winkel $< 30°$ angegeben.

Integritätsbedingungen

Zur Vermeidung von spitzen Winkeln in den Gebäudegeometrien wird im Programm 7.6 ein Minimum für den Winkel zwischen zwei aufeinanderfolgenden Liniensegmenten festgelegt.

Wertebereich		Prozentualer Anteil		
Von [°]	Bis [°]	Chicago	Osm-Cz	Osm-Fr
0	10	0,01	0,10	0,14
10	20	0,01	0,09	0,16
20	30	0,02	0,12	0,18
...	...	...	...	...
80	90	93,10	91,91	89,13
90	100	5,37	1,37	0,95
...	...	...	...	...
160	170	0,00	0,02	0,04
170	180	0,01	0,00	0,01

Tabelle 7.17: Übersicht über die minimalen Winkel zwischen zwei aufeinanderfolgenden Liniensegmenten

```
context Gebaeude
  inv MinVxAngle: geom.exteriorRing().interiorAngles()->forAll(a | a >= 30) -- [Grad]
```

Programm 7.6: Integritätsbedingungen für den minimalen Winkel zwischen zwei aufeinanderfolgenden Liniensegmenten

Fläche

Die Fläche eines Gebäudegrundrisses ist ein wichtiges geometrisches Maß, nicht zuletzt weil viele weitere Maße aus der Fläche abgeleitet werden können. Um eine feingranulare Aussage über die Attributwerte zu ermöglichen, wird für die Bestimmung der Fläche wird eine Bereichsgröße von $10\,\mathrm{m}^2$ gewählt. Die Pentagramme für die Flächen der Gebäudegrundrisse sind in Abbildung 7.20 für die drei Datensätze dargestellt. Abgesehen vom Minimum weisen die Lagemaße wenig Gemeinsamkeiten auf. Die Unterschiede zwischen den Datensätzen sind zudem anschaulich aus den Stabdiagrammausschnitten in Abbildung 7.21 ersichtlich. Der Datensatz Chicago weist eine geringe Anzahl an Flächen $< 300\,\mathrm{m}^2$ und ein Maximum bei etwa $500\,\mathrm{m}^2$ auf. In den OpenStreetMap-Daten sind dagegen viele Flächen $< 400\,\mathrm{m}^2$ vorhanden, was auch aus den oberen hinges der Pentagramme ersichtlich ist. Sowohl die Gebäudeflächen Tschechiens als auch Frankreichs weisen im dargestellten Bereich zwei Maxima auf, wobei das erste Maximum bei $45\,\mathrm{m}^2$ (Osm-Cz) beziehungsweise $15\,\mathrm{m}^2$ (Osm-Fr) liegt. Auffällig ist zudem, dass für den Datensatz Osm-Fr die Verteilung der Attributwerte der Fläche bereits für geringe Werte vergleichweise hohe relative Häufigkeiten aufweist.

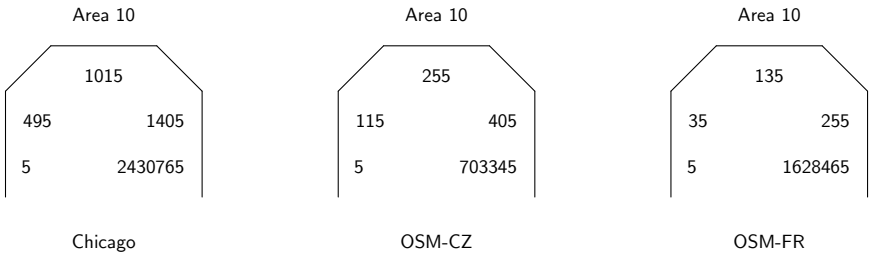


Abbildung 7.20: Pentagramme der Fläche

Integritätsbedingungen

Aufgrund der geringen Übereinstimmung der Lagemaße und der Verteilung der Attributwerte wird für die Fläche lediglich ein Minimalwert und ein Maximalwert gefordert. Im Programm 7.7 wird eine Mindestfläche von $10\,\mathrm{m}^2$ als Integritätsbedingung formuliert, da Gebäude mit einer geringeren Fläche nur eingeschränkt nutzbar sind. Ebenso wird eine Maximalfläche von $100\,000\,\mathrm{m}^2$ festgelegt, da nur sehr wenige Gebäude eine größere Grundfläche aufweisen.

Kompaktheit und fraktale Dimension

Die Komplexität einer Objektgeometrie kann über die beiden Maße Kompaktheit und fraktale Dimension ausgedrückt werden. Die Kompaktheit bezieht sich auf ein Quadrat als Referenzfläche und weist bei einer Bereichsgröße von 0,1 für die drei Datensätze die Lagemaße aus Abbildung 7.23 auf. Die Lagemaße weisen bis

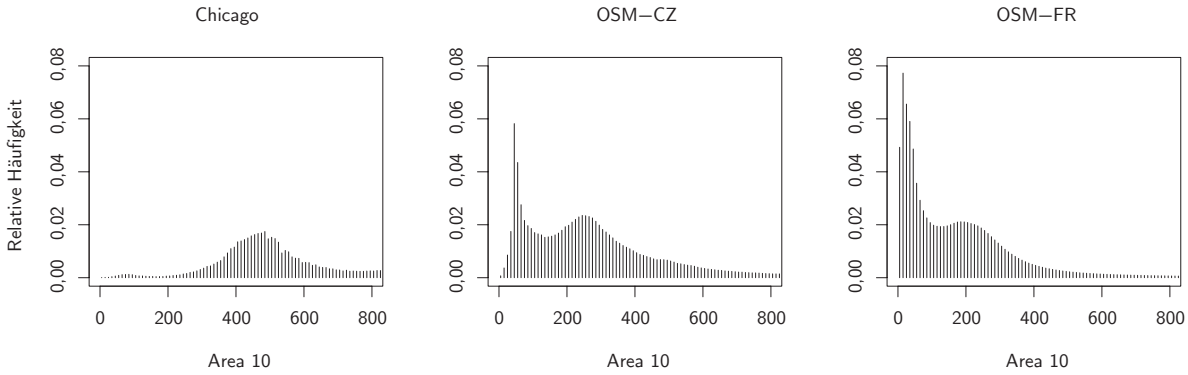


Abbildung 7.21: Stabdiagrammausschnitte der Fläche

```
context Gebaeude
  inv Area: geom.area() >= 10 and geom.area() < 100000 -- [Quadratmeter]
```

Programm 7.7: Integritätsbedingungen für die Fläche

auf das Maximum eine hohe Übereinstimmung auf. Die Verteilung der Attributwerte, die in den Stabdiagrammausschnitten in Abbildung 7.23 dargestellt ist, zeigt jedoch deutliche Unterschiede zwischen dem Datensatz Chicago und den beiden OpenStreetMap-Datensätzen.

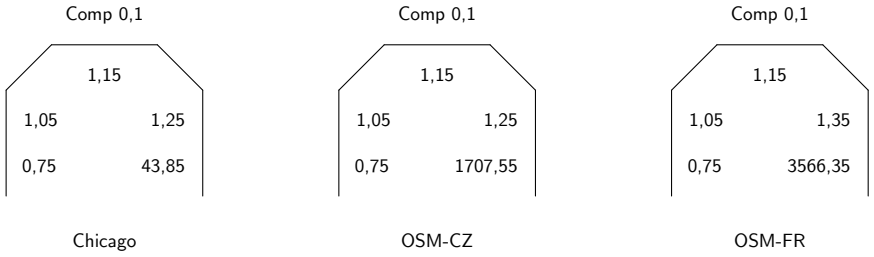


Abbildung 7.22: Pentagramme der Kompaktheit bezogen auf ein Quadrat

Für die fraktale Dimension wird wie bei der Kompaktheit eine Bereichsgröße von 0,1 gewählt. Die in Abbildung 7.24 dargestellten Lagemaße der fraktalen Dimension sowie die Verteilung in den Stabdiagrammausschnitten in Abbildung 7.25 zeigen auf, dass die Gebäudegrundrisse im Datensatz Chicago die geringste fraktale Dimension aufweisen. Etwa 97 % der Werte liegen im Bereich von 1,3 bis 1,5. Dagegen weist die Verteilung der Attributwerte für die beiden OpenStreetMap-Datensätze eine größere Bandbreite auf.

Integritätsbedingungen

Für die Kompaktheit und die fraktale Dimension werden im Programm 7.8 jeweils Integritätsbedingungen an das Minimum und das Maximum der Attributwerte gestellt. Diese folgen aus einer manuellen Inspektion der Gebäudegrundrisse. Aufgrund der Übereinstimmung der Werte für den Median der Kompaktheit, wird für diesen eine statistische Bedingung aufgestellt. Für die fraktale Dimension werden dagegen Bedingungen an die hinges aufgestellt.

Rechtwinkligkeit

Die Lagemaße der Rechtwinkligkeit sind bei einer Bereichsgröße von 0,1 für die drei Datensätze identisch, weshalb in Abbildung 7.26 auch nur ein einzelnes Pentagramm dargestellt ist. Da bereits der untere hinge bei einem Wert von 0,95 liegt, weist die Mehrheit der Gebäudegrundrisse eine hohe Rechtwinkligkeit auf.

Integritätsbedingungen

Als Bedingungen wird im Programm 7.9 für alle Gebäudegrundrisse eine minimale Rechtwinkligkeit gefordert sowie der Wert des unteren hinge festgelegt.

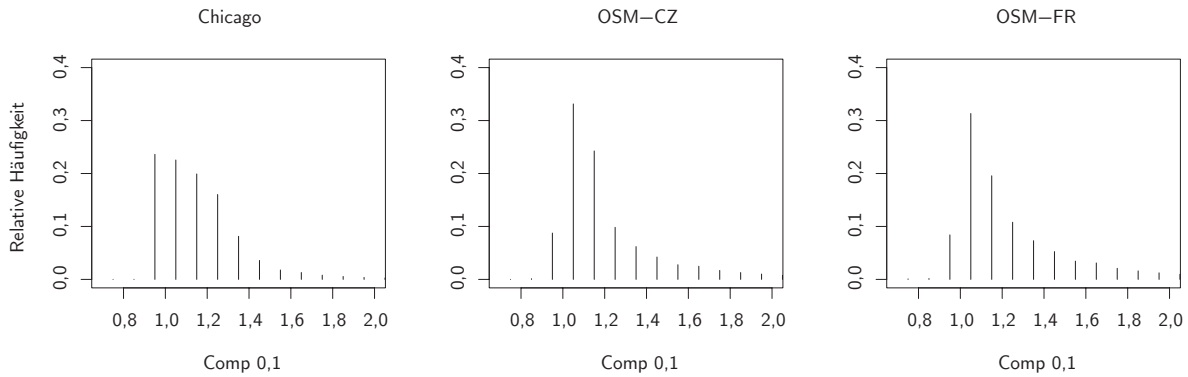


Abbildung 7.23: Stabdiagrammausschnitte der Kompaktheit bezogen auf ein Quadrat

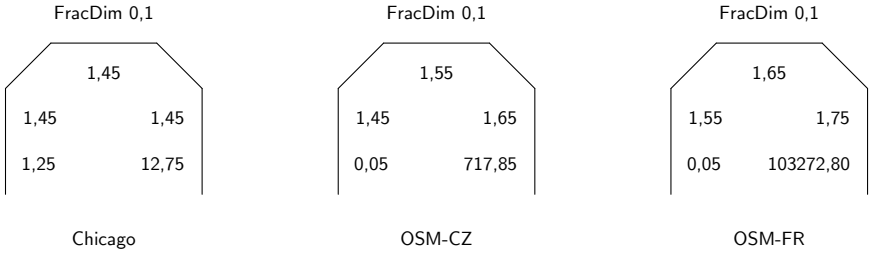


Abbildung 7.24: Pentagramme der fraktalen Dimension

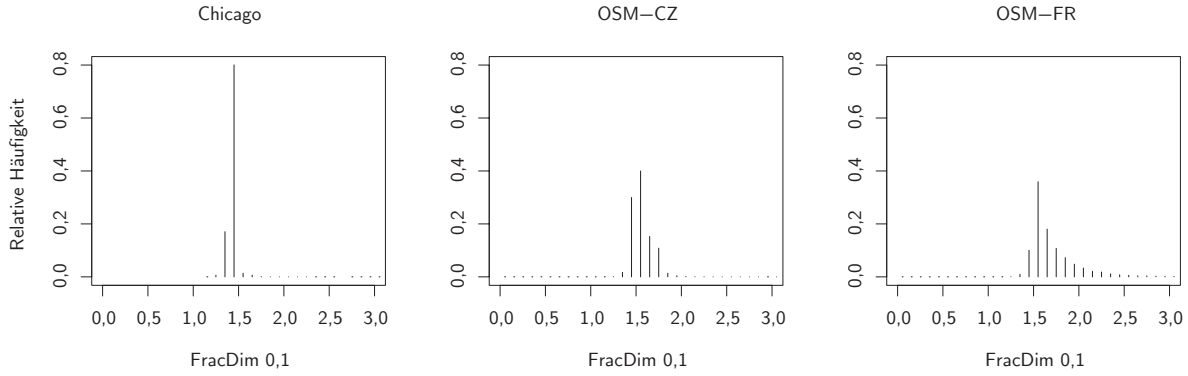


Abbildung 7.25: Stabdiagrammausschnitte der fraktalen Dimension

```
context Gebaeude
  inv Comp: geom.compactnessSquare() >= 0.7 and geom.compactnessSquare() < 10 -- []
  inv CompMedian: let compFiveNum: FiveNumberSummary = Gebaeude.allInstances()->collect(geom.
      compactnessSquare())->fiveNumberSummary() in compFiveNum.median() > 1.1 and compFiveNum.
      median < 1.2 -- []
  inv FracDim: geom.fractalDimension() >= 1.2 and geom.fractalDimension() < 2.5 -- []
  inv FracDimHinges: let fracDimFiveNum: FiveNumberSummary = Gebaeude.allInstances()->collect(geom.
      fractalDimension())->fiveNumberSummary() in fracDimFiveNum.lowerHinge() > 1.4 and
      fracDimFiveNum.upperHinge() < 1.8
```

Programm 7.8: Integritätsbedingungen für die Kompaktheit und die fraktale Dimension

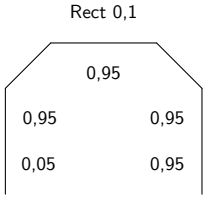


Abbildung 7.26: Pentagramm der Rechtwinkligkeit

```
context Gebaeude
  inv Rect: geom.exteriorRing().rectangularity(10) >= 0.20 -- [], Toleranz 10 Grad
  inv RectHinges: Gebaeude.allInstances()->collect(geom.exteriorRing().rectangularity(10))->
      fiveNumberSummary().lowerHinge() > 0.9
```

Programm 7.9: Integritätsbedingungen für die Rechtwinkligkeit

Minimaler Durchmesser des umschließenden Rechtecks

Der minimale Durchmesser ist das erste abgeleitete Maß, das sich statt auf die Geometrie selbst auf dessen umschließendes Rechteck bezieht. Aus einer Bereichsgröße von 1 m folgen die Pentagramme in Abbildung 7.27 sowie die Stabdiagrammausschnitte in Abbildung 7.28. Aus den Pentagrammen lässt sich ein deutlicher Unterschied zwischen den drei Datensätzen bestimmen. Insbesondere verringern sich die Werte für den Median und die beiden hinges deutlich vom Datensatz Chicago, zum Datensatz OSM-CZ bis zum Datensatz OSM-FR. Die Verteilung der Attributwerte kann wieder zwischen dem Datensatz Chicago und den beiden OpenStreetMap-Datensätzen unterschieden werden. Der minimale Durchmesser des umschließenden Rechtecks weist für die Gebäudegrundrisse des Datensatzes Chicago nur eine geringe Anzahl an Werten < 16 m und ein deutliches Maximum bei etwa 22 m auf. Die beiden Verteilungen der Datensätze OSM-CZ und OSM-FR sind ähnlich, jedoch sind das erste und zweite lokale Maximum vertauscht. Auffällig ist zudem, dass für den Datensatz OSM-FR die Verteilung der Attributwerte des minimalen Durchmessers des umschließenden Rechtecks wie bei der Fläche bereits für geringe Werte vergleichsweise hohe relative Häufigkeiten aufweist.

