Lehrstuhl für Bauinformatik
Fakultät für Bauingenieur- und Vermessungswesen
Technische Universität München

Computerunterstützung verteilt-kooperativer Bauplanung durch Integration interaktiver Simulationen und räumlicher Datenbanken

André Borrmann

Vollständiger Abdruck der von der Fakultät für Bauingenieur- und Vermessungswesen der Technischen Universität München zur Erlangung des akademischen Grades eines

Doktor-Ingenieurs

genehmigten Dissertation.

Vorsitzender:		Univ.-Prof. Dr.-Ing. M. Schilcher
Prüfer der Dissertation:	1.	Univ.-Prof. Dr. rer. nat. E. Rank
	2.	Univ.-Prof. Dr.-Ing. habil. R. Hübler, Bauhaus-Universität Weimar

Die Dissertation wurde am 17.04.2007 bei der Technischen Universität München eingereicht und durch die Fakultät für Bauingenieur- und Vermessungswesen am 22.06.2007 angenommen.

5., leicht überarbeitete Fassung

April 2008

Zusammenfassung

Die vorliegende Arbeit befasst sich mit der Computerunterstützung verteilt-kooperativer Planungsprozesse im Bauwesen. Dabei werden Lösungsansätze für die synchrone und die asynchrone Form der kooperativen Arbeit präsentiert, die der besonderen Bedeutung von Bauteilgeometrie und -topologie für Planungsentscheidungen gerecht werden. Das im ersten Teil vorgestellte System zur Unterstützung synchroner Zusammenarbeit auf Basis eines 3D-Mehrbenutzereditors erlaubt die dynamische Einbeziehung interaktiver Simulationen in eine kooperative Sitzung. Während dabei die Geometrie den generischen Kern des gemeinsam genutzten Modells bildet, werden die Parameter der dynamisch koppelbaren Simulationsserver mithilfe eines Metamodells modelliert und im verteilten System propagiert. Im zweiten Teil der Arbeit wird eine räumliche Anfragesprache vorgestellt, die die Dekomposition eines Bauwerkmodells anhand geometrisch-topologischer Kriterien erlaubt und damit eine Voraussetzung für konfliktfreies paralleles Arbeiten schafft. Hierfür werden metrische, direktionale und topologische Operatoren auf Grundlage von Punktmengentheorie und Punktmengentopologie formal definiert, eine oktalbaumbasierte Implementierung dieser Operatoren vorgestellt und mit objekt-relationalen Datenbanksystemen eine mögliche Implementierungsbasis für die Anfragesprache aufgezeigt.

Abstract

This thesis addresses computer support for collaborative planning processes in the construction industry. It presents approaches for the synchronous and the asynchronous form of collaborative engineering that cope with the particular significance of geometry and topology of building components for planning decisions. The software system for synchronous collaborative engineering presented in the first part allows for a dynamic integration of interactive simulations in a shared engineering session. While geometric information forms the generic core of the shared model, the steerable parameters of dynamically connectable simulation servers are modeled and announced using an explicitly available meta-model. The second part of the thesis presents a 3D spatial query language which enables the decomposition of digital building models on the basis of geometric and topological criteria and thus lays the foundation for reducing conflicts in parallel work. Metric, directional and topological operators of the language are formally defined using point-set theory and point-set topology and an octree-based implementation of these operators is presented. Finally, the advantages of using object-relational database systems as an implementation base for the spatial query language are discussed.

Vorwort

Diese Arbeit entstand von Dezember 2003 bis April 2007 im Rahmen eines von der Siemens AG gewährten Industriestipendiums während meiner Tätigkeit am Lehrstuhl für Bauinformatik der Technischen Universität München.

An dieser Stelle möchte ich mich bei all jenen bedanken, die zum Gelingen dieser Arbeit beigetragen haben. Mein erster Dank gilt Herrn Prof. Dr. rer. nat. Ernst Rank, der mir mit der Aufnahme in sein exzellentes Team die Chance zur intensiven Auseinandersetzung mit wissenschaftlichen Fragestellungen gegeben und für hervorragende Rahmenbedingungen während meiner Promotionszeit gesorgt hat. Sein Interesse an den Inhalten meiner wissenschaftlichen Arbeit waren mir immer ein großer Ansporn. Herausheben möchte ich vor allem seine enorme Konzentrationsfähigkeit, die trotz häufig knapper Zeit die Möglichkeit zu äußerst konstruktiven fachlichen Diskussionen eröffnete.

Herrn Prof. Dr.-Ing. habil. Reinhard Hübler möchte für die Übernahme des Zweitgutachtens, sein aufrichtiges Interesse an meiner Arbeit und die detaillierten inhaltlichen Hinweise danken. Bei der Siemens AG bedanke ich mich für die finanzielle Unterstützung meiner Arbeit und das damit entgegengebrachte Vertrauen.

Für die nahezu familiäre Atmosphäre und das freundschaftliche Miteinander am Lehrstuhl bedanke ich mich bei allen Kollegen, im Besonderen jedoch bei Frau Hanne Cornils, die mit ihrer Geduld und ihrem Charme einen wesentlichen Anteil am außerordentlich guten Arbeitsklima hat. Besonders zu schätzen weiß ich die zahlreichen fachlichen Diskussionen mit meinen Kollegen, aus denen viele gute Ideen für diese Arbeit hervorgegangen sind. In diesem Zusammenhang möchte vor allem Andreas Niggl für die vielen spannenden und anregenden Gespräche danken.

Meinen Eltern danke ich für ihre stetige Unterstützung während meines Studiums und der Promotionszeit, durch die sie mir immer die Sicherheit gegeben haben, auf dem richtigen Weg zu sein. Schon in meiner frühen Kindheit legten sie durch die geduldige Beantwortung fast aller meiner Fragen den Grundstein für meine fortwährende Begeisterung am wissenschaftlichen Arbeiten.

Ganz besonders danke ich Julie Rousset, die mir während der mitunter recht arbeitsreichen Promotionszeit zur Seite gestanden und mir Rückhalt gegeben, aber auch immer für willkommene Abwechslung gesorgt hat. Natürlich soll ihr wertvoller Rat bei der visuellen Gestaltung dieser Arbeit nicht unerwähnt bleiben.

Schließlich möchte ich mich ganz besonders bei Julie Metzdorf, Robert Schmidt und Ziad Wassouf bedanken, die diese Arbeit aufmerksam gelesen und viele Hinweise für die Korrektur gegeben haben.

Inhaltsverzeichnis

Kapitel 1

Einführung

Die vorliegende Arbeit befasst sich mit der Computerunterstützung verteilt-kooperativer Planungsprozesse im Bauwesen. Dabei werden Lösungsansätze für zwei Formen der kooperativen Arbeit vorgestellt: Ein System zur Unterstützung der *synchronen* Zusammenarbeit unter Einbeziehung interaktiver Simulationen und ein Ansatz zur Unterstützung der *asynchronen* Zusammenarbeit auf Basis einer räumlichen Datenbank. Dabei wird die These vertreten, dass der geometrischen Form von Bauteilen bzw. den geometrisch-topologischen Zusammenhängen zwischen verschiedenen Bauwerkskomponenten eine entscheidende Bedeutung für einen Großteil der Planungsentscheidungen und damit auch für die Zusammenarbeit der Fachplaner zukommt. Diese Sichtweise spiegelt sich in der geometrieorientierten Konzeption der zwei vorzustellenden Systeme wider.

1.1 Motivation

Die Bauplanung ist durch eine große Zahl stark spezialisierter Beteiligter aus unterschiedlichen Fachdisziplinen gekennzeichnet. Immer kürzer werdende Planungszeiträume erhöhen stetig die Zahl der gleichzeitig in einem Projekt involvierten Fachplaner.

Aus der hohen Komplexität des Planungsgegenstands *Bauwerk* ergibt sich dabei eine Vielzahl von Abhängigkeiten zwischen den Planungsleistungen der einzelnen Beteiligten. Diese zunehmende Verzahnung der Planungsarbeit führt zur Notwendigkeit eines hohen Maßes an Kooperation zwischen den Planenden. Eine wesentliche Grundlage hierfür bilden geeignete Formen von Informationsaustausch, deren Gegenstand das zu planende Bauwerk bzw. die zu erbringende Planungsleistung ist.

Die Unterstützung dieser Kooperation durch Softwarewerkzeuge kann einen wesentlichen Beitrag zur Effizienzsteigerung leisten. Zwar existiert bereits eine Reihe wissenschaftlicher Arbeiten zur Computerunterstützung verteilt-kooperativer Arbeit im Bauwesen. Gerade aber die für eine Unterstützung bei ingenieurstechnischen Planungsentscheidungen zwingend erforderliche Einbindung von Simu-

lationswerkzeugen in kooperative Umgebungen wurde bislang nicht eingehend erforscht. Des Weiteren stellte bisher die geometrieorientierte Analyse und Partitionierung von digitalen Bauwerksmodellen ein ungelöstes Problem dar. Diese Arbeit stellt Lösungsmöglichkeiten zu beiden Aspekten der verteilt-kooperativen Arbeit in der Bauplanung vor.

1.2 Ausgangspunkt

In den letzten Jahren wurden vielfältige Applikationen entwickelt, die fachspezifische Teilprozesse der Planung in ausreichendem Maß unterstützen. Jedoch mangelt es nach wie vor an den für die kooperative Zusammenarbeit von Fachplanern verschiedener Fachbereiche notwendigen Möglichkeiten des Datenaustauschs auf hohem semantischen Niveau. Hannus illustrierte diesen Zustand eindrücklich mit dem Bild der „Islands of Automation in Construction“.[1]

Die mangelnde Unterstützung des Datenaustauschs äußert sich u.a. darin, dass Informationen größtenteils immer noch in Papierform ausgetauscht und manuell in das jeweilige Zielsystem eingepflegt werden. Zur Erhöhung der Effizienz des Planungsprozesses und zur Vermeidung von Planungsfehlern ist eine durchgängig computergestützte Erhebung, Be- bzw. Verarbeitung und Weitergabe aller im Planungsprozess anfallenden Daten jedoch unabdingbar.

Zwar beginnt sich in der Praxis der Einsatz von Dokumentenverwaltungssystemen durchzusetzen, die dem Austausch und der zentralen Verwaltung von Dateien dienen. Die Granularität des Zugriffs auf einzelne Planungsinformationen ist bei dateibasierten Lösungen jedoch sehr grob, woraus sich fundamentale Schwächen derartiger Systeme hinsichtlich des Änderungsmanagements und der Unterstützung synchroner Arbeitsphasen ergeben.

Einen Ansatz mit weitaus größerem Potential stellt die Integration aller verwendeten Applikationen durch die Nutzung eines gemeinsamen Informationsraums dar. Ein solcher gemeinsamer Informationsraum wird in beiden Teilen dieser Arbeit propagiert und bildet die Grundlage der jeweiligen technischen Lösung.

1.3 Aufbau der Arbeit

Im sich anschließenden Kapitel 2 werden zunächst die für das Verständnis dieser Arbeit notwendigen Grundlagen diskutiert. Hierbei werden verschiedene Formen der verteilt-kooperativen Arbeit, Möglichkeiten der Computerunterstützung sowie die im Bereich der Bauplanung auftretenden Problematiken und verschiedenen Lösungsansätze besprochen.

Der **erste Teil** der Arbeit befasst sich mit Lösungsmöglichkeiten zur Unterstützung des *synchronen* kooperativen Engineerings unter Anbindung von interaktiven Simulationswerkzeugen.

[1] `http://cic.vtt.fi/hannus/islands/index.html`

Der **zweite Teil** behandelt als einen wesentlichen Aspekt der *asynchronen* Zusammenarbeit der Planungsbeteiligten die geometriebasierte Zerlegung des Gesamtmodells in exklusiv bearbeitbare Teilmodelle. Hierbei wird als Basis die Verwendung einer räumlichen Datenbank für digitale Bauwerksmodelle propagiert. Eine solche räumliche Datenbank erlaubt die Extraktion von Partialmodellen aus einem Gesamtmodell auf Grundlage von räumlich-topologischen Kriterien und trägt auf diese Weise zu einer geometrieorientierten Unterstützung kooperativer Planungsvorgänge bei.

Kapitel 2

Computergestützte kooperative Arbeit in der Bauplanung

Als Grundlage für die Diskussion in den folgenden Kapiteln werden in diesem Kapitel zunächst die verschiedenen Begrifflichkeiten im Bereich des verteilt-kooperativen Arbeitens definiert sowie eine Kategorisierung der verschiedenen Formen der Computerunterstützung kooperativer Arbeit vorgestellt. Im Anschluss daran werden die besondere Bedeutung der verteilt-kooperativen Arbeit im Bauwesen, die bei der Realisierung einer Computerunterstützung auftretenden Problematiken sowie auf verschiedenen Methoden der Bauwerksmodellierung beruhende Lösungsansätze dargelegt.

2.1 Verteiltes Arbeiten und Arten der Computerunterstützung

2.1.1 Begriffsdefinitionen

Nach (Bretschneider, 1998) ist unter Kooperation "*die Zusammenarbeit mehrerer Personen, Gruppen oder Institutionen [...] an einem gemeinsamen Material und auf ein gemeinsames Ziel hin*" zu verstehen. Bei der Planung von Bauwerken besteht das gemeinsame Ziel in der Fertigstellung dieser Planung. Das gemeinsame Material bzw. der gemeinsame Betrachtungsgegenstand ist das Modell des zu planenden Bauwerks, das bei klassischer papiergebundener Planungsweise implizit in 2D-Plänen enthalten ist und bei Nutzung moderner computergestützter Systeme als digitales Bauwerksmodell vorliegt.

Computerunterstützte Zusammenarbeit wird im Englischen mit *Computer Supported Cooperative Work* (CSCW) bezeichnet. Dieser Ausdruck wurde erstmals 1984 von den beiden Wissenschaftlern Greif und Cashman verwendet und bezeichnet inzwischen ein weitreichendes Forschungsgebiet mit Anleihen aus den

Sozial- und Arbeitswissenschaften, der Psychologie sowie der Informatik (Teufel *et al.*, 1995; Borghoff & Schlichter, 2000).

Unter *Groupware* wird Software zur Unterstützung kooperativer Arbeit verstanden. Johansen definiert den Begriff wie folgt: *„Groupware is a generic term for specialized computer aids that are designed for the use of collaborative groups.“* (Johansen, 1988). Ellis *et al.* betonen hingegen den Aspekt einer gemeinsamen Arbeitsumgebung in ihrer Definition von Groupware: *„We define groupware as computer-based systems that support groups of people engaged in a common task (or goal) and that provide an interface to a shared environment.“* (Ellis *et al.*, 1991).

2.1.2 Formen von Kooperation

Man kann anhand ihrer Bestandteile verschiedene Stufen von Zusammenarbeit unterscheiden. Die folgenden Definitionen dieser Stufen sind (Bretschneider, 1998) entnommen:

Kommunikation ist Austausch von Informationen zwischen verschiedenen Mitgliedern des Planungsteams bzw. zwischen verschiedenen CAE-Applikationen. Reine Kommunikation setzt also weder die Existenz eines gemeinsamen Betrachtungsgegenstandes noch eines gemeinsamen Ziels voraus.

Koordination basiert auf Kommunikation und hat zum Ziel, neben Informationen auch andere Ressourcen, insbesondere gemeinsames Material bereitzustellen. Durch Koordination werden konkurrierende Planungstätigkeiten am gemeinsamen Material so aufeinander abgestimmt, dass ein weitgehend konfliktfreies Zusammenwirken erreicht wird. Koordination befasst sich also mit dem gemeinsamen Material, setzt aber kein gemeinsames Ziel voraus.

Kooperation bedeutet, dass die Mitglieder des Teams ein gegenseitiges Verständnis aufbauen, um gemeinsam Entscheidungen treffen zu können. Man unterscheidet implizite Kooperation, die durch die Arbeit am gemeinsamen Material geprägt ist, von der expliziten Kooperation, bei der ein bewusstes Austauschen von Informationen im Sinne einer Konversation stattfindet.

Weitere Ansätze zur Klassifikation kooperativer Arbeit, die beispielsweise auf der Teamgröße, den genutzten Interaktionsmedientypen oder dem Grad der Handlungsinterdependenzen beruhen, werden ausführlich in (Hauschild, 2003) diskutiert.

2.1.3 Klassifikation rechnergestützter Kooperation

Rechnergestützte Kooperation wird häufig nach zeitlichen und räumlichen Aspekten der Zusammenarbeit klassifiziert. Diese Klassifikation geht zurück auf (Johansen, 1988), einer frühen Arbeit aus dem Forschungsbereich *Computer Supported*

Cooperative Work. Hierbei werden Ort und Zeit als orthogonale Dimensionen rechnergestützter kooperativer Arbeit definiert. In die entstehende Matrix lassen sich die verschiedenen Formen des kooperativen Arbeitens einordnen (Abb. 2.1). Bezüglich der Dimension *Ort* lassen sich demnach zwei Formen der Kooperation unterscheiden:

- ***zentrale** Kooperation* findet am selben Ort statt, zum Beispiel innerhalb eines Raums,
- ***verteilte** Kooperation* findet an verschiedenen Orten statt, beispielsweise in verschiedenen Räumen eines Gebäudes (lokal verteilte Kooperation) bis über verschiedene Kontinente hinweg (global verteilte Kooperation).

Die vorliegende Arbeit befasst sich ausschließlich mit verteilter Kooperation, bezieht diesen Begriff jedoch weniger auf die Position der Kooperierenden als vielmehr auf die Art der Computernutzung. Danach kann verteilte Kooperation auch innerhalb eines Raumes stattfinden, muss allerdings unter Zuhilfenahme mehrerer unabhängiger Computersysteme realisiert werden.

Bezüglich der Dimension *Zeit* unterscheidet Johansen ebenfalls zwei Formen der Kooperation:

- ***synchrone** Kooperation* findet zur selben Zeit statt,
- ***asynchrone** Kooperation* findet zu verschiedenen Zeitpunkten statt.

Diese Definitionen sind jedoch zu ungenau. Nach Ansicht des Verfassers ist der entscheidende Aspekt bei der zeitlichen Klassifizierung von Kooperation die bewusste Inkaufnahme einer zeitlichen Verzögerung zwischen dem Versenden einer für die Gruppenarbeit relevanten Information und ihrer Rezeption durch den Empfänger. Im Rahmen dieser Arbeit soll daher von *synchroner Zusammenarbeit* gesprochen werden, wenn der Empfänger eine Information rezipiert, unmittelbar[1] nachdem der Sendende sie verschickt hat.

Wird hingegen bewusst das Verstreichen einer unbestimmten Menge von Zeit zwischen Senden und Rezipieren in Kauf genommen oder ist im Arbeitsablauf die Möglichkeit einer zeitlichen Verschiebung vorgesehen, handelt es sich um *asynchrone Zusammenarbeit*. Die asynchrone Zusammenarbeit muss zwangsläufig mit Hilfe eines Zwischenspeichers realisiert werden, wie er beispielsweise bei E-Mail-Systemen oder elektronischen Foren zum Einsatz kommt. Es ist zu beachten, dass es sich bei der übermittelten Information nicht nur um explizite Kommunikation in Form von Nachrichten handeln kann, sondern ebenso um eine Änderung am gemeinsam verwendeten Material, wie beispielsweise einem digitalen Bauwerksmodell.

Im direkten Zusammenhang mit der Form der Informationsübermittlung steht der Faktor der gemeinsamen Aufmerksamkeit (engl. *shared attention*) für den Gegenstand des Informationsaustausches. Bei synchroner Kooperation ist die Aufmerksamkeit der Kooperierenden auf den gleichen Teil des Arbeitsmaterials gerichtet,

[1] „unmittelbar“ schließt Verzögerungen infolge technischer Zwangsbedingungen (Latenzen) mit ein

	synchron	**asynchron**
ortsinzident	Gespräch, Vortrag, gemeinsame Rechnernutzung	Pinnwand, gemeinsame Ablage
verteilt	Telefonat, Chat, Videokonferenz, Shared Application Mehrbenutzereditor	Anrufbeantworter, E-Mail, Internetforum Datenbank

Abbildung 2.1: Zeit-Raum-Klassifikation verschiedener Mittel zur Unterstützung kooperativer Arbeit nach (Johansen, 1988).

bei asynchroner im Allgemeinen nicht. Damit verbunden ist auch die wechselseitige Blockade hinsichtlich der Weiterarbeit der Beteiligten: Bei synchroner Kooperation wartet der Sendende darauf, dass der Empfangende die Information erhalten und verarbeitet hat. Er setzt seine Arbeit nicht fort, da er hierzu eine Stellungnahme seines Kooperationspartners benötigt. In typischen synchronen Kooperationsszenarien wie verteilten Video-Konferenzen, Telefonaten oder der Verwendung eines Mehrbenutzereditors wechseln die Beteiligten ständig die Rollen zwischen Informationssendenden und -empfangenden.

Im Fall asynchroner Kooperation ist die Arbeit der einzelnen Beteiligten hingegen weitestgehend entkoppelt, d.h. eine Weiterarbeit des Sendenden ist grundsätzlich möglich. Abb. 2.1 zeigt Beispiele für die verschiedenen Formen der Zusammenarbeit.

Die vorliegende Arbeit wird Systeme für die beiden zeitlichen Formen von kooperativer Arbeit vorstellen. Während das im ersten Teil dieser Arbeit beschriebene System für die synchrone Zusammenarbeit konzipiert ist, eignet sich das im zweiten Teil dargelegte Datenbank-basierte System vor allem für die Unterstützung asynchroner Kollaboration.

2.1.4 Arten von CSCW-Applikationen

Je nach Form und Inhalt der kooperativen Arbeit gibt es unterschiedliche Klassen von Computersystemen zu ihrer Unterstützung. Nach (Sauter *et al.*, 1995) lassen sich CSCW-Applikationen hinsichtlich des Grads der Unterstützung von Kommunikation, Kooperation und Koordination in vier grundlegende Systemklassen einordnen:

Kommunikationssysteme dienen vor allem dem expliziten Informationsaustausch zwischen Teammitgliedern durch Überwindung von Raum- und/oder Zeitdifferenzen. Beispiele für Ausprägungen dieser Klasse sind Mail- und Konferenzsysteme, bei denen textuelle oder audiovisuelle Informationen ausgetauscht werden.

Gemeinsame Informationsräume unterstützen die implizite Kommunikation zwischen Teammitgliedern und können Informationen über einen längeren Zeitraum hinweg vorhalten. Zu dieser Klasse gehören beispielsweise verteilte Hypertextsysteme (wie Foren, Weblogs u.ä.) oder gemeinsam genutzte Kalender.

Workflow Management Systeme dienen der Ausführung und Koordination von Arbeitsabläufen (engl. workflows). Die Arbeitsabläufe werden dabei zunächst mit Hilfe von Prozessdefinitionswerkzeugen formalisiert. Bei der anschließenden Ausführung wird ein Beteiligter informiert, sobald der vorangegangene Prozessschritt abgearbeitet ist und die Erledigung seiner Aufgabe ansteht. Er informiert das System nach Abschluss seiner Arbeit, woraufhin dieses den oder die Nachfolgenden benachrichtigt. Applikationen dieser Systemklasse basieren in der Regel auf einer Integration von E-Mail- und Datenbanksystemen. Als Beispiele seien die weitverbreiteten Produkte *SAP Business Workflow* und *Oracle Workflow* genannt.

Workflow Computing Systeme unterstützen das Team bei der Erfüllung komplexer Arbeitsaufgaben, die durch Zusammenarbeit realisiert werden müssen. Vertreter dieser Klasse sind elektronische Entscheidungsunterstützungssysteme, Planungsysteme sowie kooperative Dokument- und Zeichnungseditoren.

Wie Abb. 2.2 illustriert, kann eine CSCW-Applikation dabei auch mehreren dieser Klassen angehören. Nach dieser Einteilung gehört das im ersten Teil der Arbeit vorzustellende System der Klasse der *Workgroup Computing Systems* an, während das im zweiten Teil beschriebene datenbankgestützte System den *Shared Infomation Systems* zuzuordnen ist. Beide Formen von Groupware können jedoch ebenso zu den *Shared Information Systems* gezählt werden.

2.2 Kooperative Bauplanung auf Basis einer integrativen Datenhaltung

2.2.1 Charakteristika von Bauplanungsprozessen

Infolge der zunehmend hohen Komplexität von Bauprojekten ist eine ausgeprägte Spezialisierung der an der Planung Beteiligten feststellbar. Das daraus resultierende hohe Maß an Arbeitsteilung ermöglicht zum einen immer kürzere Planungszeiträume, verlangt jedoch gleichzeitig eine erhöhte Kooperation zwischen den Beteiligten. Dies erfordert sowohl eine verstärkte Abstimmung der Arbeitsabläufe, als auch einen intensiveren Informationsaustausch.

Da bereits in nahezu jeder der an der Planung beteiligten Fachdomänen computergestützte Systeme zum Einsatz kommen, ist es wünschenswert, diesen Informationsaustausch im überwiegenden Maße durch den Austausch von computergenerierten Daten zu realisieren. Der computergestützte Datenaustausch ermöglicht durch die Vermeidung der zeitintensiven und fehleranfälligen manuellen

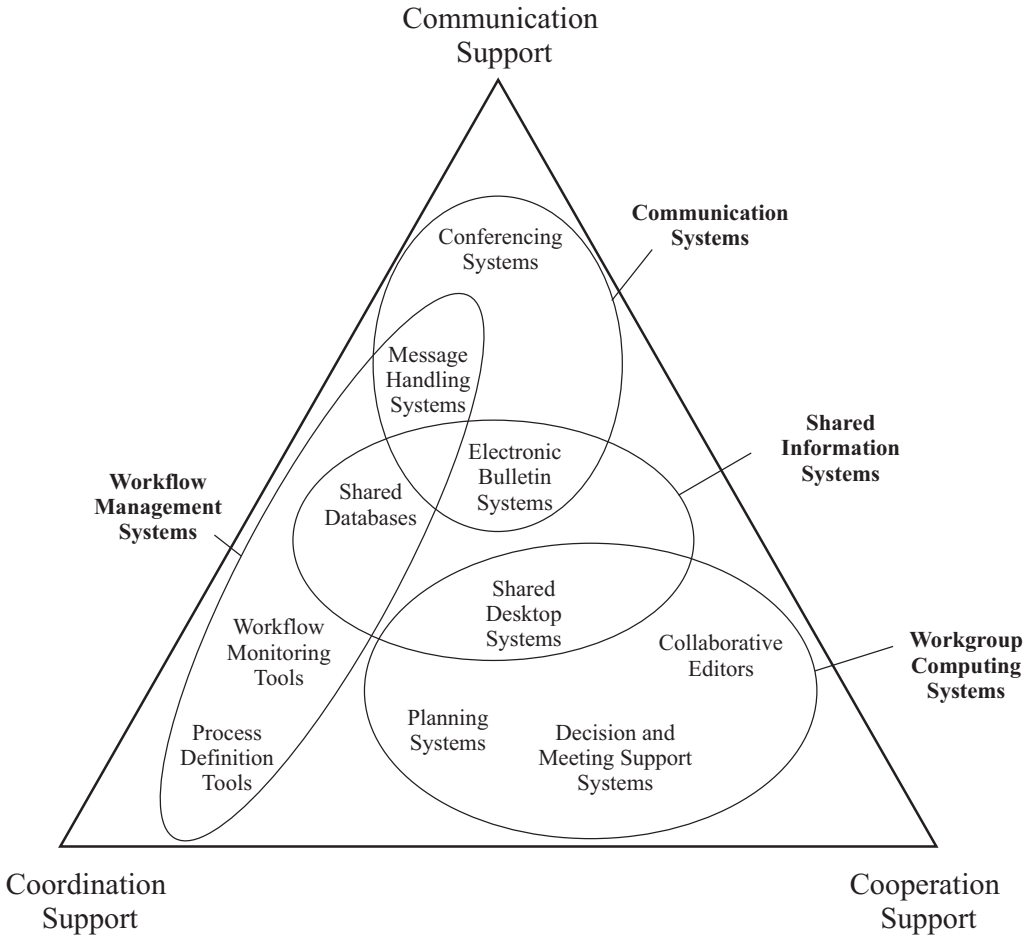


Abbildung 2.2: Klassifikation von CSCW-Systemen nach (Sauter *et al.*, 1995).

Dateneingabe eine deutliche Effizienzsteigerung, was letztlich zur Reduzierung der Gesamtplanungskosten führt.

Die Realisierung einer „computergestützten interprozessualen Daten- und Informationspräsenz“ wird als *Integration* bezeichnet (Willenbacher, 2002). Willenbacher schreibt weiter: *„Prinzipielles Anliegen der Integration im Bauwerkslebenszyklus ist die allzeit adäquate Bereitstellung relevanter Daten und Informationen und die Verbesserung der Kooperation und Kommunikation der Beteiligten . . . “.*

Gegenüber anderen Industriezweigen ist das Bauwesen durch spezifische Charakteristika gekennzeichnet, die eine derartige Integration besonders erschweren. Dazu zählen die starke Fragmentierung der Bauindustrie in eine Vielzahl kleiner und mittelständischer Unternehmen, die Existenz hochspezialisierter heterogener Fachapplikationen sowie der Unikatcharakter des Produkts Bauwerk.

Im Unterschied zur Automobil- oder Maschinenbauindustrie existieren bei Bauprojekten nur in seltenen Fällen große produktionsleitende Unternehmen, die für Zulieferer verbindliche Standards hinsichtlich des Datenaustauschs und der Infomationsflüsse vorgeben. Vorherrschend sind vielmehr temporäre unternehmensübergreifende Zusammenschlüsse gleichberechtigter Partner, sogenannter „Virtual Enterprises“ (Hauschild, 2003), die oftmals nur für die Dauer der Planung eines Bauwerks zusammenarbeiten.

Ein ebenfalls bedeutender Aspekt, der die Bemühungen zur Integration erschwert, besteht darin, dass in der Regel von jedem geplanten Bauwerk nur genau ein Exemplar tatsächlich hergestellt wird. Entsprechend hoch ist der Anteil der Planungskosten an den Gesamtproduktionskosten. Im Unterschied zur in anderen Industriezweigen üblichen Massenproduktion ist daher ein erhöhter Aufwand bei der Planung, beispielsweise durch Schaffung eines höherwertigen Modells, nicht in jedem Fall lohnenswert. Des Weiteren erschweren die von Projekt zu Projekt divergierenden Anforderungen an die vorzuhaltenden und auszutauschenden Daten das Finden allgemeingültiger Standards.

2.2.2 Deskriptiver und prozessualer Modellierungsansatz

Der auf einem gemeinsam genutzten Bauwerksmodell beruhende Ansatz zur Realisierung einer computergestützten Integration im Bauplanungsprozess wird auch als *deskriptiver Modellierungsansatz* bezeichnet. Ein Bauwerksmodell stellt dabei einen gemeinsamen Informationsraum gemäß der in Abschnitt 2.1.4 dargestellten Klassifizierung dar.

Im Unterschied dazu wird beim *prozessualen* Ansatz die explizite Abbildung des Planungsprozesses, d.h. die Modellierung einer Prozesskette einschließlich der Abhängigkeiten zwischen den Teilprozessen verfolgt. Problematisch hierbei ist, dass aktivitäts- und rollenspezifische Prozesse in der Bauplanung schwach strukturiert und hochdynamisch sind (Willenbacher, 2002) und daher eine statische a-priori Modellierung nahezu unmöglich ist. Entsprechend wurden in den letzten Jahren verschiedene Ansätze für eine dynamisierte Prozessmodellierung entwickelt, auf

die hier jedoch nicht näher eingegangen werden soll. Stattdessen wird in diesem Zusammenhang auf (Anumba *et al.*, 1998; König, 2003; Klauer, 2005; Greb, 2005; Berkhahn *et al.*, 2005; Huhnt *et al.*, 2005) verwiesen. Die verschiedenen Möglichkeiten der deskriptiven Modellierung werden hingegen im folgenden Abschnitt diskutiert.

2.2.3 Bauwerksmodelle als Grundlage kooperativer Arbeit

Geometrische Modelle

Die Entwicklung der computergerechten Abbildung von existierenden oder zu planenden Bauwerken hat ihre Ursprünge in der Verwendung des Computers als elektronisches Zeichenbrett. Das daraus resultierende „Modell“ ist eine Ansammlung von 2D-Plänen, die ihren nicht-elektronischen Pendants gleichen. Problematisch ist hierbei vor allem die Sicherung der Konsistenz zwischen den einzelnen Zeichnungen (wie verschiedenen Schnitten und Grundrissen), da entsprechende Abhängigkeiten nicht explizit modelliert werden. Für Downstream[2]-Applikationen, wie Programme zur statischen Berechnung, zur Massenermittlung oder Liegenschaftsverwaltung, sind derartige elektronische Zeichnungen nur selten als Ausgangspunkt nutzbar. In der Folge müssen die geometrischen Daten für das Zielprogramm erneut manuell erfasst und entsprechend aufbereitet werden.

Da auf diese Weise große Potentiale der Computernutzung im architektonischen bzw. ingenieurstechnischen Entwurf verschenkt werden, wird seit Beginn der Verfügbarkeit von leistungsfähiger 3D-Grafik-Hardware die Nutzung von dreidimensionalen geometrischen Modellen vorangetrieben. Dabei wird das Bauwerk von Beginn an dreidimensional modelliert, d.h. alle Bauteile durch Volumenkörper beschrieben. Schnitte und Grundrisse lassen sich anschließend im Prinzip vollständig aus diesem 3D-Modell ableiten. Da Änderungen immer das 3D-Modell betreffen, kann die Konsistenz zwischen Schnitten und Grundrissen gewährleistet werden. Problematisch ist jedoch die durch bestehende Normen geforderte symbolhafte Darstellung bestimmter geometrischer Informationen in Bauplänen, die auf diese Weise nicht realisiert werden kann.

Neben der mangelnden Generierbarkeit von normgerechten Bauplänen spricht jedoch auch die in nahezu jeder Fachdisziplin auftretende Notwendigkeit des Vorhaltens nicht-geometrischer Daten, wie Materialparameter, physikalisches Verhalten, Kosten usw. gegen die Verwendung von ausschließlich geometrischen 3D-Modellen. Diese Anforderungen können nur durch die Verwendung von semantischen Bauwerksmodellen erfüllt werden.

Dennoch wird der Einsatz von reinen Geometriemodellen bzw. primär auf geometrischen Informationen beruhenden Modellen nach wie vor von einigen Forschungsgruppen verfolgt. Fischer *et al.* propagieren beispielsweise die Verwendung von sogenannten 4D-Modellen, also dreidimensionalen geometrischen Modellen,

[2] Anwendungsprogramme, die im Gesamtprozess zeitlich nachgelagert eingesetzt werden

die um zeitliche Aspekte erweitert werden (McKinney *et al.*, 1996; Fischer, 2000; Koo & Fischer, 2006).

Semantische Bauwerksinformationsmodelle

Ausgangspunkt der semantischen Bauwerksmodellierung ist der Wunsch, die Eigenschaften von Bauwerkskomponenten auf abstrakte Art und Weise abzubilden, d.h. unabhängig von einer konkreten geometrischen oder nicht-geometrischen Repräsentation. Ein semantisches Bauwerksmodell – in der Literatur auch als Bauwerksinformationsmodell (engl. *Building Information Model*) oder Produktmodell bezeichnet – kann die Interpretierbarkeit der auszutauschenden Daten gewährleisten und damit eine Verständigungsplattform bilden, die eine effektive Kooperation der Beteiligten ermöglicht (Meissner *et al.*, 1995; Scherer, 1995; Willenbacher, 2002).

Bakkeren & Tolman definieren den Begriff *Produktmodell* wie folgt (Bakkeren & Tolman, 1995): *„A product model is a computer-interpretable, complete and unambigous description of the product being designed, constructed, and operated. It supports the communication between computer-aided activities by storing all the information that needs to be exchanged within the project ... “*

Diese Definition lässt offen, ob es sich bei einem Produktmodell um die Spezifikation einer Datenstruktur oder das computerinterne Abbild eines konkreten Gegenstandes (des Produkts) handelt. Entsprechend uneinheitlich ist die Verwendung dieses Begriffs in der Literatur. Aus diesem Grund unterscheidet (Björk, 1995) explizit zwischen dem *building product data model*, das einen Bauwerkstyp bzw. dessen Komponenten mit Hilfe eines konzeptionellen Schemas beschreibt, und dem *building product model*, das eine Ausprägung dieses Schemas umfasst und damit ein konkretes Bauwerk abbildet. Willenbacher schließt sich dieser Unterscheidung an, verwendet jedoch anstelle von *building product data model* den Begriff *Bauwerksmodell* und statt *building product model* den Begriff *Bauwerksmodelldaten* (Willenbacher, 2002). Im Rahmen dieser Arbeit wird der Begriff *Bauwerksmodell* verwendet, um eine Datenstruktur zu beschreiben, *Produktmodell* und *Bauwerksinformationsmodell* werden synonym benutzt.

Zur Reduzierung der Komplexität bei der Modellierung einer Datenstruktur, die die große Zahl der bei Bauwerken auftretenden Entitäten und Beziehungen abbildet, hat sich die Verwendung des objektorientierten Paradigmas durchgesetzt. Die Objekte des resultierenden Modells repräsentieren tatsächlich oder gedanklich existierende Entitäten wie Bauteile, Personen oder Termine und bilden sämtliche sie beschreibende Eigenschaften ab (Eastman, 1999). Das objektorientierte Paradigma erlaubt dabei den Einsatz von abstraktionsreduzierenden Modellierungskonzepten wie Klassifikation, Kapselung, Beziehungen und Vererbung (Booch, 1994).

Einige Wissenschaftler propagieren darüber hinaus die Aufteilung in domänenspezifische Partialmodelle, auf die hier jedoch nicht näher eingegangen werden soll. Teile dieser Betrachtungsweise finden Eingang in die Konzeption des in Teil I

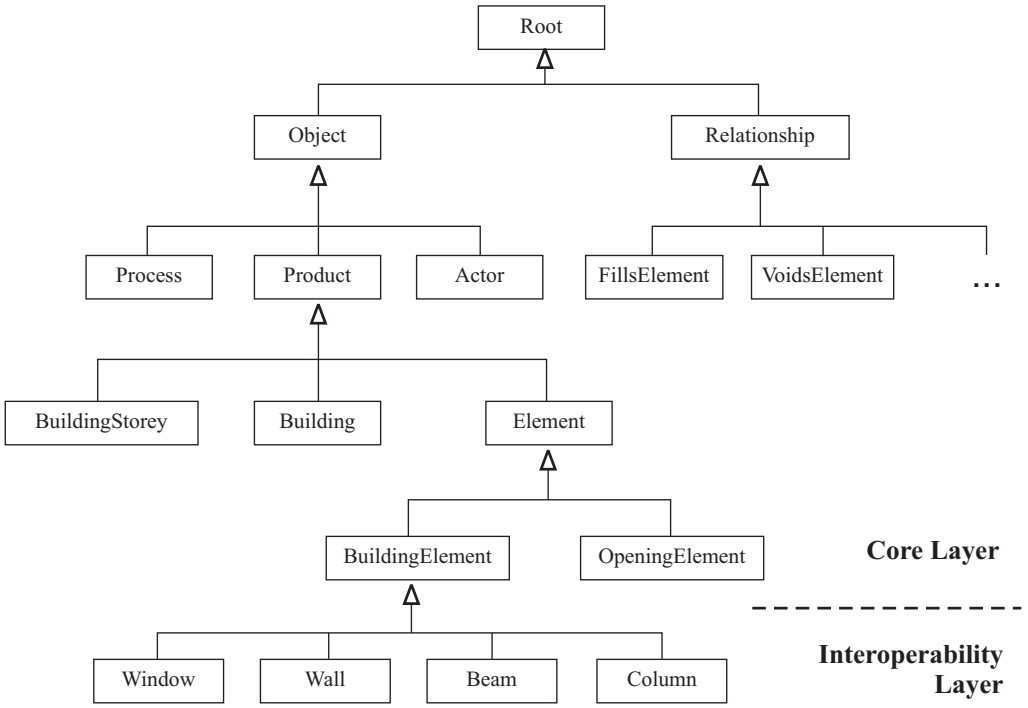


Abbildung 2.3: Ausschnitt aus dem semantischen Bauwerksmodell der IFC.

dieser Arbeit vorzustellenden Systems, das ebenfalls von der Existenz domänen- bzw. simulationsspezifischer Modelle ausgeht.

Semantische objektorientierte Bauwerksmodelle mit dem Ziel einer Integrationsunterstützung wurden u.a. im Rahmen der EU-Projekte COMBI (Katranuschkov, 1994), ATLAS (Tolman & Poyet, 1995), COMBINE (Augenbroe, 1995), VEGA (Junge *et al.*, 1997) sowie dem ARMILLA-Projekt (Hovestadt & Hovestadt, 1999) entwickelt. Sie bildeten zudem eine Grundlage der DFG-Forschungsschwerpunkte 694 (Hartmann, 2000) und 1103 (Rüppel, 2007).

Auf einer objektorientierten Modellierung von Bauwerken basieren ebenfalls die Standards STEP[3], CIMsteel[4] und IAI-IFC[5], deren Ziel die Kompatibilität beim Austausch von Daten im Bauwerkslebenszyklus ist. Die am weitesten fortgeschrittenen Bemühungen zur Spezifikation eines semantischen Bauwerkmodells stellen dabei die *Industry Foundation Classes* dar (Abb. 2.3), die bereits Eingang in eine Vielzahl von kommerziellen Produkten gefunden haben.

Wegen des erhöhten Aufwands beim Erstellen von dreidimensionalen semantischen Modellen überwiegt in großen Teilen der Bauindustrie nach wie vor die Erstellung und der Austausch von zweidimensionalen elektronischen Zeichnun-

[3]Standard for the Exchange of Product Data Model - ISO Standard 10303

[4]Computer Integrated Manufacturing for Constructional Steelwork (CIMSteel) Integration Standard CIS/2, `http://www.cis2.org`

[5]International Alliance for Interoperability - Industry Foundation Classes, ISO PAS16579, `http://www.iai-international.org`

gen. Gründe hierfür liegen zum einen in den bestehenden rechtsverbindlichen Normen zur Erstellung von Bauplänen, zum anderen aber auch an der bereits angesprochenen Fragmentierung der Bauindustrie entlang der Prozesskette, die die Wahrnehmung einer erhöhten Wirtschaftlichkeit des Gesamtplanungsprozesses bei Verwendung höherwertiger Modelle verhindert. Schließlich ist die Amortisierung der erhöhten Planungskosten durch den bereits diskutierten Unikatcharakter von Bauwerken nicht in jedem Fall gegeben.