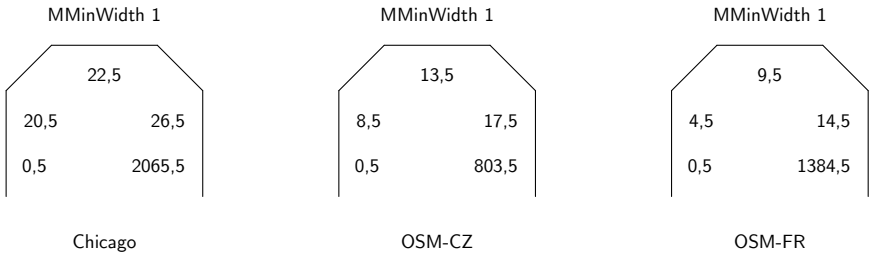


Abbildung 7.27: Pentagramme des minimalen Durchmessers des umschließenden Rechtecks

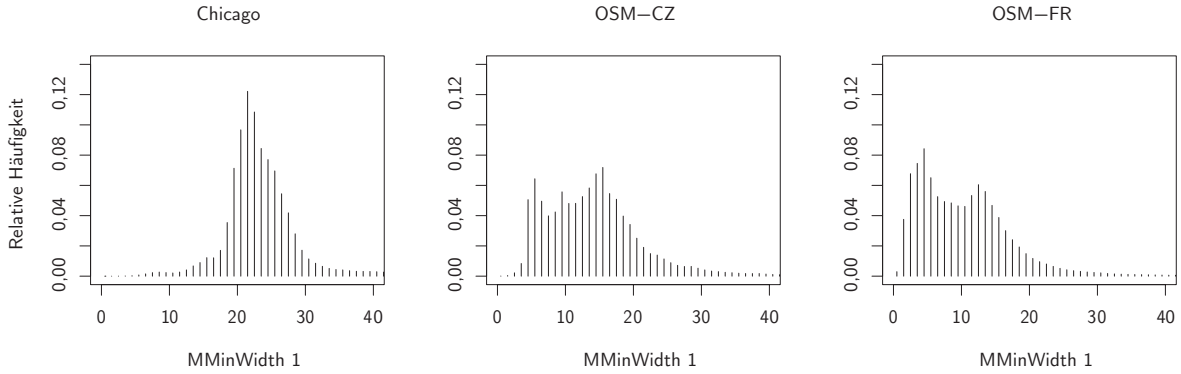


Abbildung 7.28: Stabdiagrammausschnitte des minimalen Durchmessers des umschließenden Rechtecks

Integritätsbedingungen

Für den minimalen Durchmesser des umschließenden Rechtecks wird im Programm 7.10 nur für das Minimum und Maximum eine Integritätsbedingung aufgestellt. Die Werte folgen aus dem Aspekt der Nutzbarkeit der zugehörigen Gebäudegrundrisse.

Azimut der längeren Seite des umschließenden Rechtecks

Das Azimut der längeren Seite des umschließenden Rechtecks wird ermittelt um bestimmen zu können, ob die Ausrichtung von Gebäuden einer statistischen Regelmäßigkeit unterliegt. Die Stabdiagrammausschnitte[6] in

[6] Der letzte dargestellte Wert in den Stabdiagrammausschnitten ist 185°. Bei einer Bereichsgröße von 10° enthält dieser demnach alle Werte im Intervall [180°, 190°). Der maximale Rückgabewert des Algorithmus zur Bestimmung des Azimuts liegt bei 180°.

```
context Gebaeude
  inv MMinWidth: geom.minimumEnclosingAdaptedRectangle().width() >= 2 and geom.
      minimumEnclosingAdaptedRectangle().width() < 500 -- [Meter]
```

Programm 7.10: Integritätsbedingungen für den minimalen Durchmesser des umschließenden Rechtecks

Abbildung 7.29 zeigen wiederum die klare Trennung zwischen dem Datensatz Chicago und den beiden OSM-Datensätzen. Bei einer Bereichsgröße von 10° weisen die Azimute für den Datensatz Chicago zwei dominante Maxima bei 85° (52 % der Objekte) sowie bei 175° (30 % der Objekte) auf. Die beiden Azimute sind um 90° zueinander versetzt. Die Beobachtung von solch deutlichen Maxima ist auf die regelmäßige Anordnung der Gebäude in einem vorwiegend rechtwinklig ausgelegten Straßennetz zurückzuführen, die typisch für amerikanische Großstädte ist. Dagegen weisen die Stabdiagrammausschnitte für die beiden OSM-Datensätze nur geringfügige Maxima bei 5° sowie 85° auf. Diese sind wiederum um 90° zueinander versetzt. Daraus kann eine geringfügige Präferenz für die Ausrichtung von Gebäuden in die Himmelsrichtungen Nord-Süd beziehungsweise Ost-West abgeleitet werden. Nichts desto trotz ist die Ausrichtung nicht ausreichend statistisch signifikant, um für diese entsprechende Integritätsbedingungen aufstellen zu können.

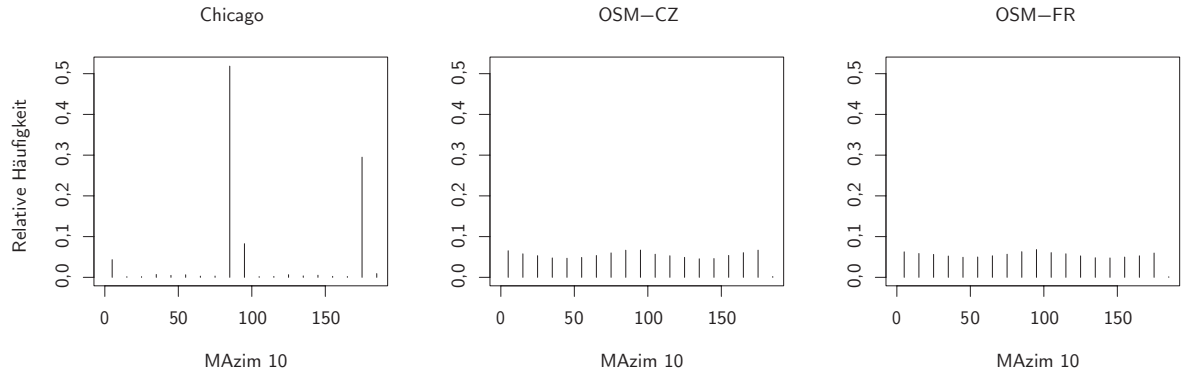


Abbildung 7.29: Stabdiagrammausschnitte des Azimuts der längeren Seite des umschließenden Rechtecks

Elongation des umschließenden Rechtecks und Flächenverhältnis zum umschließenden Rechteck

Die Elongation und das Flächenverhältnis basieren auf dem umschließenden Rechteck und sind zwei wichtige Maße zur Beschreibung der Form eines Objekts. In den Pentagrammen in Abbildung 7.30 und Abbildung 7.32 sind die Lagemaße für die beiden Attribute dargestellt. Der minimale Wert für die beiden Maße ist jeweils 1, weshalb die Minima von 0,95 in den Pentagrammen auf Rundungsfehler zurückzuführen sind.

Ein Wert von 1 für die Elongation entspricht einem quadratischen umschließenden Rechteck. Die Mediane von 1,55 sowie 1,65 für die Elongation entsprechen einem Breiten-Längen-Verhältnis der umschließenden Rechtecke von etwa 1:1,6. Die Maxima in den Pentagrammen sind jedoch für Gebäudegrundrisse atypisch, da diese extrem langgestreckten Geometrien entsprechen. Die in Abbildung 7.31 dargestellten Verteilungen der Attributwerte zeigen wiederum eine Ähnlichkeit der beiden OSM-Datensätze und Unterschiede zum Datensatz Chicago auf. Die Elongation der Gebäudegrundrisse weist für den Datensatz Chicago deutlich mehr nahezu quadratische umschließende Rechtecke auf.

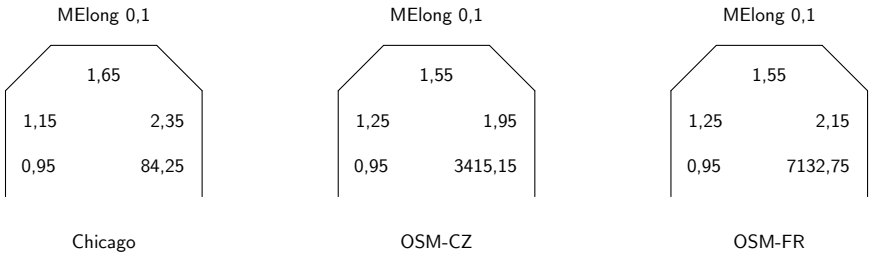


Abbildung 7.30: Pentagramme der Elongation des umschließenden Rechtecks

Dementsprechend enthält dieser Wertbereich nur einen einzelnen Wert, was zu einer vergleichbar geringen relativen Häufigkeit der Attributwerte führt.

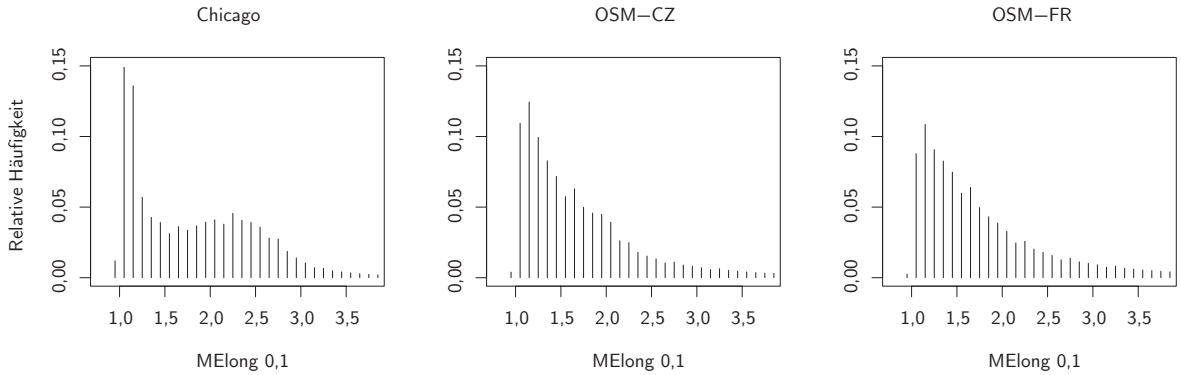


Abbildung 7.31: Stabdiagrammausschnitte der Elongation des umschließenden Rechtecks

Beim Flächenverhältnis zum umschließenden Rechteck sagt ein Wert von 1 aus, dass die Objektgeometrie exakt dem umschließenden Rechteck entspricht, also perfekt rechteckig ist. Auch hier weist der Datensatz Chicago einen höheren Anteil an nahezu perfekt rechteckigen Objektgeometrien auf, der sich durch einen geringen Wert für den oberen hinge auszeichnet.

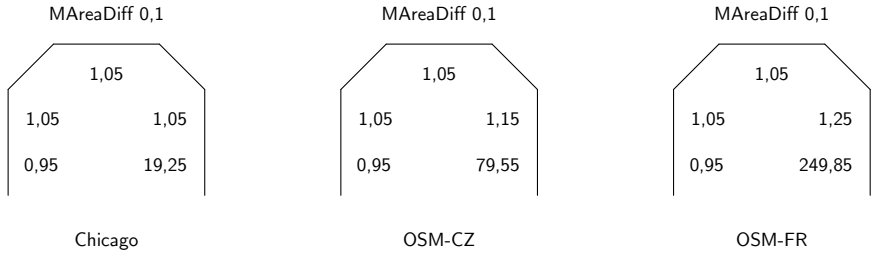


Abbildung 7.32: Pentagramme des Flächenverhältnisses zum umschließenden Rechteck

Integritätsbedingungen

Da die minimalen Werte der Elongation des umschließenden Rechtecks und des Flächenverhältnisses zum umschließenden Rechteck per Definition den Wert 1 aufweisen, werden in den Integritätsbedingungen im Programm 7.11 lediglich die Maxima der beiden Attribute beschränkt. Diese folgen aus einer manuellen Inspektion der Gebäudegrundrisse.

```
context Gebaeude
  inv Elong: geom.minimumEnclosingAdaptedRectangle().elongation() < 10 -- []
  inv MerDiff: (geom.minimumEnclosingAdaptedRectangle().area() - geom.area()) / geom.area() < 6 --
      []
```

Programm 7.11: Integritätsbedingungen für die Elongation des umschließenden Rechtecks und das Flächenverhältnis zum umschließenden Rechteck

7.6.2 Topologische Maße

Die Werte und Verteilungen der zwei verwendeten topologischen Maße werden im Folgenden basierend auf ihrer Definition in Unterabschnitt 7.4.2 analysiert. Dabei werden die drei Datensätze Chicago, Osm-Cz und Osm-Fr wiederum gemeinsam untersucht, um Unterschiede aufdecken beziehungsweise Gemeinsamkeiten identifizieren zu können.

Größe der Schnittfläche mit benachbarten Objekten

Die Größe der Schnittfläche mit benachbarten Objekten entspricht bei überlappungsfreien Objektgeometrien exakt $0\,\mathrm{m}^2$. Um eine möglichst exakte Aussage darüber treffen zu können, ob eine Gebäudegeometrie überlappungsfrei ist, wird die Bereichsgröße auf $0{,}1\,\mathrm{m}^2$ festgelegt. Die Pentagramme in Abbildung 7.33 zeigen, dass die Anforderung für alle drei Datensätze mehrheitlich erfüllt ist. Jedoch weisen die Maxima auf große Überlappungen zwischen Objektgeometrien hin. Eine Trennung der Attributwerte in den Histogrammen in überlappungsfrei

(Bereich entspricht 0,05 m^2) und nicht überlappungsfrei (alle anderen Bereiche) ergibt, dass im Datensatz Chicago 99,9 % und in den Datensätzen Osm-Cz 98,9 % und Osm-Fr 99,6 % der Geometrien überlappungsfrei sind.

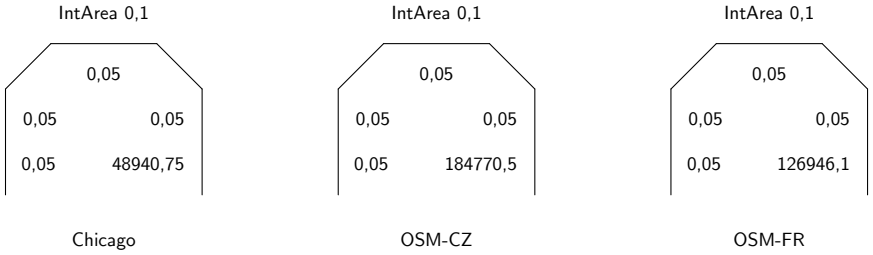


Abbildung 7.33: Pentagramme der Größe der Schnittfläche mit benachbarten Objekten

Integritätsbedingungen

Als Integritätsbedingung wird gefordert, dass die Gebäudegrundrisse überlappungsfrei sind. Aufgrund der prinzipiell inexakten rechnerinternen Speicherung von Fließkommazahlen, wird im Programm 7.12 lediglich gefordert, dass die Größe der Schnittfläche kleiner als ein Schwellwert von 0,01 m^2 ist.

```
context Gebaeude
  inv IntArea: Gebaeude.allInstances()->forAll(g1, g2 | g1 <> g2 implies g1.geom.intersection(g2.
      geom).area() < 0.01) -- [Quadratmeter]
```

Programm 7.12: Integritätsbedingungen für die Größe der Schnittfläche mit benachbarten Objekten

Minimale Distanz zu benachbarten Objekten

Die Attributwerte für die minimale Distanz zwischen benachbarten Objekten werden in drei aneinandergrenzenden Wertebereichen untersucht. Bei der Bestimmung der Häufigkeit der Attributwerte wird dabei eine Bereichsgröße von 0,1 m gewählt, um möglichst exakte Aussagen treffen zu können.