Ein weiteres, nicht vollständig gelöstes Problem beim Einsatz semantischer Bauwerksmodelle mit dem Ziel, Interoperabilität und Durchgängigkeit im Planungsprozess zu erreichen, ist die Notwendigkeit der Standardisierung, da daraus ein enormer Aufwand im Standardisierungsprozess selbst sowie bei der Implementierung des resultierenden Standards in den Anwendungsprogrammen der verschiedenen Hersteller resultiert. Als besonders erschwerend wirkt dabei das breite Spektrum der in der Realität existierenden Bauwerkstypen, für die es auch mittelfristig nur bedingt Aussichten auf eine Standardisierung gibt. Für viele Spezialbauten bzw. Spezialdomänen stellt sich zudem grundsätzlich die Frage, ob der hohe Aufwand einer durchgehend objektorientierten Modellierung lohnenswert ist. Diesem Einwand wird durch die Verwendung dynamischer Bauwerksmodelle zu begegnen versucht, die im Folgenden beschrieben werden sollen.

Dynamische Bauwerksmodelle

Dynamische Bauwerksmodelle basieren auf dem Konzept einer sukzessiven Erweiterung bzw. Anpassung der objektorientierten Datenstruktur an die Bedürfnisse der Anwendungsdomäne. Beispielsweise empfiehlt Steinmann zur Unterstützung von Entwurfsaktivitäten im Rahmen des sogenannten Innovativ-Designs die Nutzung von dynamischen Bauwerksmodellen und betrachtet deren Anwendung ausführlich für frühe Phasen des architektonischen Entwurfs (Steinmann, 1997).

Hauschild überträgt das Konzept dynamischer Bauwerksmodelle auf die Verwendung über den gesamten Lebenszyklus eines Bauwerks hinweg und nennt dabei folgende Vorteile (Hauschild, 2003):

- **Lebenszyklus-Aspekt.** Ein dynamisches Bauwerksmodell kann von den ersten Entwurfshandlungen über mehrere Nutzungs- und Umbauphasen bis zum Abbruch verwendet werden.
- **Wechsel der Bauwerksnutzung.** Nach dem Wechsel der Nutzungsart können teilweise andere Informationen für das Facility Management relevant sein.
- **Know How-Aspekt.** Ein dynamisches Bauwerksmodell ermöglicht dem Entwerfenden, eigenes Entwurfswissen in der Entwurfsumgebung abzulegen.
- **Aufgaben-Aspekt und Vollständigkeitsaspekt.** Der Unikatcharakter von Bauwerken und der geringe Standardisierungsgrad der eingesetzten

Bauteile erhöht die Wahrscheinlichkeit, mit statischen Modellen nicht alle relevanten Informationen abbilden zu können.

– **Softwarelebenszyklus-Aspekt.** Dynamische Bauwerksmodelliersysteme besitzen eine potentiell längere Nutzungsdauer als Applikationen auf Grundlage statischer Modelle, da sie leichter an neue Technologien oder Materialien sowie Änderungen von Bauvorschriften, Normen etc. angepasst werden können.

Willenbacher schlägt ebenfalls den Einsatz dynamischer Bauwerksmodelle während des gesamten Lebenszyklus vor, empfiehlt aber zur weiteren Komplexitätsreduzierung die Aufteilung in domänenspezifische Partialmodelle, deren Kohärenz mittels explizit zu definierender Verknüpfungen gesichert wird (Willenbacher, 2002).

Wesentliche technische Aspekte bei der Verwendung dynamischer Bauwerksmodelle sind die Einführung eines explizit verfügbaren Metamodells, das die laufzeitdynamische Modifikation des Bauwerkmodells ermöglicht, sowie der Austausch von projektbezogenen Bauwerksdaten mit Hilfe einer generischen Schnittstelle (Hauschild *et al.*, 2003).

Ein ähnlicher Ansatz wird in der Arbeit von Kowalczyk verfolgt (Kowalczyk, 1997). Sie beschreibt einen evolutionären Modellierkern, der sich ebenfalls auf die Modelliermächtigkeit des objektorientierten Paradigmas stützt.

Darüber hinaus entstand im Kontext der EU-Projekte VEGA und COMBI die Plattform O.P.E.N. (Object-oriented Product model Engineering Network) als industrielle Implementierung einer Middleware für computergestützte Kooperation im Bauwesen (Beetz *et al.*, 1998). Um die Flexibilität der Produktmodellinformationen zu gewährleisten, wurde hier ebenfalls ein dynamischer Produktmodellierungskern entwickelt, der die Möglichkeit der dynamischen Schemaevolution bietet, also Erzeugen, Löschen und Modifizieren von Schemainformationen zur Laufzeit. Trotz dieser Bemühungen konnten sich modelldynamische Ansätze in der Praxis bislang nur wenig durchsetzen.

2.2.4 Verwaltung von Bauwerksmodellen

Orthogonal zur Methode der Bauwerksmodellierung ist die eingesetzte Technik der Bauwerksmodellverwaltung. Sie bildet die technische Basis zur Realisierung des Datenaustauschs und damit der Unterstützung der kooperativen Arbeit. Man kann hierbei nach der Art der vorzuhaltenden Informationen zwischen Dokumentenverwaltungssystemen und Modellverwaltungssystemen unterscheiden.

Dokumentenverwaltungssysteme

Die dokumentenbasierte Verwaltung der anfallenden Daten resultiert daraus, dass viele Fachapplikationen ihre Daten in Dokumenten (Dateien) ablegen. Mit Hilfe

sogenannter Dokumentenverwaltungssysteme (engl. *Document Management System*, DMS) können alle bei einem Projekt anfallenden Dokumente zentral abgelegt und verwaltet werden (Matheu, 2005). DMS verfügen in der Regel über eine ausgefeilte Steuerung der Zugangsberechtigung, bieten verschiedene Möglichkeiten der automatisierten Benachrichtigung bei Neueinstellen oder Verändern eines Dokuments und erlauben die Verfolgung von Revisionsständen unter Angabe des Namens des Bearbeiters und dem Zeitpunkt der Modifikation. Häufig wird ihre Bedienung mit Hilfe von Web-Portalen realisiert, was ein wesentlicher Aspekt für die technisch reibungslose Integration in bereits existierende Softwaresysteme ist.

Dank der vergleichsweise einfachen Integration in bestehende Softwarelandschaften und Arbeitsabläufe sind DMS bereits bei vielen Projekten im praktischen Einsatz und beginnen sich bei größeren Bauvorhaben als Standardwerkzeug durchzusetzen (Björk, 2001; Hjelt & Björk, 2006; Gabrielaitis & Baušys, 2006). Allerdings ist der Grad der Unterstützung von kooperativen Prozessen durch DMS von vornherein limitiert, da die kleinste von ihnen verarbeitbare Informationseinheit eine Datei ist. Das bedeutet, dass das System lediglich feststellen kann, dass eine Datei geändert wurde. Welche Änderungen genau vorgenommen wurden, bleibt einem DMS jedoch verborgen. Entsprechend schwierig gestaltet sich das Management von Konflikten nach der parallelen Bearbeitung eines Dokuments durch mehrere Benutzer. Es ist daher festzustellen, dass sich Dokumentenverwaltungssysteme hauptsächlich zur Unterstützung rein sequentieller Arbeitsabläufe eignen. Synchrone Kooperation auf Basis eines DMS ist nicht möglich.

Modellverwaltungssysteme

Diesbezüglich eine deutlich bessere Unterstützung kann durch eine Verwaltung des Bauwerkmodells durch Datenbankmanagementsysteme bzw. sogenannte Produktmodellserver erreicht werden (s.a. Kapitel 9), da hierbei der direkte Zugriff auf einzelne Daten, wie beispielsweise Attribute eines Bauteils, möglich ist. Dies erlaubt eine weitaus feinere Steuerung von konkurrierenden Zugriffen und eröffnet erweiterte Möglichkeiten der Versionierung (Firmenich, 2002) und automatisierten Zusammenführung von infolge getrennter Bearbeitung divergierenden Modellzuständen.

Erste Konzepte zur modellbasierten Verwaltung wurden bereits Ende der 90er Jahre entwickelt (Katranuschkov & Hyvärinen, 1998; Khemlani & Kalay, 1997; Debras *et al.*, 1998). Neben einigen ausgereiften Forschungsprototypen wie dem *IFC Model Server* (Adachi, 2002) existieren mittlerweile bereits erste kommerzielle Produkte, darunter der *Eurostep ModelServer for IFC*[6] und der *Express Data Manager*[7] von EPM Technology.

Zur Verwaltung von dynamischen Bauwerksmodellen kommen sogenannte Modellverwaltungssysteme zum Einsatz (Hauschild *et al.*, 2003), die sich in ihrer kooperationsunterstützenden Charakteristik von Produktmodellservern jedoch le-

[6] `http://www.eurostep.com`
[7] `http://www.epmtech.jotne.com/products`

diglich durch die dezentrale Verwaltung von Domänenmodellen unterscheiden (Willenbacher, 2002).

2.3 Zusammenfassung

In diesem Kapitel wurden die Grundlagen der Computerunterstützung verteilt-kooperativer Arbeit in der Bauplanung dargelegt und dabei zwei grundsätzliche Kooperationsformen definiert: die *synchrone* Zusammenarbeit, die Gegenstand des sich anschließenden ersten Teils der Arbeit ist, und die *asynchrone* Zusammenarbeit, mit der sich der zweite Teil der Arbeit befasst. Hinsichtlich der verschiedenen Möglichkeiten der computerinternen Abbildung von Bauwerken bietet die Verwendung eines objektorientierten semantischen Bauwerkmodells eine Reihe von Vorteilen, darunter die Ableitbarkeit unterschiedlichster Repräsentationen wie normgerechter Pläne und 3D-Simulationsmodelle.

Im Unterschied zu den vorgestellten Methoden stellen die in den beiden Teilen der Arbeit vorgestellten Ansätze die dreidimensionale Geometrie der Bauteile in den Mittelpunkt der Bauwerksmodellierung. Die Motivation hierfür liegt in der überproportional hohen Bedeutung von geometrischen Informationen für den Entwurf von Bauwerken und vor allem für den Anschluss von Simulationswerkzeugen.

In das in Teil I der Arbeit vorzustellende Konzept wurden Ideen der dynamischen Bauwerksmodellierung aufgenommen, darunter die Verfügbarkeit eines expliziten Metamodells. Es dient im vorgeschlagenen System jedoch nicht zur laufzeitdynamischen Modifikation des Datenmodells durch den Nutzer, sondern zum Austausch von Modellinformationen zwischen den Komponenten des verteilten Systems.

Das in Teil II besprochene Konzept einer räumlichen Datenbank sieht ebenfalls die Existenz eines semantischen Bauwerkmodells vor, ist jedoch weitestgehend unabhängig von dessen konkreter Struktur. Dies eröffnet wiederum die Möglichkeit der Dynamisierung des verwendeten semantischen Modells bei gleichzeitigem Erhalt der räumlich-geometrischen Analysefunktionalitäten.

Teil I

Collaborative Computational Steering

Kapitel 3

Einführung und Motivation

3.1 Integration von Entwurfs- und Simulationswerkzeugen

Computergestützte Simulations- und Analysewerkzeuge spielen eine entscheidende Rolle im Bauplanungsprozess. Zu den am häufigsten eingesetzten numerischen Berechnungswerkzeugen gehören Programme zur Strukturanalyse beim Tragwerksentwurf, die in der Regel auf Basis der Finite-Elemente-Methode (FEM) arbeiten (Zienkiewicz & Taylor, 2005). Daneben gibt es eine ganz Reihe weiterer Simulationsprogramme, die für die Bauplanung eingesetzt werden, darunter beispielsweise Systeme zur Berechnung von Luftströmungen innerhalb und außerhalb des Gebäudes, zur Beleuchtungsanalyse mit Hilfe von Raytracing- oder Radiosity-Verfahren sowie zur Simulation des Energieverbrauchs von Gebäuden (Clarke, 2001). Diese Berechnungsprogramme formen jedoch zu einem erheblichen Teil immer noch Automatisierungsinseln, d.h. es ist sowohl eine manuelle Aufbereitung der Eingangsdaten als auch die manuelle Überführung der Ergebnisdaten in ein geeignetes Auswertungsprogramm notwendig (s.a. Abschnitt 2.2).

Eine stärkere Integration von Entwurfsapplikationen und Simulationswerkzeugen stellt dementsprechend eine der großen Herausforderungen für die Forschung auf dem Gebiet des rechnergestützten Engineerings dar (Shephard *et al.*, 2004; Rank *et al.*, 2005). Das Ziel ist hierbei, einen weitestgehend nahtlosen, automatisierten Datenaustausch zwischen Entwurfs- und Simulationsprogrammen dergestalt zu realisieren, dass der Nutzer hierzu weder über detailliertes technisches Wissen verfügen, noch übermäßig Zeit aufwenden muss. Im günstigsten Fall sind die verschiedenen Softwarewerkzeuge in so hohem Maß integriert, dass die Nahtstellen zwischen den Systemen nur noch bei Bedarf bzw. auf Wunsch erkennbar sind. Wenn dies gelingt, können Arbeitsabläufe weitaus effizienter gestaltet und damit der zeitliche und finanzielle Aufwand für ein Gesamtprojekt deutlich reduziert werden.

3.2 Computational Steering

Eines der Schlüsselkonzepte auf dem Weg zur Integration von Entwurf und Berechnung ist *Computational Steering*. Der Ausdruck wurde von Liere *et al.* eingeführt und beschreibt die interaktive Veränderung von Randbedingungen durch den Nutzer zur Laufzeit einer Simulation (Liere *et al.*, 1997). Ein wichtiges Prinzip des *Computational Steering*-Paradigmas ist, dass es dem Nutzer ermöglicht wird, die Auswirkungen seiner Modifikationen auf den simulierten Prozess direkt zu beobachten. Der Anwender kann dadurch ein intuitives Verständnis für den simulierten Prozess aufbauen, also für die Zusammenhänge zwischen Modifikationen der Eingangsparameter und der Reaktion des Prozesses. Zwar kann das *Computational Steering*-Konzept grundsätzlich auf jeden beliebigen simulierbaren Prozess angewandt werden, diese Arbeit fokussiert jedoch vornehmlich auf die Simulationen raum-zeitlicher physikalischer Prozesse.

Um eine interaktive Simulation mit angemessen kurzen Reaktionszeiten realisieren zu können, müssen im Allgemeinen Abstriche hinsichtlich der erzielbaren Genauigkeit im Vergleich zu nicht-interaktiven Simulationen in Kauf genommen werden. Bei der numerischen Simulation physikalischer Prozesse resultiert die notwendige Rechenzeit zum einen aus der Feinheit der verwendeten räumlichen und zeitlichen Diskretisierung und zum anderen aus der Komplexität des zugrunde liegenden physikalischen Modells. Interaktive Simulationen müssen also in der Regel mit einem vereinfachten physikalischen Modell sowie einer gröberen Diskretisierung arbeiten. Grundsätzliche Aussagen bezüglich des betrachteten physikalischen Vorgangs können jedoch trotzdem abgeleitet werden. Auf diese Weise kann eine interaktive Simulation als Grundlage für grobe Designentscheidungen in frühen Phasen des Entwurfs dienen. In späteren Entwurfsphasen sollten sich jedoch detaillierte Untersuchungen auf Basis herkömmlicher nicht-interaktiver Simulationen anschließen.

Das herkömmliche Verfahren zur Durchführung numerischer Simulationen oder Berechnungen während des Bauplanungsprozesses ist mit hohem manuellen Aufwand verbunden. Zunächst muss die Geometrie des Bauwerks bzw. eines Teils aus Plänen oder einem digitalen Bauwerksmodell extrahiert und für die Eingabe in das Simulationswerkzeug aufbereitet werden. In vielen Fällen muss dies manuell geschehen. Danach werden die geometrischen Informationen um zusätzliche simulationsrelevante Daten angereichert, wie beispielsweise Materialparameter, Oberflächenbeschaffenheit u.ä. Diese gesamte Phase der Datenaufbereitung wird im Allgemeinen als *Pre-Processing* bezeichnet.

Auf Grundlage der entsprechend aufbereiteten Eingangsdaten kann die Simulation schließlich gestartet werden. Abhängig vom notwendigen Rechenaufwand für die Simulation und den verfügbaren Ressourcen kann die Berechnung mehrere Stunden oder sogar Tage in Anspruchung nehmen. Während dieser Zeit sind keine Änderungen an den Eingangsparametern möglich.

Wenn die Berechnung beendet ist, werden die Ergebnisse in geeigneter Form dargestellt, um ihre Interpretation durch den Fachplaner zu erleichtern. Dieser

Schritt wird als *Post-Processing* bezeichnet und umfasst ggf. eine erneute Konvertierung in das Datenformat der Visualisierungssoftware.

Zeigen sich bei der Analyse der Ergebnisse Defizite der gewählten Entwurfslösung, werden die Ausgangsdaten entsprechend verändert, die Simulation erneut gestartet und nach ihrer Beendigung wiederum visualisiert. Ein solcher Optimierungszyklus wird solange wiederholt, bis ein zufriedenstellendes Ergebnis erzielt werden kann. Weil die drei Teilschritte *(Pre-Processing, Simulation, Post-Processing)* von einander isoliert sind und der Übergang von einem Teilschritt zum nächsten in der Regel mit manuellem Datentransfer und ggf. sogar -konvertierung verbunden ist, sind derartige Optimierungszyklen langwierig und erfordern einen hohen personellen Aufwand. Eine Möglichkeit, den Zeit- und Personalaufwand zu reduzieren, liegt in der Verfolgung des *Computational Steering*-Ansatzes.

Im letzten Jahrzehnt hat sich der Kontext für computergestütztes Engineering stark gewandelt. Dank der beständig fallenden Preise auf dem Hardware-Markt und neuer Techniken der Nutzbarkeit entfernter Großrechenanlagen ist mittlerweile sogar Höchstrechenleistung für klein- und mittelständische Unternehmen erschwinglich. Die aktuellen Entwicklungen im Rahmen des sogenannten *Grid Computing* werden diesen Trend durch vereinfachten Zugang zu öffentlichen oder privaten Hochleistungsrechnern auf der ganzen Welt und Verwendbarkeit dieser Ressourcen auf Basis eines nutzungsabhängigen Entgeltes weiter verstärken (Foster & Kesselman, 1999; Brooke *et al.*, 2003; Turk *et al.*, 2004). Diese Veränderungen bereiten zusammen mit neuen numerischen Methoden und Parallelisierungstechniken den Weg für den Einsatz von *Computational Steering* in der Bauplanungspraxis. Daraus werden kürzere Entwicklungszyklen und hinsichtlich verschiedenster Designziele besser optimierte Bauwerke resultieren.

3.3 Collaborative Computational Steering

Wie in Abschnitt 2.2 bereits ausführlich dargelegt, spielt die verteilt-kooperative Zusammenarbeit mehrerer Fachplaner von derselben oder von unterschiedlichen Fachdisziplinen eine entscheidende Rolle für einen effizienten Ablauf der Bauplanung.

Collaborative Computational Steering ist eine Methode zur Unterstützung von Planungsphasen mit hauptsächlich *synchroner* verteilter Zusammenarbeit. Fachplaner derselben oder unterschiedlicher Fachdisziplinen können mit Hilfe der vorzustellenden Plattform CoCoS verteilte Konferenzen abhalten, um über Details einer Planungsentscheidung zu beraten und ggf. auftretende Planungskonflikte zu lösen. Die Besonderheit dieser Form der computergestützten Zusammenarbeit ist die Integration des *Computational Steering*-Ansatzes. Durch die Verfügbarkeit von interaktiven Simulationen während der kooperativen Sitzung können die beteiligten Fachplaner die Auswirkungen einer Planungsentscheidung direkt und unmittelbar abschätzen. Auf diese Weise wird die Entscheidungsfindung im Entwurfsprozess zum einen beschleunigt und zum anderen transparenter, da die

Gründe für eine Planungsentscheidung allen Beteiligten auf nachvollziehbare Weise dargelegt werden können.

Ein wichtiger Aspekt bei der Konzeption eines *Collaborative Computational Steering*-Systems für die Bauplanung ist die Unterstützung der Kooperation von Planern aus unterschiedlichen Fachbereichen. Da diese in der Regel unterschiedliche Anforderungen an die von ihnen verwendeten Entwurfs- und Simulationsanwendungen haben, sind einfache Ansätze für die Unterstützung synchronen Arbeitens wie *Desktop Sharing* oder *Application Sharing* als nur bedingt geeignet einzuschätzen. Zur Verdeutlichung soll als Beispiel die synchrone Zusammenarbeit eines Ingenieurs der technischen Gebäudeausstattung (TGA), der u.a. für die Platzierung und Auslegung von Ein- und Auslässen der Klimaanlage zuständig ist, mit einem Innenarchitekten, der die Position der Arbeitsplätze in einem Büro festlegt und dabei u.a. den Tageslichteinfall berücksichtigen muss, betrachtet werden. Zur Abschätzung der Auswirkungen der gemeinsamen Entwurfsentscheidungen wird der Klimaingenieur[1] eine interaktive Strömungssimulation verwenden und die Ergebnisse zum Beispiel in Form von Stromlinien visualisieren, während der Innenarchitekt beispielsweise eine interaktive Beleuchtungssimulation zur Beurteilung ergonomischer und ästhetischer Aspekte nutzen möchte.

Dem Umstand der unterschiedlichen fachspezifischen Bedürfnisse hinsichtlich der verfügbaren interaktiven Simulationen wird in der vorzustellende CoCoS-Plattform durch die Möglichkeit Rechnung getragen, dass die an einer kooperativen Sitzung Beteiligten das mit ihrem jeweiligen Frontend verbundene interaktive Simulationswerkzeug unabhängig voneinander wählen können. Das gemeinsame Arbeitsmaterial wird im Extremfall lediglich durch das gemeinsam genutzte geometrische Modell definiert.

3.4 Kooperative Planung der Klimatechnik

Als Anwendungsbeispiel für *Collaborative Computational Steering* soll die kooperative Zusammenarbeit mehrerer Klimaingenieure mit dem Ziel der Schaffung eines für die späteren Nutzer angenehmen Raumklimas betrachtet werden. Auf die Theorie des thermischen Behaglichkeitsempfindens soll hier jedoch nicht näher eingegangen, sondern stattdessen auf (van Treeck, 2004) verwiesen werden.

Der *gemeinsame Betrachtungsgegenstand* bei einer solchen Zusammenarbeit sei ein Büroraum inklusive der umgebenden Wände, der verschiedenen Wandöffnungen für Türen und Fenster sowie der für die Regulierung des Raumklimas zuständigen Gerätschaften wie Heizkörper und Klimaanlage.

[1] Im Folgenden soll der Begriff *Klimaingenieur* verwendet werden, um einen TGA-Planer zu bezeichnen, dessen Aufgabe es ist, ein optimales, d.h. für die späteren Nutzer angenehmes Raumklima zu schaffen. Er sorgt für die Auslegung von Heiz- und Klimaanlagen und für die Anordung von Heizkörpern und Auslässen der Klimanlage in den betreffenden Wohn- oder Büroräumen.

Um bestmöglichen Komfort hinsichtlich des Raumklimas für die späteren Nutzer zu erzielen, müssen sowohl die Ein- und Auslässe der Klimaanlage als auch die Heizkörper im Raum richtig bemessen und optimal positioniert werden. Auf das lokale Komfortempfinden der späteren Büronutzer hat zudem die Positionierung der Arbeitsplätze sowie weiterer beweglicher Einrichtungsgegenstände wie beispielsweise Stellwände oder Pflanzen einen Einfluss.

Die *gemeinsame Informationsmenge* (Arbeitsmaterial) der Kooperierenden setzt sich daher aus der Position und der geometrischen Form der Einrichtungsgegenstände, der Heizkörper und der Ein- und Auslässe der Klimaanlage zusammen. Diese Informationen werden im betrachteten Beispiel den an der Sitzung Beteiligten in Form einer dreidimensionalen virtuellen Welt präsentiert. Das gemeinsame geometrische Modell ist dabei um zusätzliche charakteristische Kenngrößen erweitert, wie Einströmgeschwindigkeit am Einlass, Wärmestrom der Heizkörper etc.

Zu einer präzisen Voraussage des aus diesen Randbedingungen resultierenden Komforts eignet sich ganz besonders die Verwendung einer Strömungssimulation. Mit ihrer Hilfe können typische lokale Erscheinungen wie Zugluft, vertikale Temperaturschichtung, zu warmer oder zu kalter Fußboden berücksichtigt werden.

3.5 Verwandte Arbeiten

3.5.1 Mehrbenutzereditoren und Collaborative Virtual Environments

Die Unterstützung der synchronen Zusammenarbeit verteilter Nutzer auf Basis sogenannter Mehrbenutzereditoren wurde erstmals Ende der 1980er Jahre umgesetzt. Die ersten Anwendungen waren gemeinsam nutzbare Textverarbeitungs- (Ellis *et al.*, 1991; Newman-Wolfe & Pelimuhandiram, 1991; Baecker *et al.*, 1993) und Zeichenprogramme (Greenberg *et al.*, 1992; Moran *et al.*, 1995). Später wurden Systeme für die verteilte Zusammenarbeit in einer CAD-Umgebung entwickelt (Rosenman & Gero, 1996).

Da die vorzustellende CoCoS-Plattform die synchrone Zusammenarbeit in einem virtuellen dreidimensionalen Raum unterstützt, kann sie zu den *Collaborative Virtual Environments*[2] (CVEs) gezählt werden. CVEs stellen den Teilnehmern einer kooperativen Sitzung eine immersive Schnittstelle zu einem virtuellen Raum zur Verfügung, indem sie Techniken der Virtuellen Realität (VR), darunter beispielsweise stereoskopische Projektionen, verwenden (Singhal & Zyda, 1999; Churchill *et al.*, 2002). Der Nutzer eines CVEs kann durch die virtuelle Welt navigieren und sowohl mit virtuellen Objekten als auch anderen Nutzern interagieren. Üblicherweise werden die anderen Teilnehmer dabei durch symbolhafte Darstellungen im virtuellen Raum, sogenannte Avatare, verkörpert (Capin *et al.*, 1998).

[2] in der Literatur mitunter auch als *Networked Collaborative Virtual Environments* (NCVE) bezeichnet

Infolge intensiver Forschungsanstrengungen entstanden Anfang der 1990er Jahre eine Reihe ausgereifter CVE-Systeme, darunter MASSIVE, DIVE, NPSNET und VIVA, die im Folgenden einzeln vorgestellt werden sollen.

MASSIVE ist ein verteiltes VR-System und unterstützt die Interaktion zwischen den Nutzern durch Text, Audio und Grafik (Greenhalgh & Benford, 1995). Das Grafik-Interface erzeugt einen dreidimensionalen Raum, durch den sich die Nutzer hindurchbewegen können, das Audio-Interface erlaubt Nutzern, Geräusche in diesem Raum zu hören und sich miteinander zu unterhalten, und das Text-Interface stellt eine vereinfachte 2D-Darstellung der virtuellen Welt zur Verfügung. Eine Schlüsseleigenschaft von MASSIVE ist es, dass diese Interfaces auf beliebige Art und Weise kombiniert werden können, um den Fähigkeiten der Hardware an den einzelnen Arbeitsplätzen gerecht zu werden.

DIVE steht für *Distributed Interactive Virtual Environment* und ist ein Internetbasiertes Mehrbenutzer-VR-System, in dem die Teilnehmer durch einen dreidimensionalen virtuellen Raum navigieren und andere Nutzer sehen, treffen und mit ihnen interagieren können (Hagsand, 1996). Es handelt sich bei DIVE um ein lose gekoppeltes, heterogenes verteiltes System, das auf Basis von Peer-to-Peer-Kommunikation arbeitet und daher ohne einen zentralen Server auskommt. DIVE kombiniert VR-Techniken mit Audio, der Handhabung von Dokumenten und dem WWW und stellt den Teilnehmern eine 3D-Nutzerschnittstelle zur Verfügung. Wie bei MASSIVE werden andere Nutzer mit Hilfe von Avataren dargestellt.

Die Arbeiten im Rahmen von **NSPNET** (Macedonia *et al.*, 1994) befassen sich vor allem mit Techniken zur Realisierung von *Large Scale* CVEs, also virtuellen Umgebungen, bei denen mehr als 1000 Nutzer gleichzeitig angemeldet sein können. Der Rahmen des Einsatzes von NSPNET liegt überwiegend im Bereich von Mehrbenutzer-Spielen und militärischen Simulationen.

Im Gegensatz dazu ist **VIVA** ein CVE, das explizit zur Unterstützung verteilter *Arbeit* konzipiert wurde (Pekkola *et al.*, 2000). Hierbei handelt es sich um eine verteilte Office-Anwendung, die wiederum verschiedene Medien wie VR, Audio, Video, WWW und Dokumentenhandling in sich vereinigt, um den Teilnehmern eine einheitliche Applikation für die Gruppenarbeit zur Verfügung zu stellen.

Trotz der vielfältigen und umfangreichen Forschungsanstrengungen konnte sich die Verwendung von CVEs in der Ingenieurspraxis bislang nicht durchsetzen. Gründe dafür sind unter anderem in der fehlenden Integration von Simulationswerkzeugen in die verteilten Anwendungen zu suchen. Simulationswerkzeuge stellen eine unverzichtbare Grundlage für die Entscheidungsfindung im Entwurfsprozess dar. Sind sie während einer kooperativen Sitzung nicht verfügbar, schränkt dies das Spektrum möglicher Einsatzgebiete von CVEs stark ein. Hier setzt das Konzept der vorzustellenden CoCoS-Plattform unter Nutzung der in den beschriebenen Forschungsprojekten erzielten Erkenntnisse an.

Ein erstes System, bei dem kooperativer Entwurf und Simulation integriert wurden, ist das *Shared Simple Virtual Environment* (**SSVE**), das die Kooperation

verteilt arbeitender Ingenieure unterstützen soll (Linebarger *et al.*, 2003). Bei seiner Entwicklung wurde das Augenmerk vor allem auf die durch den Zusammenbau von Einzelteilen geprägte Konstruktionsarbeit gelegt. Neben dem gemeinsamen Zusammenbau ist es möglich, einfache Simulationen innnerhalb der verteilten Konstruktionsumgebung durchzuführen. Die Funktionsweise des Systems wird anhand der Entwicklung und des virtuellen Tests einer Achterbahn demonstriert.

Im Bereich asynchronen kooperativen Arbeitens ist das **SEMPER**-Projekt eines der anspruchsvollsten Vorhaben und inhaltlich eng mit dem CoCoS-Projekt verwandt. SEMPER ist eine interdisziplinäre, raumbasierte Designumgebung mit integrierten Werkzeugen zur Simulation des Energieverbauchs, der Analyse des thermischen Komforts u.v.m. (Lam *et al.*, 2002). Das System basiert, ebenso wie CoCoS, auf einem zentral verwalteten gemeinsamen Objektmodell. Für die verschiedenen Analyse- und Simulationsmodule (Energie, Luftstrom, Klimatechnik, thermischer Komfort, Licht, Akustik usw.) wird aus diesem gemeinsamen Modell das entsprechende Domänenmodell mit Hilfe einer Homologie-basierten Abbildung abgeleitet.

3.5.2 Computational Collaboratories

Das Kunstwort *Collaboratory* setzt sich aus den Teilen *Collaboration* und *Laboratory* zusammen und beschreibt eine laborartige gemeinsam nutzbare Arbeitsumgebung, in der verschiedenste Experimente durchgeführte werden können. Im Fall von *Computational Collaboratories* werden diese Experimente weitestgehend simuliert, d.h. mit Hilfe numerischer Methoden berechnet. Die Labors stellen entsprechend Hochleistungsrechner und darauf laufende Simulations- bzw. Berechnungsprogramme bereit.

Eine frühe Arbeit in diese Richtung ist (Anupam *et al.*, 1994), in der **SHASTRA** vorgestellt wird, ein verteiltes System zur Unterstützung kooperativer Arbeit auf Basis sogenannter Problemlösungsumgebungen (engl. *Problem Solving Environment*). Im Mittelpunkt dieses Projekts steht kollaboratives geometrisches Design, zu dessen Unterstützung in der Umgebung verschiedene Freiformmodellierer zur Verfügung stehen. Zur Anbindung von Simulations- und Analysesoftware wurden Frontends für die Netzgenerierung und zur interaktiven Festlegung von Randbedingungen integriert. Ein Eingriff in eine laufende Simulation ist jedoch nicht möglich.

Eine der wichtigsten Entwicklungen im Bereich von *Computational Collaboratories* ist **DISCOVER**. Die Abkürzung steht für *Distributed Interactive Steering and Collabrative Visualization EnviRonment* und stellt Wissenschaftlern und Ingenieuren einen gemeinsam nutzbaren virtuellen Arbeitsplatz zur Verfügung, mit dessen Hilfe sie große parallele und verteilte Simulationsanwendungen kooperativ steuern können. Im Unterschied zu CoCoS bietet DISCOVER jedoch kein VR-Interface zur direkten dreidimensionalen Visualisierung von Rechenergebnissen und Interaktion mit geometrischen Randbedingungen, sondern lediglich Web-

basierte Portale mit deren Hilfe Simulationen gestartet und Simulationsparameter während der Laufzeit durch alphanumerische Eingabe geändert werden können.

In diesem Sinne ist zwar ist das *Distributed Laboratories*-Projekt (Plale *et al.*, 1999) enger verwandt mit CoCoS, da es ebenfalls die Unterstützung von Forschern und Ingenieuren hinsichtlich der Durchführung von virtuellen Experimenten zum Ziel hat. Das Projekt stellt jedoch ein Framework zur Verfügung, das Softwareentwicklern die Integration von Interaktions- und Kollaborationsfunktionalitäten in fachspezifische Simulationsanwendungen erlaubt. Die Plattform wurde unter anderem für das Steering einer Simulation des globalen Transports von chemischen Gemischen durch die Atmosphäre verwendet.

TENT, das von Forkert *et al.* entwickelt wurde, ist ebenfalls ein Komponentenbasiertes Framework für die Integration von verteilten technischen Applikationen (Forkert *et al.*, 2000). Es wurde als Simulationsumgebung für die Analyse von Turbokomponenten in Gasturbinen, für die Berechnung statischer Aeroelastik bei Flugzeugen und für die Simulation von Verbrennungskammern eingesetzt. TENT erlaubt jedoch keine interaktive Steuerung der Simulationen.

Auch das Forschungsprojekt *Collaborative Computing Frameworks* (**CCF**) von Sunderam *et al.* verfolgt die Entwicklung von Softwarekomponenten, Kommunikationsprotokollen und Werkzeugen zur Unterstützung computergestützter kooperativer Arbeit (Sunderam *et al.*, 1998). Auch hier steht die Idee eines *Collaboratory* als gemeinsam nutzbare virtuelle Arbeitsumgebung im Mittelpunkt. Das Anwendungsspektrum ist vergleichweise breit gefasst und reicht von gemeinsamer Dokumentenbearbeitung bis zu *Computational Transformation*. Ein Prototyp des Systems realisiert *Application Sharing*, also den gemeinsamen Zugriff mehrere Benutzer auf die Oberfläche einer Anwendung, und stellt eine Reihe von Telekooperationsfunktionen wie Whiteboard, Chat und Audio zur Verfügung.

3.5.3 Computational Steering

Seit Mulder *et al.* den Begriff des *Computational Steering* geprägt haben, wurde eine ganze Reihe von Forschungsprojekten durchgeführt, die das Konzept der interaktiv bedienbaren Simulation auf die unterschiedlichsten Gebiete anwenden.

Mulder *et al.* setzten ihre Vision von *Computational Steering* als Interaktion mit dreidimensionalen Körpern in einem CSE-System um, dessen wesentliches Element eine 3D-Nutzerschnittstelle ist, die sowohl zur Visualisierung von Simulationsdaten als auch zur Manipulation von Steering-Parametern dient (Mulder & van Wijk, 1995). Die wichtigste Simulationsparameter leiten sich dabei aus der Form und Lage von parametrisierten geometrischen Objekten ab. Die angeschlossen Simulationen haben vergleichsweise einfachen Charakter; beispielsweise wurde ein Applikation zur Bestimmung des Einpark-Pfades eines Fahrzeugs integriert. Die manipulierbaren Parameter sind dabei die Position und die Orientierung des Fahrzeugs, dargestellt wird der berechnete Pfad. Eine andere Applikation berechnet die Bewegung eines Partikels durch ein magnetisches Feld mit Hilfe der Lorentz-Gleichung. Der Nutzer kann das Partikel zu verschiedenen Aus-

gangspositionen verschieben und beobachten, wie es nach einigen Zeitschritten um die Attraktoren kreist. Schließlich werden in einer dritten Anwendung aufeinanderstoßende Kugeln simuliert. Hier können zur Laufzeit der Simulation die Dämpfung, die Zahl der Kugel und die Größe des Radius einer Kugel verändert werden. Außerdem können einzelne Kugeln vom Nutzer zu einer neuen Position bewegt werden.

Im Vergleich dazu ist das am Rechenzentrum der Universität Stuttgart entwickelte **CoViSE** (*Collaborative Visualization and Simulation Environment*) ein äußerst ausgereiftes und vielseitiges *Computational-Steering*-System. Es handelt sich hierbei um ein erweiterbares, verteiltes Softwarepaket, das der Integration von Höchstleistungsrechnen, Post-Processing, Visualisierung und kooperativer Arbeit in einer gemeinsamen Umgebung dient (Wierse, 1995).

Während CoViSE selbst lediglich eine zweidimensionale Schnittstelle für das visuelle Programmieren, d.h. zur visuellen Verknüpfung von Datenströmen mit Verarbeitungsmodulen bietet, erlaubt der integrierte VR-Renderer **COVER** die kollaborative Datenanalyse in einer immersiven Umgebung (Rantzau & Lang, 1998). COVER wurde u.a. für das Steering einer Fluid-Struktur-Simulation im Bereich der Turbinenoptimierung eingesetzt: Während die Simulation läuft, kann der Winkel der Turbinenblätter verändert und die Geschwindigkeit des Wasserstroms reguliert werden. Die Auswirkungen dieser Modifikationen auf den Wasserstrom sind direkt sichtbar. Bezüglich der kooperationsunterstützenden Eigenschaften des Systems ist vor allem die Visualisierung von Avataren als virtuelle Stellvertreter der entfernten Teilnehmer einer Session erwähnenswert. Mit Hilfe eines Tele-Pointers kann zudem in der gemeinsamen, immersiven Umgebung auf Objekte gedeutet werden. Die Integration von CoViSE in eine GRID-Umgebung wird in (Brooke *et al.*, 2003) präsentiert.

Die **gViz**-Bibliothek (Brodlie *et al.*, 2004) erlaubt ihren Nutzern, Daten während der Simulation oder im Post-Processing-Schritt zu visualisieren. In beiden Fällen wird die Verwendung von GRID-Ressourcen und kooperatives Arbeiten unterstützt. Wenn von der Simulation eine laufzeitdynamische Modifikation von Parametern angeboten wird, können diese mit Hilfe der gViz-Bibliothek zur Simulation gesendet werden. Allerdings sind nur einfache skalare Werte veränderbar.

CUMULVS (Collaborative, User Migration, User Library for Visualization and Steering) ist eine Middleware, die zur Realisierung von Visualisierung und Steering wissenschaftlicher Simulationen verwendet werden kann (Kohl *et al.*, 2006). Die während der laufenden Berechnung modifizierbaren Parameter beschränken sich jedoch auf einfache skalare bzw. vektorielle Größen. Geometrische Randbedingungen sind somit nicht veränderbar. CUMULVS verfügt darüber hinaus über eine Reihe von Bibliotheken, die der Unterstützung kooperativen Arbeitens dienen. Im Fall von Systemausfällen kann CUMULVS eine abgebrochene Simulation neu starten bzw. zum zuletzt gespeicherten Checkpoint zurückrollen. In (Wilde *et al.*, 2003) wird die Integration von immersiven Nutzungsschnittstellen auf Basis von VTK und AVS in CUMULVS beschrieben.

Das französische Projekt **EPSN** (*Computational Steering Environment for Distributed Numerical Simulations*) konzentriert sich wiederum verstärkt auf Visualisierungsapekte (Esnard *et al.*, 2006). Das entwickelte Framework erlaubt es, parallele Simulationen mit parallel arbeitenden Visualisierungssystemen zu verbinden. Dabei müssen vor allem Probleme der parallelen Daten-Redistribution und des zeitlich kohärenten Transfers berücksichtigt werden.

Die **RealityGrid**-Initative (Brooke *et al.*, 2003; Pickles *et al.*, 2005) hat sich zum Ziel gesetzt, durch integrierte numerische Simulationen die Entwicklung neuer Materialien zu vereinfachen sowie das Verständnis komplexer physikalische Systeme zu vertiefen. Ein zentraler Ansatzpunkt ist der vereinfachte Zugang verteilt arbeitender Forscherteams zu kollaborativ steuerbaren Simulationen, verbunden mit einer entsprechenden High-End-Visualisierung der Ergebnisdaten. Um ersteren zu gewährleisten, setzt das RealityGrid-Framework auf dem Standard *Open Grid Services Infrastructure* (OGSI) auf, der die weltweite Verwendbarkeit von Hochleistungsrechenressourcen vereinfacht. Mit Hilfe der RealityGrid-Infrastruktur wurde unter anderem ein parallelisierter Lattice-Boltzmann-Strömungssimulator um Steering-Funktionalitäten erweitert (Chin *et al.*, 2003). Zu den zur Laufzeit manipulierbaren Parametern gehören die Dichte des Fluids, verschiedene Kopplungskonstanten und die Relaxationszeit. Die Veränderung der Geometrie von Strömungshindernissen ist allerdings nicht möglich.

Das EU-Projekt **ViSiCADE** (Lee *et al.*, 2004) beabsichtigt, ein intuitiv bedienbares Rahmenwerk für Simulationen zu entwickeln, dass die Integration von CAD (Computer Aided Design) und CAE (Computer Aided Engineering) in einer virtuellen Umgebung ermöglicht, um dadurch den zeitlichen Aufwand bei der Durchführung von Simulationen und deren Auswertung zu reduzieren. Die VR-Umgebung soll es dabei dem Entwerfenden erlauben, ein CAD-Modell zu laden, daraus automatisiert und interaktiv ein Analysemodell abzuleiten (autom. Vernetzung) und dieses an die Bedürfnisse des Simulationswerkzeugs hinsichtlich der benötigten Verfeinerung anzupassen. Darüber hinaus sollen Möglichkeiten der Interaktion während der laufenden Simulation bereit gestellt werden. Besonderes Augenmerk wird dabei auf die Entwicklung neuer Visualisierungstechniken zur besseren Unterstützung des Entwerfenden während des Entwurfszyklus gelegt.