Der erste Wertebereich umfasst das Intervall von 0 m bis 0,1 m. Ist die minimale Distanz eines Objekts zu mindestens einem Nachbarn in diesem Wertebereich, dann können die entsprechenden Gebäudegrundrisse als direkt benachbart bezeichnet werden. Für die drei Datensätze ist die Zuordnung der Objekte zu den Wertebereichen in Tabelle 7.18 aufgelistet. Im Datensatz Chicago weisen lediglich 4 % der Objekte direkte Nachbarn auf, wohingegen dieser Anteil in den beiden Osm-Datensätzen deutlich höher ist. Im Datensatz Chicago sind demnach die meisten Gebäude alleinstehend, wohingegen in den beiden anderen Datensätze die Gebäude häufig Teil einer Reihen- oder Blockbebauung sind.

	Prozentualer Anteil		
Daten	$[0, 0.1)$	$[0.1, 1000)$	> 1000
Chicago	4,39	95,61	0,00
Osm-Cz	44,85	55,10	0,05
Osm-Fr	67,02	32,96	0,02

Tabelle 7.18: Verteilung der Objekte in Abhängigkeit von der minimalen Distanz zu benachbarten Objekten

Der zweite Wertebereich umfasst das Intervall von 0,10 m bis 1000 m. Aus Tabelle 7.18 ist ersichtlich, dass die prozentualen Anteile in diesem Wertebereich den ersten Wertebereich nahezu vollständig zu 100 % komplementieren. Die Pentagramme der minimalen Distanz der Objekte für diesen Wertebereich sind in Abbildung 7.34 dargestellt. Diese zeigen auf, dass die Werte für die hinges sowie den Median zwischen den drei Datensätzen deutliche Unterschiede aufweisen. Um detailliertere Aussagen treffen zu können, wird deshalb im Folgenden der Wertebereich weiter unterteilt.

Im Intervall von 0,15 m bis 1,05 m sind die prozentualen Anteile der Objekte entsprechend der Tabelle 7.19 verteilt. Anhand der Verteilung lässt sich die Vermutung nicht bestätigen, dass Gebäude einen einheitlichen funktional bedingten Mindestabstand aufweisen, beispielsweise aufgrund von Brandschutzbedingungen oder für einen Durchgang zwischen benachbarten Gebäuden. Ein relativ deutlicher Anstieg des prozentualen Anteils

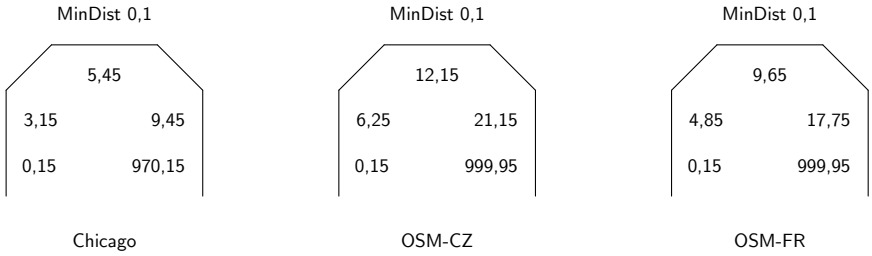


Abbildung 7.34: Pentagramme der minimalen Distanz zu benachbarten Objekten im Wertebereich [0.10, 1000)

lässt sich jedoch für den Datensatz Chicago bei 0,45 m und für den Datensatz Osm-Cz bei 0,75 m feststellen. Dagegen lässt sich ein solcher Anstieg für den Datensatz Osm-Fr nicht feststellen.

	Prozentualer Anteil		
Mittelpunkt Bereich [m]	Chicago	Osm-Cz	Osm-Fr
0,15	0,05	0,02	0,05
0,25	0,12	0,03	0,07
0,35	0,14	0,02	0,07
0,45	0,55	0,03	0,10
0,55	0,33	0,04	0,11
0,65	0,25	0,10	0,14
0,75	0,25	0,23	0,13
0,85	0,38	0,59	0,17
0,95	1,03	0,45	0,15
1,05	0,56	0,28	0,14

Tabelle 7.19: Verteilung der Objekte im Intervall [0,15, 1,05]

Im anschließenden Intervall von 1 m bis 30 m Distanz zu benachbarten Objekten ist in Abbildung 7.35 wieder ein deutlicher Unterschied zwischen dem Datensatz Chicago und den beiden Osm-Datensätzen erkennbar. Im Datensatz Chicago ist ein deutliches Maximum bei 3,5 m erkennbar. Zudem unterscheidet sich der Datensatz durch vergleichbar hohe Werte für die relative Häufigkeit geringer Distanzen. Eine weitere Auffälligkeit des Datensatzes Chicago offenbart sich bei genauer Betrachtung der minimalen Distanz bei einer Bereichsgröße von 0,1 m und einem entsprechend großen Intervall. In Abbildung 7.36 sind deutlich lokale Maxima in festen Abständen von 0,5 m erkennbar, beispielsweise an den Stellen 1,45 m, 1,95 m, 2,45 m, 2,95 m, 3,45 m. Dies lässt auf streng regulierte Mindestabstände zwischen benachbarten Gebäuden schließen.

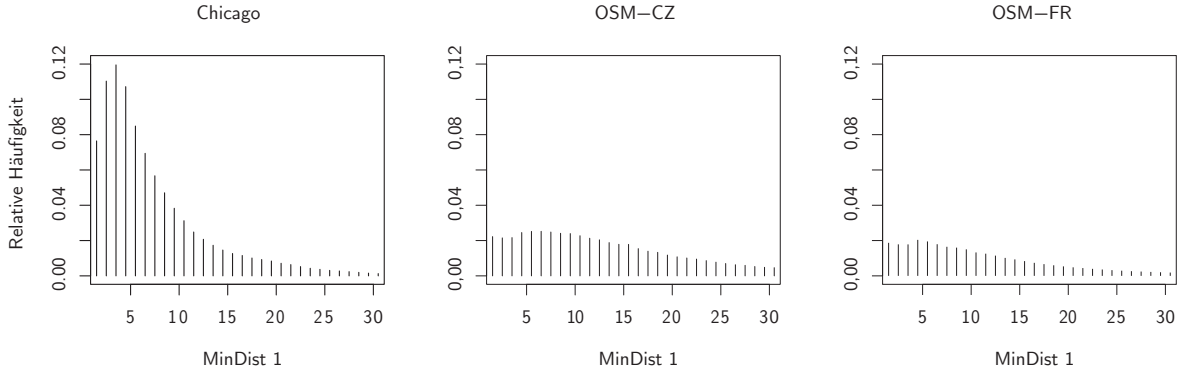


Abbildung 7.35: Stabdiagrammausschnitte der minimalen Distanz zu benachbarten Objekten im Wertebereich [1, 30]

Der dritte Wertebereich in Tabelle 7.18 umfasst alle Werte die > 1000 m sind, d.h. im Umkreis von 1 km um ein Gebäude steht kein weiteres Gebäude. Die Anzahl der Objekte ist in den drei Datensätzen entsprechend gering, mit maximal 0,05 % für den Datensatz Osm-Cz.

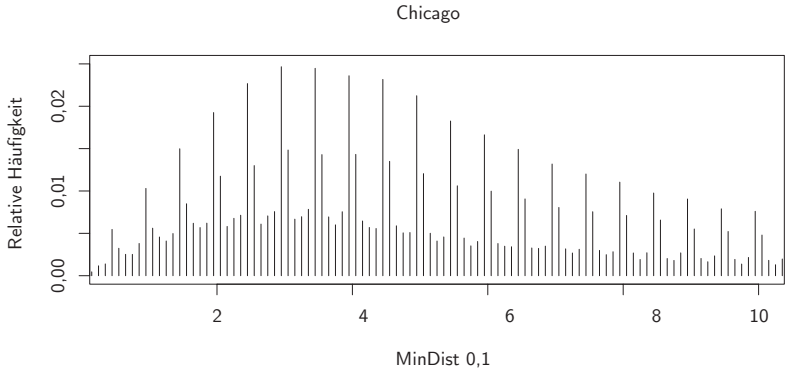


Abbildung 7.36: Stabdiagrammausschnitt der minimalen Distanz zu benachbarten Objekten

Integritätsbedingungen

Für die minimale Distanz zu benachbarten Objekten werden lediglich statistische Bedingungen aufgestellt, da weder ein Minimum noch ein Maximum sinnvoll sind. Die entsprechende Integritätsbedingung im Programm 7.13 fordert einen minimalen unteren und einen maximalen oberen hinge. Dabei wird davon ausgegangen, dass durch eine Anreicherung der Daten das Attribut minDistance für jedes Gebäude vorab bestimmt wird.

```
context Gebaeude
  inv MinDist: let minDistFiveNum: FiveNumberSummary = Gebauede.allInstances()->collect(minDistance
      )->fiveNumberSummary() in minDistFiveNum.lowerHinge() >= 3 and minDistFiveNum.upperHinge() <
       25 -- [Meter]
```

Programm 7.13: Integritätsbedingungen für die minimale Distanz zu benachbarten Objekten

7.6.3 Weitere alphanumerische Attribute

Neben den geometrischen und topologischen Maßen lassen sich mit dem in Abschnitt 7.5 vorgestellten MapReduce-Workflow auch die Häufigkeiten von beliebigen alphanumerischen Attributwerten bestimmen. Als Beispiel wird im Folgenden das Attribut building der Osm-Datensätze verwendet.

Typ der Gebäude

Der Objektartenkatalog von Osm sieht für das Attribut building mehrere mögliche Werte vor (OpenStreetMap, 2013b). Falls kein spezifischer Typ, wie beispielsweise 'hotel' oder 'garage', bestimmt werden kann, dann soll der Wert 'yes' für das Attribut gesetzt werden. In Tabelle 7.20 ist die Auswertung der relativen Häufigkeiten für die Datensätze Osm-Cz und Osm-Fr aufgelistet. Diese zeigt auf, dass selten von den im Objektartenkatalog definierten Gebäudetypen Gebrauch gemacht wird. Stattdessen weisen nahezu alle Gebäude den unspezifischen Typ 'yes' auf.

Osm-Cz		Osm-Fr	
Wert	[%]	Wert	[%]
yes	99,25	yes	99,50
garage	0,25	house	0,18
house	0,18	roof	0,13
residential	0,08	residential	0,03
garages	0,06	garage	0,03

Tabelle 7.20: Verteilung des Typs der Gebäude in den OpenStreetMap-Daten

7.6.4 Datenbereinigung

Basierend auf den in den Unterabschnitten 7.6.1 und 7.6.2 aufgestellten geometrischen und topologischen Integritätsbedingungen können die Daten nun um fehlerhafte beziehungsweise auffällige Objekte bereinigt werden. Im Folgenden werden dazu zwei MapReduce-Workflows vorgestellt, mit denen die Filterung der Objekte beziehungsweise die Erstellung einer Statistik über die Filterung der Objekte durchgeführt werden kann. Die Filterung der Objekte mit einem MapReduce-Workflow wird im Folgenden nur konzeptionell vorgestellt, wohingegen die statistischen Ergebnisse des zweiten MapReduce-Workflows detailliert diskutiert werden.

Filterung der Objekte

Der MapReduce-Workflow für die Filterung der Objekte ist in Abbildung 7.37 dargestellt. Der Workflow unterscheidet sich gegenüber den bereits vorgestellten Workflows lediglich durch unterschiedliche Map- und Reduce-Funktionen. Im Workflow ist der Fall dargestellt, dass lediglich eine einzige Ausgabedatei erstellt werden soll, beispielsweise um diese anschließend in eine einzige Datei im Esri Shapefile-Format auf das lokale Dateisystem des Benutzers kopieren zu können. Der Workflow erlaubt jedoch eine beliebige Anzahl an Ausgabedateien, die durch die Anzahl R der Reduce-Funktionen festgelegt werden kann.

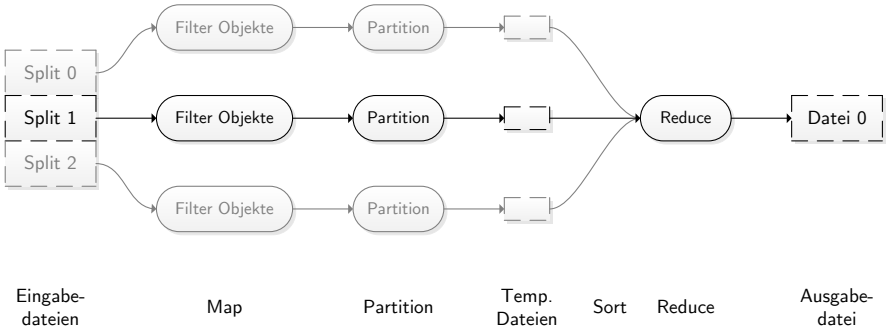


Abbildung 7.37: MapReduce-Workflow für die Filterung der Objekte

Der MapReduce-Workflow für die Filterung der Objekte besteht aus den folgenden sechs Schritten, wobei bereits detailliert ausgeführte Aspekte entsprechend verkürzt wiedergegeben werden:

1. *Unterteilung der Eingabedateien*: Die Eingabedateien für den MapReduce-Workflow bilden Dateien im GeoAvro-Format (siehe Unterabschnitt 7.3.2).

2. *Map*: Für jeden der M Splits wird ein eigener Map-Task erzeugt, der diejenigen Objekte aus den Daten filtert, die nicht den Integritätsbedingungen entsprechen. Als Eingabe erhält die Map-Funktion analog zu den anderen MapReduce-Workflows ein Objekt im GeoAvro-Format als Schlüssel, sowie einen vernachlässigbaren Wert (siehe Tabelle 7.21). Erfüllt ein Objekt die Integritätsbedingungen, dann wird es in die Ausgabe der Map-Funktion geschrieben. Der Schlüssel der in die Ausgabe geschrieben wird, entspricht dem Schlüssel der bei der Anreicherung der Objekte verwendet wird (siehe Unterabschnitt 7.4.3).

Funktion	Eingabe	Ausgabe
map	$(AvroKey<GR>, NullWritable)$	$list(AvroKey<Utf8>, AvroValue<GR>)$
reduce	$(AvroKey<Utf8>, list(AvroValue<GR>))$	$list(AvroKey<GR>, NullWritable)$

Tabelle 7.21: Schlüssel und Werte der Map- und Reduce-Funktionen für die Filterung der Objekte

3. *Partition*: Für die MapReduce-Partitionierung wird die Standardimplementierung verwendet.

4. *Shuffle und Sort*: Die beiden Schritte Shuffle und Sort entsprechen der Standardimplementierung von Hadoop MapReduce.

5. *Reduce*: Im Reduce-Schritt werden die Objekte der einzelnen Partitionen zu einem gemeinsamen Datensatz zusammengeführt.

Erstellung einer Statistik über die Filterung der Objekte

Mit diesem MapReduce-Workflow werden nicht die Objekte selbst gefiltert, sondern es wird für jede Integritätsbedingung eine Statistik erstellt wie viele Objekte die Bedingung verletzen. Damit kann eine Aussage darüber getroffen werden, welche Integritätsbedingungen am häufigsten verletzt werden.

Der in Abbildung 7.38 dargestellte MapReduce-Workflow zur Erstellung einer Statistik über die Filterung ist vergleichbar zum MapReduce-Workflow zur Bestimmung der Häufigkeit von Attributwerten (siehe Unterabschnitt 7.5.1).