3.5.4 Interaktive Strömungssimulationen

Die Idee, interaktive Techniken auf Strömungssimulationen anzuwenden, wird bereits seit Anfang der 1990er Jahre verfolgt. Während sich jedoch ein Großteil der wissenschaftlichen Arbeiten mit der interaktiven *Visualisierung* der Simulationsergebnisse durch beispielsweise verschiebbare Schnittebenen, frei positionierbare Saatpunkte für Partikelverfolgung und anpassbare Isobarenflächen beschäftigt (Bryson & Levit, 1992) (Kreylos *et al.*, 2002), hat sich bislang nur eine kleine Zahl mit der Anwendung des *Computational Steering*-Paradigmas auf die Strömungssimulation selbst befasst, also mit der Entwicklung von Strömungssimulationen, bei denen Randbedingungen während der Berechnung verändert werden können.

Zu diesen wenigen Arbeiten zählt zum Beispiel **OViD** (*Online Visualization and Computational Steering of parallel and Distributed scientific applications*), eine von Luksch *et al.* entwickelte Plattform (Luksch *et al.*, 2000). Das OViD-Projekt verfolgte das Ziel, bestehende Simulationsanwendungen und Visualisierungswerkzeuge auf einfache Weise in eine kooperative Umgebung zu integrieren. Im Quellcode der Simulationsanwendung müssen dazu die Objekte definiert werden, die visualisiert werden sollen. Außerdem sind Stellen im Programmtext festzulegen, an denen sich diese Objekte in einem konsistenten Zustand befinden, also visualisiert werden können. Das System besteht aus einem zentralen Server, der Schnittstellen für parallelisierte Simulationsprogramme und für verschiedene Visualisierungswerkzeuge zur Verfügung stellt. Auf diese Weise wirkt OViD als Vermittlungsschicht zwischen der Simulation und einer beliebigen Zahl von Visualisierungs- und Interaktionsclients. Zur Illustration wurde ein parallelisierter Strömungssimulator mit dieser Architektur verbunden. Leider lassen Luksch *et al.* jedoch offen, welche Arten von Interaktion mit den Simulationen von OViD unterstützt werden.

Basierend auf der bereits erwähnten CoVISE-Umgebung wurde im Rahmen des Projekts **VISiT** (*Virtual Intuitive Simulation Testbed*) ein *Computational Steering*-System entwickelt, das kommerzielle Gittergeneratoren und kommerzielle Strömungssimulatoren mit einer VR-Schnittstelle integriert (Klimetzek, 2001). Hier hat der Anwender die Möglichkeit, Randbedingungen der Simulation inklusive der Geometrie von Strömungshindernissen zu manipulieren. Im Unterschied zu dem in dieser Arbeit vorgestellten Simulationsverfahren, das auf der Lattice-Boltzmann-Methode beruht, werden in VISiT klassische Navier-Stokes-Löser mit blockstrukturierten Finite-Volumen-Netzen verwendet.

Eine erste Umsetzung der Idee, *Computational Steering* für Strömungssimulationen im Sinne einer Manipulation der Geometrie und Position von Strömungshindernissen aufzufassen, wurde in (Kühner, 2003) demonstriert. Die Arbeit stellt ein Konzept zur Realisierung eines virtuellen Windkanals durch Verbindung von Hochleistungsrechnern auf Simulationsseite mit VR-Systemen auf der Nutzerseite vor.

Dieser Ansatz wird in (Wenisch, 2008) fortgeführt und weiter vertieft. Die Arbeit beschäftigt sich unter anderem mit Techniken der schnellen Überführung von facettierter Ausgangsgeometrie in das numerische Berechnungsgitter, effizienten Parallelisierungsstrategien sowie der Integration von Turbulenzmodellen in den Strömungslöser. Sie bildet die Grundlage des hier vorzustellenden, um Aspekte der Mehrbenutzerfähigkeit erweiterten *Computational Steering*-Systems.

In (Fahrig *et al.*, 2006) wird die Idee der interaktiv veränderbaren Randbedingungen ebenfalls verfolgt und ein System entwickelt, mit dessen Hilfe Komfortwerte in einem Büroraum optimiert werden können. Ähnlich zu dem hier vorgestellten System können zur Laufzeit der Simulation Strömungshindernisse bewegt und verschiedene Parameter der Klimainstallation verändert werden. Zusätzlich wurden agentenbasierte Methoden integriert, um auf Grundlage der Simulationsergebnisse den Kontrollkreis der Klimainstallation zu optimieren. Kooperatives

Arbeiten wird bei diesem Ansatz jedoch nur rudimentär durch einfaches *Desktop Sharing* unterstützt.

In (Boulanger *et al.*, 2006) wird ein vergleichsweise junges Projekt vorgestellt, das sich zum Ziel gesetzt hat, große Strömungssimulationen mit Hilfe einer GRID-Infastruktur für eine Gruppe verteilt arbeitender Experten interaktiv bedienbar zu machen. Allerdings liegt auch hier der Schwerpunkt zunächst auf der Visualisierung von Ergebnisdaten zur Laufzeit der Simulation. Laufzeitdynamische Änderungen an der Geometrie von Strömungshindernissen sind vorerst nicht vorgesehen.

3.5.5 Techniken der Virtuellen Realität im Kontext von digitalem Engineering

Das Ziel des Einsatzes von Systemen der Virtuellen Realität (VR) ist das Generieren einer scheinbar echten Welt für den Nutzer. Ein wesentlicher Aspekt ist hierbei das Erzeugen eines dreidimensionalen Eindrucks von dieser künstlichen Welt. Hierzu existieren verschiedene Techniken, die auf dem Prinzip der Stereoskopie beruhen, also dem Eintreffen unterschiedlicher Bilder auf dem linken und rechten Auge des Betrachters. Aus der Größe der Verschiebung zwischen den beiden Abbildern eines beobachteten Objekts auf den Netzhäuten leitet der Betrachter den Abstand dieses Objekts von seinem Standpunkt ab, womit sich ein dreidimensionaler Eindruck einstellt. Die Simulation dieses Effekts kann mit Hilfe verschiedener Filtertechniken (chromatisch, mittels Polarisation des Lichts) oder durch direkte Projektion verschiedener Bilder auf die einzelnen Augen (zum Beispiel mit Hilfe eines *Head Mounted Display*) erfolgen.

Mit dem Begriff der *Immersivität* wird der Grad des Gelingens der Sinnestäuschung umschrieben, also die vom Nutzer subjektiv wahrgenommene Echtheit der dargestellten Welt. Um die Immersivität weiter zu steigern, kann die Stereoskopie mit zusätzlichen Techniken kombiniert werden, wie beispielsweise Tracking (Verfolgung der Kopfbewegungen), räumliches Hören und virtuelle Haptik mit Hilfe eines Datenhandschuhs. Näheres hierzu ist den Standardwerken (Stannkey, 2002) und (Sherman & Craig, 2002) sowie der Arbeit von Kühner (Kühner, 2003) zu entnehmen.

Das Potential der Verwendung von VR-Techniken zur besseren Darstellung dreidimensionaler Simulationsergebnisse wurde bereits früh erkannt. Besonders interessant hierbei ist die Möglichkeit der Exploration durch den Nutzer, also dem interaktiven Verändern sowohl des Betrachtungsortes als auch der dargestellten physikalischen Größen (Haase *et al.*, 1997).

Bryson wendete diese Idee als erster auf die Visualisierung der Ergebnisse einer Strömungssimulation an (Bryson & Levit, 1992). Er propagierte einen *Virtual Windtunnel*, in dem die Ergebnisse der Simulation in einem Virtuellen Raum dargestellt werden. Daneben gibt es eine ganze Reihe Projekte, die dieses Prinzip auf ähnliche Weise umsetzten, darunter die bereits erwähnte COVER-Umgebung.

Unter dem Schlagwort „Digitale Produktentwicklung“ wird darüber hinaus seit kurzem verstärkt untersucht, inwieweit VR-Systeme nicht mehr nur für die reine Visualisierung von Geometrie und Simulationsdaten, sondern für einen erweiterten Bereich von Engineering-Aufgaben nutzbar gemacht werden können. Eine wesentliche Anwendung besteht hierbei in Einbauuntersuchungen, dem sogenannten *Digital Mock-Up* (DMU), das vorzugsweise im Automobilbau zum Einsatz kommt (Döllner *et al.*, 2000).

Guddat und Straube stellen den Entwurf eines Systems vor, mit dessen Hilfe spontane, VR-basierte Telekonferenzen einberufen werden können, in denen große Nutzergruppen in verteilten virtuellen Umgebungen an verschiedenen Engineering-Problemstellungen arbeiten können (Guddat & Straube, 2003). Dazu wurden Techniken und Arbeitsumgebungen der Virtuellen Realität und der immersiven Telepräsenz eingesetzt, um physikalisch basierte Echtzeit-Simulationen ergänzt und in eine verteilte Konferenz eingebettet.

Im Bauwesen konnte sich der Einsatz von VR-Techniken bislang lediglich bei der Visualisierung des Endprodukts, zum Beispiel zum Zweck der Präsentation für potentielle Bauherren, durchsetzen. Im Planungsprozess finden VR-Techniken dagegen bislang kaum Anwendung. Nichtsdestotrotz gibt es einige Forschungsarbeiten in diesem Bereich, darunter das Projekt **DIVERCITY** (Christiansson *et al.*, 2002), das kollaborative virtuelle Workspaces mit Bauprozessapplikationen verbindet. Dabei konzentriert sich das Anwendungsgebiet auf die drei typischen Projektaktivitäten *Kundenberatung*, *Entwurfsbesprechung* und *Konstruktionsplanung*.

3.5.6 Fazit

Zwar gibt es eine Reihe ausgereifter Ansätze zur Nutzung von VR-Techniken für die Auswertung von Simulationsergebnissen und zur Verknüpfung von VR mit kooperativem Arbeiten sowie erste Ergebnisse bei den Bemühungen um hochwertige interaktive Simulationen. Doch gerade die Einbindung von interaktiv bedienbaren Simulationen in kollaborative *Virtual Reality*-Systeme ist ein bislang wenig erforschtes Gebiet. Diese Arbeit soll daher einen Beitrag dazu leisten, diese technologische Lücke zu schließen.

Kapitel 4

Die CoCoS-Plattform

4.1 Überblick

Die CoCoS-Plattform wurde als verteilte Mehrbenutzerapplikation für das Steering von interaktiven Simulationen entworfen. Jeder Teilnehmer einer kollaborativen Session kann diese Applikation über eine individuell konfigurierbare Schnittstelle bedienen. Gegenüber einfacheren *Shared Desktop* oder *Shared Application*-Systemen hat dieser Ansatz zwei wesentliche Vorteile: Auf der einen Seite ist die verwendete Visualisierungs- und Eingabehardware unabhängig voneinander und kann dementsprechend von einfachen Schreibtischarbeitsplätzen mit Monitor und herkömmlicher Maus-Tastatur-Bedienung bis zu immersiven Visualisierungsumgebungen mit stereoskopischer Projektion und Interaktion mittels Datenhandschuh reichen (Abb. 4.1).

Auf der anderen Seite kann jeder Teilnehmer die Modellansicht, d.h. die Beobachtungsposition und -richtung sowie die Vergrößerung des betrachteten Ausschnitts, unabhängig von den anderen wählen. Der damit verbundene Gewinn an individueller Freiheit ist für bestimmte Phasen des kooperativen Arbeitens unabdingbar,

Abbildung 4.1: Verteilte Nutzer der *Collaborative Computational Steering*-Plattform CoCoS, hier bei der Steuerung einer interaktiven Strömungssimulation

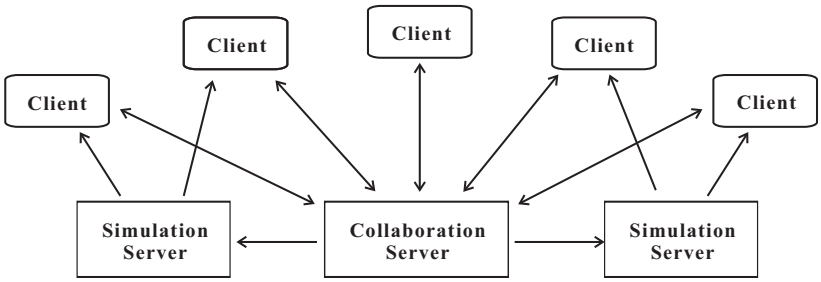


Abbildung 4.2: Die Architektur der CoCoS-Plattform sieht einen zentralen Kollaborationsserver und eine beliebige Anzahl von Simulationsservern und Clients vor. Die verschiedenen Clients können von unterschiedlichen Simulationsservern mit Daten versorgt werden bzw. überhaupt keine Simulationsdaten empfangen. Sowohl Clients als auch Simulationsserver können dynamisch dem verteilten System beitreten und es wieder verlassen.

da auf diese Weise ein zumindest partiell unabhängiges Arbeiten möglich ist. Dies ist vor allem hinsichtlich der für eine fundierte Simulationsauswertung notwendigen eigenständigen Informationsbeschaffung von Bedeutung. Des Weiteren können typische Phänomene, die bei der Nutzung von *Shared Desktop* oder *Shared Application*-Systemen auftreten, wie „Mauskriege“ oder Übelkeit infolge eines fremd gesteuerten Bildschirmausschnitts, vermieden werden.

Die grundlegende Architektur des vorgeschlagenen Systems besteht aus einem zentralen *Kollaborationsserver*, einer beliebigen Anzahl von *Simulationsservern* und einer ebenfalls beliebigen Anzahl von *Visualisierungs- und Interaktionsclients*. Abb. 4.2 zeigt die Systemkomponenten und die Kommunikationspfade zwischen ihnen. Jede der Komponenten läuft in der Regel auf einem separaten Rechner.

Die Modularität des verteilten Systems erlaubt ein hohes Maß an Flexibilität: Client-Programme können einer kooperativen Session jederzeit beitreten bzw. sie verlassen. Das Gleiche gilt für die Simulationsserver: Sie können in das verteilte System integriert werden, wenn eine bestimmte Simulationsanwendung benötigt wird, und wieder freigegeben werden, wenn ihre Nutzung beendet ist. Im Extremfall können Ingenieure das System für die kollaborative, geometriegestütze Zusammenarbeit auch ohne Einbindung von Simulationswerkzeugen nutzen, zum Beispiel in frühen Phasen der Kooperation, die lediglich der Grobabstimmung dienen.

Darüber hinaus erlaubt die Struktur des Systems auch, dass verschiedene Client-Applikationen Simulationsdaten von verschiedenen Simulationsservern erhalten bzw. einzelne Client-Applikationen überhaupt keine Simulationsdaten empfangen. Letzterer Fall kann zum Beispiel bei Einbeziehung eines Konstruktionsingenieurs auftreten, der an der kooperativen Session mittels einer CAD-Anwendung teilnimmt.

Für die Kommunikation zwischen den Komponenten des verteilten Systems wird eine auf dem CORBA-Standard beruhende Middleware eingesetzt. CORBA steht

für *Common Object Request Broker Architecture* und ermöglicht den Datenaustausch nach dem objektorientierten Paradigma, d.h. mittels Aufruf von Methoden verteilt vorliegender Objekte (Object Management Group, 2004; Henning & Vinoski, 1999). Dazu gehört neben der Einführung nutzerdefinierter Typen (Schnittstellen) auch das Werfen und Fangen von Exceptions. Durch den Einsatz von CORBA wird die Komplexität der verteilten Systementwicklung stark reduziert, da die Übertragung stark strukturierter Informationen vereinfacht wird und durch die eingeführte strenge Typifizierung Fehler bereits zum Zeitpunkt der Kompilierung erkannt werden.

Wesentliche Vorteile des Einsatzes einer CORBA-Middleware gegenüber dem im Bereich des *Scientific Computing* weit verbreiteten MPI-Standard (s.a. Abschnitt 5.4) sind die Verwendbarkeit semantisch höherwertiger nutzerdefinierter Schnittstellen und die damit verbundene gesteigerte Robustheit und einfachere Wartbarkeit des Gesamtsystems sowie eine deutlich höhere Flexibilität hinsichtlich des Startens und Beendens einzelner Prozesse innerhalb des verteilten Systems. Da MPI vor allem für paralleles Rechnen entwickelt wurde, sieht der Standard vor, dass alle Prozesse zur gleichen Zeit gestartet und beendet werden. Bei der hier beschriebenen Kollaborationsplattform ist es jedoch sinnvoll und wünschenswert, dass die einzelnen Teilnehmer (und damit auch die von ihnen gesteuerten Prozesse) dynamisch der kollaborativen Session beitreten und sie wieder verlassen können.

Weitere Vorteile der Verwendung von CORBA sind die Behandlung nebenläufiger Zugriffe mehrerer Clients auf eine gemeinsam genutzte Ressource sowie die Interoperabilität CORBA-basierter Systeme auf heterogenen Hardwareplattformen und zwischen verschiedenen Programmiersprachen.

Zwar führt die Verwendung von CORBA durch die zusätzlich notwendigen Konvertierungsschritte (auch als *Marshalling* und *De-Marshalling* bezeichnet) zu einem größeren Kommunikationsoverhead und infolgedessen zu einer gesteigerten Latenz bei der Datenübertragung, jedoch konnten Untersuchungen zeigen, dass der mittels CORBA erzielbare Datendurchsatz vergleichbar mit dem einer MPI-basierten Implementation ist (Denis *et al.*, 2003; Kopyssov *et al.*, 2003). Gerade mit Hinblick auf die anfallenden Datenmengen kann der Datendurchsatz als der wesentlichere Aspekt bei der Bewertung der für *Collaborative Computational Steering* einzusetzenden Kommunikationstechnologie betrachtet werden.

CORBA findet Verwendung in einer Vielzahl von verwandten Forschungsarbeiten, beginnend bei verschiedenen *Collaborative Virtual Enviroments* (Picard *et al.*, 2001; Wilson *et al.*, 2001; Linebarger *et al.*, 2003) über das *Distributed Engineering Framework* TENT (Forkert *et al.*, 2000) und die Gebäudesimulationsplattform S2 (Mahdavi *et al.*, 1999) bis zum Einsatz im Rahmen von *Distributed Computational Steering* (Muralidhar & Parashar, 2000), *Parallelem Rechnen* (Keahey & Gannon, 1997; Denis *et al.*, 2003; Kopyssov *et al.*, 2003) und in Kombination mit der GRID-Technologie (Priol, 2002).

4.2 Das gemeinsam verwendete Modell

Das gemeinsam verwendete Modell ist eines der wesentlichen Merkmale der CoCoS-Plattform. Es wird vom zentralen Kollaborationsserver verwaltet und reflektiert den aktuellen Zustand der geometrischen und nicht-geometrischen Randbedingungen sowie Start- und Steuerungsparameter der angeschlossenen Simulationen.

Wie in Abschnitt 2.2.3 besprochen, wird in der modernen Datenaustauschtheorie und -praxis in zunehmendem Maße die Technik der Produktmodellierung zur Strukturierung der gemeinsam genutzten Informationmenge eingesetzt (Eastman, 1999; Hartmann, 2000). Im Bereich des Bauwesens werden derartige Produktmodelle auch als Bauwerksinformationsmodelle (BIM) bezeichnet. Ein BIM modelliert das Bauwerks nach dem *objektorientierten* Paradigma, also mittels semantischer Entitäten wie *Wand*, *Fenster*, *Decke* und deren Beziehungen untereinander wie *eine Wand hat einen Durchbruch*, *der Durchbruch wird von einem Fenster gefüllt* (IAI, 2004).

Da ein BIM überwiegend durch die semantische Modellierung geprägt ist, wird die Geometrie der Bauteile aus semantischen Attributen des betreffenden Objekts (wie *Höhe, Länge, Dicke*) generiert. Dieses Konzept der *„attribute-driven geometry“* bietet eine Reihe von Vorteilen, darunter die Möglichkeit der Generierung unterschiedlicher geometrischer Repräsentationen physischer Objekte, wie sie beispielsweise für 2D-Pläne und virtuelle 3D-Modelle benötigt werden. Dies ist vor allem deshalb notwendig, da nach wie vor große Teile der Bauindustrie ausschließlich mit 2D-Plänen arbeiten und diese zur rechtlich verbindlichen Dokumentation einsetzen.

Als Grundlage moderner Simulations- und Berechnungsanwendungen muss hingegen ein präzises dreidimensionales Modell der Gebäudegeometrie zum Einsatz kommen. Zu solchen Anwendungen zählen 3D-Struktur-Berechnungen (Romberg *et al.*, 2004), 3D-Strömungssimulationen (Kühner *et al.*, 2004), 3D-Fluid-Struktur-Simulationen (Scholz *et al.*, 2004) sowie Tools für die fotorealistische Visualisierung und zur Beleuchtungsanalyse. Die für diese Applikationen herausragende Bedeutung geometrischer Informationen ist der wesentliche Grund dafür, im Rahmen dieser Arbeit einen Ansatz zur geometrischen Modellierung zu wählen, der sich von jenem der Produktmodellierung unterscheidet: In CoCoS wird der Kern des gemeinsam genutzten Modells durch eine *explizite* dreidimensionale geometrische Beschreibung der physischen Objekte geformt. Dieser Ansatz wird auch als *volumenorientierte Modellierung* bezeichnet (Niggl, 2007).

Im Bauwesen ist die geometrische Beschreibung physischer Objekte einer der wichtigsten Aspekte einer computerinternen Modellierung. Die geometrische Gestaltung bestimmt den architektonischen, aber auch den ingenieurtechnischen Entwurfsprozess; Funktionalität, Nutzbarkeit und ästhetische Erscheinung eines Gebäudes werden durch seine Gestalt bzw. die Form seiner Bauteile bestimmt. Für einen großen Teil der Simulationsaufgaben ist die Geometrie ein wesentlicher Ausgangspunkt für die Definition von Rändern bzw. Randbedingungen.

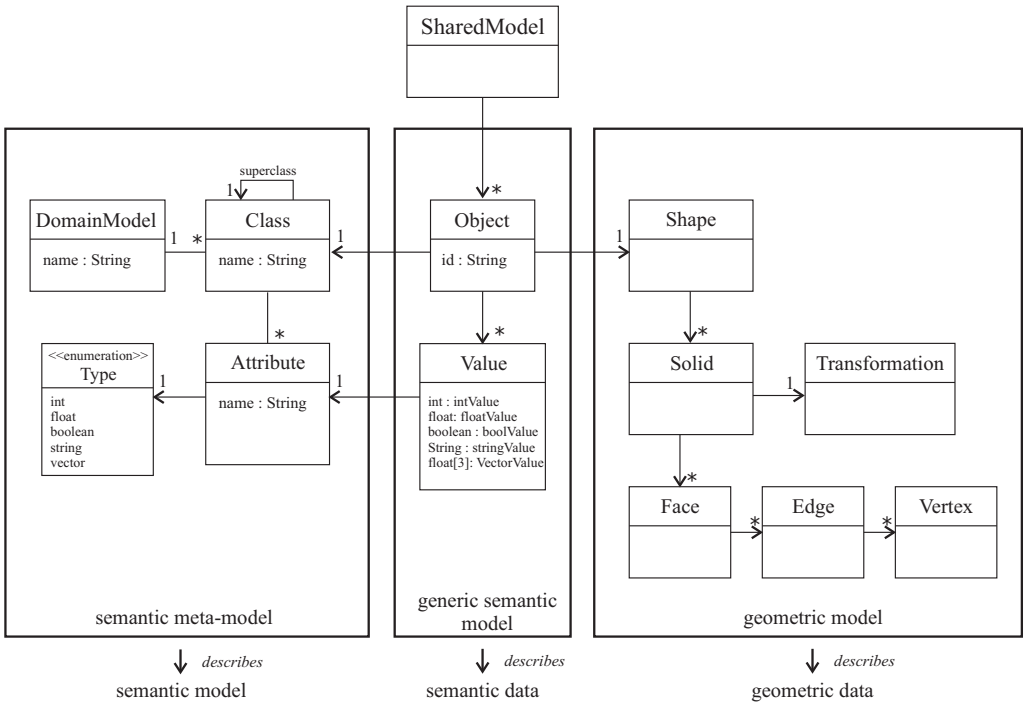


Abbildung 4.3: Das zentral verwaltete Modell vereinigt geometrische und semantische Informationen. Das explizit verfügbare Metamodell auf der linken Seite beschreibt die Struktur des semantischen Modells einer Domäne (Domänenmodell). Da CoCoS für die Unterstützung interdisziplinärer Zusammenarbeit konzipiert ist, können mehrere Domänenmodelle vorgehalten werden. Die Instanz eines Domänenmodells trägt die semantischen Daten und wird mit Hilfe der generischen Klassen in der Mitte modelliert. Die Geometrie und die Position eines Objekts wird durch Instanzen der Klassen auf der rechten Seite abgebildet.

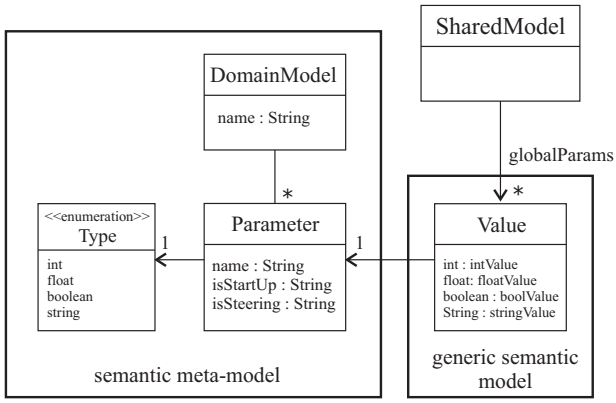


Abbildung 4.4: Auch globale Parameter einer Simulationsanwendung, die nicht an ein bestimmtes geometrisches Objekt geknüpft sind, werden im zentralen Modell vorgehalten. Sie werden ebenso wie die semantischen Attribute geometrischer Module mit Hilfe des Metamodells auf der linken Seite beschrieben.

Während nicht erwartet werden kann, dass in naher Zukunft semantische Produktmodelle für alle denkbaren Simulationsanwendungen entwickelt und standardisiert werden, existiert bereits eine überschaubare Zahl ausgereifter expliziter Geometriemodelle, darunter die *Constructive Solid Geometry* (CSG), die auf der Verknüpfung von einfachen Ausgangskörpern durch boolesche Operationen beruht, und verschiedene *Boundary Representation*-Modelle (BRep), die einen Körper anhand seiner Oberfläche beschreiben (s.a. Abschnitt 9.3.4) (Mortenson, 1985; Mäntylä, 1988).

Neben der geometrischen Beschreibung gibt es jedoch immer auch eine Reihe nicht-geometrischer Daten, die für eine anzuschließende Simulation von Bedeutung sind und damit zu der von den Teilnehmern der kooperativen Sitzung gemeinsam genutzten Informationsmenge gehören. Ein reines Geometriemodell kann derartige Daten nicht aufnehmen. Aus diesem Grund wurde sich bei dem hier vorzustellenden Ansatz für ein hybrides Modell entschieden, dessen Kern zwar von einem geometrischen Modell gebildet wird, das aber um semantische Aspekte nach dem objektorientierten Paradigma erweitert werden kann. Der geometrische Kern formt innerhalb der gesamten CoCoS-Plattform die kleinste gemeinsame Informationsmenge, die von allen angeschlossenen Clients und Simulationsservern interpretiert werden kann.

Die Geometrie wird im gemeinsamen Modell mit Hilfe einer klassischen BRep-Datenstruktur auf Basis eines *vef*-Graphen[1] beschrieben (s.a. Abschnitt 9.3.4), wie sie in Abb. 4.3 auf der rechten Seite dargestellt ist. Um nicht-geometrische Informationen mit einem geometrischen Objekt zu verknüpfen, kann letzterem eine Klasse der Anwendungsdomäne zugewiesen werden, die die entsprechenden Attribute zur Aufnahme nicht-geometrischer Daten aufweist.

[1] vertex – edge – face

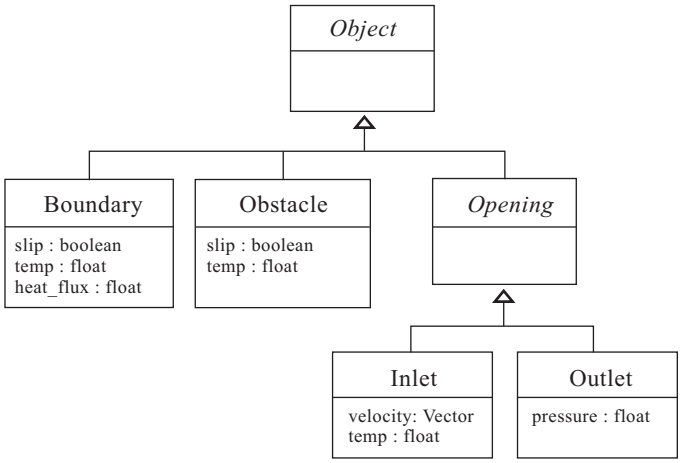


Abbildung 4.5: Das Domänenmodel *CFD*, das von dem in Kapitel 5 vorgestellten Strömungssimulator verwendet wird.

Jeder Teilnehmer wählt mit seiner Zugehörigkeit zu einer Fachdisziplin bzw. durch Angabe der von ihm verwendeten Simulationsressource das für ihn verfügbare Domänenmodell[2]. Als Beispiel ist in Abb. 4.5 das Domänenmodell des in Kapitel 5 besprochenen Strömungssimulators dargestellt.

Alle Teilnehmer einer Domäne bzw. alle Nutzer derselben Simulationsressource verwenden das gleiche Domänenmodell und teilen die semantische Sicht auf ein geometrisches Objekt, also die ihm zugewiesene Klasse sowie die Attributbelegung. Teilnehmer einer anderen Fachdisziplin werden demselben geometrischen Objekt jedoch eine Klasse *ihres* Domänenmodells zuweisen. Einem tischförmigen geometrischen Objekt kann beispielsweise aus einem innenarchitektonischen Domänenmodell die Klasse *Tisch* und aus einem strömungsmechanischen Domänenmodell die Klasse *Hindernis* zugewiesen werden (Abb. 4.6). Aus der Sicht des zentralen Modells kann ein Objekt somit mit mehrere Klassen aus unterschiedlichen Domänen verknüpft sein.

Um die Flexibilität des Systems hinsichtlich der zu verwendenden Domänenmodelle zu erhöhen, wurde ein Metamodell in das zentral verwaltete gemeinsame Modell integriert. Mit Hilfe des Metamodells ist es möglich, das verwendete Domänenmodell flexibel an die Bedürfnisse der jeweiligen Simulationsanwendung anzupassen. Dies hat wesentliche Vorteile vor allem im Hinblick auf eine kontinuierliche Weiterentwicklung der Simulatoren und der damit einhergehenden Änderungen an der Parameterstruktur: Sollten bei einer Weiterentwicklung zusätzliche Parameter eingeführt bzw. die Struktur der bereits vorhandenen Parameter verändert werden, können diese dynamisch, d.h. ohne erneute Kompilierung, im Kollaborationsserver und den Clients zur Verfügung gestellt werden.

Ähnliches gilt für a priori nicht bekannte Simulationstypen, die dem allgemeinen Konzept einer Steering-Applikation mit geometriebezogenen Randbedingungen

[2]Der Begriff *Domänenmodell* beschreibt das innerhalb einer Fachdisziplin verwendete Objektmodell. Eine ausführliche Definition ist beispielsweise (Willenbacher, 2002) zu entnehmen.

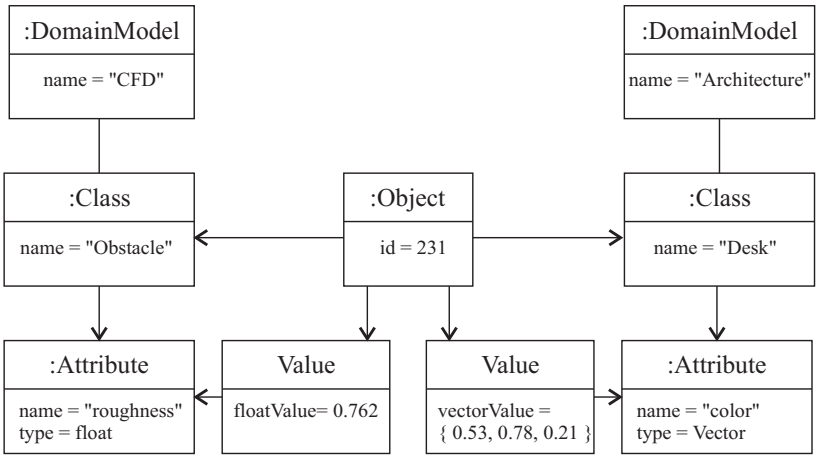


Abbildung 4.6: Einem Objekt können mehere Klassen aus unterschiedlichen Domänenmodellen zugewiesen werden. Das Beispiel zeigt Objekt 231, dem im Domänenmodell *CFD* die Klasse *Obstacle* zugewiesen wird und im Domänenmodell *Architecture* die Klasse *Desk*. Alle Objekte im Diagramm sind Instanzen der in Abb. 4.3 gezeigten Klassen. Die Domänenmodelle sind nur ausschnittsweise dargestellt.

genügen. Auch sie können dank des Metamodells in die Plattform integriert werden, ohne Code-Anpassungen an den CoCoS-Komponenten vornehmen zu müssen. Sobald eine Simulationsanwendung für die CoCoS-Plattform zur Verfügung steht, legt sie mit Hilfe des Metamodells das zu verwendende Modell der Randbedingungen und damit ihr Domänenmodell fest. Abb. 4.5 zeigt somit eine Instanz des Metamodells, die als solche vom CFD-Simulationsserver an den Kollaborationsserver übermittelt wird.

Häufig werden derartige Erweiterungsmechanismen durch einfache Name-Wert-Listen realisiert, wie sie beispielsweise die *Property Sets* der IFC darstellen (Neuberg, 2003). Die Probleme dieses Ansatzes liegen jedoch vor allem in der geringen Strukturierbarkeit der Informationsmenge (eine Kapselung mehrerer Datensätze ist nicht möglich) und der fehlenden Typisierung (in der Regel werden alle Daten als Zeichenkette abgelegt). Die mit der Verwendung eines (dynamisch anpassbaren) Objektmodells einhergehende strenge Typisierung erlaubt hingegen, dass hinsichtlich des Typs fehlerhafte Eingaben als solche erkannt werden.

Darüber hinaus ermöglicht der explizite Zugang zum Metamodell den Austausch von Metainformationen (Klassenname, Attributname, Attributtyp) zwischen den Komponenten der CoCoS-Plattform. Auf diese Weise kann vom Simulationsserver an den zentralen Kollaborationsserver Wissen über die im Simulationsmodell vorhandenen nicht-geometrischen Parameter weitergegeben werden. Zum Austausch tatsächlicher Instanzdaten, also simulationsrelevanter Eigenschaften konkreter physischer Objekte, dient eine generische Schnittstelle, die in Abb. 4.3 in der Mitte dargestellt ist. Mit ihrer Hilfe können die angeschlossenen Clients die nicht-geometrischen Daten eines Körpers abfragen und manipulieren. In den meisten Fällen werden sie dem Nutzer in einer zweispaltigen Tabelle präsentiert, mit dem Namen des Attributs in der linken und seiner aktuellen Wertbelegung in

der rechten Spalte (Abb. 4.12). Im Unterschied zu dem von Hübler *et al.* verfolgten Ansatz (siehe Abschnitt 2.2.3) dient das Metamodell hier nicht zur laufzeitdynamischen Modifikation des Domänenmodells, sondern lediglich der initialen Definition.

4.3 Eine adaptierte Client-Server-Architektur

Beim Entwurf eines verteilten Systems gibt es zwei grundsätzliche Möglichkeiten der systemarchitektonischen Gestaltung (Tanenbaum & van Stean, 2002). Die erste Möglichkeit sieht eine zentrale Komponente mit dezidierten Verwaltungsaufgaben vor, den sogenannten Server, und eine Reihe von Anwendungsprogrammen, sogenannten Clients, die von verschiedenen physisch getrennten Orten auf den zentralen Server zugreifen. Dabei kommunizieren die Clients nicht untereinander. Die Bezeichnungen *Client* und *Server* leiten sich von den unterschiedlichen Rollen der Systemkomponenten ab: Der Server bietet eine Dienstleistung an, die der Client in Anspruch nimmt. Diese Architektur wird entsprechend als Client-Server-Architektur bezeichnet.

Die zweite Möglichkeit verwendet keine zentrale Komponente, sondern sieht die direkte Kommunikation der Endanwendungsprogramme untereinander vor. Diese Architektur wird auch als Peer-to-Peer-Architektur bezeichnet. Dabei sind aus Sicht des verteilten Systems alle Komponenten identisch, d.h. keine von ihnen übernimmt dezidierte Aufgaben.

Bei der Umsetzung von CoCoS wurde sich für eine Client-Server-Architektur entschieden. Zwar gelten Client-Server-Architekturen durch ihre Abhängigkeit von der zentralen Komponente als anfälliger für Systemausfälle (Borghoff & Schlichter, 2000). Im vorliegenden Fall ist jedoch auch der Ausfall eines einzigen Clients (bzw. Peers) als systemkritisch anzusehen, da dies die Teilnahme eines der kooperierenden Partner an der gemeinsamen Sitzung verhindert. Die Umsetzung als Peer-to-Peer-System würde also nur unwesentlich zur Stabilität des Gesamtsystems beitragen. Auch die häufig gegen Client-Server-Architekturen ins Feld geführten Ressourcenprobleme (Server als „Flaschenhals“) treten bei CoCoS so nicht auf, da die Anzahl der Teilnehmer in der Regel vier oder fünf nicht übersteigt und die Modifikationshäufigkeit bei unter einer Modifikation pro Sekunde liegen wird. Gleichzeitig liegt ein wesentlicher Vorteil von Client-Server-Architekturen darin, dass die für die Umsetzung eines verteilten Mehrbenutzersystems essentiellen Techniken wie Konsistenzsicherung und Nebenläufigkeitskontrolle deutlich einfacher und robuster implementierbar sind (Singhal & Zyda, 1999).

Obwohl die Architektur von CoCoS also im Wesentlichen dem Client-Server-Muster entspricht, mussten an verschiedenen Punkten bestimmte Prinzipien dieses Paradigmas aufgegeben werden, um eine geeignete Form der Implementierung für die kollaborative Simulationsplattform zu finden. So besteht eine der Charakteristiken des Client-Server-Musters darin, dass der Server die Identität der Clients nicht kennt (Henning & Vinoski, 1999). Dies hat den Nachteil, dass auf Serverseite weder festgestellt werden kann, welcher Client einen Anforderung

sendet, noch ob ein Client abgestürzt oder eine Verbindung unterbrochen ist. Da jedoch im vorliegenden Fall eine Modifikationsaktion eindeutig dem ausführenden Nutzer und damit einem Clientprozess zugewiesen werden muss, ist dieses Prinzip der clientseitigen Anonymität für die Umsetzung von CoCoS nicht haltbar. Um dieses Prinzip zu umgehen und die Clients identifizierbar zu machen, weist der Server jedem Clientprozess einen eindeutigen Identifikator (ID) zu. Diese ID wird bei jedem Aufruf des Clients an den Server übermittelt.

Des Weiteren ist es entgegen dem reinen Client-Server-Modell notwendig, dass Clients selbst Nachrichten empfangen können. Diese Benachrichtigungen betreffen u.a. Modifikationen am zentralen Modell, Sperren, die eingerichtet und entfernt werden, sowie Nutzer, die der Konferenz beitreten und sie verlassen. Dies kann dadurch realisiert werden, dass Clients selbst eine aufrufbare Schnittstelle zur Verfügung stellen, die vom Serverprozess angesteuert wird. Aus Sicht der Interprozesskommunikation tauschen Client und Server in diesem Fall ihre Funktion bzw. nehmen, über einen längeren Zeitraum betrachtet, beide Rollen ein.

Um Zweideutigkeiten zu vermeiden, sollen daher die Begriffe *Client* und *Server* im Kontext der weiteren Betrachtungen als grobgranulare Beschreibung der Funktion eines Moduls auf Dienstleistungsebene verwendet werden: Simulationsserver und Kollaborationsserver bieten Dienste an und Clients nehmen diese in Anspruch.

Eine andere Möglichkeit der Umsetzung von Benachrichtigungen ist der Einsatz eines sogenannten *Event Service* (Abb. 4.7). Dabei handelt es sich um einen dritten Prozess, der Benachrichtigungen vom Server entgegennimmt und selbständig an die Clients verteilt. Zwar kann durch den Einsatz eines Event Service die Implementierung einer stabilen und gut skalierenden Notifikationslösung vermieden werden, jedoch birgt er einige Nachteile. Zum einen wird durch die Einbeziehung eines weiteren Prozesses der Kommunikations-Overhead vergrößert, was zu höheren Latenzzeiten führt. Zum anderen hat der Server durch die eingeführte Anonymisierung keine Möglichkeit mehr, einzelne Empfänger für eine Benachrichtigung auszuwählen.

4.4 Nebenläufigkeitskontrolle

Die Nebenläufigkeitskontrolle hat zur Aufgabe, simultane Zugriffe parallel laufender Programme auf eine gemeinsam genutzte Ressource, wie beispielsweise das zentral verwaltete Modell, zu koordinieren. In Hinblick auf eine verteilte virtuelle Umgebung gilt es, die Konsistenz des zugrunde liegenden Modells zu sichern, also zum Beispiel der Daten, die die Position eines Objekts im virtuellen Raum beschreiben. Dies wird durch den Umstand erschwert, dass das Modell bei jedem Client repliziert vorgehalten wird.

Die Nebenläufigkeitskontrolle ist ein intensiv untersuchter Gegenstand im Bereich verteilter Betriebssysteme und Datenbankverwaltungssysteme. Die Standardliteratur der jeweiligen Gebiete wie (Borghoff & Schlichter, 2000), (Tanenbaum &

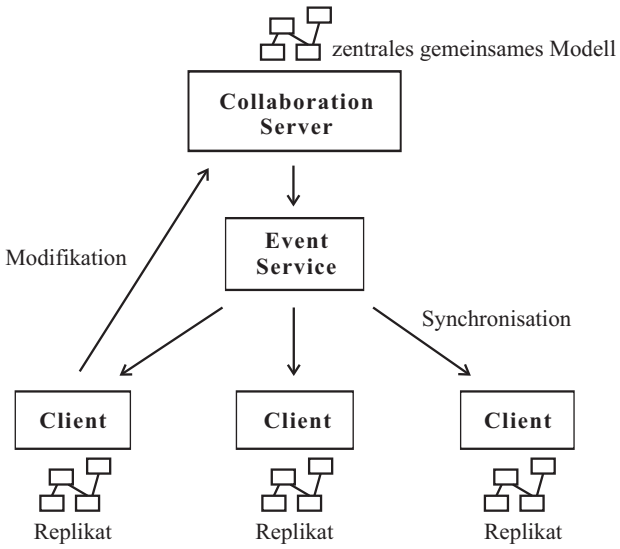


Abbildung 4.7: Einsatz eines Event Service zur Synchronisation der verteilten Replikate des gemeinsamen Modells.

van Stean, 2002) und (Elmasri & Navathe, 2003) diskutiert dieses Thema unter verschiedenen Überschriften wie Nebenläufigkeitskontrolle, Konsistenzsicherung, Synchronisation und Transaktionsverarbeitung.

In verteilten virtuellen Umgebungen ist es Aufgabe der Nebenläufigkeitskontrolle, den simultanen Zugriff auf ein gemeinsam genutztes Objekt zu regeln. Die Notwendigkeit einer Nebenläufigkeitskontrolle für CVEs wurde bereits früh erkannt und detailliert in (Greenberg *et al.*, 1992), (Dewan & Shen, 1998) und (Singhal & Zyda, 1999) diskutiert.

Die Methoden zur Nebenläufigkeitskontrolle können in optimistische und pessimistische Ansätze unterschieden werden. Optimistische Ansätze gehen davon aus, dass Konflikte vergleichsweise selten auftreten, und machen Änderungen im Nachhinein rückgängig, wenn ein Konflikt erkannt wird. Pessimistische Ansätze vermeiden hingegen das Auftreten jeglicher Konflikte.

Eine optimistische Strategie toleriert also zeitweise Inkonsistenzen, bis diese erkannt und ausgeräumt werden und kann dadurch kürzere Interaktionszeiten gewährleisten. Bei Einsatz einer pessimistischen Strategie werden hingegen Verzögerungen auftreten, wenn potentiell simultane Zugriffe erfolgen und diese zur Wahrung der Konsistenz serialisiert werden müssen. Dies führt zu einer geringeren Interaktivität des verteilten Mehrbenutzersystems. Die Abwägung zwischen Konsistenzsicherung und Interaktivität wird in der Literatur intensiv diskutiert, allgemein für Groupware in (Greenberg & Marwood, 1994) und speziell für Verteilte Virtuelle Umgebungen in (Singhal & Zyda, 1999).

Nach (Singhal & Zyda, 1999) bestimmt die Architektur des verteilten Systems wesentlich die anzuwendenden Methode der Nebenläufigkeitskontrolle. Der einfachste Fall liegt vor, wenn eine zentrale Komponente eingesetzt wird, um neben-

läufige Schreibzugriffe auf das gemeinsame Modell mit Hilfe von Sperren zu verhindern. Dieses Verfahren wird zum Beispiel im Shastra-System (Anupam *et al.*, 1994) eingesetzt, kommt aber auch bei der hier vorzustellenden CoCoS-Plattform zum Einsatz. Da für große verteilte Systeme die Verwendung einer zentralen Komponente jedoch zu inakzeptablen Performanzeinbußen führen kann, gibt es daneben auch dezentrale Ansätze, bei denen ein verteiltes Konsistenzprotokoll für die Nebenläufigkeitskontrolle sorgt. Da es in diesem Fall keinen zentralen Server gibt, der die Sperren verwaltet, kommunizieren die Clients direkt miteinander und determinieren eine globale Ordnung für die schreibenden Zugriffe.

In Mehrbenutzerspielen kommt in der Regel eine vereinfachte Methode zum Einsatz, die sogenannte *Frequent State Regeneration*, die auch als *Blind Broadcasting* bekannt ist. Dabei wird der lokale Statusvektor jedes Clients in regelmäßigen Intervallen an alle anderen Clients versandt. Der Empfänger schreibt den eintreffenden Zustandsvektor ohne weitere Prüfung in seine lokale Kopie des Modells. Bei diesem Vorgehen werden temporäre Inkonsistenzen zugelassen, was jedoch bei dieser Anwendungsklasse durch eine hohen Update-Zyklus kompensiert werden kann. Wegen der damit verbundenen hohen Netzwerklast ist dieser Ansatz jedoch für CoCoS nur wenig geeignet, da hier Netzwerkressourcen vor allem für die Übertragung von Simulationsdaten reserviert werden müssen. Darüber hinaus liegen die Vorteile eines dezentralen Systems vor allem in seiner guten Skalierbarkeit hinsichtlich der Zahl der Nutzer. Typische CoCoS-Sitzungen werden jedoch nur selten mehr als vier oder fünf Teilnehmer aufweisen.

Die in CoCoS realisierte Nebenläufigkeitskontrolle basiert auf Sperrungen (engl. *Locks*), die zentral verwaltet werden. Eine Sperrung gilt jeweils für ein gemeinsam genutztes geometrisches Objekt. Ein Locking für globale Parameter existiert hingegen nicht, da hier das Setzen eines Wertes als atomare Aktion angesehen werden kann. Auf diese Weise wird eine pessimistische Strategie umgesetzt. Da die Clients mit Replikaten des gemeinsam genutzten Modells arbeiten, wird dieses Modell in der Literatur auch als „Zentralisiertes Sperren mit vollständiger Replikation“ bezeichnet (Borghoff & Schlichter, 2000).

Der Kollaborationsserver serialisiert und synchronisiert alle Modifikationen mit Hilfe von Sperrungen. Bevor ein Nutzer ein physisches Objekt verschieben oder andersweitig modifizieren kann, wird von seiner Client-Applikation eine Sperre vom Kollaborationsserver angefordert. Wird diese Sperre genehmigt, können anderer Nutzer das betreffende Objekt so lange nicht verändern, bis die Sperre wieder aufgehoben wird. Dies geschieht, sobald der Nutzer die Modifikation beendet hat und seine Client-Applikation eine entsprechende Nachricht an den Kollaborationsserver sendet. Beide Operationen, das Anfordern und das Freigeben eines Locks, geschehen implizit, d.h. für den Nutzer vollkommen transparent.

Da die Aufrufe zur Sperrung eines Objekts auf Seiten des Kollaborationsservers serialisiert werden, ist es nicht möglich, dass mehrere Nutzer gleichzeitig ein Objekt modifizieren. Die Sperren werden ohne Priorisierungen, d.h. nach dem Eintreffen der Anforderungen vergeben. Dieses Prinzip wird auch mit *„first come, first served“* umschrieben.

Wenn ein Objekt gesperrt wird, werden alle anderen Client-Applikationen darüber informiert. Ihre Nutzerschnittstelle muss dem Anwender die Möglichkeit geben, ein gesperrtes Objekt als solches zu erkennen, und Modifikationen daran verhindern. In der prototypisch implementierten Client-Applikation CoFluids, werden Objekte, die gerade von anderen Nutzern modifiziert werden und daher gesperrt sind, in einer anderen Farbe dargestellt und können vom Nutzer nicht selektiert werden.

Die zeitliche und räumliche Granularität der Sperren hat wesentliche Auswirkungen auf die Interaktivität eines Mehrbenutzersystems (Greenberg & Marwood, 1994). Werden zu große Teile des gemeinsam genutzten Modells zu lange gesperrt, hindert dies die anderen Nutzer an der Weiterarbeit. Die Wahl des geometrischen Objekts als kleinste sperrbare Einheit und der Transformationsoperationen als kürzeste Transaktionsdauer stellt einen geeigneten Kompromiss mit guter Nutzerakzeptanz dar, wie ausführliche praktische Untersuchungen zeigen konnten.

4.5 Awareness-Aspekte

Im Gegensatz zu klassischen transaktionsbasierten Datenbanksystemen, die simultan zugreifende Nutzer voneinander isolieren und ihnen den Eindruck vermitteln, sie wären alleinige Nutzer, muss eine Umgebung für synchrone Zusammenarbeit Bewusstsein für die Aktionen der anderen Beteiligten schaffen (Ellis *et al.*, 1991; Dourish & Bellotti, 1992).

In CoCoS wird gegenseitiges Bewusstsein *(Awareness)* dadurch geschaffen, dass die Modifizierungsaktionen der Beteiligten sofort für alle anderen sichtbar sind, virtuelle Zeigewerkzeuge zur Verfügung stehen und Avatare die Beobachtungspositionen der anderen Teilnehmer symbolisieren. Daneben ist der Einsatz eines separaten Audio- oder sogar Videokanals unbedingt zu empfehlen, um einen höheren Grad von *Awareness* zu erzielen.

Zwischen absoluter Isolation und gleichgeschalteter Visualisierung (auch mit „What you see is what I see“, kurz WYSIWIS umschrieben) gibt es allerdings mehrere Zwischenstufen, von denen jede für eine unterschiedliche Form der kooperativen Arbeit geeignet ist.

Die CoCoS-Plattform bietet drei verschiedene Ebenen der Unterstützung gegenseitigen Bewusstseins:

- Auf der **untersten Ebene** ist lediglich das gemeinsam genutzte Modell identisch. Die Methoden zur Visualisierung der Simulationsdaten können die Teilnehmer unabhängig voneinander wählen, ebenso wie den eigenen Beobachtungsstandpunkt und die Position der virtuellen Zeigewerkzeuge.
- Auf der **mittleren Ebene** bleibt der Beobachtungsstandort individuell wählbar, allerdings ist hier die Visualisierung der Simulationsdaten gekoppelt. Das heißt, dass die Art der Visualisierung (bspw. Stromlinien) sowie die verschiedenen Parameter (u.a. Position und Färbung) bei allen Nutzern identisch sind, die sich für diese Ebene der Awareness entschieden haben.

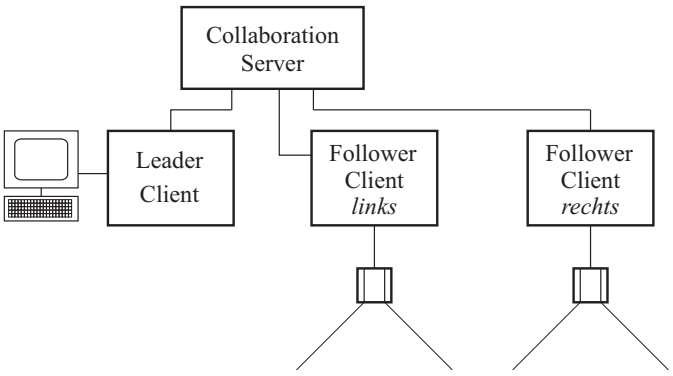


Abbildung 4.8: Verwendung der Leader-Follower-Funktionalität von CoCoS zur Umsetzung einer speziellen Form der stereoskopischen Projektion, bei der die Projektoren für linkes und rechtes Auge von unterschiedlichen Rechnern angesteuert werden. Die Follower-Clients nehmen nicht die exakte Beobachtungsposition des Leader ein, sondern eine um jeweils einen halben Augenabstand verschobene.

Diese Ebene eignet sich vor allem für die Zusammenarbeit von Ingenieuren des gleichen Fachgebiets, da die Voraussetzung der Empfang von Simulationsdaten des gleichen Simulationsservers ist.

– Auf der **höchsten Ebene** sind außerdem auch Beobachtungsstandort und -richtung sowie die Position der virtuellen Zeigewerkzeuge aneinander gekoppelt. Diese Ebene kann für Zusammenarbeit nach dem *Leader-Follower*-Prinzip verwendet werden, bei der ein Teilnehmer die anderen zu einem speziellen *Point of Interest* führt, um ein dort sichtbares Phänomen zu diskutieren.

Zwischen diesen drei Awareness-Modi kann zu jeder Zeit beliebig gewechselt werden. Außerdem können Untergruppen der gesamten Teilnehmermenge unterschiedliche Kooperationsweisen wählen.

Technische Grundlage aller Formen der Unterstützung von Awareness ist die kontinuierliche Übermittlung von Daten der Clients an den Kollaborationsserver bezüglich der aktuellen Beobachtungsposition, der Position der Zeigewerkzeuge und der Wahl und Parametrisierung der Simulationsvisualisierung.

Diese Daten können des Weiteren dazu genutzt werden, um mit Hilfe der CoCoS-Plattform eine spezielle Form der stereoskopischen Projektion zu realisieren, bei der die Projektoren für linkes und rechtes Auge von verschiedenen Rechnern angesteuert werden (Abb. 4.8). Dazu wird auf einem dritten Rechner, dem Vorführrechner, ein Client im *Leader*-Modus gestartet. Auf den beiden mit den Projektoren verbundenen Maschinen wird jeweils ein Client in einem speziellen *Follower*-Modus gestartet, der nicht die exakte Beobachtungsposition des *Leader* einnimmt, sondern eine um jeweils einen halben Augenabstand verschobene. Der Kollaborationsserver dient in diesem Fall der Synchronisierung der verschiedenen Projektionen. Darüber hinaus können beliebig viele weitere Projektorenrechner,

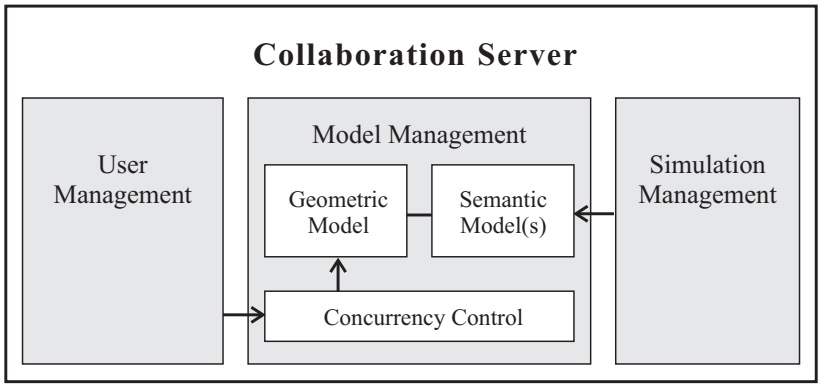


Abbildung 4.9: Die Module des Kollaborationsservers.

beispielsweise zur Erweiterung der Projektionsfläche, angesteuert werden. Auf diese Weise konnte eine stereoskopische Projektion im MediaLab der *Siemens Corporate Technology* erfolgreich realisiert werden.

4.6 Der Kollaborationsserver

Der Kollaborationsserver stellt die wichtigste Komponente der gesamten CoCoS-Plattform dar. Er übernimmt die folgenden Aufgaben:

- Verwaltung des gemeinsamen Modells und Nebenläufigkeitskontrolle,
- Verwaltung der Nutzer, ihrer aktuellen Beobachtungsstandpunkte, ihrer Rollen und Rechte sowie der Art der aktuell verwendeten Visualisierung und ihrer Parameter,
- Verwaltung der Simulationsserver, ihrer Standorte (IP), die aktuelle Belegung ihrer Start- und Steeringparameter sowie die Struktur der Simulationsdaten, die sie produzieren.

Jede dieser Aufgaben korrespondiert mit einem Modul innerhalb des Kollaborationsservers, wie Abb. 4.9 zeigt. Der Kern des Servers wird vom Modellverwaltungsmodul gebildet. Wie bereits in Abschnitt 4.2 ausführlich diskutiert, wird in CoCoS ein hybrider Ansatz gewählt, der ein geometrisches und beliebig viele semantische Modelle (Domänen- bzw. Simulationsmodelle) miteinander kombiniert. In der Beispielapplikation *Komfortanalyse* repräsentiert das geometrische Modell die Strömungshindernisse sowie die Hülle des Strömungsgebietes. Randbedingungen wie die Oberflächenrauigkeit werden als semantische Daten verwaltet, die mit dem geometrischen Modell verknüpft sind.

Nimmt ein Teilnehmer eine lokale Modifikation am Modell vor, durch Hinzufügen, Entfernen, Bewegen oder Skalieren von geometrischen Objekten oder Setzen von semantischen Attributen, werden diese Informationen von der Client-Applikation an den Kollaborationsserver gesendet. Wie in Abschnitt 4.4 dargelegt, wird zur Vermeidung von Konflikten zu Beginn einer geometrischen Transformationen eine Sperre für das betreffende Objekt verhängt, die verhindert, dass andere Nutzer

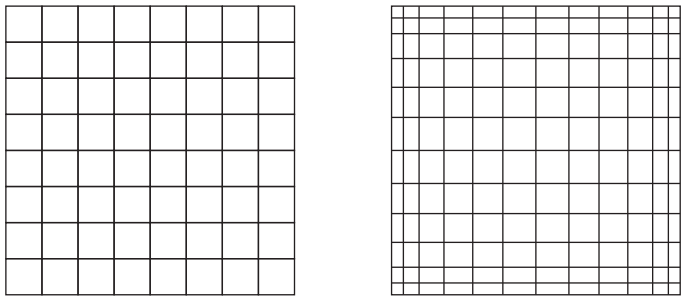


Abbildung 4.10: Von CoCoS werden uniforme (links) und rektolineare Gitter (rechts) unterstützt.

dieses Objekt zur selben Zeit transformieren. Die Sperre wird erst aufgehoben, wenn die Transformation beendet ist. Der Kollaborationsserver muss diese Sperren verwalten, d.h. festhalten, welche Objekte von welchem Nutzer gesperrt sind, und ggf. das Setzen einer Sperre verweigern. Um die Robustheit des Systems gegenüber Ausfällen einzelner Clients zu erhöhen, wird einem Client nach Verstreichen einer festgelegten Zeit eine Sperre automatisch entzogen.

Wenn ein Simulationsserver der CoCoS-Plattform beitritt, meldet er sich beim Kollaborationsserver an und stellt alle notwendigen Informationen zur Verfügung, damit sich Clients mit ihm verbinden können, darunter die IP-Adresse, die verwendeten Ports sowie das verwendete Simulationsmodell. Diese Informationen werden im Simulationsverwaltungsmodul vorgehalten. Die Clients fragen dieses ab, um Informationen zur Aufnahme einer Verbindung zum betreffenden Simulationsserver zu erhalten. Darüber hinaus verfolgt das Simulationsverwaltungsmodul die Verfügbarkeit der Simulatoren bzw. ihren aktuellen Status (*nicht gestartet, laufend, beendet, nicht erreichbar*).

Das Modul verwaltet des Weiteren Informationen über die Struktur der Ergebnisdaten der einzelnen Simulationsserver, die diese bei ihrer Anmeldung bekanntgeben. Dazu gehört der Typ des Gitters[3] (uniform oder rektolinear - Abb. 4.10), die Bezeichnung der einzelnen Feldgrößen und die Zuordung zu skalaren oder vektoriellen Größen. Diese Informationen werden von den Clients verwendet, um passende Visualisierungsmittel für die Simulationsergebnisse zur Verfügung zu stellen. Für Vektorfelder bieten sich beispielsweise Stromlinien- und Vektordarstellungen an, während Skalarfelder in der Regel mit Hilfe von Schnittebenen oder isometrischen Flächen visualisiert werden.

Das Nutzerverwaltungsmodul hält Informationen über alle Nutzer vor, die an der kollaborativen Session teilnehmen. Dazu gehört der Name des Teilnehmers und die IP-Adresse des von ihm verwendeten Clients sowie seine Zugehörigkeit zu einer Domänengruppe, aber auch seine aktuelle Beobachtungsposition, die von ihm visualisierten Simulationsdaten und die dazu verwendete Technik. Dies ist notwendig, um die verschiedenen Ebenen von Awareness, wie sie in Abschnitt 4.5

[3] Zum Zeitpunkt des Abschlusses dieser Arbeit werden von CoCoS lediglich Berechnungs*gitter* unterstützt, das Konzept erlaubt jedoch eine einfache Erweiterung um Berechnungs*netze*.

beschrieben werden, realisieren zu können. Als mögliche Visualisierungsmethode werden einfache Schnittebenen, Vektorebenen, Stromlinien und isometrische Flächen akzeptiert.

Der Kollaborationsserver bietet eine dem CORBA *Event Service* ähnliche Schnittstelle (vgl. Abschnitt 4.8), mit deren Hilfe sich Clients und Simulationsserver registrieren können, um über verschiedene Ereignisse benachrichtigt zu werden. Um die unterschiedlichen Kategorien von Ereignissen klar voneinander zu trennen, werden dabei verschiedene Benachrichtigungskanäle angeboten: Der *User Event Channel* benachrichtigt über das Eintreten und Verlassen von Nutzern, der *Model Event Channel* über Änderungen am geometrischen oder semantischen Modell, der *Concurrency Control Channel* über das Setzen und Freigeben von Sperren und der *Simulation Event Channel* über die Verfügbarkeit neuer Simulationsserver und das Starten und Stoppen einer Simulation. Diese Trennung erlaubt es den Clients bzw. den Simulationsservern, sich nur über die spezifischen Events benachrichtigen zu lassen, die für die jeweilige Anwendung von Interesse sind. Ein Simulationsserver muss beispielsweise in der Regel nicht über den Ein- und Austritt von Teilnehmern informiert werden.

4.7 Die Clients

Die CoCoS-Clients dienen als Visualisierungs- und Interaktionsschnittstelle für die Teilnehmer der kollaborativen Sitzung. Die folgenden grundlegenden Leistungen müssen von jeder Client-Applikationen realisiert werden:

- Ein- und Ausloggen beim Kollaborationsserver,
- Visualisierung geometrischer Objekte, Mittel zur Interaktion mit ihnen (Transformation), Unterstützung der Sperrmechanismen,
- Anzeige der mit einem geometrischen Objekt verknüpften semantischen Daten,
- Anzeige der aktuell Teilnehmenden; Benachrichtigungen, wenn andere Teilnehmer der Sitzung beitreten oder sie verlassen.

Dies sind die Basisdienste, die auch solche Clients anbieten müssen, die zur reinen geometrischen Manipulation des Modells dienen und keine Simulationsdaten empfangen. Für alle anderen ist es darüber hinaus notwendig, dass Möglichkeiten zur Verbindung mit einem Simulationsserver und Funktionalitäten zur Visualisierung der Simulationsdaten angeboten werden.

Im Rahmen des beispielhaften Anwendungsszenarios *Interaktive Komfortanalyse* wurde die prototypische Client-Anwendung CoFluids implementiert (Abb. 4.11). Sie erlaubt einem Nutzer, die in einem Büroraum herrschende Luftströmung in Form von Vektorebenen, isometrischen Flächen und Stromlinien zu visualisieren, die Strömungshindernisse in diesem Raum zu bewegen und zu skalieren und die Randbedingungen an Ein- und Ausströmrändern zu verändern.

Der CoFluids-Client kann im stereofähigen Ein-Fenster- oder im Mehr-Fenster-Modus laufen. Letzterer eignet sich vor allem für die 3D-Visualisierung und

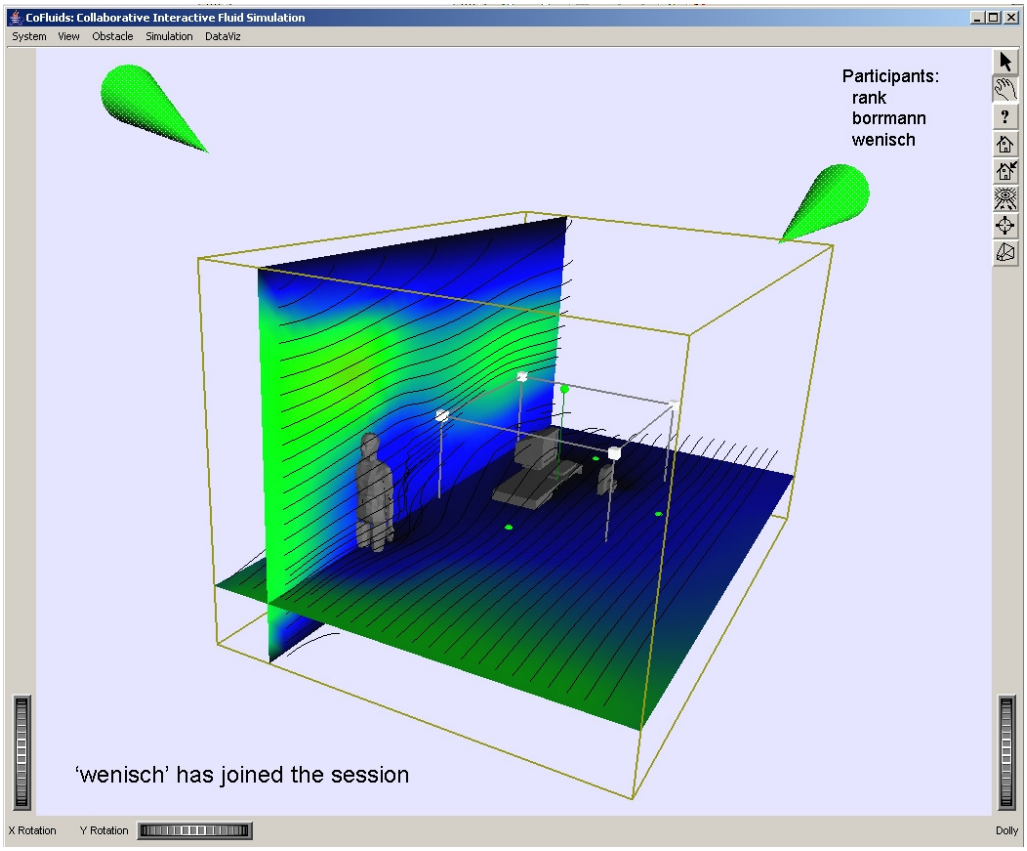


Abbildung 4.11: Screenshot der prototypisch implementierten Client-Applikation CoFluids. Gezeigt werden bewegbare Einrichtungsgegenstände sowie Dummies in einem Büroraum, die Visualisierung der Strömung in Form von Stromlinien und kegelförmige Avatare, die die Beobachtungspostionen der anderen Teilnehmer symbolisieren.

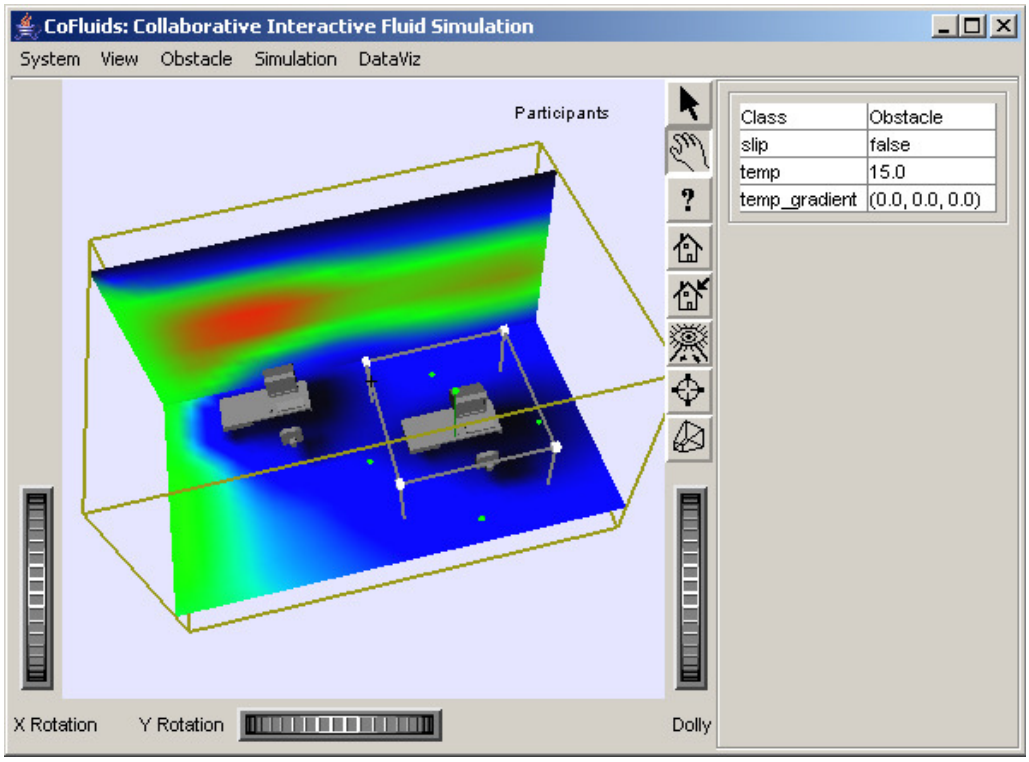


Abbildung 4.12: Screenshot der prototypisch implementierten Client-Applikation CoFluids. Die Zuweisung von Klassen aus dem Domänenmodell CFD an geometrische Objekte sowie die Belegung der dazugehörigen nicht-geometrischen Attribute mit Werten erfolgt mit Hilfe einer tabellarischen Eingabemaske am rechten Rand des Fensters.

-Navigation auf herkömmlichen Desktop-Computern. Zur Realisierung der Sperrmechanismen werden Objekte, die gerade von anderen Teilnehmern modifiziert werden, in einer anderen Farbe dargestellt und können nicht selektiert werden (s.a. Abschnitt 4.4).

Die Zuweisung von Klassen aus dem Domänenmodell *CFD* sowie die Belegung der dazugehörigen nicht-geometrischen Attribute mit Werten erfolgt mit Hilfe einer tabellarischen Eingabemaske am rechten Rand des Eingabefensters (Abb. 4.12).

Zur Unterstützung der gegenseitigen Wahrnehmung wird in CoFluids eine Liste mit den Namen aller angemeldeten Teilnehmer angezeigt und der Nutzer mit Hilfe von Textnachrichten darüber informiert, wenn ein anderer Teilnehmer der Sitzung beitritt oder sie verlässt. Darüber hinaus ist CoFluids in der Lage, die Beobachtungspositionen der übrigen Beteiligten mittels einfacher kegelförmiger Avatare zu visualisieren (s.a. Abschnitt 4.5).

4.8 Die Simulationsserver

Hauptaufgabe eines Simulationsservers ist es, eine Brücke zwischen dem verteilten kooperativen System und einem spezifischen Simulationskernel zu schaffen.

Ebenso wie die Clients lauscht ein Simulationsserver am *Model Event Channel* des Kollaborationsservers. Er wird auf diese Weise über Änderungen am geometrischen Modell und an den Simulationsparametern informiert und leitet sie an den Simulationskernel weiter.

Beim Hochfahren des Simulationsservers meldet er sich beim Kollaborationsserver an und übermittelt mit Hilfe der generischen Schnittstelle auf Metaniveau (Abb. 4.3, links) die Struktur des semantischen Modells, also der Start- und Steering-Parameter sowie der geometriebezogenen Randbedingungen der Simulation. Des Weiteren gibt er die für die Verbindungen verwendeten Ports sowie die Struktur der Ergebnisdaten bekannt, darunter die Art des Gitters, die verwendeten Variablen usw. (siehe Abschnitt 4.6). Diese Informationen werden durch den Client vom Kollaborationsserver abgefragt, um sich mit dem Simulationsserver zu verbinden und die während der Simulation erhaltenen Ergebnisdaten und die verfügbaren Steering-Parameter auf angemessene Weise darzustellen.

Zur minimalen Schnittstelle, die ein Simulationsserver nach außen anbieten muss, gehören Methoden zum Starten und Stoppen einer Simulation, zum Verändern der Steering-Parameter sowie für das An- und Abmelden von Clients. Der Simulationsserver muss darüber hinaus alle bei ihm angemeldeten Clients informieren, wenn die Simulation gestartet oder beendet wurde oder wenn ein Steering-Parameter modifiziert wurde.

Bei der Übertragung von Ergebnisdaten vom Simulationsserver zu den Clients wurde mit zwei verschiedenen Kommunikationsmodi experimentiert: einem *Push*- und einem *Pull*-Modus. Der Unterschied besteht darin, welche der beteiligten Komponenten die Datenübertragung auslöst. Beim *Push*-Modus wird sie vom Simulationsserver initiiert, d.h. er „schiebt“ die Daten zu den Clients, sobald sie verfügbar sind. Beim *Pull*-Modus bestimmt jeder Client selbst, zu welchem Zeitpunkt bzw. in welchem Intervall er Simulationsdaten empfangen will, und „zieht“ sie sich in der Folge vom Simulationsserver.

Es gibt zwei wesentliche Kriterien, mit denen die Eignung eines Kommunikationsverfahrens für das Verteilen von Simulationsdaten bewertet werden kann. Ein erstes Kriterium ist, inwieweit die Simulation selbst durch den Kommunikationsprozess abgebremst wird. Zweitens sollte die Übertragung sowohl möglichst schnell, als auch flexibel an die unterschiedlichen Bedürfnisse verschiedener Client-Applikationen anpassbar sein. Im Gegensatz zu Streaming-basierten Kommunikationslösungen wie sie zur Ausstrahlung von Audio- und Videodaten verwendet werden (Mungee *et al.*, 1999; Wu *et al.*, 2001), ist es für einen Client in einem verteilten *Computational Steering*-System nicht notwendig, jeden einzelnen Datensatz der Simulation zu empfangen. Stattdessen ist es wünschenswert, dass der empfangene Datensatz möglichst jung ist, d.h. als einer der letzten vom Simulator produziert wurde.

Bei der Umsetzung des *Push*-Modus wurde eine Implementierung des CORBA *Event Service* eingesetzt. Dabei werden die Simulationsdaten zunächst vom Simulationsserver an den *Event Service* übertragen. Dieser sorgt dann für die Ver-

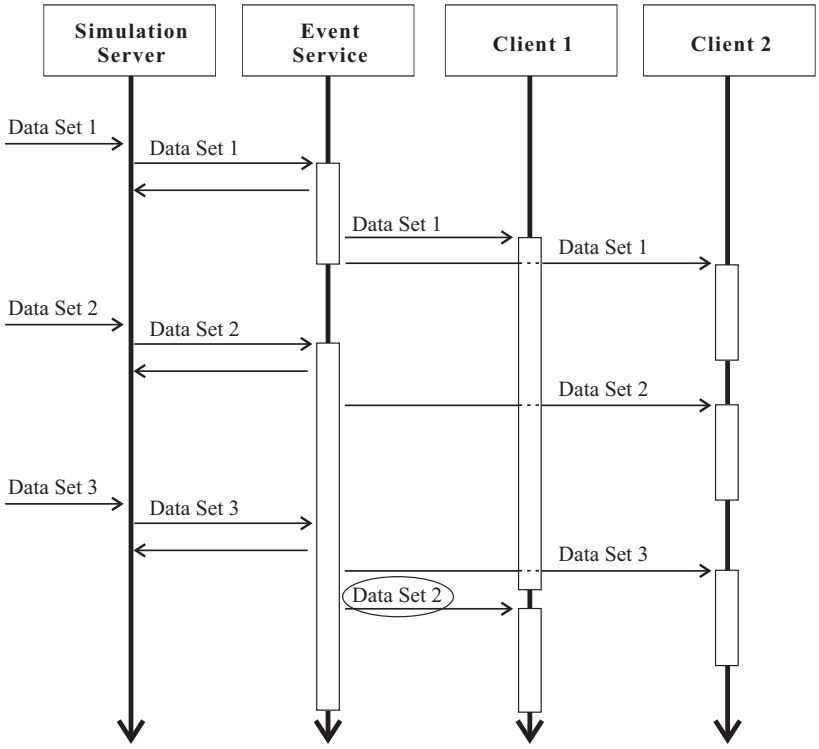


Abbildung 4.13: Im *Push*-Kommunikationsmodus werden die Simulationsdaten unter Einsatz eines *Event Service* zu den Clients „geschoben". Wie die Abbildung am Beispiel von Client 1 zeigt, kann es zum Verstopfen des *Event Service* mit veralteten Datensätzen kommen, wenn ein Client die Simulationsdaten nicht schnell genug verarbeitet.

teilung der Daten an die angemeldeten Clients. In beiden Fällen werden die Daten „geschoben". Der Vorteil dieser Lösung ist, dass die Verteilung der Simulationsdaten vollkommen vom Simulationsprozess entkoppelt ist. Ein wesentlicher Nachteil ist, dass die Daten jedes einzelnen Simulationsschrittes übertragen werden, unabhängig davon, ob diese evtl. bereits veraltet sind, weil neuere Simulationsdaten beim Simulationsserver auftreten. Dies führt vor allem bei Clients mit erhöhter Netzwerklatenz oder langsamer Verarbeitung zu einem unerwünschten Hinterherhinken in der Simulationshistorie.

Da im *Pull*-Modus hingegen jeder Client den Empfang selbst initiiert, kann er zum einen den Takt der Übertragung selbst wählen (Abb. 4.14). Zum anderen ist gewährleistet, das immer nur die neuesten Simulationsdaten übertragen werden. Aus diesem Grund ist die Kommunikation im *Pull*-Modus als geeigneter für ein verteiltes *Computational Steering*-System einzuschätzen.

Zwar bietet auch der CORBA *Event Service* die Möglichkeit der *Pull*-Kommunikation, allerdings nur mit FIFO[4]- anstatt der hier benötigten LIFO[5]-Semantik. Aus diesem Grund und dem erhöhten Kommunikationsaufwand bei Verwendung

[4]first in, first out
[5]last in, first out

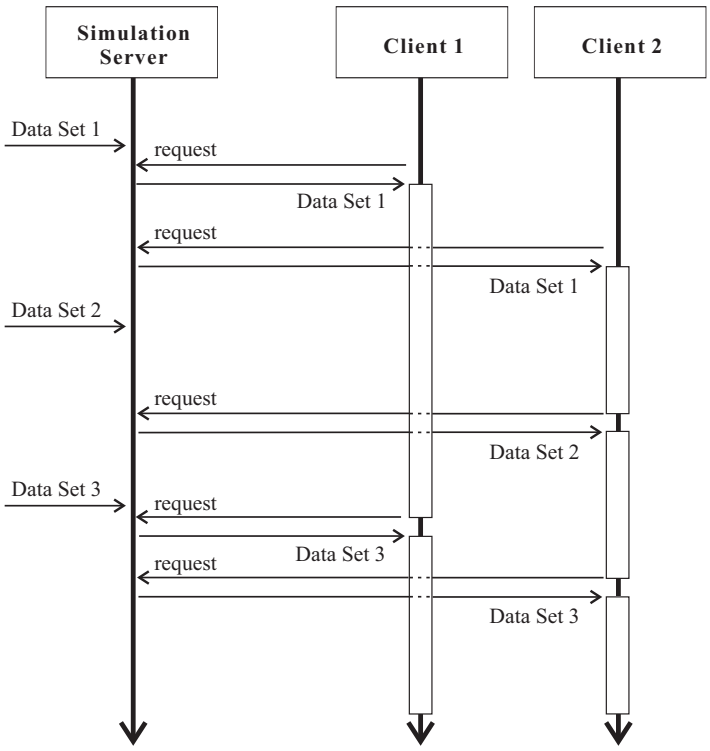


Abbildung 4.14: Im *Pull*-Kommunikationsmodus werden die Simulationsdaten von den Clients „gezogen“. Auf diese Weise ist sichergestellt, dass immer nur die aktuellsten Simulationsdaten übermittelt werden.

eines zusätzlichen Prozesses ist die Verwendung des *Event Service* als nicht zielführend einzustufen. Die vom Event Service realisierte Entkopplung des Simulationsprozesses vom Datenverteilungsprozess ist vorzugsweise in den Simulationsserver selbst zu integrieren, was allerdings nur durch eine vergleichsweise aufwändige (Nach-)Implementierung der Benachrichtungsfunktionalität möglich ist.

Kapitel 5

Ein Gitter-Boltzmann-Simulationsserver für interaktive Strömungssimulationen

Dieses Kapitel stellt die für eine kollaborativ bedienbare, interaktive Strömungssimulation notwendige CoCoS-Systemkomponente *Simulationsserver* vor, die im gewählten Anwendungsszenario *Planung der Klimatechnik*[1] zum Einsatz kommt. Dabei wird zunächst ein kurzer Überblick über Grundlagen der numerischen Simulation von Strömungsvorgängen gegeben und besonders auf die Vorzüge des Gitter-Boltzmann-Verfahrens mit Hinblick auf interaktive Simulationen eingegangen, bei denen geometrische Randbedingungen zur Laufzeit verändert werden können. Im Anschluss wird eine mögliche Realisierung der für diese Interaktivität notwendigen schnellen und vollständig automatisierten Diskretisierung der Ausgangsgeometrie im Berechnungsgitter vorgestellt sowie die Parallelisierung des Simulationscodes zur weiteren Performanzsteigerung diskutiert.

5.1 Numerische Strömungssimulation

5.1.1 Top-Down- und Bottom-Up-Ansätze

Die klassische Herangehensweise bei der numerischen Simulation raum-zeitlicher Phänomene wählt als Ausgangspunkt die makroskopischen Erhaltungssätze, im Fall der Strömungssimulation die *Navier-Stokes-Gleichungen* (Abschnitt 5.1.2).

[1] Der in diesem Kapitel vorzustellende Simulationscode ist ein reiner Strömungslöser, berücksichtigt also keinerlei Wärmetransport- und Auftriebsvorgänge. Für das im vorangegangenen Kapitel besprochene Anwendungsszenario *Komfortanalyse* sind diese jedoch von entscheidender Bedeutung. Ein entsprechender interaktiver Simulationscode befindet sich derzeit in der Entwicklung (Pfaffinger *et al.*, 2007).

Dabei handelt es sich um einen Satz partieller Differentialgleichungen, für die keine allgemeine analytische Lösung bekannt ist. Zur Überführung dieser Differentialgleichungen in ein algebraisches Gleichungssystem wird eine zeitliche und räumliche Diskretisierung mittels verschiedener numerischer Methoden vorgenommen. Zur räumlichen Diskretisierung sind beispielsweise Finite-Differenzen-, Finite-Element- und Finite-Volumen-Verfahren einsetzbar, wobei letztere sich als besonders geeignet zur Lösung von Strömungsproblemen erwiesen haben (Ferziger & Peric, 2001). Für die zeitliche Diskretisierung stehen *explizite* und *implizite* Zeitschritt-Verfahren zur Verfügung. Explizite Verfahren benötigen zur Ermittlung des Systemzustandes im Zeitschritt t_n ausschließlich Systeminformationen vom vorangegangenen Zeitschritt t_{n-1}. Implizite Zeitschritt-Verfahren hingegen ermitteln den Zustand des Systems zum Zeitpunkt t_n mittels Gleichgewichtsbetrachtungen, bei denen der Systemzustand durch Lösen eines Gleichungssystems gewonnen wird.

Dieser Ansatz zur numerischen Lösung durch Diskretisierung eines Systems von Differentialgleichungen wird als *Top-Down-Ansatz* bezeichnet. Dem Top-Down-Ansatz konzeptionell entgegengesetzt sind die sogenannten *Bottom-Up-Ansätze*, bei denen durch eine geeignete Konstruktion der Interaktionen auf mikroskopischer Ebene die makroskopischen Erhaltungssätze erfüllt werden.