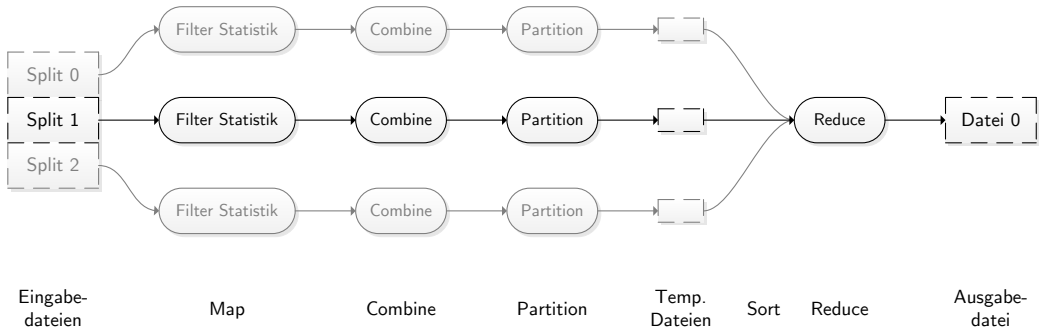


Abbildung 7.38: MapReduce-Workflow für die Erstellung einer Statistik über die Filterung der Objekte

Der MapReduce-Workflow besteht aus den folgenden sechs Schritten:

1. *Unterteilung der Eingabedateien*: Die Eingabedateien für den MapReduce-Workflow bilden Dateien im GeoAvro-Format (siehe Unterabschnitt 7.3.2).
2. *Map*: Für jeden der insgesamt M Splits wird ein eigener Map-Task erzeugt, der für alle Objekte des Splits überprüft ob diese die Integritätsbedingungen erfüllen. Als Eingabe erhält die Map-Funktion jeweils ein Objekt im Avro-Format als Schlüssel, sowie einen vernachlässigbaren Wert (siehe Tabelle 7.22).

Funktion	Eingabe	Ausgabe
map	$(AvroKey<GR>, NullWritable)$	$list(Text, LongWritable)$
reduce	$(Text, list(LongWritable))$	$list(Text, LongWritable)$

Tabelle 7.22: Schlüssel und Werte der Map- und Reduce-Funktionen für die Erstellung einer Statistik über die Filterung der Objekte

   Das Avro-Objekt muss dabei über genau die Attribute verfügen, die auch in den Integritätsbedingungen überprüft werden. Dabei ist es wichtig, dass sowohl die Attributnamen als auch die Datentypen zwischen den Daten und den Bedingungen übereinstimmen. Attribute an die keine Integritätsbedingungen gestellt werden, können über eine Projektion des Eingabeschemas bereits aus den Eingabedaten der Map-Funktion entfernt werden (siehe auch Unterabschnitt 2.8.4).

   Verletzt ein Objekt eine Integritätsbedingung bezüglich eines bestimmten Attributs, dann erstellt die Map-Funktion ein Schlüssel-Wert-Paar vom Datentyp $(Text, LongWritable)$ mit den Werten $(Attributname, 1)$. Verletzt das Objekt eine weitere Integritätsbedingung, dann wird auch für diese ein entsprechendes Schlüssel-Wert-Paar erzeugt. Um zudem die Anzahl der Objekte ermitteln zu können, die mindestens eine Integritätsbedingung verletzen, erzeugt die Map-Funktion für jedes betroffene Objekt ein zusätzliches Schlüssel-Wert-Paar mit dem Schlüssel *invalid* und dem Wert *1*.
3. *Combine*: Der Combine-Schritt entspricht einem lokalen Reduce, bei dem die Ergebnisdaten eines einzelnen Map-Tasks zusammengefasst werden.
4. *Partition*: Für die MapReduce-Partitionierung wird die Standardimplementierung verwendet.
5. *Shuffle und Sort*: Die beiden Schritte Shuffle und Sort entsprechen der Standardimplementierung von Hadoop MapReduce.
6. *Reduce*: Da die Statistik über die Filterung der Objekte global für alle Objekte eines Datensatzes bestimmt werden soll, wird lediglich ein einziger Reduce-Task erzeugt. Dieser summiert die Werte für jeden Schlüssel auf und bestimmt damit die absolute Anzahl der Objekte die eine bestimmte Integritätsbedingung verletzen sowie die absolute Anzahl der Objekte die mindestens eine Integritätsbedingung verletzen.

Statistik über die Filterung der Objekte

Die statistische Auswertung über die Filterung ermöglicht Aussagen über die Anzahl der fehlerhaften Objekte sowie über die Verteilung der Fehler bezüglich der aufgestellten Integritätsbedingungen. Die Tabelle 7.23 gibt einen Überblick über die fehlerhaften Objekte der drei Datensätze Chicago, Osm-Cz und Osm-Fr. Der prozentuale Anteil der fehlerhaften Objekte liegt für den Datensatz Chicago bei lediglich 0,25 %. Dabei weisen die meisten Objekte nur einen einzigen Fehler auf, da die Gesamtanzahl der Fehler nur geringfügig höher ist. Für den OpenStreetMap-Datensatz Tschechiens ist der prozentuale Anteil der Fehler mit 1,61 % deutlich höher. Auch für diesen Datensatz weisen die meisten Objekte nur einen einzigen Fehler auf. Für den OpenStreetMap-Datensatz Frankreichs ist der prozentuale Anteil der Fehler mit 7,35 % schließlich am höchsten. Da die Anzahl der Fehler deutlich über der Anzahl der fehlerhaften Objekte liegt, weisen viele Objekte mehr als einen Fehler auf. Für die aufgestellten Integritätsbedingungen lässt sich zusammenfassend also feststellen, dass die Qualität der Datensätze deutlich unterschiedlich ist.

	Chicago		Osm-Cz		Osm-Fr	
	Absolut	Prozentual	Absolut	Prozentual	Absolut	Prozentual
Anzahl der Objekte	820 146	—	2 791 447	—	31 782 785	—
Anzahl der fehlerhaften Objekte	2054	0,25	45 330	1,62	2 335 752	7,35
Anzahl der Fehler	2279	—	49 681	—	4 484 503	—

Tabelle 7.23: Überblick über die fehlerhaften Objekte

Die Verteilung der Fehler ist für die drei Datensätze in Tabelle 7.24 aufgelistet. Die prozentualen Fehler folgen dabei aus dem Verhältnis der absoluten Anzahl der Fehler zur Gesamtanzahl der Fehler aus Tabelle 7.23. Bei allen Datensätzen sind wenige Integritätsbedingungen für die Mehrzahl der Fehler verantwortlich. Für den Datensatz Chicago sind die drei Bedingungen an den minimalen Winkel zwischen aufeinanderfolgenden Liniensegmenten (MinVxDist), an die Fläche (Area) und an die Größe der Schnittfläche mit benachbarten Objekten (IntArea) zusammen für 76,2 % der Fehler verantwortlich. Für den Datensatz Osm-Cz sind sogar nur die beiden Bedingungen an den minimalen Winkel zwischen aufeinanderfolgenden Liniensegmenten (MinVxDist) sowie an die Größe der Schnittfläche mit benachbarten Objekten (IntArea) für insgesamt 80,7 % der Fehler verantwortlich. Für den Datensatz Osm-Fr sind die drei Bedingungen an die Fläche (Area), an die fraktale Dimension (FracDim) sowie an den minimalen Durchmesser des umschließenden Rechtecks (MMinWidth) für insgesamt 85,1 % der Fehler verantwortlich. Daraus wird ersichtlich, dass für die Überprüfung von Gebäudeumrissen bereits wenige Integritätsbedingungen zu einer deutlichen Verbesserung der geometrischen und topologischen Qualität führen. Aus den prozentualen Werten für die Fehler folgt jedoch auch, dass die Verteilung der Fehler abhängig vom überprüften Datensatz ist. Aussagen die anhand der Verteilung der Fehler für einen Datensatz getroffen werden sind damit nur bedingt auf andere Datensätze übertragbar.

	Chicago		Osm-Cz		Osm-Fr	
Integritätsbedingung	Absolut	Prozentual	Absolut	Prozentual	Absolut	Prozentual
NumVx	27	1,2	531	1,1	143 181	3,2
MinVxDist	0	0,0	703	1,4	71 884	1,6
MinVxAngle	324	14,2	8957	18,0	155 007	3,5
Area	646	28,3	2003	4,0	1 565 886	34,9
Comp	59	2,6	303	0,6	16 380	0,4
FracDim	27	1,2	1030	2,1	962 650	21,5
Rect	125	5,5	1295	2,6	21 950	0,5
MMinWidth	134	5,9	1374	2,8	1 287 409	28,7
MElong	155	6,8	2079	4,2	125 665	2,8
MAreaDiff	7	0,3	165	0,3	5398	0,1
IntArea	775	34,0	31 241	62,9	129 093	2,9

Tabelle 7.24: Verteilung der Fehler

7.7 Anforderungen

Die für die Gebäudegrundrisse in Open Data berücksichtigten Anforderungen aus dem Anforderungskatalog in Kapitel 4 sind in Tabelle 7.25 aufgelistet. Die nicht in der Tabelle enthaltenen Anforderungen sind entweder nicht berücksichtigt, da sie auf die verwendeten Daten nicht zutreffen, oder weil sie bewusst nicht ausgewählt werden. Die verbleibenden nicht berücksichtigten und nicht ausgewählten Anforderungen werden bereits in Abschnitt 6.5 diskutiert und werden an dieser Stelle nicht nochmals wiederholt. Für die Gebäudegrundrisse ist der Grund für die nicht durchgeführte Korrektur der Fehler schlicht dass keine entsprechende Software vorhanden ist und deren Entwicklung auch nicht zur Thematik dieser Arbeit passt.

Nr.	Anforderung	Diskussion
1	Diskrete und kontinuierliche Objekte	Die Gebäudegrundrisse in Open Data sind diskrete Objekte.
9	Anzahl der Attributwerte/Attribute/Objekte/Klassen	Die Attribute weisen jeweils nur einen Wert auf. Zudem verwenden nahezu alle aufgestellten Integritätsbedingungen nur ein einzelnes Attribut. An die Anzahl der Objekte werden keine Bedingungen gestellt. Da nur eine Klasse verwendet wird, werden auch keine klassenübergreifenden Bedingungen benötigt.
11	Methoden für die Ableitung von Geometrien	Es werden einige Methoden für die Ableitung von Geometrien verwendet, wie beispielsweise die Ableitung der Stützpunkte und der Liniensegmente aus den Umringen der Flächen sowie die Ableitung von umschließenden Rechtecken.
12	Funktionale Abhängigkeit	Die funktionale Abhängigkeit wird in der Anwendung implizit berücksichtigt. Keines der im Originaldatensatz vorhandenen Attribute weist eine funktionale Abhängigkeit auf. Dagegen werden alle angereicherten Attribute aus der Geometrie und Topologie der Objekte bestimmt und sind damit funktional von diesen abhängig. Dieser Zusammenhang muss jedoch nicht als Integritätsbedingungen ausgedrückt werden, da die angereicherten Attribute automatisch und damit korrekt erzeugt werden.
14	Fordern vs. Einschränken	Alle Integritätsbedingungen sind in der Anwendung als fordernd definiert, da diese jeweils weniger Vergleichswerte benötigen.
15	Wertebereich	Nahezu alle aufgestellten Integritätsbedingungen schränken den gültigen Wertebereich der Attribute ein.
16	Genauigkeit	Die Genauigkeit der Attributwerte wird durch eine geeignete Wahl von Nachkommastellen berücksichtigt. So sind beispielsweise die Bedingungen an die Flächen mit Metergenauigkeit angegeben, wohingegen die Schwellwerte für die fraktale Dimension mit einer Nachkommastelle angegeben sind.
23	Umfangreiche Datensätze	Durch die Verwendung von MapReduce-Workflows können selbst für umfangreiche Datensätze Integritätsbedingungen aufgestellt und überprüft werden.

Tabelle 7.25: Berücksichtigte Anforderungen für die Gebäudegrundrisse in Open Data

7.8 Implementierung

Für die in diesem Kapitel beschriebene Parallelisierung der Verarbeitung von Geodaten werden mehrere Softwareprodukte eingesetzt. Dabei werden die in Abschnitt 6.6 verwendeten beziehungsweise erstellten Softwareprodukte gleichermaßen hier verwendet und wo notwendig auf die Besonderheiten der Datenverarbeitung mit MapReduce-Workflows angepasst. Die verwendeten Softwareprodukte werden im Folgenden kurz vorgestellt.

Aufbereitung der Daten

Die in Abschnitt 7.2 vorgestellten Daten liegen in zwei unterschiedlichen Formaten vor. Die Gebäudegrundrisse der Stadt Chicago werden im Esri Shapefile-Format bereitgestellt und können somit nach einer Konvertierung in das GeoAvro-Format (siehe Abschnitt 7.3) direkt in den MapReduce-Workflows verwendet werden.

Dagegen müssen die Daten des OpenStreetMap-Projekts mit einem Workflow konvertiert werden (siehe Abbildung 7.39). Die vollständigen Daten werden vom Osm-Projekt unter anderem im Protocolbuffer Binary Format (Pbf) bereitgestellt (OpenStreetMap, 2013c). Dieses effiziente binäre Format verwendet zur Serialisierung der Daten das Framework Protobuf (2012) von Google (siehe auch Unterabschnitt 2.8.4). Mit dem Importer imposm

können die Dateien im PBF-Format dann in eine räumliche Datenbank importiert werden (Omniscale, 2013). Beim Import können Filter gesetzt werden, so werden beispielsweise für diese Arbeit nur Objekte vom Typ Gebäude importiert. Die Daten werden von imposm in eine PostGIS (2013) Datenbank geschrieben. PostGIS erweitert die objektrelationale Datenbank PostgreSQL (2013) um räumliche Datentypen und Funktionen. Mit dem Werkzeug pgsql2shp von PostGIS werden die Objekte wieder aus der Datenbank in ein Dateiformat exportiert. Die so erstellten Dateien im ESRI Shapefile-Format werden schließlich in das GeoAvro-Format exportiert und können dann in den MapReduce-Workflows verarbeitet werden.

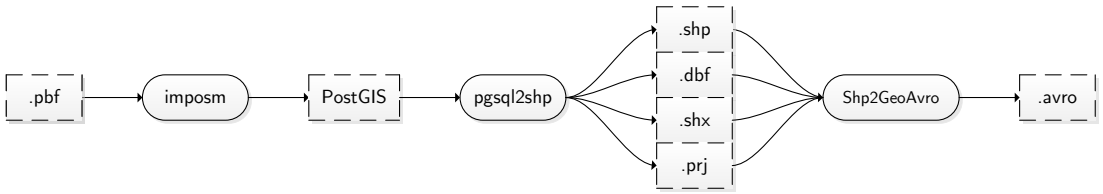


Abbildung 7.39: Workflow für die Konvertierung der OpenStreetMap-Daten

Verarbeitung der Daten mit MapReduce

Die Anreicherung der Objekte um geometrische und topologische Maße in Abschnitt 7.4 sowie die Bestimmung der Häufigkeiten von Attributwerten in Abschnitt 7.5 werden mit selbst entwickelten MapReduce-Programmen durchgeführt. Die Programme sind in der Programmiersprache Java geschrieben und verwenden zur Verarbeitung von Geodaten die beiden quelloffenen Bibliotheken GeoTools (2013) und Java Topology Suite (JTS) (Davis, 2013).

Die Softwareentwicklung und Tests werden in einer virtuellen Maschine durchgeführt, die dieselbe Software verwendet wie die Server des Hadoop-Clusters (siehe Unterabschnitt 7.4.4). Dadurch ist sichergestellt, dass MapReduce-Jobs, die lokal auf der virtuellen Maschine des Softwareentwicklers lauffähig sind, auch auf dem Cluster ohne aufwändige Anpassungen ausgeführt werden können.

Statistik und Klassenbeschreibung

Aus den Häufigkeiten der Attributwerte werden in Abschnitt 7.6 statistische Auswertungen und Klassenbeschreibungen abgeleitet. Dazu wird das quelloffene Statistikprogramm R (2013) verwendet. Dieses wird um selbst entwickelte Funktionen erweitert, um unter anderem statistische Lagemaße auf Basis der Häufigkeiten von Attributwerten bestimmen zu können.