Zu den Bottom-Up-Ansätzen zählt beispielsweise der Gitter-Gas-Ansatz (Krafczyk, 1995; Wolf-Gladrow, 2000), bei dem Teilchenbewegungen und die Interaktion zwischen einzelnen Teilchen mit Hilfe eines zellulären Automaten simuliert werden. Eine als belegt gekennzeichnete Zelle entspricht dabei einem einzelnen Teilchen. Der Nachteil dieses Ansatzes liegt u.a. darin, dass für eine realistische Simulation extrem große Gitter benötigt werden, was zu einem enormen Speicherplatzbedarf und entsprechend langen Rechenzeiten führt.

Daher wird seit Ende der 80er Jahre intensiv an einem Ansatz auf der Meso-Skala geforscht, was schließlich zur Entwicklung der *Gitter-Boltzmann-Methode* führte (Zanetti & McNamara, 1988; Krafczyk, 2001). Im Unterschied zum Gitter-Gas-Ansatz bilden hierbei nicht einzelne Teilchen die Grundlage der Simulation, sondern sogenannte Wahrscheinlichkeitsdichtefunktionen, die die Bewegung eines Teilchenensembles statistisch abbilden. Diese Methode zeigt gegenüber dem Gitter-Gas-Ansatz wesentliche Vorteile hinsichtlich numerischer Stabilität und Ressourcenbedarf und soll nach Einführung der makroskopischen Erhaltungssätze näher vorgestellt werden.

5.1.2 Die Navier-Stokes-Gleichungen

Die Navier-Stokes-Gleichungen bilden den Gleichgewichtszustand der makroskopischen Größen *Druck* und *Impuls* ab. Betrachtet werden sollen dabei im Folgenden Boussinesq-inkompressible[2] Luftströmungen im dreidimensionalen Raum,

[2]Die grundlegende Annahme der Boussinesq-Approximation ist, dass das betrachtete Fluid zwar inkompressibel gegenüber Druckanwendungen ist, durch Temperaturänderungen aber Volumenänderungen hervorgerufen werden können.

wobei Luft als chemisch nicht-reagierendes Newtonsches Fluid[3] aufgefasst werden kann. Bei der eingesetzten Notation gilt die Einsteinsche Summenkonvention.

Kontinuitätsgleichung. Die Kontinuitätsgleichung beschreibt den Massenerhalt in einem infinitisimal kleinen Volumen, drückt also die Forderung aus, dass die Summe der ein- bzw. ausfließenden Massen gleich der Massenänderung (Dichteränderung) sein muss. Für inkompressible Fluide ($\rho = const.$) reduziert sich diese Forderung auf die Divergenzfreiheit des Geschwindigkeitsfeldes:

$$\frac{\partial u_i}{\partial x_i} = 0 \quad \text{mit } i \in \{1,2,3\} \quad . \tag{5.1}$$

Impulsgleichungen. Die Impulserhaltungsgleichung fordert, dass die zeitliche Änderung des Impulses eines Fluidelements gleich der Summe der auf das Fluid wirkenden Kräfte ist, die sich aus Volumenkräften (zum Beispiel Schwerkraft) und Oberflächenkräften (Druck- und Scherkräfte) zusammensetzen.

Für den hier betrachteten Fall Newtonscher, inkompressibler Fluide ($\partial u_i / \partial x_i = 0$) ergibt sich das System partieller Differentialgleichungen zur Beschreibung des Impulserhalts zu

$$\rho \frac{\partial u_i}{\partial t} + \rho u_j \frac{\partial u_i}{\partial x_j} = -\frac{\partial p}{\partial x_i} - \frac{\partial \tau_{ij}}{\partial x_j} + \rho g_i \quad \text{mit } i,j \in \{1,2,3\} \ , \tag{5.2}$$

wobei die viskosen Spannungen (Tensor τ_{ij}) aus dem molekülbedingten Impulstransport herrühren und g_i eine auf das gesamte betrachtete Volumen wirkende Kraft, wie beispielsweise die Gravitationskraft, darstellt.

Der hier vorzustellende Simulationscode berücksichtigt weder konvektive Wärmeströmungen noch chemische Reaktionen im Fluid. Daher ist die Einbeziehung einer *Energiegleichung* in die Betrachtungen nicht notwendig. Da des Weiteren die numerische Strömungssimulation nicht im Mittelpunkt dieser Arbeit steht, soll zur Bestimmung charakteristischer dimensionsloser Größen wie Reynolds[4]-, Mach[5]- und Knudsen-Zahl[6] sowie zur Integration von Turbulenzmodellen auf (van Treeck, 2004) verwiesen werden.

5.2 Die Gitter-Boltzmann-Methode

Die von (Zanetti & McNamara, 1988) eingeführte Gitter-Boltzmann-Methode verfolgt einen Bottom-Up-Ansatz, beruht also auf der Modellierung von Interaktionen zwischen Fluid-Partikeln. Dabei besteht die wesentliche Idee darin, das

[3] Bei Newtonschen Fluiden ist die Scherspannung proportional zur Schergeschwindigkeit.

[4] Die Renolyds-Zahl gibt das Verhältnis von Trägheits- zu Zähigkeitskräften an. Überschreitet sie einen (problemabhängigen) kritischen Wert ist mit einem Umschlag von laminarer in turbulente Strömung zu rechnen.

[5] Die Mach-Zahl gibt das Verhältnis des Betrages einer Geschwindigkeit v (hier des Fluides) zur Schallgeschwindigkeit c des Fluides an.

[6] Die Knudsen-Zahl misst das Verhältnis der mittleren freien Weglänge der Gasmoleküle zu einer geometrischen Bezugslänge. Sie ist ein Maß für die Dichte einer Gasströmung.

Fluid als Ensemble von miteinander interagierenden Partikeln zu betrachten und so das Verhalten eines Vielteilchensystems zu simulieren. Der Gitter-Boltzmann-Ansatz bedient sich dabei der Methoden der statistischen Physik bzw. der kinetischen Gastheorie. Es kann gezeigt werden, dass die gewählte Konstruktion geeignet ist, um die makroskopischen Erhaltungsgleichungen abzubilden.

Zwar ist es grundsätzlich denkbar, die Interaktion einzelner Teilchen auf Grundlage der Hamiltonschen Bewegungsgleichungen zu simulieren, die die individuellen Wechselwirkungen zwischen einzelnen Partikeln mit den Gesetzen der Molekulardynamik beschreiben, jedoch führt dieser Ansatz aufgrund der hohen Teilchenzahl ($\mathcal{O}(10^{19})$ Partikel je cm^3) zu einem enormen Ressourcenbedarf hinsichtlich Speicherplatz und Rechenkapazität und ist daher für praktische Anwendungen, bei denen makroskopische Strömungsphänomene simuliert werden sollen, nicht geeignet.

Für den Nutzer von Simulationswerkzeugen sind vielmehr vor allem solche Verfahren brauchbar, bei denen möglichst stufenlos zwischen den benötigten Rechenressourcen und der erzielbaren Genauigkeit abgewogen werden kann (Skalierung). Der Gitter-Boltzmann-Ansatz wird dieser Anforderung gerecht. Des Weiteren bestehen nach (van Treeck, 2004) die Vorteile des Gitter-Boltzmann-Verfahrens gegenüber einem Finite-Differenzen-Verfahren auf Grundlage der Navier-Stokes-Gleichungen in einer deutlich größeren maximalen Element-Reynoldszahl und dem Fehlen jeglicher numerischer Viskosität. Schließlich erlaubt die unmittelbare Verfügbarkeit des Spannungstensors die vergleichsweise einfache und ressourcenschonende Integration von Turbulenzmodellen.

Die Struktur der folgenden Abschnitte entspricht den Darstellungen in (Krafczyk, 2001), (Tölke, 2001) und (van Treeck, 2004).

5.2.1 Die Boltzmann-Gleichung

Grundlage des Gitter-Boltzmann-Verfahrens bildet die Boltzmann-Gleichung. Sie beschreibt die Wahrscheinlichkeit, ein Partikel mit dem Geschwindigkeitsvektor $\vec{\xi}_i$ zu einem gegebenen Zeitpunkt an einem gegebenen Ort $(t, \vec{x}_i)$ vorzufinden. Bei der Herleitung der Boltzmann-Gleichung (5.3) werden folgende Vereinfachungen angenommen:

- Es werden nur binäre Kollisionen berücksichtigt. Dies ist anzunehmen, wenn der Durchmesser der Teilchen sehr viel kleiner als der mittlere Abstand zwischen zwei Teilchen ist.
- Es herrscht molekulares Chaos, d.h. die Geschwindigkeiten der Partikel sind statistisch unkorreliert. Diese Annahme ist gerechtfertigt, wenn die Zeitdauer zwischen zwei Kollisionen sehr viel größer ist als die Dauer der Kollision selbst.
- Der lokale Kollisionsprozess wird nicht durch externe Kräfte beeinflusst, d.h. diese sind klein gegenüber den Kollisionskräften.

Boltzmann-Gleichung. Unter Berücksichtigung dieser Vereinfachungen ergibt sich die Integro-Differentialgleichung für die Verteilungsfunktion $f(t, \vec{x}, \vec{\xi})$ nach Boltzmann zu

$$\frac{\partial f}{\partial t} + \vec{\xi} \frac{\partial f}{\partial \vec{x}} + \vec{F} \frac{\partial f}{\partial \vec{\xi}} = \Omega(f) \ , \tag{5.3}$$

mit der Änderung der Verteilungsfunktion $\partial f / \partial t$ bezüglich der mikroskopischen Geschwindigkeit $\vec{\xi}$ und den äußeren Kräften $\vec{F}$ auf der linken Seite und dem Kollisionsintegral $\Omega(f)$ auf der rechten Seite.

Dieses Kollisionsintegral modelliert dabei den Kollisionsprozess zwischen je zwei Partikeln mit den Geschwindigkeiten $\vec{\xi}$, $\vec{\xi}_1$ vor bzw. $\vec{\xi}'$, $\vec{\xi}'_1$ nach dem Stoß mit dem differentiellen Wirkungsquerschnitt $\sigma(\Omega)$ (Wolf-Gladrow, 2000):

$$\Omega(f) = \int\int \sigma(\Omega) \, |\vec{\xi} - \vec{\xi}_1| \left[f(\vec{\xi}') f(\vec{\xi}'_1) - f(\vec{\xi}) f(\vec{\xi}_1) \right] d\Omega \, d\vec{\xi}_1 \quad . \tag{5.4}$$

BGK-Approximation. Bedingt durch den Kollisionsoperator ist die Boltzmann-Gleichung eine Integro-Differentialgleichung, die zunächst aufwändiger zu lösen ist als die Navier-Stokes-Gleichungen. Der von Bhatnagar, Gross und Krook eingeführte Ansatz (Bhatnagar *et al.*, 1954) vereinfacht daher den Kollisionsoperator unter der Annahme, dass sich das Gas als Kontinuum verhält und die Abweichungen vom Gleichgewichtszustand sehr gering sind, womit sich

$$\Omega(f) \approx \frac{1}{\tau} \left(f^{(0)} - f \right) \tag{5.5}$$

ergibt. Dabei beschreibt $f^{(0)}$ die lokale Gleichgewichtsverteilung (Maxwell-Verteilung) für den Zustand einheitlicher Dichte- und Geschwindigkeitsverteilung im Fluid.

Makroskopische Größen. Bildet man die Momente der Verteilungsfunktionen bezüglich der mikroskopischen Geschwindigkeit $\vec{\xi}$, so erhält man die makroskopischen Größen *Dichte* ρ und *Impulsdichte* ρu_α als Momente nullter und erster Ordnung zu

$$\rho(t, \vec{x}) = \int_{-\infty}^{+\infty} f(t, \vec{x}, \vec{\xi}) \, d\vec{\xi} \ , \tag{5.6}$$

$$\rho(t, \vec{x}) \, u_\alpha(t, \vec{x}) = \int_{-\infty}^{+\infty} \xi_\alpha \, f(t, , \vec{x}\vec{\xi}) \, d\vec{\xi} \ . \tag{5.7}$$

Somit kann die makroskopische Geschwindigkeit $u_\alpha(t, \vec{x})$ als Mittelwert der mikroskopischen Teilchengeschwindigkeiten dargestellt werden. Der *Impulsstromtensor* ergibt sich als Moment zweiter Ordnung.

Makroskopische Erhaltungsgleichungen. Es kann gezeigt werden, dass das Kollisionsintegral genau fünf Invarianten aufweist. Diese sind konsistent mit der Forderung nach Massen-, Impuls- und Energieerhaltung während des Kollisionsprozesses.

Abbildung 5.1: Zweistufige Diskretierung der kontinuierlichen Boltzmann-Gleichung. Ergebnis ist die Gitter-Boltzmann-Gleichung.

Durch Multiplikation der Boltzmann-Gleichung mit den Invarianten ψ_k (mit $k = 0, ..., 4$) und durch anschließende Integration über den mikroskopischen Geschwindigkeitsraum ergeben sich makroskopische Erhaltungsgleichungen.

Mit Hilfe des *Chapman-Enskog-Verfahrens* (Chapman & Cowling, 1939), einer aufwändigen Multi-Skalen-Analyse, lassen sich schließlich die Navier-Stokes-Gleichungen direkt aus der Boltzmann-Gleichung ableiten. Damit ist die Gleichwertigkeit der beiden Ansätze unter bestimmten Voraussetzungen wie kleine Mach- und Knudsen-Zahl nachweisbar. Näheres hierzu ist (Krafczyk, 2001), (Tölke, 2001) und (van Treeck, 2004) zu entnehmen.

5.2.2 Von der kontinuierlichen Boltzmann- zur Gitter-Boltzmann-Gleichung

Wie Abb. 5.1 zeigt, erfolgt die Diskretisierung der Boltzmann-Gleichung in zwei Schritten. Zunächst werden die Bewegungen des Teilchenensembles im mikroskopischen Raum hinsichtlich Geschwindigkeit und Richtung diskretisiert, wobei jedoch Zeit und Raum weiterhin als kontinuierlich betrachtet werden. Die Verteilungsfunktionen $f(t, \vec{x}, \vec{\xi})$ werden somit durch die Werte $f(t, \vec{x}, \vec{e}_i)$ an N sogenannten *Kollokationspunkten* bzw. *-richtungen* $\vec{e}_i$ dargestellt.

Mit der Ausbreitungsgeschwindigkeit c des numerischen Modells und der Einheitsbasis $\vec{e}_i$ ergibt sich damit für die diskreten miroskopischen Geschwindigkeiten $\vec{\xi}_i$

$$\vec{\xi}_i \;=\; c\,\vec{e}_i \quad . \tag{5.8}$$

Die Boltzmann-Gleichung geht damit in die *diskreten Boltzmann-Gleichungen* über, einem System von N partiellen Differentialgleichungen, deren Variablen lediglich vom Ort $\vec{x}$ und der Zeit t abhängen:

$$\frac{\partial f_i}{\partial t} \;+\; \vec{\xi}_i \,\frac{\partial f_i}{\partial \vec{x}} \;=\; -\frac{1}{\tau}\left(f_i - f_i^{(0)}\right) \quad \text{mit} \quad i = 1, ..., N \quad . \tag{5.9}$$

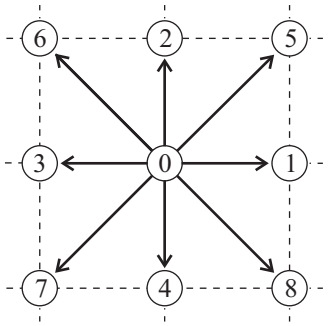


Abbildung 5.2: Mögliche Wahl der Kollokationspunkte im 2D-Fall (D2Q9-Modell). Gezeigt sind alle neun möglichen Positionen, die ein zum Zeitpunkt t_0 auf Punkt 0 befindliches (statistisches) Teilchenensemble zum Zeitpunkt $t_1 = t_0 + 1$ einnehmen kann.

Wahl der Kollokationspunkte. Die diskreten mikroskopischen Geschwindigkeiten müssen so gewählt werden, dass zum einen die im diskreten Phasenraum durch $\vec{e}_i$ definierten Zellen ein raumfüllendes numerisches Rechengitter bilden und sich zum anderen die korrekten makroskopischen Gleichungen aus der Chapman-Enskog-Analyse ergeben. Für eine ausführliche Darstellung sei hierbei auf (Shan & He, 1998) und (Tölke, 2001) verwiesen. Die am häufigsten verwendeten Modelle sind das $d2q9$-, das $d3q15$- und das $d3q19$-Modell, wobei in der Nomenklatur $dAqB$ A der Anzahl der Raumdimensionen und B der Anzahl der Kollokationspunkte entspricht. Abb. 5.2 zeigt beispielhaft das d2q9-Modell.

Für die räumliche und zeitliche Diskretisierung der diskreten Boltzmann-Gleichungen (5.9) verwendet man nun in der Regel ein Finite-Differenzen-Schema[7] mit explizitem Zeitschrittverfahren, was zur *Gitter-Boltzmann-Gleichung* (5.11) führt. Die räumlichen Differenzenquotienten entsprechen in ihrer Schrittweite dabei genau der durch die Kollokationspunkte aufgespannten Zelle, womit eine Übereinstimmung zwischen dem Gitter der mikroskopischen Geschwindigkeiten und dem numerischen Gitter erzwungen wird.

$$\frac{f_i(t+\Delta t,\vec{x}) - f_i(t,\vec{x})}{\Delta t} + c\,\frac{f_i(t+\Delta t,\vec{x}+\vec{e}_i\Delta x) - f_i(t+\Delta t,\vec{x})}{\Delta x} = -\frac{1}{\tau}\left(f_i(t,\vec{x}) - f_i^{(0)}(t,\vec{x})\right) \quad . \tag{5.10}$$

Mit $\Delta x = c\Delta t$ erhält man die *Gitter-Boltzmann-Gleichung*

$$f_i(t+\Delta t,\vec{x}+\vec{\xi}_i\Delta t) = f_i(t,\vec{x}) - \frac{\Delta t}{\tau}\left(f_i(t,\vec{x}) - f_i^{(0)}(t,\vec{x})\right) \quad . \tag{5.11}$$

Da im Rahmen dieser Arbeit ausschließlich äquidistante Gitter zum Einsatz kommen, gilt $\Delta x = c\Delta t = 1$, womit sich eine numerische Informationsausbreitungsgeschwindigkeit von $c = 1$ ergibt.

[7] Jüngere Arbeiten konnten auch die Anwendbarkeit eines Finite-Elemente-Schemas zur räumlichen Diskretisierung der diskreten Boltzmann-Gleichungen zeigen (Düster *et al.*, 2006).

5.2.3 Algorithmus zur Lösung der Gitter-Boltzmann-Gleichung

Der Algorithmus zur Lösung von Gleichung 5.11 teilt sich in zwei Unterschritte pro zu berechnendem Zeitschritt:

- Die Berechnung der jeweils neuen Verteilungsfunktionen hinsichtlich der rechten Seite von Gleichung 5.11, der sogenannten *Kollision*, und
- der Weiterleitung der Verteilungsfunktionen zu den Nachbarknoten, üblicherweise als *Propagation* bezeichnet.

Die für ingenieurtechnische Anwendungen relevanten makroskopischen Größen werden als Momente der Verteilungsfunktionen bestimmt und näherungsweise durch numerische Integration berechnet. Sie ergeben sich zu

$$\rho = \sum_{i=0}^{N-1} f_i \quad \text{und} \tag{5.12}$$

$$\rho\vec{u} = \sum_{i=0}^{N-1} \vec{e}_i f_i \quad . \tag{5.13}$$

Methoden zur effizienten programmiertechnischen Umsetzung dieses Algorithmus sind (Krafczyk, 2001), (Kühner, 2003) und (Crouse, 2003) zu entnehmen.

5.2.4 Weitere Entwicklungen

Neben der Forderung kleiner Mach- und kleiner Knudsen-Zahlen beschränkt die BGK-Approximation das Verfahren aus Stabilitätsgründen auf einen schmalen Anwendungsbereich. Das von d'Humières eingeführte Momentenmodell weist hingegen eine höhere numerische Stabilität auf und ermöglicht darüber hinaus deutlich reduzierte Werte für die kinematische Viskosität (d'Humières *et al.*, 2002; Krafczyk, 2001). Dabei werden die Verteilungen für die Kollision zunächst in einen physikalisch äquivalenten Momentenraum transformiert. Daraufhin werden anstelle der Verteilungsfunktionen die entstehenden Momente relaxiert, wobei die Relaxationsparameter für die Momente jeweils individuell eingestellt werden können. Anschließend werden die Momente wieder in den Raum der Verteilungen zurück transformiert.

Jüngere Arbeiten beschäftigen sich unter anderem mit der Integration von Turbulenzmodellen, wie beispielsweise dem Large-Eddy-Modell (Hou *et al.*, 1996) sowie mit der Anwendung von Multi-Grid- (Mavriplis, 2004) und adaptiven Verfahren (Crouse, 2003; Geller *et al.*, 2006).

5.3 Diskretisierung der Ausgangsgeometrie im numerischen Gitter

Um den Anforderung des *Computational Steering*-Aspekts hinsichtlich der Veränderbarkeit der geometrischen Randbedingungen bei laufender Strömungssimu-

lation gerecht zu werden, ist ein Verfahren zur performanten und vor allem automatisierten Diskretisierung der Ausgangsrepräsentation im numerischen Gitter notwendig.

Herkömmliche Finite-Volumen- und Finite-Element-Ansätze scheiden daher zur Umsetzung einer in diesem Sinne *steuerbaren* Strömungssimulation im Prinzip aus, da das hierbei verwendete Berechnungsnetz bei komplizierten geometrischen Randbedingungen in der Regel ein manuelles Eingreifen erfordert.

Das dem Gitter-Boltzmann-Verfahren zugrunde liegende uniforme kartesische Gitter ist hingegen hervorragend geeignet für eine automatisierte Diskretisierung der Ausgangsrepräsentation. Ziel ist es hierbei, in Anlehnung an den *Marker-and-Cell*-Ansatz (Harlow & Welch, 1965) diejenigen Knoten im Berechnungsgitter mit entsprechenden Randbedingungen zu markieren, die innerhalb bzw. auf dem Rand eines Hindernisses liegen (Kühner, 2003).

Für den Prozess der Überführung der Oberflächen-basierten Ausgangspräsentation[8] in diese spezielle Form des Normzellaufzählungsschemas (vgl. Abschnitt 8.1) hat sich der Begriff der *Voxelisation* durchgesetzt. Hintergrund ist die Bezeichnung *Voxel* (kurz für *Volume Element*) für eine würfelförmige Zelle im uniformen kartesischen Gitter analog zum *Pixel* in 2D.

Die von Wenisch & Wenisch entwickelte Vorgehensweise beruht auf dem Einsatz der hierarchischen raumpartionierenden Datenstruktur *Oktalbaum* (vgl. Abschnitt 8.2) für die Gittergenerierung (Wenisch & Wenisch, 2004). Dabei werden beginnend beim Ursprungsoktanten nur diejenigen Kindoktanten weiter verfeinert, die von einer Facette der Ausgangsgeometrie geschnitten werden. Damit nicht jeder Oktant auf einen Schnitt mit allen Facetten getestet werden muss, wird des Weiteren an jeden Kindoktanten eine Kandidatenliste übergeben, die nur diejenigen Facetten enthält, für die der Schnitttest mit dem Vateroktanten erfolgreich war.

Dieser rekursive Vorgang wird bis zum Erreichen der maximalen Verfeinerungsstufe ausgeführt, die sich aus der vorgegebenen Maschenweite des Gitters ergibt. Das Verfahren zeigt eine erstaunliche hohe Performanz auch bei komplexen Geometrien[9], was sich in einer gesteigerten Responsivität des gesamten *Computational Steering*-Systems bemerkbar macht.

Einschränkend ist zu erwähnen, dass der aus der Diskretisierung im kartesischen Gitter resultierende abgetreppte Rand die Genauigkeit des Gitter-Boltzmann-Verfahrens in diesem Bereich auf erste Ordnung beschränkt. Eine Erhöhung der Genauigkeit ist jedoch durch geeignete Interpolationsverfahren möglich (Ginzburg & d'Humières, 1996).

[8] Boundary Representation (B-Rep)

[9] Auf einem *Dual Opteron*-Rechner mit 2.4 GHz Prozessortakt konnte ein Objekt mit 30014 Facetten in lediglich 190 ms in eine Voxel-Repräsentation überführt werden (Wenisch *et al.*, 2007).

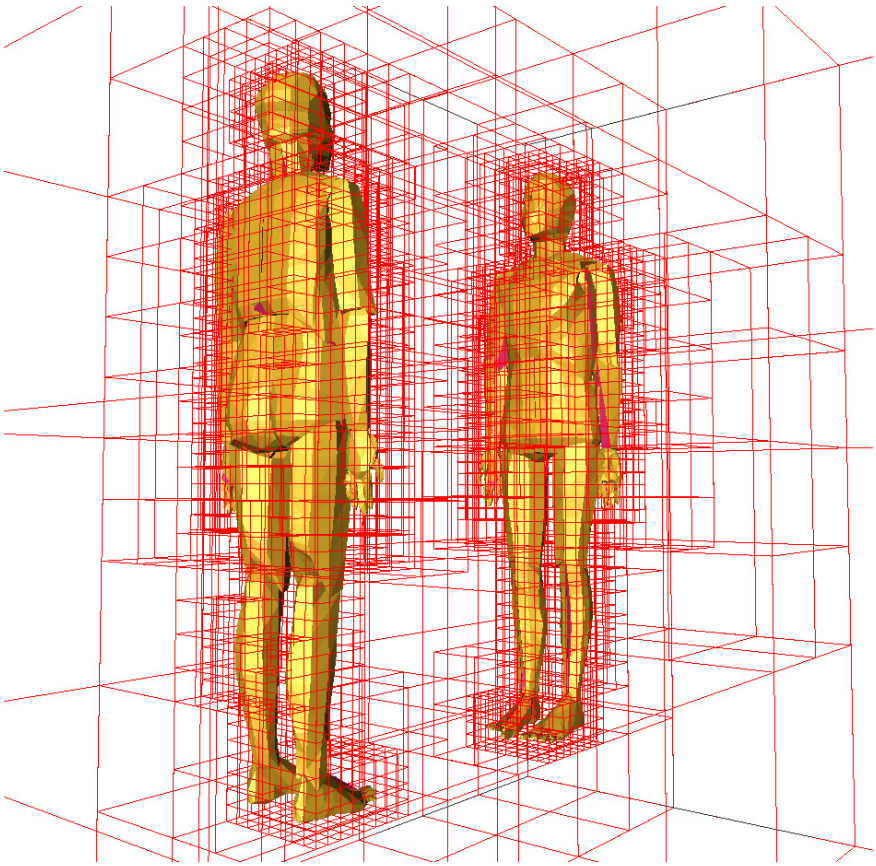


Abbildung 5.3: Diskretisierung von komplexen Dummy-Geometrien im numerischen Gitter mit Hilfe einer Oktalbaumstruktur.

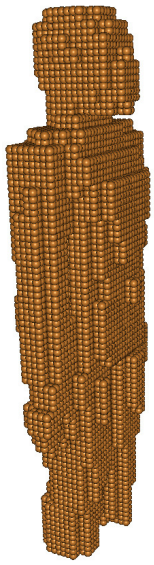


Abbildung 5.4: Ergebnis der Diskretisierung einer Dummy-Geometrie. Aus Gründen der besseren Darstellbarkeit werden die als Hindernis markierten Voxel kugelförmig visualisiert.

Wie bereits in Abschnitt 4.8 erwähnt, hält der Gitter-Boltzmann-Simulationsserver eine lokale Kopie des gemeinsamen (Teil-)Modells, inklusive aller geometrischen Informationen. Sobald der Simulationsserver vom Kollaborationsserver eine Benachrichtigungen über Veränderungen an der geometrischen Szene empfängt, wird der Voxelisierungsprozess für die gesamte Szene erneut angestoßen.

5.4 Parallelisierung

Unter Parallelisierung versteht man die Aufteilung eines Prozesses auf mehrere parallel arbeitende Teilprozesse. Führt man diese Teilprozesse auf verschiedenen physischen Prozessoren aus, kann eine deutlich höhere Gesamtleistung des Systems erreicht werden.

Mit der Parallelisierung des Gitter-Boltzmann-Simulationscodes werden zwei Ziele verfolgt:

- Erhöhung der Responsivität des *Computational Steering*-Systems durch möglichst kurze Reaktionszeiten, d.h. kurze Zeiträume bis zum Einschwingen der Simulation in einen stationären Zustand,
- Erhöhung der Genauigkeit durch Verwendung eines Berechnungsgitters mit engerer Maschenweite, d.h. Erhöhung der Anzahl der Berechnungsknoten.

Hardwarearchitekturen für Parallelrechner werden nach der Art des Speicherzugriffs in *Distributed Memory*- und *Shared Memory*-Systeme unterschieden. Beim *Shared Memory*-System (SMP) handelt es sich um eine Mehrprozessorarchitektur, bei der alle Prozessoren auf einen gemeinsamen Speicher zugreifen. Im Ge-

gensatz dazu besitzt bei der *Distributed Memory*-Architektur jeder Prozessor seinen eigenen exklusiv verfügbaren Speicher. Bei Verwendung einer großen Zahl von Prozessoren, werden Rechner letzteren Typs auch als MPP-Systeme (Massively Parallel Processing) bezeichnet. Unter den sogenannten *Supercomputern*, den 500 derzeit schnellsten Rechnern weltweit, finden sich sowohl Vertreter von *Distributed Memory*- als auch *Shared Memory*-Systemen. Parallelsysteme gewinnen jedoch auch außerhalb des Höchstleistungsrechnens zunehmend an Bedeutung, *Distributed Memory*-Architekturen vor allem im Form von Workstation-Clustern und *Shared Memory*-Systeme durch die seit kurzem verfügbaren Multi-Core-Prozessoren für Arbeitsplatzrechner (Bode, 2006).

Im Rahmen des Forschungsprojekts ViSimLab[10] wurde ein Gitter-Boltzmann-Code für den damaligen Bundeshöchstleistungsrecher vom Typ Hitachi SR-8000 optimiert (Wenisch, 2008). Dabei handelte es sich auf der oberen Ebene um ein MPP-System mit 168 Knoten. Jeder einzelne Knoten stellte wiederum ein SMP-System mit acht Prozessoren und gemeinsamem Speicherbereich dar. Die Einzelprozessoren arbeiteten auf Grundlage eines reduzierten Befehlssatzes (RISC-Prozessoren) und konnten während eines Prozessortakts mehrere Operationen gleichzeitig ausführen (Pseudo-Vektorisierung).

Um die Leistungsfähigkeit des Gesamtsystems voll auszuschöpfen, musste der auszuführende Code für jede dieser Schichten optimiert bzw. angepasst werden. Um die Funktionalität der *Vektorisierung* zu nutzen, muss beispielsweise die Struktur der Programmschleifen so gewählt werden, dass eine fließbandartige Verarbeitung möglich ist. Die Vektorisierung selbst wird dann durch spezielle Compiler-Anweisungen aktiviert, ebenso wie die Nutzung mehrerer Prozessoren innerhalb des SMP-Systems. Auf der MPP-Ebene ist hingegen eine explizite Programmierung des Datenaustauschs zwischen den Knoten erforderlich, wie sie beispielsweise mit Hilfe von *Message Passing*-Bibliotheken wie MPI (s.u.) realisiert werden kann.

Hinsichtlich der Parallelisierung auf MPP-Ebene bietet sich bei numerischen Berechnungen grundsätzlich die Aufteilung des Berechnungsgebietes in geometrische Zonen an. Jede dieser Zonen wird dabei einem dezidierten Prozess im parallelen Verbund zugewiesen. Bei dem hier vorgestellten Gitter-Boltzmann-Code wurde eine Aufteilung des Berechnungsgebietes entlang der längsten Achse des Strömungsgebiets in (etwa) gleich große Abschnitte realisiert (scheibenweise Aufteilung in Domänen) (Wenisch *et al.*, 2005a,b). Damit wird eine weitestgehend ausgeglichene Lastverteilung auf die einzelnen Prozesse erzielt.

Der Gitter-Boltzmann-Algorithmus erlaubt eine einfache und effiziente Parallelisierung, wie eine nähere Betrachtung seiner zwei Teilschritte zeigt: Die für die Berechnung der *Kollision* notwendigen Größen, die Verteilungsfunktionen, liegen jeweils lokal am betrachteten Gitterpunkt vor. Ein Zugriff auf Daten über Domänengrenzen hinweg ist demnach nicht notwendig. Beim *Propagationsschritt* werden zudem Daten nur zwischen benachbarten Knoten ausgetauscht. Liegt ein

[10] `http://konwihr.in.tum.de/projekte/visimlab.html`

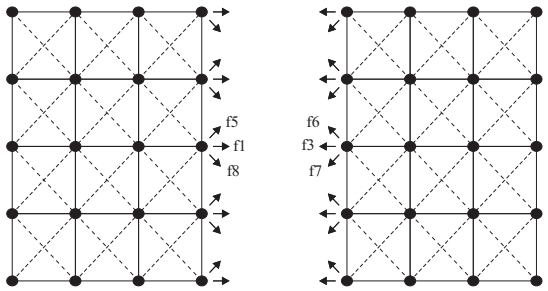


Abbildung 5.5: Austausch von Verteilungsfunktionen über die Ränder eines Teilgebiets hinweg. Bei einer Gebietszerlegung entlang der x-Achse müssen nur Verteilungsfunktionen mit einer in x-Komponente ausgetauscht werden.

Knoten also am Rand eines Teilberechnungsgebietes müssen dessen entsprechende Verteilungsfunktionen an einen Knoten in einem anderen Teilberechnungsgebiet weitergegeben werden. Gerade im Vergleich zur aufwändigen Parallelisierung von Gleichungslösern ist hier ein einfaches und robustes Kommunikationsschema implementierbar. Wie Abb. 5.5 für den 2D-Fall zeigt, müssen bei einer Gebietszerlegung entlang der x-Achse nur diejenigen Verteilungen zwischen den parallelen Prozessen ausgetauscht werden, die einen Anteil an der x-Koordinate besitzen.

Für den bei der Parallelisierung notwendigen Datenaustausch wird eine Softwarebibliothek verwendet, die die Standardschnittstelle MPI (Message Passing Interface) implementiert (Pacheco, 1996). Dabei handelt es sich im Gegensatz zu der innerhalb der CoCoS-Plattform verwendeten CORBA-Technologie (siehe Abschnitt 4.1), um eine statische Schnittstelle mit einem fest vorgegebenen Umfang aufrufbarer Bibliotheksfunktionen, die vor allem für prozedurale Sprachen wie C und FORTRAN geeignet ist.

Aufgabe des Gitter-Boltzmann-Simulationsservers ist es daher, eine Brücke zwischen den verschiedenen Kommunikationstechniken herzustellen, d.h. die übertragenen Daten anzunehmen und in das jeweils andere Kommunikationssystem zu übersetzen. Abb. 5.6 zeigt die Anbindung des parallelisierten Rechenkernels an das verteilte CORBA-basierte System von CoCoS.

Wie Abb. 5.6 weiterhin zeigt, wurde auf der Seite des Simulationskerns ein zusätzlicher Kommunikationsknoten eingeführt. Dieser Prozess sammelt die Simulationsdaten aller Rechenknoten und leitet sie in einem gemeinsamen Datenpaket weiter an den Simulationsserver. Der Kommunikationsknoten übernimmt des Weiteren die Domänenaufteilung zu Beginn des Simulationsprozesses sowie die Diskretisierung der triangularisierten Oberflächengeometrie im Berechnungsgitter (Abschnitt 5.3) beim Einfügen und Bewegen von Hindernissen bzw. Verändern von Randbedingungen (Wenisch *et al.*, 2007).

Für Effizienzanalysen des parallelisierten Gitter-Boltzmann-Codes sei hier auf (Kühner, 2003), (Wenisch *et al.*, 2005b) und (Wenisch *et al.*, 2007) verwiesen.

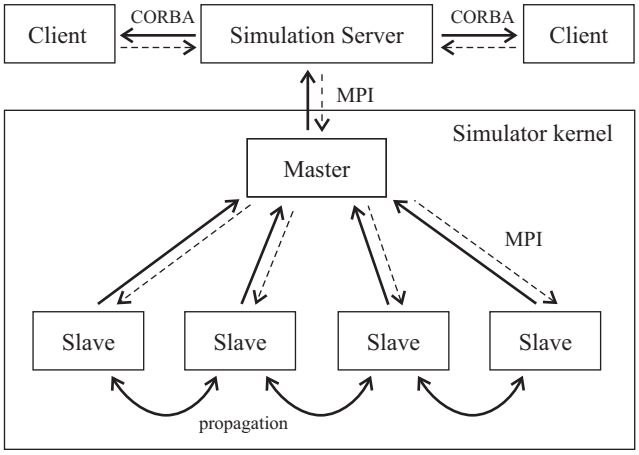


Abbildung 5.6: Anbindung des parallelisierten Rechenkernels an die verteilte CoCoS-Plattform. Während der Rechnenkernel MPI zum Datenaustausch verwendet, wird die Kommunikation innerhalb der CoCoS-Plattform mittels CORBA realisiert.

5.5 Zusammenfassung

In diesem Kapitel wurde die Umsetzung eines Strömungssimulationsservers für die CoCoS-Plattform vorgestellt. Wie gezeigt werden konnte, ist das Gitter-Boltzmann-Verfahren als Grundlage für eine interaktive Strömungssimulation besonders geeignet. Der wesentliche Vorteil dieser numerischen Methode gegenüber der weit verbreiteten Finite-Volumen-Methode liegt dabei in der Verwendung eines kartesischen numerischen Gitters, das die automatisierte Diskretisierung der Geometrie von Strömungshindernissen und Randobjekten ermöglicht. Dies ist für interaktive Strömungssimulationen mit zur Laufzeit änderbaren geometrischen Randbedingungen von essentieller Bedeutung. Gleichzeitig erlaubt sie eine deutlich höhere maximale Element-Reynoldszahl als bei Anwendung des Finite-Differenzen-Verfahrens und zeigt keinerlei numerische Viskosität.

Zudem ist dank der algorithmischen Struktur der Gitter-Boltzmann-Methode eine Vektorisierung der Programmschleifen und eine effiziente Parallelisierbarkeit auf MPP-Ebene mit vergleichsweise geringem Datenübertragungsaufkommen möglich. Dies unterstreicht die Eignung der Gitter-Boltzmann-Methode vor allem im Kontext des Hochleistungsrechnens.

Teil II

Eine räumliche Anfragesprache für digitale Bauwerksmodelle

Kapitel 6

Überblick

6.1 Motivation und Einführung

Bauwerke werden vom Menschen in erster Linie als eine strukturierte Menge physischer Objekte mit wohldefinierter Gestalt wahrgenommen. In der Regel leitet sich die Funktion eines Bauteils aus seiner Form ab oder hängt stark mit ihr zusammen. Auch für die an der Planung beteiligten Architekten, Ingenieure und Fachplaner spielen räumlich-geometrische Zusammenhänge eine wesentliche Rolle bei der Lösung von Design- und Konstruktionsaufgaben. Bislang fehlt es jedoch an Software-Werkzeugen, die eine räumliche Analyse von Bauwerksmodellen in angemessener Weise unterstützen.

Die fehlende Unterstützung geometrisch-topologischer Analysefähigkeiten von Software zur Verwaltung von Bauwerksmodellen ist u.a. damit zu erklären, dass sich die Forschung im Bereich computergestützter Bauplanung in den letzten Jahren verstärkt auf die Entwicklung eines semantischen objektorientierten Bauwerkmodells konzentriert hat[1] (siehe Abschnitt 2.2.3). Dieses Modell beschreibt die Geometrie von Bauteilen nicht explizit, d.h. nicht mit Hilfe von BRep- oder CSG-Methoden, sondern auf Basis von Objekt-Attributen, die geometrische Semantik tragen. Die wesentliche Motivation dieses Ansatzes der „attribute-driven geometry“ liegt darin, dass sich aus einer abstrakten Beschreibung sowohl Symbole für zweidimensionale Pläne als auch dreidimensionale geometrische Modelle ableiten lassen. Die Notwendigkeit des Erzeugens dimensionsreduzierter bzw. symbolischer Beschreibungen von Bauteilen ist vor allem mit der nach wie vor starken Abhängigkeit der Bauindustrie von druckbaren Plänen zu erklären.

In der Regel können die zur Verwaltung dieser abstrakten Modelle eingesetzten Systeme, die sogenannten *Produktmodellserver*, die implizit im Modell enthaltenen geometrischen Informationen jedoch nicht interpretieren, da ihnen die räumliche Semantik einzelner Objekte bzw. ihrer Attribute und Beziehungen nicht

[1] Diese Bemühungen resultierten im standardisierten Objektmodell *Industry Foundation Classes* (IFC) (IAI - International Alliance for Interoperability, 2004).

bekannt ist. Das führt dazu, dass die Mächtigkeit der Anfragesprache dieser Modellverwaltungssysteme auf die Behandlung einfacher numerischer Relationen (größer, kleiner, gleich) zwischen einzelnen geometrischen Attributen beschränkt bleibt. Komplizierte topologische Situationen (wie Bauteil A *berührt* Bauteil B) sind auf diese Weise nur mühsam und zum Teil gar nicht formulierbar.

Im Folgenden soll daher das Konzept einer Softwarelösung vorgestellt werden, die diese Lücke schließt, d.h. die es dem Anwender ermöglicht, mit Hilfe einer Anfragesprache Gebäudemodelle hinsichtlich räumlicher Kriterien zu analysieren. Zur Illustration der Mächtigkeit einer solchen Lösung sollen einige informelle Beispiele für analytische Anfragen mit räumlich-topologischer Semantik aufgeführt werden. Zum einen sind rein analytische Anfragen möglich, die eine boolesche Aussage oder einen numerischen Wert als Ergebnis liefern:

- „*Durchlaufen* Gasleitungen den Sektor 23?“
- „Wie viele Heizkörper befinden sich *in* Raum 107?“
- „Gibt es Objekte *im Umkreis von* 10 Metern?“

Daneben bietet die vorzustellende Anfragesprache jedoch auch die Möglichkeit der *Selektion* von Bauteilen, die bestimmten Bedingungen mit räumlich-topologischer Semantik genügen:

- „Finde alle Wände *im* ersten Geschoss.“
- „Welche Stützen *berühren* Decke 1?“
- „Finde alle Stahlträger *unterhalb* der ersten Platte.“

Mit Hilfe derartiger Selektionen können Partialmodelle aus einem zentral verwalteten Bauwerksmodell gewonnen werden. Im Kontext asynchroner kooperativer Arbeit (vgl. Abschnitt 2.1.3) ist die Partialmodellbildung eine wesentliche Voraussetzung für eine reibungsfreie Zusammenarbeit, da Partialmodelle quasi die „Hoheitsgebiete“ der einzelnen Planer markieren, also die Teile des Modells, in denen der einzelne Planer weitestgehend konfliktfrei, d.h. ohne weitere Abstimmungen, Entscheidungen treffen kann.

Dank der Möglichkeit der Definition des Partialmodells durch Bedingungen mit räumlich-topologischer Semantik, können aus einem Gesamtmodell beispielsweise Bauteile extrahiert werden, die sich innerhalb eines bestimmten Raums oder einer bestimmten Etage befinden oder an ein bestimmtes Bauteil angrenzen. Die Möglichkeit der Bildung von Partialmodellen anhand von räumlich-topologischen Kriterien ist als ein wesentlicher Baustein zur besseren Unterstützung der Kooperation im Planungsprozess zu sehen.

Eine Zerlegung des Bauwerksmodells auf Grundlage räumlich-topologischer Bedingungen ist darüber hinaus auch für die Ableitung von Eingangsmodellen für numerische Analysen und Simulationen, wie beispielsweise für die Strukturanalyse (Romberg, 2005), zonale thermische Simulationen sowie Strömungssimulationen (van Treeck, 2004) von Bedeutung. Die räumliche Partitionierung ist in diesem Zusammenhang zum einen mit Hinblick auf eine in der Regel beschränkte verfügbare Rechenleistung sinnvoll, zum anderen kann mit ihrer Hilfe eine Aus-

wahl der für die Simulation relevanten Objekte erfolgen. Bei einer Innenraumströmungssimulation sind dies beispielsweise die Objekte, die sich *innerhalb* des Raums befinden, für den die Luftströmung simuliert werden soll.

Schließlich kann eine deklarative Sprache mit räumlichen Semantik auch für die Definition von Konsistenzbedingungen verwendet werden. Ihre Einhaltung kann am Ende des isolierten Bearbeitens eines Partialmodells geprüft werden, also bevor eine Wiedereingliederung in das Gesamtmodell (Merging) erfolgt, um auf diese Weise die Konsistenz des Gesamtmodells zu gewährleisten (Borrmann *et al.*, 2004).

Notwendige Voraussetzung für die Entwicklung einer räumlichen Anfragesprache ist die formale Definition einer abstrakten Algebra mit räumlichen Typen und auf sie anwendbaren Operatoren. Diese Definitionen werden in Kapitel 7 zur Verfügung gestellt.

In Kapitel 8 werden Möglichkeiten zur Umsetzung der räumlichen Operatoren auf Basis von oktalbaumcodierter Rastergeometrie aufgezeigt. Nach der vorangegangen Definition der Semantik dieser Operatoren im kontinuierlichen Raum wird auf diese Weise eine Diskretisierung, d.h. eine Überführung in den diskreten Raum vollzogen. In Analogie zu Diskretisierungsverfahren der numerischen Mathematik (Finite Differenzen, Finite Elemente etc.) kann mit der Wahl einer feineren Diskretisierung auch die Größe des Fehlers begrenzt werden, der bei Verwendung der Algorithmen potentiell eintritt. Die Oktalbaumcodierung unterstützt diesen Ansatz durch die ihr inhärente Hierarchie der Raumpartitionierung.

Eine mögliche Umsetzung der eigentlichen Anfragefunktionalität auf der Basis von Datenbanktechnologie wird schließlich in Kapitel 9 vorgestellt. Datenbanken spielen traditionell eine wichtige Rolle bei der Verwaltung und Analyse großer Datenmengen, die von einer Vielzahl von Beteiligten gleichzeitig gelesen und geschrieben werden. Neben ihrer Hauptfunktionalität, der Gewährleistung einer dauerhaften Speicherung der Daten (Persistenz), verfügen Datenbanken in der Regel über leistungsfähige Mechanismen zur Nebenläufigkeitskontrolle und Konsistenzsicherung. Der im Rahmen dieser Arbeit wichtigste Aspekt ist jedoch die Verfügbarkeit einer deklarativen Anfragesprache, die es dem Nutzer erlaubt, ohne Wissen interner Implementierungsdetails eine Ergebnismenge zu spezifizieren. Die effiziente Umsetzung der entsprechenden Suche obliegt allein der Datenbank. Diese klare Trennung sorgt zum einen dafür, dass der Nutzer jederzeit mit „optimalen“ Antwortzeiten rechnen kann, zum anderen können die zugrundeliegenden Suchalgorithmen modifiziert werden, ohne dass sich auf Nutzerseite Änderungen ergeben.

Die im Bereich (objekt-) relationaler Datenbanksysteme etablierte und weit verbreitete deklarative Sprache SQL (Structured Query Language) bietet eine geeignete Basis zur Umsetzung der abstrakten Algebra in einer Anfragesprache. Die jüngste Version des ISO-Standards, SQL:1999, erweitert das relationale Modell um objektorientierte Aspekte, darunter die Möglichkeit der Definition eigener Da-

tentypen und dazugehöriger Methoden. Datenbanksysteme, die diesen Standard unterstützen, werden als *objekt-relationale* Datenbanksysteme bezeichnet.

Diese Arbeit geht vom Vorliegen eines expliziten Geometriemodells in einer Datenbank aus. Techniken zur Gewinnung einer expliziten Geometriebeschreibung aus einem abstrakten Objektmodell (wie beispielsweise den IFC) liegen nicht im Fokus dieser Arbeit und können anderen Veröffentlichungen wie (Romberg *et al.*, 2004) entnommen werden.

6.2 Verwandte Arbeiten

Das hier vorzustellenden Gesamtkonzept zur Definition und Umsetzung einer räumlichen Anfragesprache lehnt sich weitestgehend an Technologien und Konzepte aus dem Bereich Geographischer Informationssysteme (GIS) an. Diese Systeme verwalten geographische Daten wie die Lage und Ausdehnung von Straßen, Städten etc. und stellen Funktionalitäten zur räumlich-topologischen Analyse zur Verfügung. Der Natur der Fachdomäne bedingt, unterstützen die meisten GI-Systeme lediglich zweidimensionale geometrische Objekte, die jedoch dank ihrer Ausdehnung in mehr als einer Dimension ebenso als räumliche Objekte (engl. *spatial objects*) bezeichnet werden (Shekhar & Chawla, 2003).

Aus dem GIS-Bereich stammen ebenfalls die ersten Ansätze zur Umsetzung einer räumlichen Anfragesprache auf Basis von SQL: Es entstand eine Vielzahl von Dialekten, darunter die *pictorial query language* PSQL von (Roussopoulos *et al.*, 1988), Spatial SQL von (Egenhofer, 1987), GEOQL von (Ooi *et al.*, 1989), KGIS von (Ingram & Phillips, 1987) und TIGRIS von (Herring *et al.*, 1988). Einen guten Überblick über die einzelnen Dialekte sowie Vor- und Nachteile der Umsetzung einer räumlichen Anfragesprache auf Basis von SQL gibt (Egenhofer, 1992).

Im Rahmen der Entwicklung von GI-Systemen wurde auch der Begriff der räumlichen Datenbank (engl. *spatial database*) geprägt. Hierbei handelt es sich um Datenbanksysteme, die durch entsprechende Datentypen und Indizierungstechniken die Speicherung und Analyse von räumlichen Datentypen vereinfachen und beschleunigen. Eine eingehende Diskussion zu räumlichen Datenbanken und den damit verbundenen Techniken ist in Kapitel 9 zu finden. Mittlerweile gibt es eine ganze Reihe kommerziell verfügbarer räumlicher Datenbanken, darunter *PostGIS*, *Oracle Spatial* und *Informix Geodetic Datablade*, von denen die meisten dem vom OpenGIS Consortium (OGC) entwickelten Standard genügen. Dieser legt eine einheitliche Schnittstelle für den Zugriff auf zweidimensionale räumliche Daten fest und ermöglicht damit die Austauschbarkeit der einer GIS-Applikation zugrunde liegenden räumlichen Datenbank (OpenGIS Consortium (OGC), 1999).

Systeme, die dreidimensionale räumliche Daten und Beziehungen verarbeiten können, sind bislang jedoch ausschließlich im Forschungsbereich zu finden. Im Kontext geographischer Informationssysteme ist vor allem die dreidimensionale Modellierung des Geländereliefs von Interesse, daneben aber auch die Darstellbarkeit von Gebäuden und unterirdischen Ablagerungen. Zu den wichtigsten Arbei-

ten hierzu zählen die Veröffentlichungen von Breunig *et al.* (Breunig *et al.*, 1994; Breunig, 1995; Breunig *et al.*, 2001; Balovnev *et al.*, 2004), die mit dem *GeoToolkit* ein objektorientiertes System zur effizienten Speicherung und Verarbeitung dreidimensionaler geographischer und geologischer Daten entwickelt haben. Das System bietet objektorientierte Schnittstellen, stellt jedoch keine deklarative Anfragesprache zur Verfügung.

Die Umsetzung von räumlicher Anfragefunktionalität auf der Basis eines objektrelationalen Datenbanksystems, wie sie auch in der vorliegenden Arbeit propagiert wird, wurde erstmals in (Vijlbrief & van Oosterom, 1992) diskutiert, wo mit GEO++ ein GIS-System vorgestellt wird, das auf dem erweiterbaren Datenbanksystem PostGRES beruht. GEO++ wurde später um dreidimensionale Typen und entsprechende Analysefunktionen erweitert (van Oosterom *et al.*, 1994).

Auch außerhalb des GIS-Bereichs gab es Ansätze zur Entwicklung von räumlicher Anfragefunktionalität. Ein frühe Arbeit in Richtung der Integration von räumlicher Analyse und CAD ist (Rosenthal *et al.*, 1984). Hier wird gezeigt, wie zweidimensionale CAD-Modelle in einer Datenbank gespeichert werden können. Die für diese CAD-Datenbank entwickelte Anfragesprache stellt u.a. geometrische Prädikate zur Verfügung wie *schneidet* und *beinhaltet*.

Die Idee, geometrische Prädikate in einer Anfragesprache bereitzustellen, wird von Güting *et al.* vertieft, die eine *Algebra* für geometrische Objekte und dazugehörige Operationen als Erweiterung der relationalen Algebra entwickelten (Güting, 1988). Dieses Konzept wurde in den sich anschließenden Forschungsprojekten intensiv verfolgt und führte u.a. zur Entwicklung einer Algebra für geometrische Objekte im diskreten zweidimensionalen Raum $\mathbb{Z}^2$ (Güting & Schneider, 1995) und für unscharfe räumliche Objekte (Behr & Güting, 2005).

Kriegel *et al.* haben ein datenbankbasiertes System zur Unterstützung räumlicher Analysefähigkeiten für Modelle des Automobilbaus entwickelt (Kriegel *et al.*, 2003; Pötke, 2001; Pfeifle, 2004), von dem jedoch weder topologische noch direktionale Prädikate, sondern lediglich einfache Volumen-, Kollisions- und Abstandsanfragen unterstützt werden. Zur Umsetzung der Anfragefunktionalität wird die Geometrie der Bauteile als Voxeldarstellung approximiert in der Datenbank vorgehalten und eine spezielle Indexstruktur geschaffen.

In Hinblick auf die Verwendung räumlicher Analysefunktionalität für Bauwerksmodelle sind die in den besprochenen Arbeiten entwickelten Konzepte nur bedingt geeignet. Beispielsweise sind die Methoden zur geometrischen Beschreibung geologischer Formationen nicht uneingeschränkt auf Bauwerksmodelle anwendbar. In den folgenden Kapiteln soll daher die Anpassung bzw. Weiterentwicklung der bereits bekannten Konzepte auf die Problemstellung einer räumlichen Anfragesprache für Bauwerksmodelle dargelegt werden.

Kapitel 7

Definition einer abstrakten räumlichen Algebra

Die umgangssprachliche Verwendung räumlicher Begrifflichkeiten bzw. Beziehungen ist oftmals ungenau bzw. vieldeutig und kann zu Missverständnissen sowie im Bereich der computergestützten Informationsverarbeitung zu falschen Ergebnissen führen. Daher soll in diesem Kapitel die Semantik der räumlichen Typen und der für sie verfügbaren räumlichen Operatoren mit Hilfe mathematischer Methoden formal definiert werden. Dies stellt den Ausgangspunkt der Entwicklung einer räumlichen Anfragesprache dar.

Ein System aus räumlichen Typen und Operatoren wird auch als „Räumliche Algebra“ (Güting & Schneider, 1995) bezeichnet. Es reflektiert die grundlegenden Abstraktionen räumlicher Entitäten und ihrer Beziehungen.

7.1 Der Algebra-Begriff im Kontext von Anfragesprachen

Nach (Goos, 1997) wird ein Tripel *(Menge, Operationen, Gesetze)* als *Algebra* bezeichnet. Ein typischer Vertreter ist die *Elementare Algebra* der Schulmathematik, bei der die Menge durch die ganzen, gebrochenen bzw. reelen Zahlen gebildet wird, die verfügbaren Operationen Addition, Subtraktion, Multiplikation und Division umfassen und als anwendbare Gesetze das Distributiv-, das Kommunikativ- und das Assoziativgesetz zur Verfügung stehen. Weitere Beispiele sind die boolesche und die Mengenalgebra.

Das Konzept der Algebra wird im vorliegenden Kontext verwendet, um räumliche Datentypen und Operatoren formal zu beschreiben und damit die Grundlage für eine SQL-basierte räumliche Anfragesprache zu schaffen. Ein wesentlicher Aspekt der Anwendung des Algebra-Konzepts bei der Datenmodellierung ist die Geschlossenheit des Typsystems, die auch als *algebraische Abgeschlossenheit* bezeichnet wird. Sie impliziert, dass keiner der Operatoren ein Ergebnis generieren

kann, das nicht durch die Definitionen der in der Algebra definierten Typen abgedeckt ist.

Im Kontext von Datenbanken ist die Verwendung des Algebra-Begriffs stark durch die von Codd entwickelte *relationale Algebra* (Codd, 1970) geprägt, auf die in Kapitel 9 näher eingegangen wird. Sie bildet die formale Grundlage der weit verbreiteten Anfragesprache SQL.

In (Güting, 1988) wird der Begriff Algebra eingesetzt, um eine Sammlung von Datentypen und zugehörigen Operatoren für einen spezifischen Anwendungsbereich zu beschreiben. Als Beispiel wird eine „Geo-relationale Algebra“ als Erweiterung der relationalen Algebra um geometrische Datentypen mit entsprechenden Operatoren vorgestellt. Aufbauend auf diesem Konzept wurde später die *Realm*-basierte räumliche Algebra ROSE entwickelt, die der finiten Natur der rechnerinternen Repräsentation von Geometrie Rechnung trägt (Güting & Schneider, 1995; Schneider, 1997).

Die Idee der Erweiterung der relationalen Algebra um geometrische bzw. räumliche Aspekte wurde auch von anderen Forschungsgruppen verfolgt. So wird in (Gargano *et al.*, 1991) die SHAPES-Algebra eingeführt, die u.a. den abstrakten Datentyp *Geometry* beinhaltet. In (Breunig, 1995) wird die räumliche Algebra EKOM vorgestellt, die auf der Verwendung von Zellkomplexen zur geometrischen Beschreibung basiert (siehe Abschnitt 7.2.2). In (Schneider & Weinrich, 2004) wird von der Entwicklung einer räumlichen Algebra für den dreidimensionalen Raum mit Namen SPAL3D berichtet, jedoch werden ausschließlich die verfügbaren Datentypen vorgestellt.

Die Idee der Unabhängigkeit eines Datenbanksystems von einer spezifischen Algebra resultierte schließlich in der Entwicklung von SECONDO, einem Datenbanksystem, das dem Anwendungsprogrammierer auf generische Weise erlaubt, die der Anfragesprache zugrundeliegende Algebra selbst festzulegen (Dieker & Güting, 2000; Güting *et al.*, 2005). Beispielhaft wurde eine Algebra für unscharfe räumliche Objekte entwickelt und mit Hilfe von SECONDO umgesetzt (Behr & Güting, 2005).

Im Rahmen dieser Arbeit wird das Konzept der Algebra verwendet, um die Semantik der in der räumlichen Anfragesprache verfügbaren Datentypen und Operatoren unabhängig von der konkreten Syntax einer Datendefinitionssprache zu beschreiben.

7.2 Räumliche Modellierungstechniken

Grundlage für die Definition räumlicher Beziehungen ist die Modellierung räumlicher Objekte mit Hilfe formaler Methoden. Die eingesetzte Modellierungsmethodik ist dabei entscheidend für die Verwendbarkeit des Modells bzw. für seine Aussagekraft. Im Folgenden sollen einige in der Forschung verbreitete Ansätze zur räumlichen Modellierung vorgestellt werden.

7.2.1 Punktmengentheorie

Die Verwendung der Punktmengentheorie ist ein weit verbereiteter Ansatz zur formalen Definition räumlicher Typen. Dabei wird jedes räumliche Objekt als eine Menge von Punkten im Raum aufgefasst. Diese Menge ist zwar im Allgemeinen unendlich groß, es ist aber möglich, sie mit finiten Methoden zu beschreiben. Eine der am häufigsten verwendeten finiten Methoden zur Beschreibung solcher Punktmengen ist die *Boundary Representation*, bei der ein räumliches Objekt über seinen Rand definiert wird. Beispielsweise wird eine Region oft durch das sie umgebende Polygon beschrieben.

Da Punktmengen *Mengen* im mathematischen Sinne sind, können elementare Operationen der allgemeinen Mengentheorie angewandt werden. Mengenoperationen wie *Vereinigung, Schnitt* und *Differenz* erlauben es, neue Objekte aus bereits bestehenden zu konstruieren. Eine Region mit einem Loch kann beispielsweise als die mengentheoretische Differenz zwischen der äußeren und der inneren Region betrachtet werden.

Güting modelliert in seiner geo-relationalen Algebra (Güting, 1988) räumliche Objekte als Punktmenge im zweidimensionalen Raum. Er definiert dabei die drei Datentypen *points*, *lines* und *regions*. Für die Anwendung der booleschen Operationen *Schnitt, Vereinigung* und *Differenz* auf Instanzen dieser Typen werden verschiedene Funktionen zur Verfügung gestellt, die sich hinsichtlich des Typs der Operanden und des erwarteten Ergebnistyps unterscheiden. Beispielsweise gibt die Funktion *lrsect(l,r)* den Schnitt zwischen einem *lines*- und einem *regions*-Objekt als eine Menge von *lines*-Objekten zurück. Da es möglich ist, dass das Ergebnis dieser Operation einzelne Punkte enthält und damit kein reguläres *lines*-Objekt darstellt, wird der Durchschnitt als „regularisiert“ definiert (siehe Abschnitt 7.4.5).

Darauf aufbauend können die räumlichen Beziehungen *equal, unequal, inside, outside* und *intersects* mit Hilfe von Mengenoperationen beschrieben werden (Güting, 1988):

$$
\begin{aligned}
x = y &\Leftrightarrow \mathit{points}(x) = \mathit{points}(y) \\
x \neq y &\Leftrightarrow \mathit{points}(x) \neq \mathit{points}(y) \\
x \text{ inside } y &\Leftrightarrow \mathit{points}(x) \subseteq \mathit{points}(y) \\
x \text{ outside } y &\Leftrightarrow \mathit{points}(x) \cap \mathit{points}(y) = \varnothing \\
x \text{ intersects } y &\Leftrightarrow \mathit{points}(x) \cap \mathit{points}(y) \neq \varnothing \quad .
\end{aligned}
$$

Der Nachteil dieser Definitionen ist jedoch, dass sie weder das gesamte Spektrum topologischer Beziehungen abdecken noch eineindeutig sind. Beispielsweise werden die Konstellationen *equal* und *inside* ebenfalls durch die Definition von *intersects* abgedeckt. Die wesentliche Schwäche des rein punktmengentheoretischen Ansatzes liegt darin, dass er keine Definitionen erlaubt, die auf bestimmten topologisch relevanten Teilmengen der betrachteten räumlichen Objekte beruhen, wie ihrem *Inneren* oder ihrem *Rand*.

Rein mengentheoretische Konzepte zur Beschreibung geometrischer Typen werden ebenfalls in (Orenstein, 1990), (Gargano *et al.*, 1991) und (Huang *et al.*, 1992) verwendet. Mangels ausreichender Modelliermächtigkeit, vor allem mit Hinblick auf topologische Beziehungen, konnte sich die ausschließliche Verwendung der Mengentheorie jedoch nicht durchsetzen.

7.2.2 Punktmengentopologie

Die Punktmengentopologie – siehe u.a. (Gaal, 1964) – ist ein mächtiges Instrument der Mathematik zur Beschreibung von Nachbarschaften im kontinuierlichen Raum. Mit Hilfe der *Umgebung* eines Punktes können die höheren Konzepte *Inneres*, *Rand*, *Äußeres*, *Abschluss*, *Separation* und *Verbundenheit* eindeutig beschrieben werden. Darauf aufbauend können wiederum topologische Beziehungen zwischen geometrischen Objekten formal definiert werden. Diese Vorgehensweise wurde von der Forschung im GIS-Bereich eingeführt (Pullar, 1988; Egenhofer & Herring, 1990; Clementini *et al.*, 1993; Schneider & Weinrich, 2004) und hat sich mittlerweile weitestgehend durchgesetzt.

Da die Punktmengentopologie auch im Rahmen dieser Arbeit intensive Anwendung bei der Definition räumlicher Typen und Beziehungen findet, soll hier auf eine überblicksartige Einführung verzichtet und auf die Abschnitte 7.3 und 7.4 verwiesen werden.

7.2.3 Algebraische Topologie

Die Methoden der Algebraischen Topologie (Armstrong, 1983; Hatcher, 2001) spielen schon seit langem[1] eine wichtige Rolle auf dem Gebiet der mathematischen Algebra. Dieses befasst sich mit der abstrakten Untersuchung der Eigenschaften von Zahlensystemen und den darin definierten Operationen und wird daher auch als „Abstrakte Algebra“ bezeichnet. Mittlerweile hat sich die Algebraische Topologie selbst zu einem großen, unabhängigen Gebiet innerhalb der Mathematik mit eigenen Anwendungsmöglichkeiten entwickelt.

Eine dieser Möglichkeiten liegt in der formalen Beschreibung von räumlichen Typen und Beziehungen. Die räumliche Modellierung auf Grundlage der Algebraischen Topologie hat gegenüber der punktmengenorientierten Herangehensweise den Vorteil, dass Probleme mit der Endlichkeit und Ungenauigkeit der Zahlensysteme in Rechensystemen und der damit einhergehenden Möglichkeit des Auftretens topologischer Fehler bei Algorithmen von vornherein vermieden werden, da topologische Eigenschaften explizit im Modell enthalten sind.

Die Algebraische Topologie beschreibt topologische Strukturen mit Hilfe algebraischer Methoden, um damit Punktmengen klassifizieren und formal beschreiben zu können. Die dabei eingesetzten Techniken beruhen auf der algebraischen Ma-

[1] Poincaré schuf bereits 1895 in seiner *analysis situs* die Konzepte *Homologie* und *Homotopie* und brachte damit algebraische Ideen in die Topologie ein.

nipulation von Symbolen, die räumliche Elemente und ihre Beziehungen repräsentieren. Der grundlegende Ansatz besteht aus drei Schritten:

1. Umformung des Problems von einer räumlichen in eine algebraische Form
2. Lösung der algebraischen Form des Problems
3. Rückumwandlung der algebraischen Lösung in eine räumliche Umgebung

Auf diese Weise wird beispielsweise das Problem des Schnitts zweier Linien in eine Suche nach gemeinsamen Knoten umgewandelt.

Notwendige Voraussetzung zur Anwendung der Algebraischen Topologie ist die Zerlegung des gesamten betrachteten Raums in regulär geformte Teilgebiete. Diese Gebiete bilden ein geometrisches Rahmenwerk, mit dessen Hilfe alle räumlichen Objekte der Anwendungsdomäne beschrieben werden. Nach der Art der Zerlegung werden die zwei Hauptzweige der Algebraischen Topologie unterschieden: die *Simpliziale Topologie* und die *Zelluläre Topologie*.

Simpliziale Topologie

Die Darstellungen in diesem und dem folgenden Abschnitt sind an (Schneider, 1997) angelehnt. Der überwiegende Teil der Ansätze zur Modellierung räumlicher Objekte, die Algebraische Topologie verwenden, basiert auf der *Simplizialen Topologie*, da es sich um die einfachere Theorie handelt und Datenstrukturen und Algorithmen leichter zu implementieren sind. Die Grundlagen der Simplizialen Topologie sollen im Folgenden kurz vorgestellt werden. Wichtigstes Element ist der *Simplex*, der wie folgt definiert ist:

Seien $\nu_0, \ldots, \nu_k \in \mathbb{R}^n$ und $\nu_0, \ldots, \nu_k$ linear unabhängig. Dann heißt die Menge

$$\sigma = \sigma_k = \{x \in \mathbb{R}^n \mid x = \sum_{i=0}^{k} \lambda_i \nu_i \text{ mit } \sum_{i=0}^{k} \lambda_i = 1,\ \lambda_i \in \mathbb{R}, \lambda_i \geq 0\}$$

(geschlossener) Simplex der Dimension k (oder kurz k-Simplex) und $\nu_0, \ldots, \nu_k$ heißen Knoten dieses Simplex.

Ein k-Simplex ist homöomorph zu einer konvexen Hülle von $k+1$ geometrisch unabhängigen Punkten und kann in einen euklidischen Raum der Dimension k oder größer eingebettet werden. Für eine gegebene Dimension k ist ein k-Simplex das elementare räumliche Objekt, d.h. ein Baustein mit dessen Hilfe alle komplexeren Objekte dieser Dimension gebildet werden können. Im dreidimensionalen Raum ist ein 0-Simplex ein einzelner Punkt, ein 1-Simplex eine gerade Linie bzw. eine Kante zwischen zwei Knoten inklusive dieser Endpunkte, ein 2-Simplex ein Dreieck und ein 3-Simplex ein Tetraeder (Abb. 7.1).

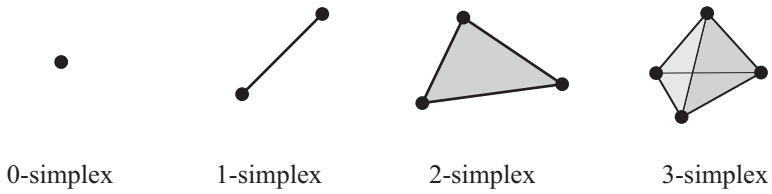


Abbildung 7.1: Simplizes in $\mathbb{R}^3$. Ein 0-Simplex ist ein einzelner Punkt, ein 1-Simplex eine gerade Strecke, ein 2-Simplex ein Dreieck und ein 3-Simplex ein Tetraeder.

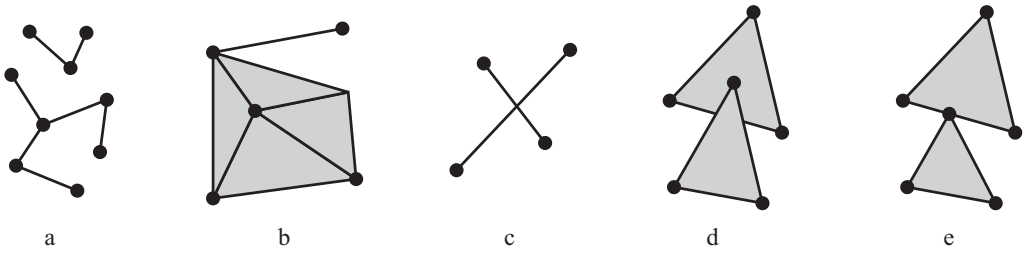


Abbildung 7.2: Gültige und ungültige Simplizialkomplexe. **(a)** Ein Komplex aus 1-Simplizes. **(b)** Ein Komplex aus 1- und 2-Simplizes. **(c)**, **(d)** und **(e)** Keine Simplizialkomplexe, da der jeweilige Schnitt nicht Facette beider Simplizes ist.

Jedes k-Simplex wird durch $k+1$ geometrisch unabhängige Simplizes der Dimension $k-1$ definiert. Beispielsweise wird ein 2-Simplex (ein Dreieck) aus drei geometrisch unabhängigen 1-Simplizes[2] gebildet. Eine *Facette* eines Simplex ist jedes Simplex einer niedrigeren Dimension, das das betrachtete Simplex definiert. Beispielsweise sind die Endknoten einer Strecke die Facetten dieses 1-Simplex.

Ein Simplizialkomplex C ist eine finite Menge von Simplizes, für die gilt: Jede Facette eines Simplex in C ist ebenfalls in C enthalten und der Schnitt zweier Simplizes in C ist entweder leer oder eine Facette beider Simplizes. Die Dimension von C ist die größte Dimension der Simplizes in C. Abb. 7.2 zeigt verschiedene Beispiele für gültige und ungültige Simplizialkomplexe. Mit Hilfe von derartigen Zellkomplexen können komplexe Objekte modelliert werden.

Einen *orientierten* k-Simplex erhält man aus einem k-Simplex σ mit den Knoten $\nu_0 \ldots, \nu_k$, indem man eine Ordnung für die Knoten wählt.

Der *Rand* eines k-Simplex σ_k wird mit $\partial\sigma_k$ bezeichnet und ist die Vereinigung aller $k+1$ $(k-1)$-Simplizes. Der *Rand* eines k-Komplexes C, bezeichnet mit ∂C, ist die symmetrische Differenz[3] der Ränder der k-Simplizes, aus denen C besteht. Das *Innere* eines k-Komplexes C, bezeichnet mit C°, ist die Vereinigung aller $(k-1)$-Simplizes, die nicht Teil des Randes von C sind.

[2] Drei 1-Simplizes sind geometrisch unabhängig genau dann, wenn keine zwei Kanten parallel sind und keine Kante die Länge 0 hat.

[3] Die symmetrische Differenz zweier Mengen ist die Menge derjenige Elemente, die in genau einer der beiden enthalten sind.

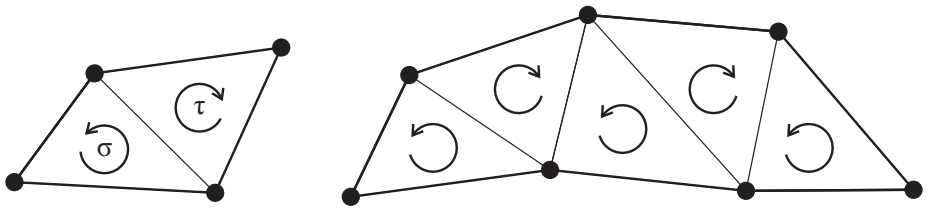


Abbildung 7.3: Illustration der Berechnung des Randes eines 2-Komplexes. Durch die gegenläufige Orientierung benachbarter Simplizes hebt sich der Zwischenrand bei der algebraischen Summation auf.

Mit Hilfe des Konzepts der Orientierung von Simplizes lassen sich der Rand und das Innere eines jeglichen k-Komplexes algebraisch bestimmen (Schubert, 1968).

Die Methoden der simplizialen Topologie dienen unter anderem in (Egenhofer *et al.*, 1989; Egenhofer & Herring, 1992) als Grundlage für die Definition zweidimensionaler räumlicher Typen und ihrer topologischen Beziehungen. Aber auch bei der Modellierung von dreidimensionalen räumlichen Typen ist die simpliziale Zerlegung vorherrschend. Beispielsweise definieren Wei *et al.* die räumlichen Typen *Point*, *Line*, *Surface* und *Body* als 0-, 1-, 2- bzw. 3-Komplexe und können zeigen, wie das *9-Intersection Model* (siehe Abschnitt 7.4.4) zur Beschreibung topologischer Beziehungen auf diese Typen angewendet werden kann (Wei *et al.*, 1998).

Im 3D-GIS-Kernel *GeoToolkit* werden Geometrie und Topologie von Gesteinsschichten mittels Oberflächen modelliert, die durch simpliziale 2-Komplexe repräsentiert werden (Breunig, 1995; Balovnev *et al.*, 2004). Da jedoch simpliziale Komplexe rein topologische, abstrakte Strukturen ohne Geometrie sind, wird in (Breunig *et al.*, 1994) der Begriff *Komplex* um eine Metrik und 3D-Koordinaten erweitert. Auf Basis dieses erweiterten *Komplex*-Begriffes werden Algorithmen für metrische, direktionale und topologische Operatoren sowie für das Verschneiden zweier Komplexe entwickelt (Breunig, 1995).

Im OO3D-Datenmodell (Shi *et al.*, 2003) wird ein *Knoten* als 0-Simplex, ein *Segment* als 1-Simplex und ein *Dreieck* als 2-Simplex modelliert. Aus diesen werden räumliche Objekte vom Typ *Surface* oder *Volume* zusammengesetzt. Die topologischen Beziehungen zwischen räumlichen Objekten werden auf Grundlage der topologischen Beziehungen zwischen den konstituierenden Dreiecken beschrieben (Abb. 7.4). Da auch hier die Konzepte *Inneres*, *Rand* und *Äußeres* nicht verwendet werden, bleibt die Ausdruckskraft und Vielfalt topologischer Beziehungen deutlich eingeschränkter als bei Verwendung des *9-Intersection Model* (Abschnitt 7.4.4).

Zelluläre Topologie

Die *zelluläre Zerlegung* des Raums ist eine Generalisierung der simplizialen Zerlegung, die zu den sogenannten k-Zellen führt. Eine k-Zelle ist eine Punktmen-

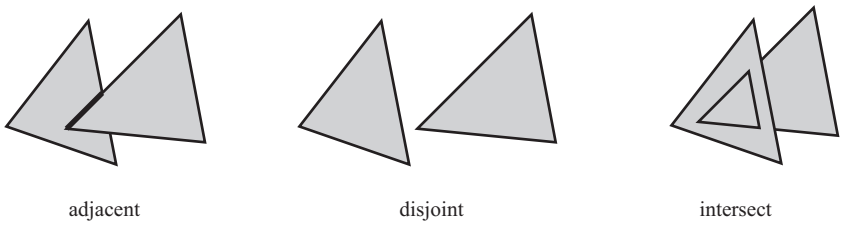


Abbildung 7.4: Die topologischen Beziehungen zwischen räumlichen Objekten des OO3D-Modells (Shi *et al.*, 2003) werden auf Basis der topologischen Beziehungen von Dreiecken beschrieben.

ge, die homöomorph[4] zu einer geschlossenen k-Scheibe ist. Eine k-Scheibe beschreibt die Menge aller Punkte im Abstand $d \leq r$ zu einem festen Punkt im n-dimensionalen euklidischen Raum $\mathbb{R}^n$. Ein k-dimensionaler zellulärer Komplex (k-Zellenkomplex) ist eine Verbindung von k-Zellen, bei der Teile der jeweiligen Ränder einander zugeordnet sind.

Das zelluläre Modell unterscheidet sich vom simplizialen Modell dahingehend, dass Simplizes immer geradlinig berandet sind, während Zellen ein beliebig geformtes Inneres aufweisen, und damit auch krummlinig berandet sein können. Darüber hinaus erlauben Zellkomplexe die Modellierung von Objekten mit Löchern.

Als einer der ersten schlägt Kovalevsky eine reguläre Zellzerlegung der $\mathbb{R}^2$-Ebene zur Untersuchung von topologischen Problemen vor (Kovalevsky, 1989). In (Egenhofer & Herring, 1992) werden ebenfalls die Methoden der zellulären Topologie zur Beschreibung von räumlichen Typen verwendet. Dabei werden *Regionen* als 2-Komplexe in $\mathbb{R}^2$ mit einem nicht-leeren, verbundenem Inneren definiert. Egenhofer unterscheidet *Regionen ohne Löcher* als Regionen mit verbundenem Äußeren und verbundenem Rand von *Regionen mit Löchern* als Regionen mit geteiltem Äußeren und geteiltem Rand. Des Weiteren werden Linien als verbundene 1-Komplexe im $\mathbb{R}^2$ definiert, die sich weder selbst schneiden noch geschlossene Ringe bilden. Dabei ist eine *einfache Linie* eine Linie mit zwei unverbundenen Rändern und eine *komplexe Linie* eine Linie mit mehr als zwei unverbundenen Rändern.

Winter verwendet den Begriff des „Hyperrasters“ (Winter, 1995) und bezeichnet damit ein 2D-Raster, das durch eine uniforme zellulären Zerlegung von $\mathbb{R}^2$ in quadratische 2-Zellen mit Seitenlänge 1 und Hinzunahme ihrer Skelette erzeugt wird (Abb. 7.5). Diese Zerlegungsmethode stellt sowohl räumliche Elemente der Dimensionen 0 bis 2 zur Verfügung (0-Zellen, 1-Zellen, 2-Zellen), als auch eine klare topologische Struktur. Winter zeigt, dass mit Hilfe dieser hybriden Repräsentation die Möglichkeit gegeben ist, auch für Rastermodelle topologische Beziehungen auf Basis des *9-Intersection Model* von Egenhofer (siehe Abschnitt 7.4.4) zu definieren.

[4]Ein Homöomorphismus ist eine bijektive stetige Abbildung zwischen zwei Objekten, deren Umkehrabbildung ebenfalls stetig ist.

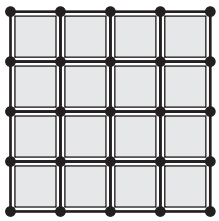


Abbildung 7.5: Das 'Hypperraster' aus (Winter, 1995). Es besteht aus 0-Zellen (die Knoten), 1-Zellen (die Kanten) und 2-Zellen (die Quadrate). Mit seiner Hilfe sind auch für Rastermodelle topologische Beziehungen auf Basis des 9-Intersection-Modells formal definierbar.

Im Projekt „Architektonische Komplexe“ der Universität Karlsruhe werden architektonische Räume formal als dreidimensionale Zellkomplexe beschrieben, um daraus ein relationales Datenmodell zu entwickeln, mit dessen Hilfe sich topologische Beziehungen direkt abbilden lassen (Paul & Bradley, 2003, 2005). Grundlage bildet dabei die Theorie der endlichen CW-Komplexe, ein Teilgebiet der Algebraischen Topologie.

Bewertung

Die Algebraische Topologie ist ein mächtiges Werkzeug für die Modellierung von räumlichen Typen. Vor allem die simpliziale Zerlegung erlaubt eine einfache Implementierung topologischer Operatoren. Die Voraussetzung zur Anwendung dieser Theorie sind jedoch als zu strikt für das hier besprochene Anwendungsgebiet einzuschätzen. Eine der Einschränkungen liegt in der Linearität eines Simplexes: 1-Simplizes sind Linien, die keinerlei Krümmung aufweisen, 2-Simplizes sind Dreiecke mit geraden Strecken als Berandung und 3-Simplizes sind Tetraeder mit planaren Dreiecks-Flächen als Berandung.

Mit dem Hintergrund aktueller Architektur-Strömungen, bei denen „organische“ Formen für Gebäude im Vordergrund stehen, sind jedoch krumm berandete Bauteile nicht mehr als Ausnahmefall zu betrachten. Zwar stellt die Tetraedervermaschung von krumm berandeten Körpern (Weatherill & Hassan, 1994) bzw. die Dreieckszerlegung von gekrümmten Flächen kein technisches Problem mehr dar. Jedoch werden durch die dabei durchgeführte Diskretisierung Fehler eingeführt. Damit setzt ein auf Algebraischer Topologie beruhender Ansatz zur Definition von räumlichen Typen und ihren Beziehungen ein bereits fehlerbehaftetes geometrisches Modell voraus.

Dies soll bei der in dieser Arbeit besprochenen Herangehensweise vermieden werden, indem die Gültigkeit der räumlichen Typen und ihrer Operatoren auch für krummlinig bzw. -flächig berandete räumliche Elemente sichergestellt wird. Die im nächsten Abschnitt vorgestellten Definitionen der räumlichen Typen beruhen daher nicht auf den Methoden der Algebraischen Topologie, sondern auf denen der Punktmengentopologie. Damit ist eine beliebige Geometrie der räumlichen Objekte präzise beschreibbar. Die Approximation erfolgt somit nicht auf der konzeptionellen bzw. theoretischen Ebene, sondern erst bei der Implementierung, bei

der beispielsweise, wie in Kapitel 8 beschrieben, Normzellen zur approximativen Geometriebeschreibung zum Einsatz kommen können. Dies erlaubt dem Anwender einen direkten Einfluss auf die Genauigkeit des Ergebnisses.

Die in Abschnitt 7.3 aufgeführten Definitionen verfolgen dabei das Ziel, dass *Inneres* und *Rand* der auf Basis von Punktmengentheorie und Punktmengentopologie beschriebenen Typen äquivalent zu den Begrifflichkeiten der Algebraischen Topologie festgelegt sind. Dies gewinnt besondere Bedeutung bei dimensionsreduzierten Objekten, da diese bei strikter Anwendung der reinen Punktmengentopologie ausschließlich aus Randpunkten bestehen. Um dies zu vermeiden, werden Randpunkte dieser Typen zunächst in einem niedrigdimensionalen Raum spezifiziert und dann mit dem Objekt selbst in $\mathbb{R}^3$ projiziert.

7.2.4 Einfache oder komplexe Typen

Die Forschung im GIS-Bereich unterscheidet zwischen einfachen und komplexen räumlichen Objekten. Die Unterscheidung beruht darauf, ob eine Separation des *Inneren*, des *Äußeren* und/oder des *Randes* eines Objekts zugelassen wird. Wird bei der Definition der Typen keinerlei Separation[5] zugelassen, spricht man von *einfachen* räumlichen Typen, ansonsten von *komplexen* räumlichen Typen (Clementini & Di Felice, 1996).

Definiert man beispielsweise den Typ *Punkt* als komplexen räumlichen Typ, darf das Innere separiert sein (Abb. 7.6). Das bedeutet, dass ein Objekt vom Typ *Punkt* aus mehreren einzelnen geometrischen Punkten bestehen kann. Eine komplexe Linie kann entsprechend aus mehreren Kurvensegmenten bestehen, die nicht notwendigerweise verbunden sind. Komplexe Linien können einen separierten Rand aufweisen und zusätzlich auch ein separiertes Inneres.

Hintergrund dieser Festlegungen ist die gewünschte Geschlossenheit des Typsystems bei Anwendung der booleschen Operationen *Vereinigung*, *Differenz* und *Schnitt*. Sie verlangt, dass beispielsweise die Vereinigung zweier Punkte in einem Objekt resultieren muss, das im Typsystem existiert. Dies kann durch die Einführung von Typen realisiert werden, die mehrere Objekte aufnehmen können. Im OGC-Standard gibt es aus diesem Grund beispielsweise den Typ *MultiPolygon* (OpenGIS Consortium (OGC), 1999). Die Einführung von multiplen Typen führt jedoch zu dem Problem, dass die Differenz zweier multipler Objekte in einem einfachen Objekt resultieren kann, was wiederum der Forderung nach der Eindeutigkeit des Ergebnistyps einer Operation widerspricht. Eine Unterscheidung zwischen einfachen und multiplen Typen im Typsystem erscheint daher aus Modellierungssicht nicht sinnvoll. Besser ist es, nur jeweils einen Typ zu definieren, der sowohl einfache als auch komplexe Typen aufnehmen kann.