7.9 Übertragbarkeit

Als Abschluss des Kapitels wird nochmals diskutiert, was durch die Analyse der Daten erreicht wurde und wie sich das vorgestellte Vorgehen auf gleichartige Datensätze übertragen lässt. Als gleichartig sind dabei alle Datensätze anzusehen, die punkt-, linien- oder flächenhaft und im Gegensatz zu Kapitel 6 nicht flächenüberdeckend sind. Bereits in Abschnitt 6.7 diskutierte Aspekte werden dabei nicht wiederholt.

Die Analyse der Daten verfolgt in diesem Kapitel zwei Ziele. Das erste Ziel ist die Klassenbeschreibung, mit der die charakteristischen Eigenschaften der Gebäudegrundrisse der drei verwendeten Datensätze beschrieben werden. Wenn die Integritätsbedingungen aufgestellt sind, dann können weitere Gebiete sowie neu erfasste oder modifizierte Objekte auf die Einhaltung der Bedingungen überprüft werden. Das zweite Ziel der Analyse der Daten ist die Parallelisierung, mit der aufgezeigt wird wie Integritätsbedingungen selbst für Daten aufgestellt werden können, deren serielle Prozessierung aufgrund der Komplexität der Bedingungen oder des Umfangs der Daten zu viel Zeit oder zu viele Ressourcen in Anspruch nehmen würde.

Das methodische Vorgehen für die Klassenbeschreibung entspricht weitgehend den in Abschnitt 6.7 beschriebenen Schritten. So verwendet die Anreicherung der Daten in Abschnitt 7.4 größtenteils dieselben geometrischen Maße wie in Unterabschnitt 6.2.2. Die topologischen Maße sind jedoch unterschiedlich, da Gebäudegrundrisse nicht flächenüberdeckend sind und dadurch andere Maße sinnvoll sind. Für andere punkt-, linien- oder flächenhaft Datensätze können die aufgestellten Maße teilweise wiederverwendet und um weitere sinnvolle Maße ergänzt werden. So können beispielsweise auch Maße für die Beschreibung von topologischen Netzwerken aufgestellt werden, wie der minimale und maximale Grad eines Knotens.

Mit Methoden der deskriptiven Statistik und der explorativen Datenanalyse wird schließlich in Abschnitt 7.6 die Klassenbeschreibung für die Gebäudegrundrisse abgeleitet. Mit Hilfe von Pentagramm und Stabdiagramm werden Integritätsbedingungen in der GeoOcl erstellt. Für die Gebäudegrundrisse ergeben sich insgesamt 15 Integritätsbedingungen basierend auf elf geometrischen und zwei topologischen Maßen. Die Maße zeigen auf, dass die Ähnlichkeit zwischen den beiden Osm Datensätzen Frankreichs und Tschechiens größer ist als zum Datensatz Chicago. Dies lässt den Schluss zu, dass sich die Bauweise einer amerikanischen Großstadt deutlich von der europäischen Bauweise unterscheidet. Insgesamt weisen die verwendeten Datensätze einen Anteil an fehlerhaften beziehungsweise auffälligen Objekten zwischen 0,3 % und 7,4 % auf. Für andere Datensätze ist das Vorgehen für die Klassenbeschreibung analog, d.h. es können die gleichen Werkzeuge und Überlegungen verwendet werden um entsprechende Integritätsbedingungen in der GeoOcl abzuleiten.

Der Aspekt der Parallelisierung wird in Abschnitt 7.1 allgemein bezüglich der Daten- und Task-Parallelität sowie der räumlichen Partitionierung diskutiert. Dabei werden Stragien vorgestellt wie Geometrien geeignet partitioniert werden. Um die Gebäudegrundrisse der verwendeten Datensätze mit dem Hadoop-Framework verarbeiten zu können, werden diese in Abschnitt 7.3 zuerst in das neu entworfene und standardbasierte Dateiformat GeoAvro konvertiert. Die Anreicherung der Daten wird in Abschnitt 7.4 mit einem MapReduce-Workflow durchgeführt. Dieser reichert die Daten in den Map-Tasks um die geometrischen Maße an. Die Anreicherung um die topologischen Maße erfolgt nach einer Partitionierung der Daten in den Reduce-Tasks. Als geeignete Partitionsgröße werden quadratische Partitionen mit einer Seitenlänge von 10 km und einem Randbereich von 2 km verwendet. Die Anreicherung der Daten skaliert dabei annähernd linear, was eine Übertragbarkeit des MapReduce-Workflows auf beliebig umfangreiche Datensätze ermöglicht. Um die Klassenbeschreibung mit den gleichen Werkzeugen wie in Abschnitt 6.3 durchführen zu können, muss vorab die Häufigkeit der Attributwerte bestimmt werden. Dazu werden in Abschnitt 7.5 die angereicherten Daten durch einen MapReduce-Workflow geeignet diskretisiert. Sind Integritätsbedingungen für die Daten aufgestellt, dann können die Daten, wie in Unterabschnitt 7.6.4 gezeigt, durch zwei MapReduce-Workflows zuerst gefiltert und dann statistisch ausgewertet werden. Die gesamte Prozessierung der Daten zeigt eindrucksvoll auf, wie mehrere MapReduce-Workflows sinnvoll nacheinander ausgeführt beziehungsweise orchestriert werden. Dabei wird auch diskutiert wie durch eine geeignete Wahl der Anzahl der Map- und Reduce-Tasks die Gesamtlaufzeit optimiert werden kann. Die beschriebenen MapReduce-Workflows können mit einer geeigneten Parametrisierung und gegebenenfalls geringfügigen Anpassungen auf beliebige Datensätze übertragen werden.

8 Zusammenfassung und Ausblick

8.1 Zusammenfassung

Mit den in dieser Arbeit vorgestellten Verfahren wird eine einfache, effiziente und standardkonforme Definition, Modellierung und Überprüfung von Integritätsbedingungen ermöglicht. Damit werden sowohl Produzenten als auch Nutzern von Geodaten die notwendigen Grundlagen und Werkzeuge an die Hand gegeben um Integritätsbedingungen für ihre Datensätze verwenden zu können.

Integritätsbedingungen sind für alle Geodaten von Interesse, wie die unterschiedlichen Anwendungen in den beiden Bereichen amtlicher Geobasisdaten und Fachdaten aufzeigen. Wie am Beispiel der Flächennutzung aufgezeigt wurde, sind von amtlicher Seite erfasste Geobasisdaten nicht fehlerfrei, sondern müssen mit der gleichen Sorgfalt untersucht werden wie Datensätze anderer Herkunft. Die Auflistung einiger Klassifikationen von Integritätsbedingungen dient vor allem der Sensibilisierung für die Bereiche in denen Bedingungen auftreten können. Zudem können die Klassifikationen die Grundlage für eine gemeinsame Terminologie in einer bestimmten Domäne bilden.

Der erarbeitete Anforderungskatalog gibt mit seinen 27 Anforderungen nicht nur einen einzigartigen Überblick über alle wichtigen Aspekte von Integritätsbedingungen, sondern legt durch die detaillierte Beschreibung und Diskussion auch die Grundlage für weitere Forschungen. Der Katalog deckt von der Modellierung, über die Definition von Bedingungen und die Aufstellung einzelner Bedingungen bis zur Prüfung von Daten mit Bedingungen und schließlich bis zur Änderung von Daten und Bedingungen alle Schritte ab. Aus den umfangreichen Anforderungen können von Produzenten und Nutzern dabei genau diejenigen ausgewählt werden, die für die jeweilige Anwendung von Relevanz sind.

Die Modellierung der Integritätsbedingungen basiert auf der Model Driven Architecture (MDA) und insbesondere auf der Object Constraint Language (OCL). Mit der Erweiterung der OCL zur GeoOCL können räumliche Bedingungen eindeutig und plattformunabhängig formalisiert werden. Die GeoOCL befindet sich dabei auf der geeignetsten Abstraktionsebene der Formalisierung und kann zudem alle aufgestellten Anforderungen abdecken. Durch die Berücksichtigung entsprechender Standards des Open Geospatial Consortium (OGC) und der International Organization for Standardization (ISO) erfüllt die GeoOCL wichtige Grundvoraussetzungen um selbst als Standard etabliert werden zu können. Die vielen Beispiele von Bedingungen in der GeoOCL in der gesamten Arbeit belegen schließlich deutlich die Praxistauglichkeit der Sprache.

Die Untersuchung der Flächennutzung in Geobasisdaten zeigt mehrere Anwendungen von Integritätsbedingungen auf. Das vorgestellte Vorgehen ist dabei repräsentativ für vollständig oder nahezu vollständig flächenüberdeckende Datensätze. Durch die Anreicherung der Objekte um geometrische und topologische Maße sowie deren Auswertung mittels deskriptiver Statistik und explorativer Datenanalyse können geeignete Klassenbeschreibungen für die Flächennutzungen in der GeoOCL erstellt werden. Diese unterscheiden sich signifikant zwischen den einzelnen Nutzungen und zeigen so interessante Muster und Wissen in den Daten auf. Für die zehn ausgewählten Objektarten der Automatisierten Liegenschaftskarte (ALK) ergeben sich insgesamt 92 Integritätsbedingungen basierend auf acht geometrischen und 33 topologischen Maßen. Insgesamt weisen die verwendeten Datensätze der ALK einen Anteil an fehlerhaften beziehungsweise auffälligen Objekten zwischen 1,3 % und 1,9 % auf. Die aufgestellten Integritätsbedingungen sind dabei nicht auf die verwendeten Gebiete beschränkt, sondern sind auf Teile oder die Gesamtheit der Objekte der ALK Deutschlands übertragbar. Das Data Mining kann jedoch auch für die Klassifikation eingesetzt werden, wobei die Zusammenhänge der Flächennutzung zwischen der ALK und dem Amtlichen Topographisch-Kartographischen Informationssystem (ATKIS) aufgedeckt werden. Durch die Klassifikation wird die Frage beantwortet, ob aus den Daten der ALK automatisch Flächennutzungen in ATKIS abgeleitet werden können. Dabei zeigt sich, dass die Klassifikationsgüte proportional zur Komplexität des verwendeten Modells ansteigt und nicht alle Flächennutzungen gleich gut prädiziert werden können. Die auf den Daten des Gebiets Hildesheim-West erlernten Modelle erreichen gute Erkennungsraten für das Testgebiet Hildesheim-Ost. Der OneRule-Algorithmus erreicht mit elf Regeln bereits eine Erkennungsrate von 71,6 % und mit einer Steigerung der Komplexität durch den Entscheidungsbaum-Algorithmus kann eine Erkennungsrate von 74,4 % bei 42 Regeln erreicht werden. Sind die Gebiete also vergleichbar, dann liefert die Klassifikation mit den genannten Algorithmen gute Erkennungsraten. Sind die Gebiete jedoch nicht vergleichbar, wie bei dem Testgebiet Hannover, dann ist die Erkennungsrate mit maximal 51,8 % nicht zufriedenstellend. Es ist demnach

wichtig entweder einen entsprechend homogenen Gesamtdatensatz als Ausgangsbasis für die Klassifikation zu haben, oder mehrere Modelle für mehrere charakteristische Typen von Gebieten aufzustellen. Leider konnten solche Daten für diese Arbeit nicht bereitgestellt werden. Mit den verwendeten Algorithmen lassen sich aus dem Ergebnis der Klassifikation ebenso Integritätsbedingungen in der GeoOcl ableiten.

Die Praktikabilität von Integritätsbedingungen selbst für umfangreiche Datenmengen zeigt die Untersuchung der Gebäude in Open Data mit bis zu 31 Millionen Objekten pro Datensatz. Das vorgestellte Vorgehen ist dabei repräsentativ für punkt-, linien- oder flächenhafte Datensätze. Um die Daten geeignet parallel verarbeiten zu können, müssen diese jedoch zuerst räumlich partitioniert werden. Die Anreicherung der Objekte um geometrische und topologische Maße, die Bestimmung der Häufigkeit von Attributwerten sowie die Filterung erfolgt anschließend mit MapReduce-Workflows. Die Anreicherung der Daten skaliert dabei annähernd linear, womit sich der gewählte Ansatz dazu eignet auf beliebig umfangreiche Daten angewendet zu werden. Die gesamte Prozessierung der Daten zeigt eindrucksvoll auf, wie mehrere MapReduce-Workflows sinnvoll nacheinander ausgeführt beziehungsweise orchestriert werden. Dabei wird auch diskutiert wie durch eine geeignete Wahl der Anzahl der Map- und Reduce-Tasks die Gesamtlaufzeit optimiert werden kann. Die beschriebenen MapReduce-Workflows können mit einer geeigneten Parametrisierung und gegebenenfalls geringfügigen Anpassungen auf beliebige Datensätze übertragen werden. Die anschließende Klassenbeschreibung zeigt wiederum interessante Muster in den Daten auf, wobei diese teilweise charakteristische Besonderheiten für die verwendeten Datensätze aufzeigen. Für die Gebäudegrundrisse ergeben sich insgesamt 15 Integritätsbedingungen basierend auf elf geometrischen und zwei topologischen Maßen. Die Maße zeigen auf, dass die Ähnlichkeit zwischen den beiden OpenStreetMap (Osm) Datensätzen Frankreichs und Tschechiens größer ist als zum Datensatz Chicago. Dies lässt den Schluss zu, dass sich die Bauweise einer amerikanischen Großstadt deutlich von der europäischen Bauweise unterscheidet. Insgesamt weisen die verwendeten Datensätze einen Anteil an fehlerhaften beziehungsweise auffälligen Objekten zwischen 0,3 % und 7,4 % auf. Für andere Datensätze ist das Vorgehen für die Klassenbeschreibung analog, d.h. es können die gleichen Werkzeuge und Überlegungen verwendet werden um entsprechende Integritätsbedingungen in der GeoOcl abzuleiten.

8.2 Ausblick

Trotz der tiefgehenden und umfangreichen Betrachtung von Integritätsbedingungen bleiben Aspekte, die es wert sind weitergehend untersucht zu werden.

Die Implementierung der GeoOcl ist noch durchzuführen. Diese basiert jedoch unter anderem darauf, welches Werkzeug dafür ausgewählt wird. Wichtig wäre hier vor allem die Bereitstellung eines ausgereiften Prototyps eines Modellierungswerkzeuges, beispielsweise basierend auf dem Dresden Ocl Toolkit, um die Modellierung mit der GeoOcl weiter zu vereinfachen und damit die Eintrittsschwelle weiter zu senken.

Die Untersuchung der Flächennutzung in Geobasisdaten ist lediglich ein Beispiel für die Aufstellung von Integritätsbedingungen für amtliche Daten. Eine Erweiterung auf alle Flächennutzungen kann zu einem besseren Verständnis der Zusammenhänge führen. Die verwendeten Algorithmen für die Klassifikation lassen sich auf andere Anwendungen übertragen. Zusätzlich eignen sich einige weitere Verfahren des Data Mining für die Aufstellung von Integritätsbedingungen, wie beispielsweise Assoziationsanalysen. Eine Untersuchung welche Verfahren für welche Fragestellungen geeignet sind und welche Anforderungen und Einschränkungen die Verfahren jeweils bieten, kann eine wichtige Entscheidungshilfe für die Wahl eines Data Mining Verfahrens liefern.

Die Übertragung der entwickelten MapReduce-Workflows auf noch umfangreichere Datensätze und die Untersuchung der daraus resultierenden Laufzeiten kann weitere Einblicke in die Skalierbarkeit liefern. Dazu eignen sich unter anderem die Daten des Osm-Projekts, da diese weltweit kostenlos verfügbar sind. Interessant wäre zudem der Entwurf und die Implementierung von MapReduce-Workflows, die mehrere Datensätze beziehungsweise Objektarten parallel verarbeiten können, um beispielsweise die Zusammenhänge zwischen den Objektarten aufdecken zu können. Wenn häufig vergleichbare Workflows auf die gleichen Daten angewendet werden, dann kann eine Speicherung der Daten in einer verteilten Datenbank, wie beispielsweise Apache HBase, Vorteile bei der Speicherung und Abfrage von Daten bieten.