Während mit einfachen Typen die Geschlossenheit des Typsystem nicht realisiert werden kann, ist dies bei Verwendung von komplexen Typen unter Beachtung bestimmter Randbedingungen möglich. Diese Randbedingungen müssen sicher-

[5] für eine formale Definition separierter Punktmengen siehe Abschnitt 7.3.3 bzw. 7.3.4

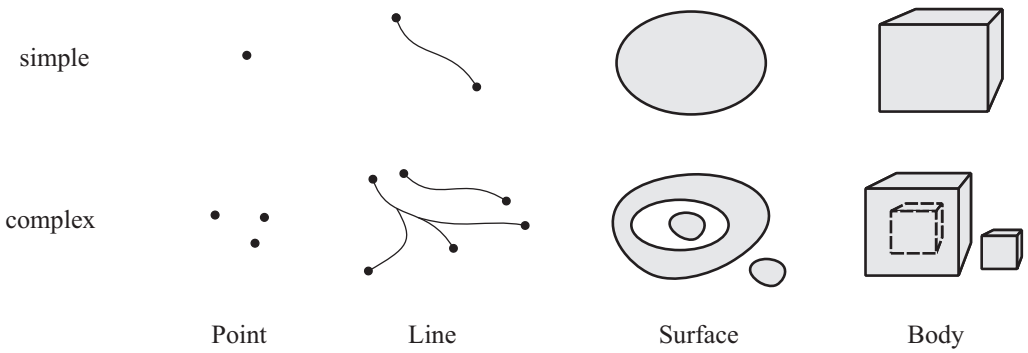


Abbildung 7.6: Der Unterschied zwischen einfachen und komplexen räumlichen Objekten. Komplexe Typen erlauben eine Separation des Inneren und des Äußeren, einfache Typen hingegen nicht.

stellen, dass bei Schnitt und Differenz zweier Objekte kein Objekt mit geringerer Dimensionalität entsteht. Ein anschauliches Beispiel hierfür ist der Durchschnitt zweier Linien. Es lässt sich a-priori nicht feststellen, ob das Ergebnis dieser Operation ein Objekt vom Typ *Punkt* oder *Linie* enthält. Aus diesem Grund wurden *regularisierte* Mengenoperationen eingeführt (Worboys & Bofakos, 1993), die dafür sorgen, dass zum einen eine Verminderung der Dimensionalität bei Durchschnitt und Differenz nicht auftritt und zum anderen bei einer Vereinigung keine irregulären Objekte wie Flächen mit angehängten Linien u.ä. erzeugt werden können. Dies hat beispielsweise zur Konsequenz, dass das Ergebnis des regularisierten Durchschnitts zweier sich kreuzender Linien kein Punkt-Objekt, sondern ein leeres Linien-Objekt ist.

Obwohl objekterzeugende räumliche Operationen in dieser Arbeit nicht behandelt werden (vgl. Abschnitt 7.4.5), ist es dennoch hinsichtlich der Allgemeingültigkeit und einer zukünftigen Erweiterbarkeit des Modells bzw. der Algebra sinnvoll, die Definitionen der räumlichen Typen auf komplexe Typen auszurichten.

Die Entscheidung zwischen einfachen und komplexen Objekten beeinflusst neben der Geschlossenheit des Typsystems maßgeblich die Darstellbarkeit unterschiedlicher räumlicher Objekte (Abb. 7.6). So sind in 2D-Regionen mit Löchern sowie Inseln innerhalb dieser Löcher nur mit Hilfe von komplexen Typen als ein Objekt darstellbar, da dies eine Separierbarkeit des Inneren voraussetzt (Worboys & Bofakos, 1993). Ähnliches gilt für linienförmige Objekte. Im dreidimensionalen Raum können darüber hinaus nur dann Körper mit Hohlräumen modelliert werden, wenn hierzu ein *komplexer* Typ *Körper* verwendet wird, der die Separation des Randes und des Äußeren erlaubt.

Eine der umfangreichsten Bemühungen zur Definition von komplexen räumlichen Typen im zweidimensionalen Raum auf Basis von Punktmengentheorie und Punktmengentopologie stellt (Schneider & Behr, 2006) dar. Zur formalen Definition von dreidimensionalen räumlichen Typen sind hingegen nur wenige Arbeiten bekannt. Die Mehrzahl von ihnen konzentriert sich auf einfache Typen, wie beispielsweise (Zlatanova, 2000). Eine Ausnahme bildet hier (Schneider &

Datentyp	Dimensionalität	Beispiel
Point	0	Lastpunkt, Steckdose, Gasverteiler
Line	1	Leitung, Tragwerksstab
Surface	2	Platten, Scheiben
Body	3	Treppen, Wände

Abbildung 7.7: Die räumlichen Typen der Algebra, ihre Dimensionalität und Beispiele für Modellierungsgegenstände in Bauwerksmodellen.

Weinrich, 2004), wo detaillierte Definitionen der komplexen räumlichen Datentypen *Point3D*, *Line3D*, *Surface*, *Relief* und *Volume* inklusive der Spezifikation von *Innerem*, *Äußerem* und *Rand* vorgestellt werden. Oosterom *et al.* verwenden die 3D-Typen *Point*, *Polyline*, *Polygon* und *Polyhedron*, geben jedoch keine formale Definition für den Rand und das Innere der Objekte an (van Oosterom *et al.*, 1994). Das Modell beschränkt sich zudem auf geradlinig begrenzte Objekte.

7.3 Formale Spezifikation des räumlichen Typsystems

Das hier vorgeschlagene räumliche Typsystem besteht aus den vier abstrakten Typen *Point*, *Line*, *Surface* und *Body*. Die Typen unterscheiden sich hinsichtlich ihrer Dimensionalität: Ein *Point*-Objekt ist nulldimensional, ein *Line*-Objekt eindimensional, ein *Surface*-Objekt zweidimensional und ein *Body*-Objekt dreidimensional. Die Einbeziehung von Typen mit geringerer Dimensionalität als der umgebende Raum ist notwendig, da im Ingenieurwesen vielfach Modelle mit dimensionsreduzierten Entitäten verwendet werden. Beispiele aus dem Bereich des Bauwesens können Abb. 7.7 entnommen werden.

Die formale Definition der Semantik der räumlichen Datentypen erfolgt unabhängig von einer konkreten Implementierung. Die Implementierung des abstrakten Typs *Body* kann beispielsweise sowohl auf Basis eines BRep-Modells mit möglicherweise gekrümmten Oberflächen, als auch auf Grundlage von *Constructive Solid Geometry* (CSG) oder Zelldekompositionsmethoden erfolgen. Die präzise Definition von Datentypen und Operatoren auf einer abstrakten Ebene erlaubt es, Aussagen über die Größe des Fehlers beim Übergang zu einer konkreten Implementierung zu treffen.

Die Semantik der Datentypen wird auf der Basis von Punktmengentheorie (Abschnitt 7.2.1) und Punktmengentopologie (Abschnitt 7.2.2) formal definiert. Wohlgemerkt handelt es sich bei der Beschreibung der Datentypen nicht um eine Definition der Datenstruktur für die Speicherung von räumlichen Objekten. Diese wird durch die unten aufgeführten Definitionen nicht festgelegt.

Neben der Spezifizierung der Menge der gültigen Punktmengen für jeden abstrakten Typ werden die zum Inneren, zum Äußeren und zum Rand gehörigen Punktmengen formal beschrieben. Dies ist notwendig, da eine direkte Anwendung der

Punktmengentopologie dazu führen würde, dass alle Punkte einer Ausprägung eines dimensionsreduzierten Typs Randpunkte wären (siehe Abschnitt 7.2.2). Das ist jedoch im Hinblick auf die darauf aufbauende formale Definition topologischer Beziehungen nicht sinnvoll.

Daher wird hier folgender genereller Ansatz verfolgt: Ein dimensionsreduziertes Objekt vom Typ *Line* oder *Surface* wird per Definition aus Objekten zusammengesetzt, die aus einer Abbildung von Objekten in einem Raum mit niedriger Dimension in den dreidimensionalen Raum resultieren. Ein Intervall in 1D wird auf eine *Curve* in 3D abgebildet, mehrere *Curves* bilden dann eine *Line*; eine *Region* in 2D wird auf eine *Superficies* in 3D abgebildet, mehrere *Superficies* bilden ein *Surface*-Objekt. Dabei werden i.a. Randpunkte auf Randpunkte, innere Punkte auf innere Punkte und äußere Punkte auf äußere Punkte abgebildet.

Die Definitionen folgen weitgehend dem von Schneider & Weinrich vorgeschlagenen Modell (Schneider & Weinrich, 2004). Eine der Abweichungen liegt in der Bezeichnung der Typen: Anstelle von *point3D*, *line3D* und *volume* werden hier, wie von (Molenaar, 1990) und (Zlatanova, 2000) vorgeschlagen, *Point*, *Line*, *Surface* und *Body* verwendet.

Für jeden abstrakten Typ werden die zum Inneren (A°), zum Rand (∂A), zum Äußeren (A^-) und zum Abschluss ($\overline{A}$) gehörigen Punktmengen spezifiert, da darauf die Definitionen der topologischen Operatoren aufbauen. Schneider & Weinrich geben für jeden räumlichen Typ sowohl eine *unstruktierte* als auch eine *struktierte* Definition an (Schneider & Weinrich, 2004). Zwar definiert die unstrukturierte Variante einen Typ als Punktmenge, die bestimmten Bedingungen genügt, aber sie stellt keine Spezifizierung des Inneren, Äußeren und des Randes zur Verfügung. Dies ist Aufgabe der strukturierten Definition, die einen Typ aus Komponenten modelliert, die durch eine Abbildung aus einem Raum mit niedrigerer Dimension gewonnen werden.

Dabei darf die strukturierte Definition jedoch nicht restriktiver als die unstrukturierte sein, d.h. alle Punktmengen, die durch die unstrukturierte Definition als zu einem bestimmten Typ gehörig klassifiziert werden, müssen auch mit Hilfe der stukturierten Definition konstruierbar sein. Da im Rahmen dieser Arbeit vor allem die Spezifikation des Inneren, Äußeren und des Randes benötigt werden, wird auf die Angabe der unstrukturierten Definitionen verzichtet.

Weiterhin fordern Schneider & Weinrich für die strukturierten Definitionen die Einhaltung einer sogenannten Eindeutigkeitsbedingung. Sie erzwingt, dass ein räumliches Objekt (als Ausprägung eines räumlichen Typs) nur auf genau eine Weise entsprechend der Regeln der strukturierten Definition konstruiert werden kann. Ein *Line*-Objekt darf beispielsweise nicht sowohl durch ein einzelnes *curve*-Objekt als auch mittels zweier zusammengesetzter *curve*-Objekte darstellbar sein. Da die Eindeutigkeitsbedingung aus Sicht des Autors für den hier beschriebenen Anwendungsfall nicht notwendig ist, wird sie in den folgenden Definitionen nicht angewandt.

7.3.1 Point

Ein *Point*-Objekt ist eine endliche Menge isolierter geometrischer Punkte im dreidimensionalen Raum. Um einen Punkt als isoliert zu definieren, wird das Konzept der Umgebung eines Punkts verwendet:

Geht man von der Existenz der euklidischen Distanzfunktion $d : \mathbb{R}^3 \times \mathbb{R}^3 \to \mathbb{R}$ mit

$$d(p,q) := d((x_p, y_p, z_p), (x_q, y_q, z_p)) := \sqrt{(x_p - x_q)^2 + (y_p - y_q)^2 + (z_p - z_q)^2}$$

aus, dann ist die Umgebung wie folgt definiert: Sei $q \in \mathbb{R}^3$ und $\varepsilon \in \mathbb{R}^+$. Dann ist die Menge $N_\varepsilon(q) = \{p \in \mathbb{R}^3 \mid d(p,q) \leq \varepsilon\}$ die (abgeschlossene) Umgebung von q mit dem Radius ε.

Darauf aufbauend lässt sich der räumliche Typ *Point* wie folgt definieren:

$$Point := \left\{ P = \bigcup_{i=1}^{n} p_i \mid n \in \mathbb{N},\, \forall i : p_i \in \mathbb{R}^3 \wedge \exists N_\varepsilon(p_i) \cap P = \{p_i\} \right\} \quad .$$

Per Definition hat ein Punkt keinen Rand ($\partial P := \varnothing$) und alle Punkte gehören zum *Inneren* ($P^\circ := P$), das außerdem dem *Abschluss* entspricht: $\overline{P} := P^\circ = P$. Das *Äußere* von P ist $P^- := \mathbb{R}^3 \setminus P$.

7.3.2 Line

Ein *Line*-Objekt ist die Vereinigung einer endlichen Anzahl von 3D-Kurven. Eine Kurve resultiert aus einer stetigen Abbildung f_i des eindimensionalen Intervalls [0,1] in den dreidimensionalen Raum und ist per Definition nicht selbstschneidend. Sie ist wie folgt definiert:

$$\begin{aligned} Curve := \Big\{ f([0,1] \;\Big|\; & \text{(i) } f([0,1] \to \mathbb{R}^3 \text{ ist eine stetige Abbildung} \wedge \\ & \text{(ii) } \forall a, b \in]0,1[\colon a \neq b \Rightarrow f(a) \neq f(b) \wedge \\ & \text{(iii) } \forall a \in \{0,1\},\, \forall b \in]0,1[\,:\, f(a) \neq f(b) \Big\} \quad . \end{aligned}$$

Bedingung (ii) verbietet, dass zwei innere Punkte des Intervalls $]0,1[$ auf einen einzelnen Punkt der Kurve abgebildet werden, wodurch selbstschneidende Kurven und Kurven, die aus nur einem geometrischen Punkt bestehen, ausgeschlossen werden. Bedingung (iii) erlaubt Ringe ($f(0) = f(1)$), aber verbietet sich selbst berührende Kurven (Abb. 7.8 a).

Die Endpunkte $f(0)$ und $f(1)$ bilden den *Rand* einer Kurve C. Wenn eine Kurve einen Ring formt, hat sie per Definition keinen Rand:

$$\partial C = \{f(0), f(1)\} \Leftrightarrow f(0) \neq f(1),\; \partial C = \varnothing \Leftrightarrow f(0) = f(1) \quad .$$

Zwei Kurven c_1 und c_2 werden als *quasi-disjunkt* bezeichnet, wenn sich ihr Inneres nicht schneidet. Es ist jedoch zulässig, dass sie sich in einem oder beiden

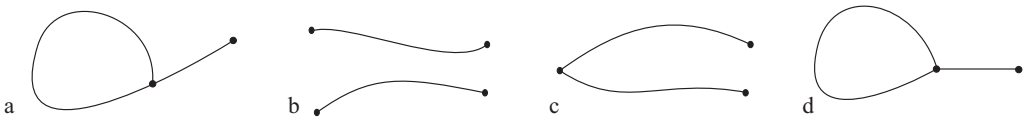


Abbildung 7.8: Kurven in $\mathbb{R}^3$. **(a)** Nicht *eine*, sondern *zwei* Kurven, da eine Kurve sich weder selbst berühren, noch selbst schneiden darf. **(b)** Zwei disjunkte Kurven. **(c)** Zwei quasi-disjunkte Kurven. **(d)** Zwei quasi-disjunkte Kurven.

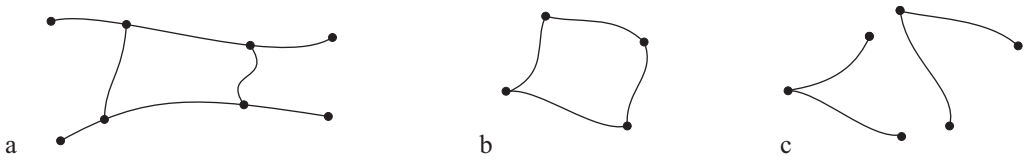


Abbildung 7.9: Blocks in $\mathbb{R}^3$. **(a)** Ein Block, der aus acht Kurven besteht. **(b)** Ein Block, der aus vier Kurven besteht. **(c)** Ein *Line*-Objekt, das aus zwei Blocks besteht.

Endpunkten berühren:

$$\begin{aligned} \textit{quasi-disjunkt}\ (c_1, c_2) \Leftrightarrow \forall a,\, b \in]0,1[\,:\, & f_1(a) \neq f_2(b) \wedge \\ & f_1(0) \neq f_2(b) \wedge f_1(1) \neq f_2(b) \wedge \\ & f_1(a) \neq f_2(0) \wedge f_1(a) \neq f_2(1) \quad , \end{aligned}$$

$$\begin{aligned} \textit{stoßen aufeinander}\ (c_1, c_2) \Leftrightarrow\ & f_1(0) = f_2(0) \vee f_1(0) = f_2(1) \vee \\ & f_1(1) = f_2(0) \vee f_1(1) = f_2(1) \quad . \end{aligned}$$

Die Definitionen von quasi-disjunkten und aufeinander stoßenden Kurven wird nun benutzt, um das Konzept des *Blocks* einzuführen, der eine Ansammlung von verbundenen Kurven beschreibt[6] (Abb. 7.9):

$$\begin{aligned} \textit{Block} := \Big\{ \bigcup_{i=1}^{m} c_i \;\Big|\; & \text{(i)}\ \forall\, i : c_i \in \textit{curve} \wedge \\ & \text{(ii)}\ \forall\, i,j \text{ mit } i<j : c_i \text{ and } c_j \text{ sind quasi-disjunkt} \\ & \text{(iii)}\ m > 1 \Rightarrow \forall\, i : \exists j \neq i : c_i \text{ und } c_j \text{ stoßen aufeinander} \Big\}. \end{aligned}$$

Eine *Line* kann nun als die Vereinigung einer endlichen Anzahl von disjunkten Blöcken definiert werden:

$$\begin{aligned} \textit{Line} := \Big\{ \bigcup_{i=1}^{m} b_i \;\Big|\; & \text{(i)}\ \forall\, i \leq m : b_i \in \textit{block} \wedge \\ & \text{(ii)}\ \forall\, i,j \text{ mit } i<j : b_i \text{ und } b_j \text{ sind disjunkt} \Big\}. \end{aligned}$$

[6] In (Schneider & Weinrich, 2004) wird zur Einhaltung der Eindeutigkeitsbedingung verlangt, dass in einem Block ein Endpunkt entweder nur genau einer oder *mehr als zwei* Kurven gehört. Letztere Einschränkung wird hier nicht übernommen.

Im Folgenden soll mit Hilfe der Endpunkte der Kurven die Definition des Randes eines *Line*-Objekts L erfolgen. Sei $E(L) = \bigcup_{i=1}^{m} \partial C_i$ die Vereinigung aller Ränder der Kurven, aus denen L besteht. Dann ist der Rand von L diese Menge minus derjenigen Endpunkte, die sich mehrere Kurven teilen. Letztere gehören zum Inneren des *Line*-Objekts.

$$\partial L := E(L) \setminus \{p \in E(L) \mid card\,(\{f_i \mid 1 \leq i \leq m \wedge f_i(0) = p\}) + card\,(\{f_i \mid 1 \leq i \leq m \wedge f_i(1) = p\}) > 1\} \quad .$$

Der *Abschluss* einer *Line* L ist die Menge aller Punkte von L inklusive der Endpunkte ($\overline{L} := L$). Das *Innere* ergibt sich zu $L^\circ := L \setminus \partial L$ und das *Äußere* zu $L^- = \mathbb{R}^3 \setminus L$.

7.3.3 Surface

Da die Definition des *Surface*-Typs auf Abbildungen von 2D-Regionen in den dreidimensionalen Raum beruht, muss zunächst der Typ *Region2D* definiert werden. Ein *Region2D*-Objekt ist eine Punktmenge im zweidimensionalen euklidischen Raum mit speziellen Eigenschaften (Schneider & Behr, 2006). Um das *Innere*, das *Äußere* und den *Rand* einer *Region2D* festzulegen, wird wiederum das Konzept der *Umgebung* eines Punktes, hier allerdings in $\mathbb{R}^2$, verwendet.

Geht man von der Existenz einer euklidischen Distanzfunktion $d : \mathbb{R}^2 \times \mathbb{R}^2 \to \mathbb{R}$ mit

$$d(p, q) := d((x_p, y_p), (x_q, y_q)) = \sqrt{(x_p - x_q)^2 + (y_p - y_q)^2}$$

aus, dann ist die Umgebung wie folgt definiert: Sei $q \in \mathbb{R}^2$ und $\varepsilon \in \mathbb{R}^+$. Dann ist die Menge $N_\varepsilon(q) := \{p \in \mathbb{R}^2 \mid d(p, q) \leq \varepsilon\}$ die (abgeschlossene) Umgebung von q mit dem Radius ε.

Mit Hilfe dieser Festlegung lassen sich alle Punkte als innere, äußere oder Randpunkte klassifizieren: Sei $X \subseteq \mathbb{R}^2$ und $q \in \mathbb{R}^2$. Dann ist q ein *innerer* Punkt von X, wenn es ein ε gibt, so dass $N_\varepsilon(q) \subseteq X$ erfüllt ist. q ist ein *äußerer* Punkt von X, wenn es ein ε gibt, für das $N_\varepsilon(q) \cap X = \varnothing$ gilt. q ist ein *Randpunkt* von X, wenn q weder ein innerer noch ein äußerer Punkt von X ist. Schließlich ist q ein *Abschlusspunkt* von X, wenn er entweder ein innerer oder ein Randpunkt ist. Die Menge aller inneren Punkte / äußeren Punkte / Randpunkte / Abschlusspunkte von X bildet das Innere / das Äußere / den Rand / den Abschluss von X.

Eine beliebige Punktmenge im zweidimensionalen Raum ist nicht zwangsläufig eine Region. Schneider *et al.* verwenden daher den Begriff der *regulär abgeschlossenen* Punktmenge (Tilove, 1980) für die Definition des *Region2D*-Typs, um geometrische Anomalien zu vermeiden (Abb. 7.10, links): Eine Punktmenge $X \subseteq \mathbb{R}^2$ ist genau dann regulär abgeschlossen, wenn $X = \overline{X^\circ}$ gilt. Die Operation X° würde isolierte oder „baumelnde“ Punkte bzw. Linien entfernen, aber in jedem Fall auch den Rand von X. Die Operation $\overline{X^\circ}$ fügt den Rand wieder hinzu und würde darüber hinaus einzelne fehlende Punkte bzw. Linien innerhalb der Region hinzufügen. Nur solche Punktmengen X, die nach Anwendung dieser beiden

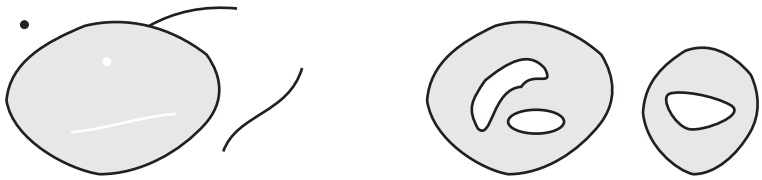


Abbildung 7.10: Links: Geometrische Anomalien, die ausgeschlossen werden: Isolierte oder baumelnde Punkte bzw. Linien sowie fehlende Punkte bzw. Linien. **Rechts:** Ein einzelnes *Region2D*-Objekt. *Region2D*-Objekte können Löcher aufweisen und aus mehreren Komponenten bestehen.

Operationen keine Veränderung erfahren haben, gelten als *regulär abgeschlossen*; sie besitzen also weder isolierte oder „baumelnde“ Punkte noch fehlen einzelne Punkte innerhalb des Gebiets.

Weitere Voraussetzungen für die Spezifikation des *Region2D*-Typs sind die Definitionen der *Beschränktheit* und des *Zusammenhangs* von Punktmengen. Zwei Punktmengen $X, Y \subseteq \mathbb{R}^2$ sind *separiert* genau dann, wenn gilt: $X \cap \overline{Y} = \varnothing = \overline{X} \cap Y$. Eine Punktmenge $X \subseteq \mathbb{R}^2$ ist genau dann *zusammenhängend*, wenn sie nicht durch eine Vereinigung einer endlichen Zahl von nicht-leeren separierten Punktmengen erzeugt werden kann.

Sei $q = (x, y) \in \mathbb{R}^2$ und die Norm $\|q\| = \sqrt{x^2 + y^2}$. Eine Menge $X \subset \mathbb{R}^2$ ist beschränkt genau dann, wenn es eine Zahl $r \in \mathbb{R}^+$ gibt, so dass gilt $\|q\| < r$ für alle $q \in X$.

Der räumliche Typ *Region2D* kann nun definiert werden als:

$$\begin{aligned} \mathit{Region2D} := \{R \subset \mathbb{R}^2 \mid\ & R \text{ ist regulär abgeschlossen und beschränkt} \wedge \\ & R \text{ besteht aus einer endlichen Zahl} \\ & \text{zusammenhängender Punktmengen}\} \quad . \end{aligned}$$

Nach dieser Definition können Regionen aus mehreren Komponenten bestehen und Löcher aufweisen (Abb. 7.10, rechts). In der Terminologie der GIS-Forschung handelt es sich also um *komplexe* Objekte.

Der räumliche Typ *Surface* kann nun basierend auf dieser Definition als die Vereinigung einer endlichen Zahl von Abbildungen von 2D-Regionen in den dreidimensionalen Raum beschrieben werden. Das Resultat einer einzelnen Abbildung wird als *Superficies* bezeichnet. *Superficies* sind Komponenten, aus denen ein *Surface*-Objekt zusammengesetzt ist. Sie sind nicht selbstschneidend, dürfen aber Löcher besitzen (Abb. 7.11).

Sei *ConnectedRegion2D* $\subset$ *Region2D* die Menge aller 2D-Regionen, die nur aus einer einzelnen Komponente bestehen. Die Menge der *Superficies* ist dann definiert

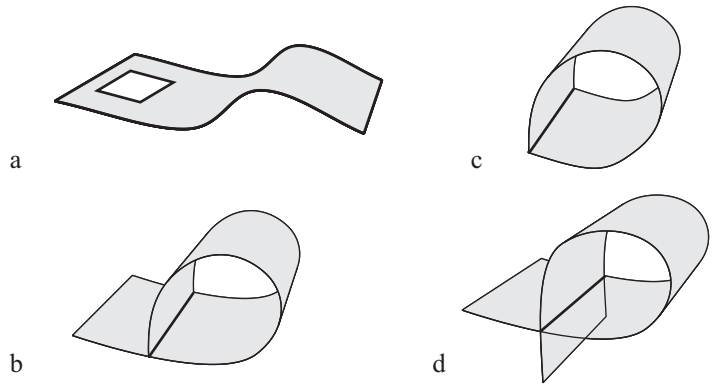


Abbildung 7.11: Zulässige und unzulässige *Superficies*. **(a)** Eine *Superficies* mit einem Loch. **(b)** Das ist keine *Superficies*, da Randpunkte der zugrunde liegenden Region auf dieselben Punkte abgebildet werden wie Punkte aus dem Inneren der Region. Das gleiche Objekt kann jedoch durch zwei *Superficies* dargestellt werden. **(c)** Das ist eine teilweise geschlossene *Superficies*. **(d)** Das ist keine *Superficies*, da Bedingung (iii) verletzt ist: Verschiedene innere Punkte der zugrunde liegenden Region werden auf den selben Punkt der *Superficies* abgebildet. Das gleiche Objekt kann jedoch durch drei *Superficies* dargestellt werden.

als

$$
\begin{aligned}
\mathit{Superficies} := \big\{\, s(R) \mid\ & \text{(i)}\ \ R \in \mathit{ConnectedRegion2D} \wedge \\
& \text{(ii)}\ \ x : R \to \mathbb{R}^3 \text{ ist eine stetige Abbildung} \wedge \\
& \text{(iii)}\ \ \forall\,(x_1, y_1), (x_2, y_2) \in R^\circ\ : \\
& \qquad (x_1, y_1) \neq (x_2, y_2) \Rightarrow s((x_1, y_1)) \neq s((x_2, y_2)) \ \wedge \\
& \text{(iv)}\ \ \forall\,(x_1, y_1) \in \partial R,\ \forall\,(x_2, y_2) \in R^\circ\ : \\
& \qquad s((x_1, y_1)) \neq s((x_2, y_2))\big\} \quad .
\end{aligned}
$$

Bedingung (iii) verhindert, dass zwei innere Punkte von R auf einen einzelnen Punkt in der *Superficies* abgebildet werden. Auf diese Weise werden selbstschneidende und degenerierte *Superficies* vermieden, die nur aus einem Punkt bestehen und damit nulldimensionale Objekte sind, und solche, die einer eindimensionalen *Curve* entsprechen. Bedingung (iv) verbietet, dass ein Randpunkt und ein innerer Punkt von R auf den selben Punkt der *Superficies* abgebildet werden.

Nach dieser Definition können *Superficies* geschlossen oder teilweise geschlossen sein. Eine *Superficies* S ist *geschlossen*, wenn zwei Punktmengen B_1 und B_2 existieren, für die gilt: $\partial R = B_1 \cup B_2$ und $B_1 \cap B_2 = \varnothing$, so dass $s(B_1) = s(B_2)$. S ist teilweise geschlossen, wenn für B_1 und B_2 gilt: $\partial B_1 \cup B_2 \subset R$ und $B_1 \cap B_2 = \varnothing$, so dass $s(B_1) = s(B_2)$. In beiden Fällen gehört die Menge $I(S) = s(B_1) = s(B_2)$ per Definition zum Inneren von S, andernfalls ist $I(S) = \varnothing$.

Der *Rand* einer *Superficies* ergibt sich zu $\partial S = s(\partial R) - I(S)$, ihr *Inneres* ist $S^\circ = s(R^\circ) \cup I(S)$. Entsprechend ist ihr Abschluss $\overline{S} = \partial S \cup S^\circ$ und ihr Äußeres $S^- = \mathbb{R}^3 - \overline{S}$.

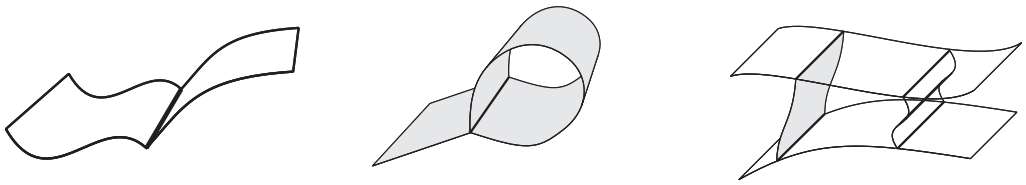


Abbildung 7.12: Links: Zwei quasi-disjunkte Superficies. **Mitte:** Ein Superficies-Block bestehend aus zwei Superficies. Quasi-disjunkte Superficies dürfen an der „Naht" einer geschlossenen Superficies aufeinanderstoßen. **Rechts:** Eine Surface-Objekt, das aus einem Superficies-Block mit acht Superficies besteht.

Sei T die Menge aller *Superficies* über $\mathbb{R}^3$. Zwei *Superficies* $S_1, S_2 \in T$ sind *quasi-disjunkt*, wenn sie keine gemeinsamen inneren Punkte haben. Sie können jedoch gemeinsame Randpunkte besitzen. Formal geschrieben:

$$\begin{aligned} \textit{quasi-disjunkt}\ (S_1, S_2) \Leftrightarrow\ & s_1(R_1^\circ) \cap s_2(R_2^\circ) = \varnothing\ \wedge \\ & s_1(\partial R_1) \cap s_2(R_2^\circ) = \varnothing\ \wedge \\ & s_1(R_1^\circ) \cap s_2(\partial R_2) = \varnothing \quad . \end{aligned}$$

Nach dieser Definition sind zwei *Superficies* auch dann quasi-disjunkt, wenn eine oder beide (teilweise) geschlossen sind und sie gemeinsame Punkte an der „Naht" haben. Zwei *Superficies stoßen aufeinander* genau dann, wenn sie quasi-disjunkt sind und $s_1(\partial R_1) \cap s_2(\partial R_2) = \varnothing$ gilt.

Die Definitionen von quasi-disjunkten und aufeinander stoßenden *Superficies* können nun benutzt werden, um das Konzept eines *Superficies-Blocks* einzuführen, welcher eine verbundene Komponente innerhalb einer Surface darstellt (Abb. 7.12). Ein *Superficies-Block* ist wie folgt formal definiert:

$$\begin{aligned} &\textit{Superficies-Block} := \\ &\Big\{ \bigcup_{i=1}^{m} S_i \ \Big|\ \ \text{(i)}\ \forall\, i : S \smallsetminus I \in T,\ \wedge \\ &\qquad\qquad \text{(ii)}\ \forall\, i, j \text{ mit } i < j : S_i \text{ und } S_j \text{ sind quasi-disjunkt}\ \wedge \\ &\qquad\qquad \text{(iii)}\ m > 1 \Rightarrow \forall\, i : \exists\, j \text{ mit } i \neq j : \\ &\qquad\qquad\qquad S_i \text{ und } S_j \text{ stoßen aufeinander} \Big\} \quad . \end{aligned}$$

Bedingung (ii) garantiert, dass alle *Superficies* innerhalb des Blocks untereinander quasi-disjunkt sind. Auf diese Weise wird verhindert, dass der *Rand* einer *Superficies* mit dem *Inneren* einer anderen zusammenfällt. Bedingung (iii) stellt sicher, dass jede *Superficies* mit mindestens einer anderen verbunden ist.

Zwei *Superficies-Blöcke* c_1, c_2 sind genau dann disjunkt, wenn $c_1 \cap c_2 = \varnothing$. Darauf aufbauend kann der räumliche Typ *Surface* wie folgt formal definiert werden:

$$\begin{aligned} \textit{Surface} := \Big\{ \bigcup_{i=1}^{n} c_i \ \Big|\ & \text{(i)}\ \forall\, i : c_i \in \textit{Superficies-Block}\ \wedge \\ & \text{(ii)}\ \forall\, i \text{ mit } i < j : c_i \text{ und } c_j \text{ sind disjunkt} \Big\} \quad . \end{aligned}$$

Dabei stellt Bedingung (ii) sicher, dass die *Superficies* innerhalb eines Blocks keine gemeinsamen Punkte mit denen eines anderen Blocks besitzen. Um den Rand einer *Surface* zu spezifizieren, wird die Hilfsmenge $L(S)$ definiert. Sei S eine *Surface* mit den *Superficies* $S_1, \ldots, S_m$. Dann ist L(S) die Punktmenge, die sich aus allen Kurven zusammensetzt, die den Rand von mehr als einer *Superficies* formen:[7]

$$L(S) := \Big\{ \bigcup_{i=1}^{m} l_i \;\Big|\; m \in \mathbb{N}, \forall 1 \leq i \leq m : l \in curve \wedge l \subseteq \bigcup_{i=1}^{m} \partial S_i \wedge card\big(\{S_i \mid 1 \leq i \leq m \wedge l \subseteq \partial S_i\}\big) \geq 2 \Big\} \; .$$

Der *Rand* von S ergibt sich damit zu $\partial S := \bigcup_{i=1}^{m} \partial S_i \setminus L(S)$ und das *Innere* zu $S^\circ := \bigcup_{i=1}^{m} S_i^\circ \cup L(S)$. Entsprechend ist der *Abschluss* $\overline{S} = \partial S \cup S^\circ$ und das *Äußere* $S^- := \mathbb{R}^3 \setminus S$.

7.3.4 Body

Der Typ *Body* repräsentiert dreidimensionale Objekte (Körper), die in $\mathbb{R}^3$ eingebettet sind. Sie werden als spezielle unendliche Punktmengen modelliert. Es ist möglich, dass ein *Body*-Objekt aus mehreren Komponenten besteht und dass es Hohlräume aufweist. Wiederum wird der Begriff der *Umgebung* verwendet (siehe Abschnitt 7.3.1), um das *Innere*, das *Äußere*, den *Rand* und den *Abschluss* eines *Body*-Objekts zu bestimmen.

Sei $X \subseteq \mathbb{R}^3$ und $q \in \mathbb{R}^3$. Dann ist q ein *innerer* Punkt von X, wenn es ein ε gibt, so dass $N_\varepsilon(q) \subseteq X$ gilt. q ist ein *äußerer* Punkt von X, wenn es ein ε gibt, so dass $N_\varepsilon(q) \cap X = \varnothing$ erfüllt ist. q ist ein *Randpunkt* von X, wenn q weder ein innerer, noch ein äußerer Punkt von X ist. Schließlich ist q ein *Abschlusspunkt* von X, wenn er entweder ein innerer oder ein Randpunkt ist. Die Menge aller inneren Punkte / äußeren Punkte / Randpunkte / Abschlusspunkte von X bildet das Innere / das Äußere / den Rand / den Abschluss von X.

Eine beliebige Punktmenge in $\mathbb{R}^3$ ist nicht zwangsläufig ein *Body*. Daher wird analog zur Definition des *Region2D*-Typs der Begriff der *regulär abgeschlossenen* Punktmenge verwendet, um geometrische Anomalien zu vermeiden: Eine Punktmenge $X \subseteq \mathbb{R}^3$ ist genau dann regulär geschlossen, wenn $X = \overline{X^\circ}$ gilt. Das bedeutet, dass nur solche Punktmengen als *regulär geschlossen* gelten, die nach Anwendung der beiden Operationen keine Veränderung erfahren haben, die also weder isolierte oder „baumelnde“ Punkte besitzen noch einzelne fehlende Punkte im Inneren aufweisen.

Ebenso werden Definitionen für *beschränkte* und *zusammenhängende* Punktmengen in $\mathbb{R}^3$ benötigt. Zwei Punktmengen $X, Y \subseteq \mathbb{R}^3$ sind *separiert* genau dann, wenn gilt: $X \cap \overline{Y} = \varnothing = \overline{X} \cap Y$. Eine Punktmenge $X \subseteq \mathbb{R}^3$ ist genau dann *zusammenhängend*, wenn sie nicht durch eine Vereinigung einer endlichen Zahl von nicht-leeren separierten Punktmengen erzeugt werden kann.

[7] Schneider & Weinrich geben eine stark abweichende Definition von $L(S)$ an (Schneider & Weinrich, 2004).

Sei $q = (x, y) \in \mathbb{R}^3$ und die Norm $\|q\| = \sqrt{x^2 + y^2 + z^2}$. Eine Menge $X \subset \mathbb{R}^3$ ist genau dann *beschränkt*, wenn es eine Zahl $r \in \mathbb{R}^+$ gibt, so dass $\|q\| < r$ für alle $q \in X$ gilt.

Der räumliche Typ *Body* wird nun definiert als:

$$\begin{aligned} \mathit{Body} := \{\, B \subset \mathbb{R}^3 \mid\ & B \text{ ist regulär abgeschlossen und beschränkt } \wedge \\ & B \text{ besteht aus einer endlichen Zahl} \\ & \text{zusammenhängender Punktmengen}\} \end{aligned}$$

7.3.5 Der Supertyp SpatialObject

Der Typ *SpatialObject* ist die Obermenge der bereits definierten Typen *Point*, *Line*, *Surface* und *Body* und ist wie folgt formal definiert:

$$\mathit{SpatialObject} := \{\mathit{Point} \cup \mathit{Line} \cup \mathit{Surface} \cup \mathit{Body}\}\,.$$

7.4 Formale Definition räumlicher Operatoren

7.4.1 Überblick

Räumliche Operatoren arbeiten auf räumlichen Typen und besitzen räumliche Semantik. Sie lassen sich entsprechend ihrer Semantik in vier Gruppen einteilen:

- metrische Operatoren (wie *Entfernung, näher, ferner* etc.)
- direktionale Operatoren (wie *über, unter, nördlich, südlich* etc.)
- topologische Operatoren (wie *berührt, beinhaltet, innerhalb* etc.)
- objekterzeugende Operatoren

7.4.2 Metrische Operatoren

Metrische Operatoren basieren auf der Metrik des zugrunde liegenden Raums. Sie dienen zur Bestimmung des Abstands zweier Objekte bzw. der Abmessungen eines Objekts.

Verwandte Arbeiten

Sowohl Güting als auch van Oosterom *et al.* schlagen drei Distanz-Operationen vor: Die Operation *dist* berechnet die Distanz zwischen zwei einfachen Punkten, die Operationen *mindist* und *maxdist* berechnen die minimale bzw. maximale Distanzen zwischen zwei ausgedehnten räumlichen Objekten jeglichen Typs (Güting, 1988; van Oosterom *et al.*, 1994).

Des Weiteren kann die Operation *diameter* verwendet werden, um die größte Distanz zwischen den Punkten eines räumlichen Objekts zu bestimmen. Die Operation *area* berechnet die Größe des von einer Region bedeckten Gebiets und die Operation *perimeter* ermittelt den Durchschnitt eines *Region*-Objekts.

Definitionen

Der Operator *distance* gibt den (minimalen) Abstand zweier räumlicher Objekte als reelle Zahl zurück und ist wie folgt formal definiert: Seien A und B räumliche Objekte vom Typ *Spatial* und $a \in A, b \in B$. Dann ist

$$distance(A, B) := \min_{a,b}(d(a, b)) \ .$$

distance gibt 0 zurück, wenn sich die Objekte berühren oder durchdringen.

Der Operator *maxdist* dient dazu, den maximalen Abstandswert zweier räumlicher Objekte zu bestimmen. Er gibt ebenfalls eine reelle Zahl als Ergebnis zurück und ist wie folgt formal definiert:

$$maxdist(A, B) := \max_{a,b}(d(a, b)) \ .$$

Die Operatoren *isCloser* und *isFarther* basieren auf der minimalen Distanz. Die Argumente dieser Operatoren sind zwei räumliche Objekte A und B sowie ein positiver reeller Wert c. Sie geben einen booleschen Ergebniswert zurück und sind wie folgt formal definiert:

$$isCloser(A, B, c) \Leftrightarrow \min_{a,b}(d(a, b)) < c \ ,$$
$$isFarther(A, B, c) \Leftrightarrow \min_{a,b}(d(a, b)) > c \ .$$

Diese Operatoren dienen der Auswahl von Objekten, die sich innerhalb bzw. außerhalb eines bestimmten Umkreises eines Referenzobjekts befinden.

Der Operator *diameter* gibt den maximalen Abstand zweier Punkte eines Objekts zurück. Parameter ist ein räumliches Objekt A, Rückgabewert eine reelle Zahl. Seien $a, b \in A$. Dann gilt:

$$diameter(A) = \max_{a,b}(d(a, b) \ .$$

Der Operator *volume* gibt das Volumen eines Körpers zurück. Entsprechend ist der Operand ein Objekt vom Typ *Body* und Rückgabewert eine reelle Zahl.