A Anhang

A.1 Integritätsbedingungen: Flächennutzung in Geobasisdaten

Objektart	Anzahl
Campingplatz	1
BetriebsflächeVersorgungsanlage	2
Obstanbaufläche	2
See	2
VerkehrsflächeUngenutztEinschlBegleitfläche	2
BetriebsflächeUngenutzt	3
Übungsgelände	5
Bahngelände	7
LandwirtschaftlicheBetriebsfläche	7
GuFEntsorgungsanlagen	9
Brachland	10
HistorischeAnlage	12
BetriebsflächeLagerplatz	14
Fluss	14
Nadelwald	14
FlussEinschlBegleitfläche	17
Friedhof	17
Verkehrsbegleitfläche	19
Schutzfläche	25
Teich	25
GuFErholung	30
Bach	32
Unland	34

Objektart	Anzahl
Mischwald	37
BahngeländeEinschlBegleitfläche	40
Sportfläche	40
GuFUngenutzt	51
Gehölz	53
Laubwald	71
GuFLandForstwirtschaft	75
Platz	86
Graben	121
Grünland	128
Gartenland	170
GuFÖffentlicheZwecke	202
GuFVersorgungsanlagen	207
GuFMischnutzungMitWohnen	211
Ackerland	219
GuFVerkehrsanlagen	236
GuFGewerbeundIndustrie	270
GuFHandelDienstleistungen	292
Grünanlage	491
StrasseEinschlBegleitfläche	1149
Strasse	1166
Weg	1235
GuFWohnen	1664

Tabelle A.1: Objekte und Objektarten in der ALK für das Gebiet Hildesheim

Objektart	Anzahl
Deponie	1
FlugLandeplatz	1
Hafen	1
Hafenbecken	1
HaldeAufschüttung	1
TruppenStandortübungsplatz	1
BrückeÜberUnterführung	2
TagebauGrubeSteinbruch	2
Wasserwerk	2
Campingplatz	3
SchwimmFreibad	3
Umspannstation	3
KläranlageKlärwerk	4
NasserBoden	5
Schwimmbecken	6
Bergbaubetrieb	7
Naturschutzgebiet	7
KanalSchifffahrt	8
Schiessanlage	10
Sonderkultur	10
Freizeitanlage	13

Objektart	Anzahl
Bahnhofsanlage	14
Gärtnerei	19
Platz	20
StromFlussBach	21
Ortslage	48
BinnenStauseeTeich	63
Friedhof	84
Sportanlage	85
Gehölz	95
SpielfeldSpiel	101
Grünanlage	110
Gartenland	111
BesondereFunktionalePrägung	157
IndustrieGewerbe	181
GemischteNutzung	279
Unbestimmbar	293
Grünland	314
WaldForst	583
Ackerland	585
Wohnbau	1419

Tabelle A.2: Objekte und Objektarten in ATKIS für das Gebiet Hildesheim

Objektart	LAck	LGan	LGar	LGla	LHaD	LInG	LMis	LOef	LVer	LWhn	None
LAck	12	11	3	8	0	2	0	1	0	4	59
LGan	5	1	4	3	2	4	0	5	0	42	33
LGar	9	15	4	6	3	3	1	4	0	44	12
LGla	27	8	8	2	1	1	0	2	0	16	37
LHaD	1	6	1	0	0	14	9	5	0	44	20
LInG	3	8	1	1	12	2	4	3	0	43	23
LMis	0	2	0	0	17	6	0	1	0	70	3
LOef	1	16	1	0	5	5	0	0	0	39	31
LVer	3	14	0	0	4	14	1	7	0	33	23
LWhn	1	15	2	2	6	6	4	5	0	1	58

Tabelle A.3: Dominante Objektart am Umring der Objekte der ALK für das Gebiet Hildesheim

Objektart	LAck	LGan	LGar	LGla	LHaD	LInG	LMis	LOef	LVer	LWhn	None
LAck	67	9	1	4	0	3	0	3	0	3	11
LGan	10	10	1	2	2	3	0	6	0	56	10
LGar	15	16	4	5	1	2	0	5	0	49	2
LGla	38	7	2	18	0	2	0	1	0	18	14
LHaD	0	6	1	0	17	11	1	7	0	49	8
LInG	3	4	0	0	14	23	0	2	0	46	8
LMis	0	1	0	0	11	4	0	3	0	76	4
LOef	2	11	0	0	6	4	0	8	0	58	9
LVer	4	6	0	0	3	17	1	7	0	54	8
LWhn	2	6	1	1	3	3	4	4	0	76	3

Tabelle A.4: Dominante Objektart im 50 m Puffer um die Objekte der ALK für das Gebiet Hildesheim

Objektart	LAck	LGan	LGar	LGla	LHaD	LInG	LMis	LOef	LVer	LWhn	None
LAck	84	4	0	1	0	2	0	0	0	2	6
LGan	7	36	1	2	1	4	0	5	0	39	5
LGar	16	5	13	5	1	4	0	3	0	41	14
LGla	27	6	3	40	0	0	0	1	0	12	12
LHaD	1	5	0	0	31	6	2	5	0	41	8
LInG	2	1	0	0	8	39	0	1	1	40	7
LMis	0	1	0	0	11	4	5	5	0	68	6
LOef	3	9	0	0	2	3	0	45	0	29	7
LVer	3	7	0	0	2	16	1	6	20	42	2
LWhn	2	3	1	1	3	3	2	4	0	80	3

Tabelle A.5: Dominante Objektart in den Blöcken der Objekte der ALK für das Gebiet Hildesheim

Objektart	TAck	TBFP	TGan	TGar	TGla	TInG	TMis	TWhn	None
LAck	87	2	0	2	5	1	0	1	2
LGan	2	9	10	13	7	3	3	43	10
LGar	5	8	1	13	4	10	10	45	6
LGla	13	4	1	9	47	2	12	6	7
LHaD	0	2	0	3	1	12	26	52	4
LInG	0	8	1	0	0	36	14	39	1
LMis	0	0	2	0	0	4	14	79	0
LOef	0	41	1	0	0	2	9	43	2
LVer	2	9	2	1	3	17	4	50	12
LWhn	1	2	0	1	0	3	7	86	0

Objektart	TOrt
LAck	8
LGan	76
LGar	72
LGla	22
LHaD	96
LInG	99
LMis	99
LOef	97
LVer	88
LWhn	97

Tabelle A.6: Zusammenhang zwischen den Objekten der ALK und ATKIS für das Gebiet Hildesheim

```
BLAckARel <= 0.17
|   BLInGARel <= 0.14
|   |   L50LHaD <= 0.24
|   |   |   BLGarARel <= 0.26
|   |   |   |   BLGlaARel <= 0.14
|   |   |   |   |   BLOefARel <= 0.54
|   |   |   |   |   |   BLVerARel <= 0.23
|   |   |   |   |   |   |   BLWhnARel <= 0.2
|   |   |   |   |   |   |   |   Area <= 8252.64
|   |   |   |   |   |   |   |   |   BLWhnARel <= 0.09
|   |   |   |   |   |   |   |   |   |   L50LWhn <= 0.34
|   |   |   |   |   |   |   |   |   |   |   BComp <= 1.25: TWhn (36.0/23.0)
|   |   |   |   |   |   |   |   |   |   |   BComp > 1.25: None (33.0/19.0)
|   |   |   |   |   |   |   |   |   |   L50LWhn > 0.34: TWhn (26.0/8.0)
|   |   |   |   |   |   |   |   |   BLWhnARel > 0.09: TWhn (34.0/5.0)
|   |   |   |   |   |   |   |   Area > 8252.64: TGan (23.0/15.0)
|   |   |   |   |   |   |   BLWhnARel > 0.2: TWhn (1026.0/72.0)
|   |   |   |   |   |   BLVerARel > 0.23: None (27.0/14.0)
|   |   |   |   |   BLOefARel > 0.54
|   |   |   |   |   |   Objektart = Ackerland: TWhn (0.0)
|   |   |   |   |   |   Objektart = Gartenland: TWhn (2.0)
|   |   |   |   |   |   Objektart = Gruenanlage: TWhn (9.0/5.0)
|   |   |   |   |   |   Objektart = Gruenland: TWhn (0.0)
|   |   |   |   |   |   Objektart = GuFGewerbeundIndustrie: TInG (1.0)
|   |   |   |   |   |   Objektart = GuFHandelDienstleistungen: TWhn (8.0/3.0)
|   |   |   |   |   |   Objektart = GuFMischnutzungMitWohnen: TWhn (1.0)
|   |   |   |   |   |   Objektart = GuFOeffentlicheZwecke: TBFP (38.0/13.0)
|   |   |   |   |   |   Objektart = GuFVersorgungsanlagen: TBFP (4.0)
|   |   |   |   |   |   Objektart = GuFWohnen: TWhn (34.0/9.0)
|   |   |   |   BLGlaARel > 0.14
|   |   |   |   |   BLGlaARel <= 0.72
|   |   |   |   |   |   BLGarARel <= 0.02: TMis (43.0/16.0)
|   |   |   |   |   |   BLGarARel > 0.02: TWhn (22.0/8.0)
|   |   |   |   |   BLGlaARel > 0.72: TGla (27.0/10.0)
|   |   |   BLGarARel > 0.26
|   |   |   |   BElong <= 2.75: TBFP (27.0/13.0)
|   |   |   |   BElong > 2.75: TGar (20.0/7.0)
|   |   L50LHaD > 0.24: TMis (60.0/17.0)
|   BLInGARel > 0.14
|   |   BLInGARel <= 0.79
|   |   |   BLOefARel <= 0.06
|   |   |   |   BLHaDARel <= 0.2
|   |   |   |   |   BLWhnARel <= 0.26: TInG (54.0/14.0)
|   |   |   |   |   BLWhnARel > 0.26
|   |   |   |   |   |   L50LMis <= 0.02
|   |   |   |   |   |   |   BArea <= 27630.45: TWhn (52.0/22.0)
|   |   |   |   |   |   |   BArea > 27630.45: TInG (20.0/5.0)
|   |   |   |   |   |   L50LMis > 0.02: TWhn (22.0)
|   |   |   |   BLHaDARel > 0.2
|   |   |   |   |   BArea <= 24982.21: TMis (22.0/4.0)
|   |   |   |   |   BArea > 24982.21: TInG (21.0/1.0)
|   |   |   BLOefARel > 0.06: TMis (27.0/10.0)
|   |   BLInGARel > 0.79
|   |   |   BLGanARel <= 0.01: TInG (32.0/15.0)
|   |   |   BLGanARel > 0.01: TBFP (22.0/3.0)
BLAckARel > 0.17
|   Objektart = Ackerland: TAck (105.0/9.0)
|   Objektart = Gartenland: TBFP (23.0/17.0)
|   Objektart = Gruenanlage: TBFP (37.0/23.0)
|   Objektart = Gruenland: TGla (35.0/12.0)
|   Objektart = GuFGewerbeundIndustrie: TBFP (4.0/1.0)
|   Objektart = GuFHandelDienstleistungen: TWhn (5.0/3.0)
|   Objektart = GuFMischnutzungMitWohnen: TAck (0.0)
|   Objektart = GuFOeffentlicheZwecke: TBFP (11.0/4.0)
|   Objektart = GuFVersorgungsanlagen: TAck (4.0/1.0)
|   Objektart = GuFWohnen: TWhn (26.0/10.0)
```

Programm A.1: Entscheidungsbaum für alle Objektarten der ALK des Gebiets Hildesheim-West

Objektart	TBFP	TGan	TGar	TGla	TInG	TMis	TWhn	None
LGan	9	25	8	2	1	24	23	8
LGar	0	0	0	0	33	44	22	0
LGla	0	0	0	100	0	0	0	0
LHaD	6	0	2	0	3	63	23	2
LInG	3	1	0	0	18	56	21	1
LMis	1	0	0	0	0	43	55	0
LOef	44	0	0	0	1	30	22	2
LVer	9	7	0	0	13	33	27	11
LWhn	0	1	0	0	1	42	55	0

Objektart	TOrt
LGan	98
LGar	89
LGla	0
LHaD	100
LInG	100
LMis	100
LOef	100
LVer	98
LWhn	100

Tabelle A.7: Zusammenhang zwischen den Objekten der ALK und ATKIS für das Gebiet Hannover

Abbildungsverzeichnis

Tabellenverzeichnis

Abkürzungsverzeichnis

9IM . . . . 9 Intersection Method

ALK . . . Automatisierte Liegenschaftskarte
ATKIS . . . Amtliches Topographisch-Kartographisches Informationssystem

CAD . . . Computer Aided Design
CBM . . . Calculus Based Model
CPU . . . Central Processing Unit

DDL . . . Data Definition Language
DE-9IM . . Dimensionally Extended 9 Intersection Method

EDA . . . Explorative Datenanalyse
EPSG . . . European Petroleum Survey Group
ESRI . . . Environmental Systems Research Institute

FN . . . . Falsch negativ
FP . . . . Falsch positiv

GFS . . . Google File System
GIS . . . . Geoinformationssystem
GKD . . . Geographic Knowledge Discovery
GML . . . Geography Markup Language
GPS . . . Global Positioning System
GPU . . . Graphic Processing Unit
GR . . . . Avro GenericRecord
GUF . . . Gebäude- und Freifläche

HDFS . . . Hadoop Distributed File System
HTTP . . . Hypertext Transfer Protocol

ISO . . . . International Organization for Standardization

JSON . . . JavaScript Object Notation
JTS . . . . Java Topology Suite

KDD . . . Knowledge Discovery from Data

MDA . . . Model Driven Architecture
MER . . . Minimum Enclosing Rectangle

OCL . . . Object Constraint Language
OGC . . . Open Geospatial Consortium

OMG . . . Object Management Group
OSM . . . OpenStreetMap

PIM . . . . Platform Independent Model
PSM . . . Platform Specific Model

RCC . . . Region Connected Calculus
RN . . . . Richtig negativ
RP . . . . Richtig positiv

SQL . . . . Structured Query Language

UML . . . Unified Modeling Language

WGS . . . World Geodetic System
WKB . . . Well Known Binary
WKT . . . Well Known Text

XML . . . Extensible Markup Language

Literaturverzeichnis

AdV, 2004. ATKIS-Objektartenkatalog. Arbeitsgemeinschaft der Vermessungsverwaltungen der Länder der Bundesrepublik Deutschland.

AGENT, 1998. Constraint Analysis. Technischer Bericht, Dept. of Geography, University of Zurich, Switzerland.

AGENT, 1999. Selection of Basic Measures. Technischer Bericht, Dept. of Geography, University of Zurich, Switzerland.

Aji, A., Wang, F., Saltz, J., 2012. Towards Building a High Performance Spatial Query System for Large Scale Medical Imaging Data. In: *Proceedings of the 20th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*. ACM, New York, NY, USA.

Allen, J. F., 1983. Maintaining Knowledge about Temporal Intervals. *Commun. ACM* 26 (11), S. 832–843.

Armbrust, M., Fox, A., Griffith, R., Joseph, A. D., Katz, R., Konwinski, A., Lee, G., Patterson, D., Rabkin, A., Stoica, I., Zaharia, M., 2010. A View of Cloud Computing. *Commun. ACM* 53 (4), S. 50–58.

Avro, 2012. Apache Avro 1.7.4 Documentation. URL http://avro.apache.org/docs/current/

Bartelme, N., 2005. Geoinformatik: Modelle, Strukturen, Funktionen, 4. Auflage. Springer, Berlin Heidelberg.

Bédard, Y., Larrivée, S., Proulx, M.-J., Nadeau, M., 2004. Modeling Geospatial Databases with Plug-Ins for Visual Languages: A Pragmatic Approach and the Impacts of 16 Years of Research and Experimentations on Perceptory. In: Wang, S., Yang, D., Tanaka, K., Grandi, F., Zhou, S., Mangina, E. E., Ling, T. W., Song, I.-Y., Guan, J., Mayr, H. C. (Hrsg.), *Conceptual Modeling for Advanced Application Domains*. Bd. 3289 von Lecture Notes in Computer Science. Springer, Berlin Heidelberg, S. 17–30.

Bejaoui, L., 2009. Qualitative Topological Relationships for Objects with Possibly Vague Shapes - Implications on the Specification of Topological Integrity Constraints in Transactional Spatial Databases and in Spatial Data Warehouses. Dissertation, Faculté des Études Supérieures de l'Université Laval, Québec, Canada.

Bejaoui, L., Pinet, F., Schneider, M., Bédard, Y., 2010. OCL for Formal Modelling of Topological Constraints Involving Regions with Broad Boundaries. *GeoInformatica* 14, S. 353–378.

Belussi, A., Negri, M., Pelagatti, G., 2000. An Integrity Constraints Driven System for Updating Spatial Databases. In: *Proceedings of the 8th ACM International Symposium on Advances in Geographic Information Systems*. ACM, New York, NY, USA, S. 121–128.

Belussi, A., Negri, M., Pelagatti, G., 2006. An ISO TC 211 Conformant Approach to Model Spatial Integrity Constraints in the Conceptual Design of Geographical Databases. In: Roddick, J., Benjamins, V., Si-said Cherfi, S., Chiang, R., Claramunt, C., Elmasri, R., Grandi, F., Han, H., Hepp, M., Lytras, M., Mišic, V., Poels, G., Song, I.-Y., Trujillo, J., Vangenot, C. (Hrsg.), *Advances in Conceptual Modeling – Theory and Practice*. Bd. 4231 von Lecture Notes in Computer Science. Springer, Berlin Heidelberg, S. 100–109.

Borges, K. A. V., Davis, C. A., Laender, A. H. F., 2001. OMT-G: An Object-Oriented Data Model for Geographic Applications. *Geoinformatica* 5 (3), S. 221–260.

Borges, K. A. V., Davis, Jr., C. A., Laender, A. H. F., 2002. Integrity Constraints in Spatial Databases. In: Doorn, J. H., Rivero, L. C. (Hrsg.), *Database Integrity: Challenges and Solutions*. IGI Publishing, Hershey, PA, USA, S. 144–171.

Borges, K. A. V., Laender, A. H. F., Davis, Jr., C. A., 1999. Spatial Data Integrity Constraints in Object Oriented Geographic Data Modeling. In: *Proceedings of the 7th ACM International Symposium on Advances in Geographic Information Systems*. ACM, New York, NY, USA, S. 1–6.

Bourier, G., 2011. Wahrscheinlichkeitsrechnung und schließende Statistik, 7. Auflage. Gabler, Wiesbaden.

Bravo, L., Rodríguez, M. A., 2012. Formalization and Reasoning about Spatial Semantic Integrity Constraints. *Data & Knowledge Engineering* 72, S. 63–82.

Breiman, L., Friedman, J. H., Olshen, R. A., Stone, C. J., 1984. CART: Classification and Regression Trees. Wadsworth, Belmont, CA, USA.

Brenner, C., 2005. Constraints for Modelling Complex Objects. In: Stilla, U., Rottensteiner, F., Hinz, S. (Hrsg.), *Proceedings of the ISPRS Workshop CMRT 2005, Vienna, Austria*. Bd. XXXVI von IAPRS. S. 49–54.

Cary, A., Sun, Z., Hristidis, V., Rishe, N., 2009. Experiences on Processing Spatial Data with MapReduce. In: Winslett, M. (Hrsg.), *Scientific and Statistical Database Management*. Bd. 5566 von Lecture Notes in Computer Science. Springer, Berlin Heidelberg, S. 302–319.

Casanova, M., Wallet, T., D'Hondt, M., 2000. Ensuring Quality of Geographic Data with UML and OCL. In: Evans, A., Kent, S., Selic, B. (Hrsg.), *UML 2000 - The Unified Modeling Language*. Bd. 1939 von Lecture Notes in Computer Science. Springer, Berlin Heidelberg, S. 225–239.

Chapman, P., Clinton, J., Kerber, R., Khabaza, T., Reinartz, T., Shearer, C., Wirth, R., 2000. CRISP-DM 1.0. NCR, SPSS, DaimlerChrysler. URL ftp://ftp.software.ibm.com/software/analytics/spss/support/Modeler/Documentation/14/UserManual/CRISP-DM.pdf

Chicago, 2013. Building Footprints – City of Chicago – Data Portal. URL https://data.cityofchicago.org/Buildings/Building-Footprints-deprecated-January-2013-/w2v3-isjw?

Christensen, J. V., 2007. Specifying Geographic Information - Ontology, Knowledge Representation, and Formal Constraints. Dissertation, Informatics and Mathematical Modelling, Technical University of Denmark.

Claramunt, C., 2000. Extending Ladkin's Algebra on Non-Convex Intervals towards an Algebra on Union-of Regions. In: *Proceedings of the 8th ACM International Symposium on Advances in Geographic Information Systems*. ACM, New York, NY, USA, S. 9–14.

Clementini, E., Felice, P. D., Oosterom, P. v., 1993. A Small Set of Formal Topological Relationships Suitable for End-User Interaction. In: *Proceedings of the 3rd International Symposium on Advances in Spatial Databases*. Springer, London, UK, S. 277–295.

Cockcroft, S., 1997. A Taxonomy of Spatial Data Integrity Constraints. *Geoinformatica* 1 (4), S. 327–343.

Cockcroft, S., 2001. Modelling Spatial Data Integrity Rules at the Metadata Level. In: *Proceedings of the 6th International Conference on GeoComputation*. Brisbane, Australia.

Cockcroft, S., 2004. The Design and Implementation of a Repository for the Management of Spatial Data Integrity Constraints. *Geoinformatica* 8 (1), S. 49–69.

Cohn, A., Gotts, N., 1996. Geographical Objects with Undetermined Boundaries. Bd. 2 von GISDATA. Francis Taylor, London, UK, Kap. The 'Egg-Yolk' Representation of Regions with Indeterminate Boundaries, S. 171–187.

Davis, M., 2013. JTS Topology Suite. URL http://tsusiatsoftware.net/jts/main.html

dBase, 2012. Data File Header Structure for the dBASE Version 7 Table File. URL http://www.dbase.com/Knowledgebase/INT/db7_file_fmt.htm

Dean, J., Ghemawat, S., 2004. MapReduce: Simplified Data Processing on Large Clusters. In: *Proceedings of the Sixth Symposium on Operating System Design and Implementation*. San Francisco, CA, USA.

Demuth, B., Hussmann, H., Loecher, S., 2001. OCL as a Specification Language for Business Rules in Data Base Applications. In: Gogolla, M., Kobryn, C. (Hrsg.), *UML 2001 – The Unified Modeling Language*. Bd. 2185 von Lecture Notes in Computer Science. Springer, Berlin Heidelberg, S. 104–117.

DESTATIS, 2012. Länder & Regionen – Gemeindeverzeichnis – Bundesländer mit Hauptstädten nach Fläche und Bevölkerung am 31.12.2011. Statistisches Bundesamt. URL https://www.destatis.de/DE/ZahlenFakten/LaenderRegionen/Regionales/Gemeindeverzeichnis/Administrativ/Aktuell/02Bundeslaender.html

Ditt, H., Becker, L., Voigtmann, A., Winrichs, K. H., 1997. Constraints and Triggers in an Object-oriented Geo Database Kernel. In: *Proceedings of the 8th International Workshop on Database and Expert Systems Applications*. IEEE Computer Society, Washington, DC, USA, S. 508–513.

DresdenOCL, 2013. DresdenOCL:Documentation. URL http://www.dresden-ocl.org/index.php/DresdenOCL:Documentation

Duboisset, M., Pinet, F., Kang, M.-A., Schneider, M., 2005a. Integrating the Calculus-Based Method into OCL: Study of Expressiveness and Code Generation. In: *Proceedings of the 16th International Workshop on Database and Expert Systems Applications*. IEEE Computer Society, Washington, DC, USA, S. 502–506.

Duboisset, M., Pinet, F., Kang, M.-A., Schneider, M., 2005b. Precise Modeling and Verification of Topological Integrity Constraints in Spatial Databases: From an Expressive Power Study to Code Generation Principles. In: Delcambre, L., Kop, C., Mayr, H. C., Mylopoulos, J., Pastor, O. (Hrsg.), *Conceptual Modeling – ER 2005*. Bd. 3716 von Lecture Notes in Computer Science. Springer, Berlin Heidelberg, S. 465–482.

Duden, 2011. Deutsches Universalwörterbuch: Das Umfassende Bedeutungswörterbuch der Deutschen Gegenwartssprache, 7. Auflage. Bibliographisches Institut, Mannheim.

Egenhofer, M. J., 1989. A Formal Definition of Binary Topological Relationships. In: Litwin, W., Schek, H.-J. (Hrsg.), *Foundations of Data Organization and Algorithms*. Bd. 367 von Lecture Notes in Computer Science. Springer, S. 457–472.

Egenhofer, M. J., Al-Taha, K. K., 1992. Reasoning About Gradual Changes of Topological Relationships. In: Frank, A. U., Campari, I., Formentini, U. (Hrsg.), *Theories and Methods of Spatio-Temporal Reasoning in Geographic Space*. Bd. 639 von Lecture Notes in Computer Science. Springer, Berlin Heidelberg, S. 196–219.

Egenhofer, M. J., Herring, J. R., 1990. A Mathematical Framework for the Definition of Topological Relationships. In: Brazzel, K., Kishimoto, H. (Hrsg.), *Proceedings of the 4th International Symposium on Spatial Data Handling*. Department of Geography, University of Zurich, Switzerland, S. 803–813.

Egenhofer, M. J., Herring, J. R., 1991. Categorizing Binary Topological Relationships Between Regions, Lines, and Points in Geographic Databases. Technischer Bericht, Department of Surveying Engineering, University of Maine.

Elmasri, R., Navathe, S. B., 2011. Fundamentals of Database Systems, 6. Auflage. Addison Wesley, Boston, MA, USA.

ESRI, 1998. ESRI Shapefile Technical Description. Environmental Systems Research Institute. URL http://www.esri.com/library/whitepapers/pdfs/shapefile.pdf

ESRI, 2009. ArcGIS Desktop Help 9.3 – Geoprocessing Considerations for Shapefile Output. URL http://webhelp.esri.com/arcgisdesktop/9.3/index.cfm?TopicName=Geoprocessing%20considerations%20for%20shapefile%20output

Facebook, 2012. Under the Hood: Hadoop Distributed Filesystem Reliability with Namenode and Avatarnode. URL http://www.facebook.com/9445547199/posts/10150888759153920

Fahrner, C., Marx, T., Philippi, S., 1997. DICE: Declarative Integrity Constraint Embedding into the Object Database Standard ODMG-93. *Data & Knowledge Engineering* 23 (2), S. 119–145.

Fayyad, U., Piatetsky-Shapiro, G., Smyth, P., 1996. The KDD Process for Extracting Useful Knowledge from Volumes of Data. *Commun. ACM* 39 (11), S. 27–34.

Feibel, B. J., 2003. Investment Performance Measurement. Wiley & Sons, Hoboken, NJ, USA.

Finkel, R., Bentley, J., 1974. Quad Trees A Data Structure for Retrieval on Composite Keys. *Acta Informatica* 4 (1), S. 1–9.

Foster, I., 1995. Designing and Building Parallel Programs: Concepts and Tools for Parallel Software Engineering. Addison Wesley, Boston, MA, USA.

Foster, I., 2002. What is the Grid? A Three Point Checklist. *GridToday* 1 (6).

Friis-Christensen, A., Tryfona, N., Jensen, C. S., 2001. Requirements and Research Issues in Geographic Data Modeling. In: *Proceedings of the 9th ACM International Symposium on Advances in Geographic Information Systems*. ACM, New York, NY, USA, S. 2–8.

GeoTools, 2013. About GeoTools. URL http://geotools.org/about.html

Ghemawat, S., Gobioff, H., Leung, S.-T., 2003. The Google File System. In: *Proceedings of the 19th ACM Symposium on Operating Systems Principles*. ACM, New York, NY, USA.

Gogolla, M., Büttner, F., Richters, M., 2007. USE: A UML-Based Specification Environment for Validating UML and OCL. *Science of Computer Programming* 69 (1), S. 27–34.

Guo, D., Mennis, J., 2009. Spatial Data Mining and Geographic Knowledge Discovery – An Introduction. *Computers, Environment and Urban Systems* 33 (6), S. 403–408.

Guttman, A., 1984. R-trees: A Dynamic Index Structure for Spatial Searching. In: *Proceedings of the 1984 ACM SIGMOD International Conference on Management of Data*. ACM, Boston, MA, USA, S. 47–57.

Hadoop, 2012. PoweredBy – Hadoop Wiki. URL http://wiki.apache.org/hadoop/PoweredBy

Hadzilacos, T., Tryfona, N., 1992. A Model for Expressing Topological Integrity Constraints in Geographic Databases. In: Frank, A., Campari, I., Formentini, U. (Hrsg.), *Theories and Methods of Spatio-Temporal Reasoning in Geographic Space*. Bd. 639 von Lecture Notes in Computer Science. Springer, Berlin Heidelberg, S. 252–268.

Hall, M., Eibe, F., Holmes, G., Pfahringer, B., Reutemann, P., Witten, I. H., 2009. The WEKA Data Mining Software: An Update. *SIGKDD Explorations* 11 (1), S. 10–18.

Han, J., Kamber, M., Pei, J., 2011. Data Mining: Concepts and Techniques, 3. Auflage. Morgan Kaufmann Publishers, Waltham, MA, USA.

Hartung, J., Elpelt, B., Klösener, K.-H., 2002. Statistik: Lehr- und Handbuch der angewandten Statistik, 13. Auflage. Oldenbourg, München.

Hilbert, D., 1891. Über die stetige Abbildung einer Linie auf ein Flächenstück. *Mathematische Annalen* 38, S. 459–460.

Hitz, M., Kappel, G., Kapsammer, E., Retschitzegger, W., 2005. UML @ Work, 3. Auflage. dpunkt, Heidelberg.

Hoaglin, D. C., Mosteller, F., Tukey, J. W. (Hrsg.), 2000. Understanding Robust and Exploratory Data Analysis. John Wiley & Sons, New York, NY, USA.

Holte, R. C., 1993. Very Simple Classification Rules Perform Well on Most Commonly Used Datasets. *Machine Learning* 11 (1), S. 63–90.

Hornsby, K., Egenhofer, M. J., Hayes, P. J., 1999. Modeling Cyclic Change. In: *Proceedings of the Workshops on Evolution and Change in Data Management, Reverse Engineering in Information Systems, and the World Wide Web and Conceptual Modeling*. Springer, London, UK, S. 98–109.

ISO 19101, 2002. ISO 19101:2002: Geographic Information – Reference Model. International Organization for Standardization.

ISO 19107, 2003. ISO 19107:2003: Geographic Information – Spatial Schema. International Organization for Standardization.

ISO 19108, 2002. ISO 19108:2002: Geographic Information – Temporal Schema. International Organization for Standardization.

ISO 19113, 2002. ISO 19113:2002: Geographic Information – Quality Principles. International Organization for Standardization.

ISO 19114, 2003. ISO 19114:2003: Geographic Information – Quality Evaluation Procedures. International Organization for Standardization.

ISO 19507, 2012. ISO 19507:2012: Information Technology – Object Management Group Object Constraint Language (OCL). International Organization for Standardization. URL http://www.omg.org/spec/OCL/2.3.1/

Jarque, C. M., Bera, A. K., 1987. A Test for Normality of Observations and Regression Residuals. *International Statistical Review* 55 (2), S. 163–172.

JSON, 2012. JSON. URL http://www.json.org/

Kang, M.-A., Pinet, F., Schneider, M., Chanet, J.-P., Vigier, F., 2004. How to Design Geographic Databases? Specific UML Profile and Spatial OCL Applied to Wireless Ad Hoc Networks. In: *Proceedings of the 7th AGILE Conference on Geographic Information Science*. Heraklion, Greece.

Kwon, Y., Balazinska, M., Howe, B., Rolia, J., 2012. SkewTune: Mitigating Skew in MapReduce Applications. In: *Proceedings of the 2012 SIGMOD International Conference on Management of Data*. ACM, New York, NY, USA, S. 25–36.

Liao, H., Han, J., Fang, J., 2010. Multi-dimensional Index on Hadoop Distributed File System. In: *Proceedings of the 5th International Conference on Networking, Architecture and Storage*. IEEE Computer Society, Washington, DC, USA, S. 240–249.

Lämmel, R., 2008. Google's MapReduce Programming Model – Revisited. *Science of Computer Programming* 70 (1), S. 1–30.

Louwsma, J., 2004. Constraints in Geo-information Models: Applied to Geo-VR in Landscape Architecture. Master Thesis, TU Delft, Netherlands.

Louwsma, J., Zlatanova, S., van Lammeren, R., van Oosterom, P., 2006. Specifying and Implementing Constraints in GIS – with Examples from a Geo-Virtual Reality System. *Geoinformatica* 10 (4), S. 531–550.

Miller, H. J., 2008. The Handbook of Geographic Information Science. Blackwell Publishing Ltd, Malden, MA, USA, Kap. Geographic Data Mining and Knowledge Discovery, S. 352–366.

Mondzech, J., 2009. Formale Beschreibung und Web-Service-basierte Überprüfung von Constraints in Geodaten. Diplomarbeit, Institut für Kartographie und Geoinformatik, Leibniz Universität Hannover.

Morton, G., 1966. A Computer Oriented Geodetic Data Base; and a New Technique in File Sequencing. Technischer Bericht, IBM, Ottawa, Canada.

Mäs, S., 2010. On the Consistency of Spatial Semantic Integrity Constraints. Dissertation, University of the Bundeswehr Munich.

Mäs, S., Reinhardt, W., 2009. Categories of Geospatial and Temporal Integrity Constraints. In: *Proceedings of the International Conference on Advanced Geographic Information Systems and Web Services*. IEEE Computer Society, Washington, DC, USA, S. 146–151.

Mäs, S., Wang, F., Reinhardt, W., 2005. Using Ontologies for Integrity Constraint Definition. In: Wu, L., Shi, W., Fang, Y., Tong, Q. (Hrsg.), *Proceedings of the 4th International Symposium On Spatial Data Quality*. Hong Kong Polytechnic University, Hong Kong, China, S. 304–313.

NVKV, 2003. Verwaltungsvorschrift zur Führung der Automatisierten Liegenschaftskarte in der ALK/ATKIS-Datenbank. Niedersächsische Vermessungs- und Katasterverwaltung.

OGC CityGML, 2012. City Geography Markup Language (CityGML) Encoding Standard. Open Geospatial Consortium. URL http://www.opengeospatial.org/standards/citygml

OGC Simple Feature, 2011. OpenGIS Implementation Standard for Geographic Information – Simple Feature Access – Part 1: Common Architecture. Open Geospatial Consortium. URL http://www.opengeospatial.org/standards/sfa

OMG, 2003. MDA Guide Version 1.0.1. Object Management Group. URL http://www.omg.org/cgi-bin/doc?omg/03-06-01

OMG, 2011a. OMG MOF 2 XMI Mapping Specification. Object Management Group. URL http://www.omg.org/spec/XMI/2.4.1/

OMG, 2011b. OMG Unified Modeling Language (OMG UML), Infrastructure. Object Management Group. URL http://www.omg.org/spec/UML/2.4.1/

OMG, 2011c. OMG Unified Modeling Language (OMG UML), Superstructure. Object Management Group. URL http://www.omg.org/spec/UML/2.4.1/

OMG, 2012. OMG Object Constraint Language (OCL). Object Management Group. URL http://www.omg.org/spec/OCL/2.3.1/

Omniscale, 2013. Imposm 2.5.0 Documentation. URL http://imposm.org/docs/imposm/latest/

OpenStreetMap, 2013a. DE:Map Features – OpenStreetMap Wiki. URL http://wiki.openstreetmap.org/wiki/DE:Map_Features

OpenStreetMap, 2013b. Key:Building – OpenStreetMap Wiki. URL http://wiki.openstreetmap.org/wiki/Key:building

OpenStreetMap, 2013c. PBF Format – OpenStreetMap Wiki. URL http://wiki.openstreetmap.org/wiki/PBF_Format

OpenStreetMap, 2013d. WikiProject France/Cadastre – OpenStreetMap Wiki. URL http://wiki.openstreetmap.org/wiki/WikiProject_France/Cadastre

Pelagatti, G., Negri, M., Belussi, A., Migliorini, S., 2009. From the Conceptual Design of Spatial Constraints to their Implementation in Real Systems. In: *Proceedings of the 17th SIGSPATIAL International Conference on Advances in Geographic Information Systems*. GIS '09. ACM, New York, NY, USA, S. 448–451.

Pinet, F., 2010. Modélisation des Contraintes d'Intégrité dans les Systèmes d'Information Environnementaux. Habilitation, Université Blaise Pascal, Clermont-Ferrand, France.

Pinet, F., 2012. Entity-Relationship and Object-Oriented Formalisms for Modeling Spatial Environmental Data. *Environmental Modelling & Software* 33, S. 80–91.

Pinet, F., Kang, M.-A., Vigier, F., 2005. Spatial Constraint Modelling with a GIS Extension of UML and OCL: Application to Agricultural Information Systemss. In: Wiil, U. (Hrsg.), *Metainformatics*. Bd. 3511 von Lecture Notes in Computer Science. Springer, S. 43–54.

Polasek, W., 1994. EDA Explorative Datenanalyse: Einführung in die Deskriptive Statistik, 2. Auflage. Springer, Berlin Heidelberg.

PostGIS, 2013. PostGIS – Spatial and Geographic Objects for PostgreSQL. URL http://postgis.net/

PostgreSQL, 2013. PostgreSQL: About. URL http://www.postgresql.org/about/

Protobuf, 2012. Protocol Buffers – Google's Data Interchange Format – Google Project Hosting. URL https://code.google.com/p/protobuf/

Quinlan, J. R., 1986. Induction of Decision Trees. *Machine Learning* 1 (1), S. 81–106.

Quinlan, J. R., 1993. C4.5: Programs for Machine Learning. Morgan Kaufmann, Waltham, MA, USA.

R, 2013. The R Project for Statistical Computing. URL http://www.r-project.org/

Randell, D. A., Cui, Z., Cohn, A. G., 1992. A Spatial Logic based on Regions and Connection. In: Nebel, B., Rich, C., Swartout, W. R. (Hrsg.), *Proceedings of the 3rd International Conference on Principles of Knowledge Representation and Reasoning*. Morgan Kaufmann, S. 165–176.

Rauber, T., Rünger, G., 2012. Parallele Programmierung. Springer, Berlin Heidelberg.

Reeves, T., Cornford, D., Konecny, M., Ellis, J., 2006. Modeling Geometric Rules in Object Based Models: An XML / GML Approach. In: Riedl, A., Kainz, W., Elmes, G. A. (Hrsg.), *Progress in Spatial Data Handling*. Springer, Berlin Heidelberg, S. 133–148.

Rinne, H., 2003. Taschenbuch der Statistik, 3. Auflage. Harri Deutsch, Frankfurt am Main.

Rodríguez, M. A., 2005. Inconsistency Issues in Spatial Databases. In: Bertossi, L., Hunter, A., Schaub, T. (Hrsg.), *Inconsistency Tolerance*. Bd. 3300 von Lecture Notes in Computer Science. Springer, Berlin Heidelberg, S. 237–269.

Rodríguez, M. A., Bertossi, L., Caniupán, M., 2008. An Inconsistency Tolerant Approach to Querying Spatial Databases. In: *Proceedings of the 16th SIGSPATIAL International Conference on Advances in Geographic Information Systems*. ACM, New York, NY, USA, S. 36:1–36:10.

Rodríguez, M. A., Brisaboa, N., Meza, J., Luaces, M. R., 2010. Measuring Consistency with Respect to Topological Dependency Constraints. In: *Proceedings of the 18th SIGSPATIAL International Conference on Advances in Geographic Information Systems*. ACM, New York, NY, USA, S. 182–191.

Salehi, M., 2009. Developing a Model and a Language to Identify and Specify the Integrity Constraints in Spatial Datacubes. Dissertation, Foresterie et Géodésie, Université Laval.

Salehi, M., Bédard, Y., Mostafavi, M., Brodeur, J., 2007. From Transactional Spatial Databases Integrity Constraints to Spatial Data Cubes Integrity Constraints. In: Stein, A. (Hrsg.), *Proceedings of the 5th International Symposium on Spatial Data Quality*. Bd. XXXVI-2/C43 von ISPRS Archives. Enschede, The Netherlands.

Salehi, M., Bédard, Y., Mostafavi, M., Brodeur, J., 2011. Formal Classification of Integrity Constraints in Spatiotemporal Database Applications. *Visual Languages and Computing* 22, S. 323–339.

Servigne, S., Ubeda, T., Puricelli, A., Laurini, R., 2000. A Methodology for Spatial Consistency Improvement of Geographic Databases. *Geoinformatica* 4 (1), S. 7–34.

Shannon, C. E., 1948. A Mathematical Theory of Communication. *Bell System Technical Journal* 27, S. 379–423, 623–656.

Steiniger, S., 2007. Enabling Pattern-Aware Automated Map Generalization. Dissertation, Mathematisch-naturwissenschaftliche Fakultät, Universität Zürich, Schweiz.

Steiniger, S., Weibel, R., 2005. Relations and Structures in Categorical Maps. In: *8th ICA WORKSHOP on Generalisation and Multiple Representation, A Coruña, Spain*.

Stoter, J., Morales, J., Lemmens, R., Meijers, B., Oosterom, P., Quak, C., Uitermark, H., Brink, L., 2008. A Data Model for Multi-scale Topographical Data. In: Ruas, A., Gold, C. (Hrsg.), *Headway in Spatial Data Handling*. Lecture Notes in Geoinformation and Cartography. Springer, Berlin Heidelberg, S. 233–254.

Thrift, 2012. Apache Thrift. URL http://thrift.apache.org/

Tukey, J. W., 1977. Exploratory Data Analysis. Addison Wesley, Reading, MA, USA.

Ubeda, T., 1997. Contrôle de la Qualité Spatiale des Bases de Données Géographiques: Cohérence Topologique et Corrections d'Erreur. Dissertation, Institut National des Sciences Appliquées de Lyon, France.

Ubeda, T., Egenhofer, M. J., 1997. Topological Error Correcting in GIS. In: *Advances in Spatial Databases*. Bd. 1262 von Lecture Notes in Computer Science. Springer, S. 281–297.

van Bennekom-Minnema, J., 2008. The Land Administration Domain Model 'Survey Package' and Model Driven Architecture. Master Thesis, GIMA, Netherlands.

van Oosterrom, P., 2006. Constraints in Spatial Data Models, in a Dynamic Context. In: Drummond, J. (Hrsg.), *Dynamic and Mobile GIS: Investigating Changes in Space and Time*. CRC Press, Kap. 7, S. 104–137.

VKVRP, 2010. Richtlinien zur Führung des Liegenschaftskatasters mit den Systemen ALK und ALB. Vermessungs- und Katasterverwaltung Rheinland-Pfalz.

W3C, 2004. SWRL: A Semantic Web Rule Language Combining OWL and RuleML. World Wide Web Consortium. URL http://www.w3.org/Submission/SWRL/

Wang, F., Aji, A., Liu, Q., Saltz, J., 2011. Hadoop-GIS: A High Performance Query System for Analytical Medical Imaging with MapReduce. Technischer Bericht CCI-TR-2011-3, Center for Comprehensive Informatics, Emory University. URL http://confluence.cci.emory.edu:8090/display/HadoopGIS/Publications

Wang, F., Reinhardt, W., 2007. Extending Geographic Data Modeling by Adopting Constraint Decision Table to Specify Spatial Integrity Constraints. In: Fabrikant, S. I., Wachowicz, M. (Hrsg.), *The European Information Society: Leading the Way with Geo-Information*. Lecture Notes in Geoinformation and Cartography. Springer, Berlin Heidelberg, S. 435–454.

Wang, K., Han, J., Tu, B., Dai, J., Zhou, W., Song, X., 2010. Accelerating Spatial Data Processing with MapReduce. In: *Proceedings of the 16th International Conference on Parallel and Distributed Systems*. IEEE Computer Society, Washington, DC, USA, S. 229–236.

Warmer, J., Kleppe, A., 2004. Object Constraint Language 2.0. mitp, Bonn.

Watson, P., 2007. OWS4 – Topology Quality Assessment Interoperability Program Report. Discussion Paper, Open Geospatial Consortium. URL http://www.opengeospatial.org/standards/dp

Watson, P., 2008. Formal Languages for Expressing Spatial Data Constraints and Implications for Reporting of Quality Metadata. In: *Quality Aspects in Spatial Data Mining*. CRC Press, S. 329–344.

Weka, 2013. Weka 3 – Data Mining with Open Source Machine Learning Software in Java. URL http://www.cs.waikato.ac.nz/ml/weka/

Werder, S., 2009. Formalization of Spatial Constraints. In: *Proceedings of the 12th AGILE International Conference on Geographic Information Science*. Institut für Kartographie und Geoinformatik, Leibniz Universität Hannover.

Werder, S., 2010. Data Integration in a Modular and Parallel Grid-Computing Workflow. In: *Proceedings of the 1st International Workshop on Pervasive Web Mapping, Geoprocessing and Services*. Como, Italy.

Werder, S., Kieler, B., Sester, M., 2010. Semi-Automatic Interpretation of Buildings and Settlement Areas in User-generated Spatial Data. In: *Proceedings of the 10th ACM International Symposium on Advances in Geographic Information Systems*. ACM, New York, NY, USA.

Werder, S., Krüger, A., 2009. Parallelizing Geospatial Tasks in Grid Computing. *GIS.Science* 3, S. 71–76.

White, T., 2012. Hadoop: The Definitive Guide, 3. Auflage. O'Reilly Media, Sebastopol, CA, USA.

Xu, D., 2011. Design and Implementation of Constraints for 3D Spatial Database – Using Climate City Campus Database as an Example. Master Thesis, TU Delft, Netherlands.

Yazici, B., Yolacan, S., 2007. A Comparison of Various Tests of Normality. *Journal of Statistical Computation and Simulation* 77 (2), S. 175–183.

Zhang, S., Han, J., Liu, Z., Wang, K., Xu, Z., 2009. SJMR: Parallelizing Spatial Join with MapReduce on Clusters. In: *Proceedings of the IEEE International Conference on Cluster Computing and Workshops*. IEEE Computer Society, Washington, DC, USA, S. 1–8.

Zhong, Y., Han, J., Zhang, T., Li, Z., Fang, J., Chen, G., 2012. Towards Parallel Spatial Query Processing for Big Spatial Data. In: *Proceedings of the 26th IEEE International Conference on Parallel and Distributed Processing Symposium Workshops PhD Forum*. IEEE Computer Society, Washington, DC, USA, S. 2085–2094.

Lebenslauf

Persönliche Daten

Name	Stefan Werder
Geboren am	15.12.1979 in Wangen im Allgäu
Familienstand	Verheiratet, 2 Kinder

Ausbildung

1986 – 1990	Grundschule Deuchelried
1990 – 1999	Rupert-Ness-Gymnasium Wangen
25.06.1999	Allgemeine Hochschulreife
1999 – 2000	Zivildienst bei den Fachkliniken Wangen
10/2000 – 12/2005	Diplomstudiengang Geodäsie und Geoinformatik an der Universität Karlsruhe (TH)
20.12.2005	Diplom-Ingenieur Geodäsie und Geoinformatik

Berufserfahrung

01/2006 – 12/2007	Wissenschaftlicher Mitarbeiter am Institut für Photogrammetrie und Fernerkundung der Universität Karlsruhe (TH)
01/2008 – 01/2013	Wissenschaftlicher Mitarbeiter am Institut für Kartographie und Geoinformatik der Leibniz Universität Hannover
04/2013 –	Senior Engineer bei der Advanced Driver Information Technology GmbH in Hildesheim