7.4.3 Direktionale Operatoren

Direktionale Operatoren dienen zur Abfrage direktionaler Beziehungen. Sie sind wie die topologischen Operatoren Prädikate, d.h. sie besitzen einen booleschen Ergebniswert. Eine direktionale Beziehung beschreibt die relative Position zweier Objekte im Bezug auf den umgebenden Raum. Sie ist unabhängig von der internen Struktur der Objekte und der Skalierung des umgebenden Raums.

Im geographischen Kontext werden zur Angabe direktionaler Beziehungen in der Regel entweder vier (*nördlich, östlich, südlich, westlich*) oder acht (*nördlich, nordöstlich, östlich, südöstlich, südlich, südwestlich, westlich, nordwestlich*)

Hauptrichtungen (engl. *cardinal directions*) unterschieden (Frank, 1992). Für Szenen im dreidimensionalen Raum werden im Allgemeinen zusätzlich die Prädikate *über* und *unter* verwendet (Fuhr *et al.*, 1998).

Richtung ist eine binäre Relation, die für ein geordnetes Paar von Objekten A und B festgehalten wird. Dabei ist A das Referenzobjekt und B das Zielobjekt. Der dritte Teil einer Richtungsrelation ist der Referenzrahmen, der bestimmten Raumpartitionen Richtungssymbole bzw. -namen zuweist. Wird der Referenzrahmen gewechselt, ändert sich auch die Richtungsbeziehung.

Nach (Retz-Schmidt, 1988) gibt es drei Typen von Referenzrahmen:

- Ein *intrinsischer* Referenzrahmen bezieht sich auf die einem räumlichen Objekt innewohnende Orientierung. Beispielsweise ist die Vorderseite eines Gebäudes durch seinen Haupteingang gekennzeichnet.
- Ein *deiktischer* Referenzrahmen basiert auf der Position und der Orientierung des Betrachters. Dieser teilt den Raum in der Regel in die Partitionen *links, rechts, vorn, hinten, oben* und *unten*.
- Ein *extrinsischer* Referenzrahmen wird durch äußere Referenzpunkte festgelegt. Er gilt global und ist unabhängig von der Position und Orientierung des Beobachters und der betrachteten Objekte. In geographischen Anwendungen sind immer die beiden Pole der Erde die Referenzpunkte, die den Referenzrahmen der Himmelsrichtungen aufspannen.

Man unterscheidet des Weiteren zwischen quantitativen und qualitativen Richtungsbeziehungen. Bei quantitativen Richtungsbeziehungen wird in der Regel der Winkel zwischen dem Azimuth und dem Zielobjekt bestimmt. Bei qualitativen Beziehungen werden größere Bereiche unter einem gemeinsamen Namen subsumiert, was zu einer kleinen, überschaubaren Menge von Bezeichnungen führt. Für die Anwendung in räumlichen Anfragesprachen sind qualitative Richtungsbeziehungen deutlich besser geeignet.

Verwandte Arbeiten

Zwischen punktförmigen Objekten sind direktionale Beziehungen einfach und eindeutig zu definieren (Frank, 1996; Peuquet & Zhan, 1987). Um direktionale Beziehungen auch auf ausgedehnte, d.h. ein-, zwei- oder dreidimensionale Objekte anwenden zu können, werden daher häufig punktbasierte Repräsentationen, wie beispielsweise die Schwerpunkte der beteiligten Objekte, verwendet. Dies stellt jedoch eine recht grobe Annäherung dar und kann zu Ergebnissen führen, die sich nicht mit den intuitiven Erwartungen des Nutzers decken.

Für den zweidimensionalen Raum sind im Wesentlichen zwei Modelle für Richtungsbeziehungen zwischen Punkten bekannt: das kegelbasierte und das projektionsbasierte Modell. Das kegelbasierte System zerteilt den Raum um einen Referenzpunkt entweder in vier Partitionen von 90° (Abb. 7.13, links) oder in acht

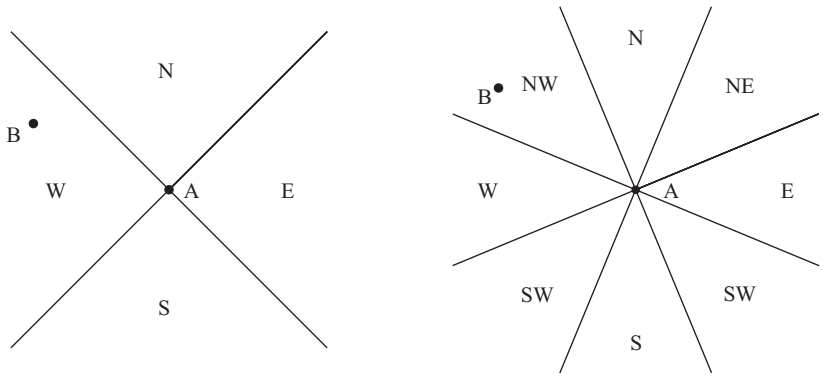


Abbildung 7.13: Das kegelbasierte Modell für Punktobjekte in 2D. **Links**: 4-Richtungssystem, **Rechts**: 8-Richtungssystem.

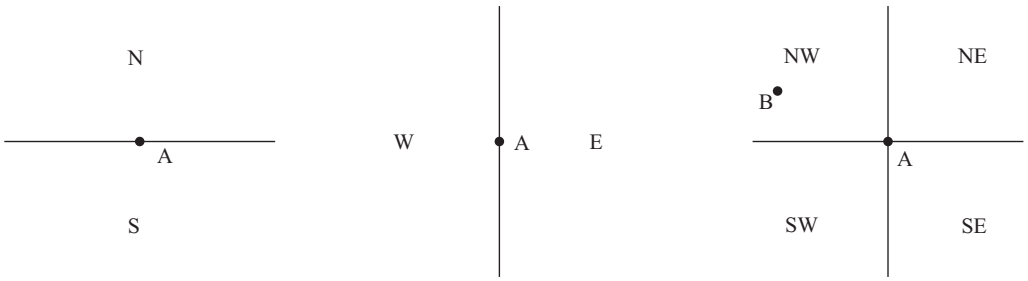


Abbildung 7.14: Das projektionsbasierte Modell für Punktobjekte in 2D.

Partitionen von 45° (Abb. 7.13, rechts) (Peuquet & Zhan, 1987; Hong, 1994; Abdelmoty, 1995; Frank, 1996; Shekhar *et al.*, 1999). Die Richtung eines Zielpunkts bezüglich des Referenzpunkts wird durch die Partition festgelegt, in der sich der Zielpunkt befindet.

Das projektionsbasierte Modell (Frank, 1996) zerteilt den Raum mittels horizontaler und vertikaler Linien, die sich im Referenzpunkt schneiden. Während eine Horizontale den Raum in nördlichen und südlichen Halbraum zerlegt, teilt eine vertikale Linie den Raum in westlichen und östlichen Halbraum. Eine Überlagerung ergibt vier Quadranten, denen die Bezeichnungen *nordwestlich, nordöstlich, südöstlich* und *südwestlich* zugewiesen werden (Abb. 7.14).

In (Zimmermann & Freksa, 1996) wird ein Framework zur Repräsentation von Richtungsbeziehungen eines Punktes bezüglich eines (einfachen) Linienobjektes vorgestellt. Demnach kann ein Punkt einen der 15 in Abb. 7.15 gezeigten qualitativ verschiedenen Orte bezüglich einer Linie einnehmen.

Mit den von Allen entwickelten Intervallrelationen können räumliche Beziehungen im eindimensionalen Raum abgebildet werden (Allen, 1983). Es ergeben sich 13 verschiedene Relationen (Abb. 7.16), die jedoch zum großen Teil topologischen Charakter besitzen. Lediglich *before* und *after* sind reine Richtungsbeziehungen.

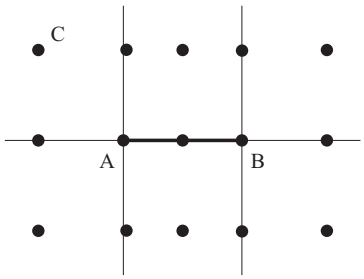


Abbildung 7.15: Nach (Zimmermann und Freksa, 1996) kann ein Punkt C bezüglich einer Linie AB 15 qualitativ verschiedene Positionen einnehmen.

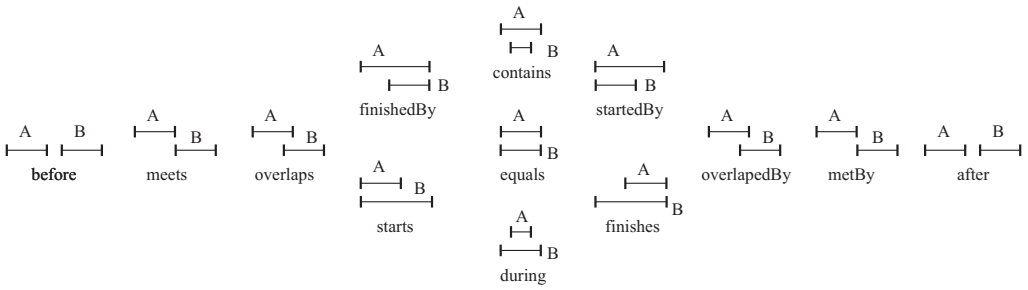


Abbildung 7.16: Die 13 von Allen identifizierten Intervallrelationen in 1D.

In (Guesgen, 1989) werden Allens Intervallrelationen zur Charakterisierung von räumlichen Beziehungen zwischen zweidimensionalen Objekten verwendet, indem diese auf die x- und die y-Achse abgebildet werden. Auf jeder Achse werden acht der 13 Intervallrelationen eingesetzt, namentlich *left*, *attached*, *overlapping* und *inside* sowie jeweils die inverse Relation. Mit diesem Modell sind damit $8 \cdot 8 = 64$ verschiedene Relationen in 2D beschreibbar. Das Modell nähert die Geometrie von ausgedehnten Objekten durch das minimale umgebende Rechteck (engl. *minimum bounding rectangle*, MBR) an, daher sind die sich ergebenden räumlichen Beziehungen nicht in jedem Fall die gleichen wie für die exakten Objekte. Sharma verwendet alle 13 Intervallrelationen, um Richtungsbeziehungen zwischen ausgedehnten Objekten zu beschreiben, und kombiniert diese mit topologischen Beziehungen um heterogenes räumliches Reasoning durchzuführen (Sharma, 1996).

Die Idee der Repräsentation von ausgedehnten räumlichen Objekten durch ihre MBRs und anschließender Projektion auf die Achsen des Koordinatensystems folgt auch der Ansatz von (Papadias *et al.*, 1995). Sie unterscheiden auf jeder Achse 13 Beziehungen im 1D-Raum. Entsprechend sind mit dieser Methode insgesamt $13 \cdot 13 = 169$ verschiedene direktionale Beziehungen darstellbar.

Im Gegensatz dazu wird in (Goyal, 2000) eine Methode vorgestellt, mit der Richtungsbeziehungen zwischen beliebig geformten Regionen festgehalten werden können, ohne ein Näherungsverfahren für ihre Geometrie einzusetzen. Er schlägt drei verschiedene Matrizen vor, mit deren Hilfe direktionale Beziehungen zwischen ausgedehnten Objekten formalisiert festgehalten werden können.

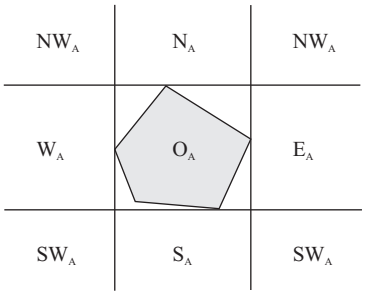


Abbildung 7.17: Die neun sich gegenseitig ausschließenden Richtungspartitionen der Richtungsmatrix von (Goyal, 2000).

Zur Formalisierung der Beziehungen zwischen Regionen wird zunächst die projektionsbasierte Methode angewendet, um den Raum um das Referenzobjekt in neun verschiedene Partitionen zu unterteilen, den so genannten *direction tiles* (Abb. 7.17). Dabei entsprechen die acht äußeren Partitionen den acht Himmelsrichtungen, während die Partition im Zentrum identisch mit dem kleinsten umgebenden Rechteck des Referenzobjekts ist und als *same* bezeichnet wird.

In einer *groben* Richtungsbeziehungsmatrix werden nun die Schnittmengen zwischen den Richtungspartitionen und dem Zielobjekt eingetragen:

$$\mathbf{dir}_{RR}(A,B) = \begin{bmatrix} NW_A \cap B & N_A \cap B & NE_A \cap B \\ W_A \cap B & O_A \cap B & E_A \cap B \\ SW_A \cap B & S_A \cap B & SE_A \cap B \end{bmatrix}.$$

Die Belegung der Matrix für das in Abb. 7.18 gezeigte Beispiel ergibt sich entsprechend zu

$$\mathbf{dir}_{RR}(A,B) = \begin{bmatrix} \varnothing & \neg\varnothing & \neg\varnothing \\ \varnothing & \neg\varnothing & \neg\varnothing \\ \varnothing & \varnothing & \neg\varnothing \end{bmatrix}.$$

Goyal entwickelt darüber hinaus noch eine zweite Matrix, mit der Richtungsbeziehungen quantitativ ausgedrückt werden können. Während in der *groben* Richtungsbeziehungsmatrix lediglich festgehalten wird, ob die sich ergebenden Schnittmengen leer oder nicht-leer sind, wird in der *detaillierten* Richtungsbeziehungsmatrix eingetragen, wie groß der Anteil des Zielobjekts ist, der sich mit der betreffenden Richtungspartition schneidet.

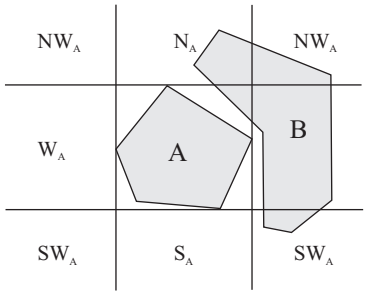


Abbildung 7.18: Beispielszene für die grobe und die detaillierte Richtungsbeziehungsmatrix.

Die detaillierte Richtungsbeziehungsmatrix ist wie folgt definiert:

$$\mathbf{dir}_{RR}(A,B) = \begin{bmatrix} \frac{area(NW_A \cap B)}{area(B)} & \frac{area(N_A \cap B)}{area(B)} & \frac{area(NE_A \cap B)}{area(B)} \\ \frac{area(W_A \cap B)}{area(B)} & \frac{area(O_A \cap B)}{area(B)} & \frac{area(E_A \cap B)}{area(B)} \\ \frac{area(SW_A \cap B)}{area(B)} & \frac{area(S_A \cap B)}{area(B)} & \frac{area(SE_A \cap B)}{area(B)} \end{bmatrix}.$$

Die detaillierte Richtungsbeziehungsmatrix für das in Abb. 7.18 gezeigte Beispiel ergibt sich entsprechend zu

$$\mathbf{dir}_{RR}(A,B) = \begin{bmatrix} 0 & 0.10 & 0.10 \\ 0 & 0.05 & 0.70 \\ 0 & 0 & 0.05 \end{bmatrix}.$$

Mit diesem quantitativen Richtungsmodell können Richtungsbeziehungen feiner aufgelöst und kleine Unterschiede zwischen verschiedenen Konstellationen besser erkannt werden. Dies ist vor allem für die Bestimmung der Ähnlichkeit verschiedener Szenen von Belang.

Mit dem „tiefen“ Richtungsmodell schlägt Goyal schließlich eine Erweiterung der groben Richtungsbeziehungsmatrix auf beliebige Kombinationen von punkt-, linien- oder regionenförmigen Objekten vor, bei dem zum einen der Referenzrahmen an den Typ des Referenzobjekts angepasst ist und zum anderen zusätzlich die Schnitte zwischen dem Zielobjekt und den Rändern der einzelnen Richtungspartitionen Berücksichtigung findet.

Zusammenfassend ist festzustellen, dass keines der vorgestellten Richtungsmodelle als adäquat für die Verwendung in einer räumlichen Anfragesprache für Bauwerksmodelle angesehen werden kann. Ziel muss es sein, dem Anwender intuitiv verständliche natürlichsprachliche Konstrukte zur Verfügung zu stellen. Die direkte Verwendung von Matrizen zur Beschreibung von Richtungsbeziehungen in einer Anfrage ist daher nicht akzeptabel. Darüber hinaus ist die mit den Richtungsmatrizen von Goyal erreichbare Unterscheidung von 512 verschiedenen

Konstellationen bei weitem zu feingranular, um von einem Nutzer sinnvoll verwendet werden zu können. Die von Guesgen und Papadias entwickelten Modelle verwenden zwar natürlichsprachliche Konstrukte, sie weisen jedoch den Makel einer Vermischung von topologischen und direktionalen Operatoren auf. Daher soll im Folgenden ein eigenständiges Framework zur Definition direktionaler Operatoren vorgestellt werden.

Definitionen

Das hier vorzustellende Modell verwendet einen extrinsischen Referenzrahmen, der durch die vom Planer gewählte Ausrichtung des Koordinatensystems gebildet wird. Das Modell gilt für beliebige Kombinationen von räumlichen Typen und beruht auf einer nach Koordinatenachsen getrennten Betrachtung direktionaler Beziehungen. Für jede Koordinatenachse gibt es genau zwei mögliche Beziehungen: in der x-Achse *eastOf* und *westOf*, in der y-Achse *northOf* und *southOf* und in der z-Achse *above* und *below*.

Die Wahl der Himmelsrichtungen für die Bezeichnungen anstelle von *links*, *rechts*, *vorn* und *hinten* dient der Abgrenzung gegenüber vom Beobachterstandort abhängigen Modellen.

Im Unterschied zu den Richtungsmodellen von (Guesgen, 1989), (Papadias *et al.*, 1995) und (Goyal, 2000) werden die Richtungsbeziehungen der einzelnen Koordinatenachsen nicht überlagert, sondern jeweils getrennt betrachtet. Im Modell wird also einem Zielobjekt nicht die Position *nordwestlich* zugewiesen, sondern sowohl *nördlich* als auch *westlich*.

Um den unterschiedlichen Anforderungen verschiedener Planungsaufgaben gerecht zu werden, bietet die räumliche Algebra zwei Gruppen direktionaler Operatoren an, denen verschiedene Modelle zur Beschreibung von Richtungsbeziehungen zugrunde liegen. Die erste Gruppe basiert auf einem projektionsbasierten Modell und berücksichtigt die tatsächliche Form von Referenz- und Zielobjekt. Die zweite hingegen verwendet Halbräume, die durch das kleinste umgebende Hexaeder (engl. *bounding box*) des Referenzobjekts gebildet werden.

Des Weiteren wird in beiden Gruppen zwischen relaxierten und strikten Operatoren unterschieden. Während die relaxierten Operatoren lediglich prüfen, ob *ein Teil* des Zielobjekts mit der gegebenen Richtungspartition überlappt, prüfen die strikten Operatoren, ob sich das *gesamte* Zielobjekt innerhalb der entsprechenden Richtungspartition befindet.

Projektionsbasierte direktionale Operatoren

Die Semantik dieser direktionalen Operatoren beruht auf einer Extrusion des Referenzobjekts entlang der betreffenden Richtung. Für *Point*-Objekte ergibt sich eine linienförmige Extrusion, für *Line*-Objekte eine Fläche, für *Surface*-Objekte ein Körper und für *Body*-Objekte ebenfalls ein Körper (Abb. 7.19).

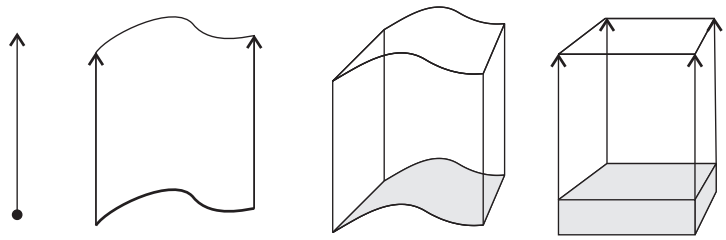


Abbildung 7.19: Extrusion von *Point*-, *Line*-, *Surface*- und *Body*-Objekten.

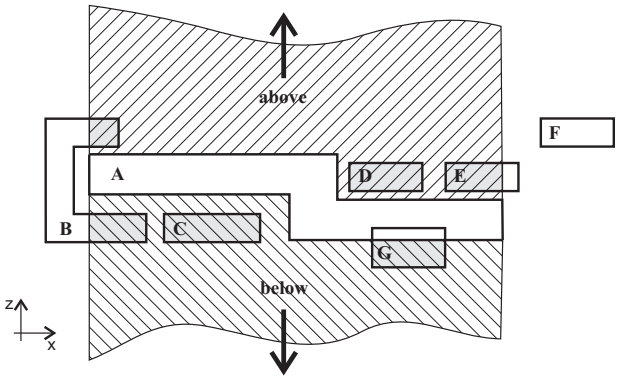


Abbildung 7.20: Das projektionsbasierte Richtungsmodell beruht auf einer Extrusion des Referenzobjekts A entlang der betrachteten Koordinatenachse. Ein projektionsbasierter Operator prüft entsprechend, ob ein Schnitt zwischen dem Zielobjekt und dem überstrichenen Raum vorliegt. Der relaxierte Operator *above_proj* gibt beispielsweise für die Zielobjekte B, D und E *true* zurück und für alle anderen Zielobjekte *false*. Der strikte Operator *above_proj_strict* gibt hingegen auch für B und E *false* zurück. Für das überlappende Zielobjekt G gibt der relaxierte Operator *true* zurück, der strikte hingegen *false*.

Der dabei überstrichene Raum bildet die Richtungspartition (Abb. 7.20). Die relaxierten Operatoren prüfen, ob es einen Schnitt zwischen der Richtungspartition und dem Zielobjekt gibt, und geben entsprechend *wahr* oder *falsch* zurück. Seien A und B räumliche Objekte vom Typ *Spatial*, wobei A das Referenz- und B das Zielobjekt sei. Weiterhin gelte $a \in A, b \in B$. Die formale Definition der relaxierten projektionsbasierten Richtungsoperatoren lautet somit:

$$
\begin{aligned}
\mathit{eastOf\_proj}(A,B) &\Leftrightarrow \exists a,b \text{ mit } a_y = b_y \wedge a_z = b_z : a_x < b_x \ , \\
\mathit{westOf\_proj}(A,B) &\Leftrightarrow \exists a,b \text{ mit } a_y = b_y \wedge a_z = b_z : a_x > b_x \ , \\
\mathit{northOf\_proj}(A,B) &\Leftrightarrow \exists a,b \text{ mit } a_x = b_x \wedge a_z = b_z : a_y < b_y \ , \\
\mathit{southOf\_proj}(A,B) &\Leftrightarrow \exists a,b \text{ mit } a_x = b_x \wedge a_z = b_z : a_y > b_y \ , \\
\mathit{above\_proj}(A,B) &\Leftrightarrow \exists a,b \text{ mit } a_x = b_x \wedge a_y = b_y : a_z < b_z \ , \\
\mathit{below\_proj}(A,B) &\Leftrightarrow \exists a,b \text{ mit } a_x = b_x \wedge a_y = b_y : a_z > b_z \ .
\end{aligned}
$$

Die *strikten* projektionsbasierten Operatoren prüfen hingegen, ob sich das *gesamte* Zielobjekt innerhalb der betreffenden Richtungspartition befindet. Sie sind daher wie folgt formal definiert:

$$
\begin{aligned}
\mathit{eastOf\_proj\_strict}(A,B) \Leftrightarrow \forall b : &(\exists a : a_y = b_y \wedge a_z = b_z \wedge a_x < b_x) \wedge \\
&(\nexists a : a_y = b_y \wedge a_z = b_z \wedge a_x \geq b_x) \ , \\
\mathit{westOf\_proj\_strict}(A,B) \Leftrightarrow \forall b : &(\exists a : a_y = b_y \wedge a_z = b_z \wedge a_x > b_x) \wedge \\
&(\nexists a : a_y = b_y \wedge a_z = b_z \wedge a_x \leq b_x) \ , \\
\mathit{northOf\_proj\_strict}(A,B) \Leftrightarrow \forall b : &(\exists a : a_x = b_x \wedge a_z = b_z \wedge a_y < b_y) \wedge \\
&(\nexists a : a_x = b_x \wedge a_z = b_z \wedge a_y \geq b_y) \ , \\
\mathit{southOf\_proj\_strict}(A,B) \Leftrightarrow \forall b : &(\exists a : a_x = b_x \wedge a_z = b_z \wedge a_y > b_y) \wedge \\
&(\nexists a : a_x = b_x \wedge a_z = b_z \wedge a_y \leq b_y) \ , \\
\mathit{above\_proj\_strict}(A,B) \Leftrightarrow \forall b : &(\exists a : a_x = b_x \wedge a_y = b_y \wedge a_z < b_z) \wedge \\
&(\nexists a : a_x = b_x \wedge a_y = b_y \wedge a_z \geq b_z) \ , \\
\mathit{below\_proj\_strict}(A,B) \Leftrightarrow \forall b : &(\exists a : a_x = b_x \wedge a_y = b_y \wedge a_z > b_z) \wedge \\
&(\nexists a : a_x = b_x \wedge a_y = b_y \wedge a_z \leq b_z) \ .
\end{aligned}
$$

Umgangssprachlich ließe sich die Semantik des Operators *above_proj_strict* beispielsweise mit „direkt darüber“ oder „ausschließlich darüber“ beschreiben. Wie Abb. 7.20 am Beispiel von Zielobjekt G zeigt, geben die strikten direktionalen Operatoren bei einer Überlappung von Referenz- und Zielobjekt *false* zurück.

Aus der Symmetrie der Definitionen ist zu erkennen, dass im Grunde die Bereitstellung zweier Operanden unter Hinzunahme einen dritten Parameters, der die betrachtete Richtung beschreibt, ausreichend wäre. Hierbei gestaltet sich jedoch zum einen die Namensfindung für die resultierenden Operatoren schwierig, zum anderen widerspricht diese Vorgehensweise dem Designziel einer möglichst hohen Benutzerfreundlichkeit, dessen Realisierung sich im Wesentlichen auf die Verwendung natürlichsprachlicher Konstrukte für die bereitgestellten Operatornamen stützt. Daher wurde eine ausführlichere Definition bevorzugt.

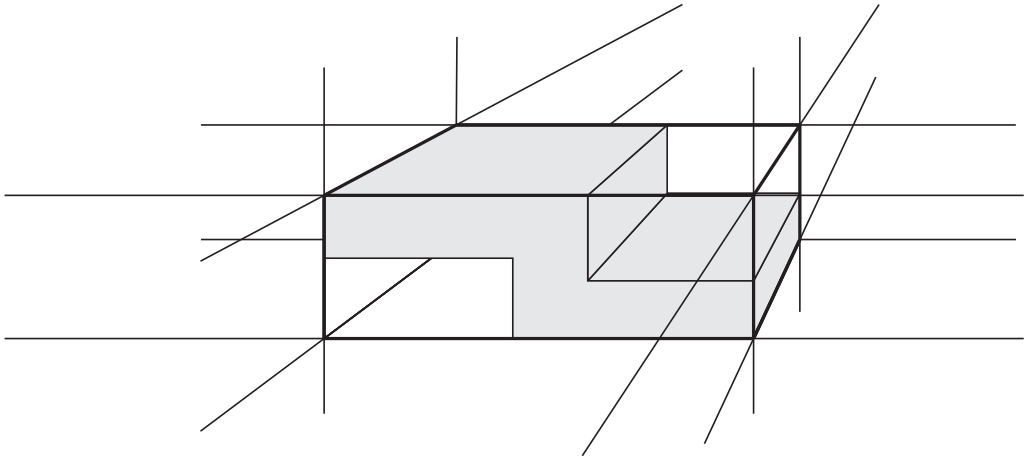


Abbildung 7.21: Die Seiten der *Axis Aligned Bounding Box* des Referenzobjekts bilden die Ebenen, durch die die Halbräume begrenzt sind, die jeweils eine Richtungspartition bilden.

Die projektionsbasierten Operatoren erfüllen nicht das Prinzip der Vollständigkeit, da ein Objekt nicht zwangsläufig in eine der beiden Richtungen pro Koordinatenachse eingeordnet werden kann. So befindet sich Zielobjekt E in Abb. 7.20 nach den Definitionen weder oberhalb noch unterhalb von A. Anders ausgedrückt, bildet die Vereinigung der beiden Richtungspartitionen einer Koordinatenrichtung in der Regel nicht den gesamten Raum $\mathbb{R}^3$.

Das Prinzip der Exklusivität, also des gegenseitigen Ausschließens der direktionalen Prädikate, wird lediglich von den *strikten* Operatoren erfüllt: Wenn eines der beiden Richtungsprädikate einer Koordinatenachse erfüllt ist, ist das andere zwangsläufig nicht erfüllt.

Halbraum-basierte direktionale Operatoren

Die zweite Gruppe direktionaler Operatoren basiert auf einem Richtungsmodell, das auf Halbräumen beruht. Diese Halbräume werden durch Ebenen gebildet, in denen die Seiten des kleinsten das Referenzobjekt umgebenden achsenparallelen Hexaeders (engl. *Axis Aligned Bounding Box*, AABB) liegen (Abb. 7.21).

Analog zu den projektionsbasierten Operatoren prüfen auch hier die *relaxierten* Operatoren lediglich, ob ein Schnitt des Zielobjekts mit dem betreffenden Halbraum vorliegt. Die strikten Operatoren hingegen prüfen, ob das *gesamte* Zielobjekt in diesem Halbraum liegt.

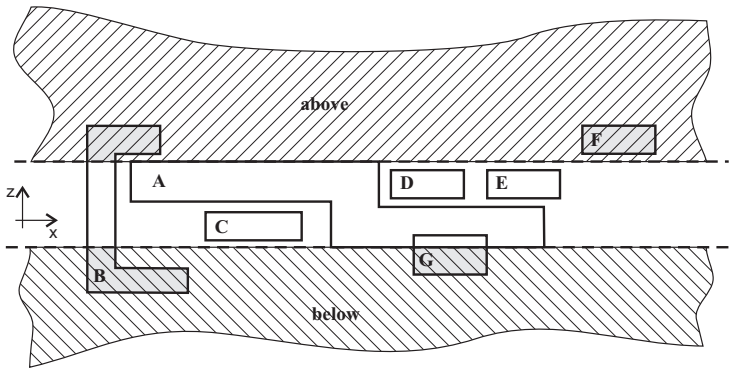


Abbildung 7.22: Halbraum-basiertes direktionales Modell. Die Auswertung der direktionalen Prädikate beruht auf einer Prüfung des Schnitts zwischen dem Zielobjekt und dem betreffenden Halbraum. Dient A als Referenzobjekt, gibt der relaxierte Operator *above_hs* für die Zielobjekte B und F *true* zurück, der strikte Operator *above_hs_strict* hingegen lediglich für F.

Seien A und B räumliche Objekte vom Typ *Spatial* und $a \in A, b \in B$. Dann gilt für die *relaxierten* Halbraum-basierten Operatoren:

$$\begin{aligned}
eastOf_hs(A,B) &\Leftrightarrow \forall a : \exists b : a_x < b_x \ , \\
westOf_hs(A,B) &\Leftrightarrow \forall a : \exists b : a_x > b_x \ , \\
northOf_hs(A,B) &\Leftrightarrow \forall a : \exists b : a_y < b_y \ , \\
southOf_hs(A,B) &\Leftrightarrow \forall a : \exists b : a_y > b_y \ , \\
above_hs(A,B) &\Leftrightarrow \forall a : \exists b : a_z < b_z \ , \\
below_hs(A,B) &\Leftrightarrow \forall a : \exists b : a_z > b_z \ .
\end{aligned}$$

Die formale Definition der *strikten* Halbraum-basierten Operatoren lautet entsprechend wie folgt:

$$\begin{aligned}
eastOf_hs_strict(A,B) &\Leftrightarrow \forall a, b : a_x < b_x \ , \\
westOf_hs_strict(A,B) &\Leftrightarrow \forall a, b : a_x > b_x \ , \\
northOf_hs_strict(A,B) &\Leftrightarrow \forall a, b : a_y < b_y \ , \\
southOf_hs_strict(A,B) &\Leftrightarrow \forall a, b : a_y > b_y \ , \\
above_hs_strict(A,B) &\Leftrightarrow \forall a, b : a_z < b_z \ , \\
below_hs_strict(A,B) &\Leftrightarrow \forall a, b : a_z > b_z \ .
\end{aligned}$$

Wie in Abb. 7.22 gezeigt, kann im Gegensatz zum projektionsbasierten direktionalen Modell den Zielobjekten *C, D* und *E* keine Richtung bezüglich A zugewiesen werden. Dafür trifft das Halbraum-basierte Modell die Aussage, dass das Zielobjekt *E oberhalb* von A liegt.

Auch das Halbraum-basierte Modell genügt nicht dem Prinzip der Vollständigkeit, da die Vereinigung der beiden Halbräume einer Koordinatenrichtung nicht zwangsläufig den gesamten Raum bildet. Das Prinzip der Exklusivität ist für die strikten Operatoren erfüllt, für die relaxierten Operatoren hingegen nicht.

7.4.4 Topologische Operatoren

Topologische Operatoren werden verwendet, um das Auftreten einer topologischen Beziehung zwischen zwei räumlichen Objekten abzufragen. Da sie einen booleschen Wert zurückgeben, werden sie auch als *topologische Prädikate* bezeichnet. Alle topologischen Operatoren besitzen genau zwei Operanden: die beiden räumlichen Objekte, für die die entsprechende Beziehung getestet wird.

Topologische Beziehungen können wie folgt formal definiert werden: Seien X und Y topologische Räume. Eine Abbildung $f : X \to Y$ wird als stetig bezeichnet, wenn für jede offene Teilmenge V von Y die Menge $f^{-1}(V)$ eine offene Teilmenge von X ist. Ist die Abbildung f eine Bijektion[8] und sind sowohl f als auch die Inverse f^{-1} stetig, dann wird f als *topologischer Isomorphismus* bezeichnet. Topologische Ismorphismen erhalten Nachbarschaftsbeziehungen zwischen Punkten während der Abbildung. Dazu zählen Translation, Rotation und Skalierung sowie eine beliebige Kombination aus diesen drei Transformationen. *Topologische Relationen* lassen sich als räumliche Beziehungen klassifizieren, die unter einem topologischen Isomorphismus invariant sind.

Verwandte Arbeiten

Topologische Beziehungen gehören zu den am intensivsten untersuchten räumlichen Beziehungen. Schnell wurde erkannt, dass die Ungenauigkeit der menschlichen Sprache bei der Beschreibung topologischer Beziehungen eine Formalisierung erfordert.

Ein erster wesentlicher Schritt zu einer solchen Formalisierung war die Entwicklung des *4-Intersection Model* (Egenhofer & Herring, 1990; Egenhofer & Franzosa, 1991). Es basiert darauf, dass für topologische Prädikate (innerhalb, außerhalb, berührt etc.) die Schnittmengen zwischen dem *Inneren* (A°) bzw. dem *Rand* (∂A) des ersten Operanden und dem *Inneren* (B°) bzw. *Rand* (∂B) des zweiten Operanden festgeschrieben werden. Für die vier entstehenden Schnittmengen werden die Werte *leere Menge* ($\varnothing$) und *nicht-leere Menge* ($\neg\varnothing$) in einem Tupel festgehalten.

Es ergeben sich theoretisch 16 mögliche Belegungen, jedoch ist je nach Art der beteiligten geometrischen Objekte nur ein Teil in der Realität vorzufinden. Dieses Prinzip wurde zunächst auf Beziehungen zwischen Intervallen im eindimensionalen Raum (Pullar & Egenhofer, 1988) und später auf Beziehungen zwischen einfachen Regionen in $\mathbb{R}^2$ angewendet (Egenhofer & Franzosa, 1991). Dabei wurden systematisch unmögliche Belegungen ausgeschlossen.

In beiden Fällen wurden acht Relationen identifiziert, denen jeweils eine natürlichsprachliche Bezeichnung zugewiesen wird: *disjoint, touch, equals, inside, contains, covers, coveredBy* und *overlap* (Abb. 7.23).

[8] umkehrbar eindeutige Zuordnung zwischen den Elementen zweier Mengen

	$\partial A \cap \partial B$	$A^\circ \cap B^\circ$	$\partial A \cap B^\circ$	$A^\circ \cap \partial B$	
r_0	($\varnothing$,	$\varnothing$,	$\varnothing$,	$\varnothing$)	disjoint
r_1	($\neg\varnothing$,	$\varnothing$,	$\varnothing$,	$\varnothing$)	touch
r_3	($\neg\varnothing$,	$\neg\varnothing$,	$\varnothing$,	$\varnothing$)	equal
r_6	($\varnothing$,	$\neg\varnothing$,	$\neg\varnothing$,	$\varnothing$)	inside
r_7	($\neg\varnothing$,	$\neg\varnothing$,	$\neg\varnothing$,	$\varnothing$)	coveredBy
r_{10}	($\varnothing$,	$\neg\varnothing$,	$\varnothing$,	$\neg\varnothing$)	contains
r_{11}	($\neg\varnothing$,	$\neg\varnothing$,	$\varnothing$,	$\neg\varnothing$)	covers
r_{14}	($\neg\varnothing$,	$\neg\varnothing$,	$\neg\varnothing$,	$\neg\varnothing$)	overlap

Abbildung 7.23: Die 4-Tupel der Interjektionen für topologische Prädikate nach (Pullar & Egenhofer, 1988) und (Egenhofer & Franzosa, 1991).

Um *Line-Line*-Relationen in $\mathbb{R}^2$ besser erfassen zu können, wird in (Egenhofer, 1991) das *4-Intersection Model* zum *9-Intersection Model* (9-IM) erweitert, indem zusätzlich die Schnittmengen zwischen dem Äußeren A^- und B^- der beteiligten Objekte Berücksichtigung finden. Die neun sich ergebenden Mengen werden in einer Matrix eingetragen:

$$\mathbf{I} = \begin{pmatrix} A^\circ \cap B^\circ & A^\circ \cap \partial B & A^\circ \cap B^- \\ \partial A \cap B^\circ & \partial A \cap \partial B & \partial A \cap B^- \\ A^- \cap B^\circ & A^- \cap \partial B & A^- \cap B^- \end{pmatrix} \quad .$$

Wendet man diese Methode für die Beziehungen zwischen einfachen Regionen an und schließt alle nicht-realisierbaren Konfigurationen aus, ergeben sich die in Abb. 7.24 gezeigten acht Prädikate.

Nach topologischen Beziehungen zwischen einfachen Regionen werden auch allgemeine Beziehungen zwischen Punkt- und Linien-Objekten in $\mathbb{R}^2$ mit Hilfe des 9-IM untersucht (Egenhofer & Herring, 1992). Dabei gibt es sechs Operandenkombinationen: *Region/Region*, *Line/Region*, *Point/Region*, *Line/Line*, *Point/Line* und *Point/Point*. Bei den Untersuchungen wird zwar weiterhin von *einfachen* Regionen und *einfachen* Punkten ausgegangen, aber bereits komplexe Linien als Linien mit mehr als zwei getrennten Rändern einbezogen. Es werden also räumliche Objekte berücksichtigt, die eine Separation des Randes aufweisen, jedoch keine, deren Inneres geteilt ist.

Abb. 7.25 zeigt die hohe Zahl der durch das 9-IM abbildbaren topologischen Konfigurationen zwischen einfachen räumlichen Objekten in 2D. Es ergibt sich das Problem, dass die Unterscheidung in derart viele Konstellationen für die meisten Anwendungen zu feingranular und vom Nutzer schwer zu handhaben ist. Das zeigt sich u.a. daran, dass es für viele Konstellationen meist nur ein einzige natürlichsprachliche Beschreibung gibt.

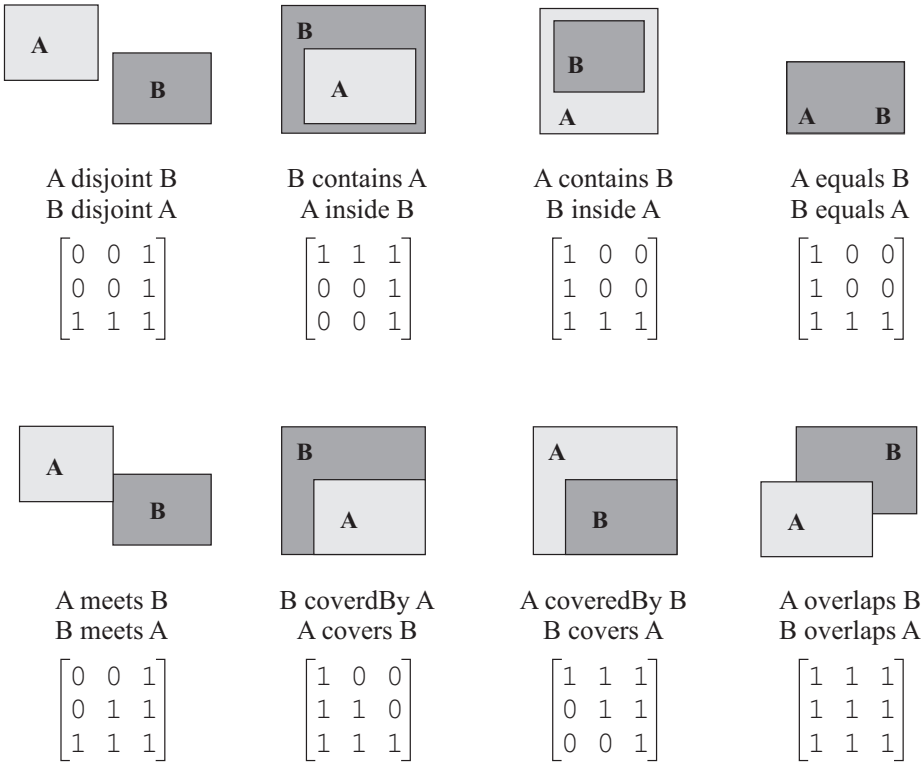


Abbildung 7.24: Die 9-Intersection-Matrizen der acht Prädikate für topologische Beziehungen zwischen zwei einfachen Regionen.

	simple point	simple line	simple region
simple point	2	3	3
simple line	3	33	19
simple region	3	19	8

Abbildung 7.25: Zahl der topologischen Konfigurationen zwischen zwei einfachen räumlichen Objekten, die durch die 9-IM beschrieben werden können.

Ein Nachteil der 9-IM ist, dass sie mitunter zwei intuitiv verschiedenartige topologische Situationen als identisch einstuft. Beispielsweise führen zwei Regionen, die sich genau einen Punkt teilen, zur gleichen Belegung der 9-IM-Matrix wie zwei Regionen, die eine gemeinsame linienförmige Grenze aufweisen. In beiden Fällen wird der Beziehung das Prädikat *(meet)* zugewiesen. Bestimmte Unterschiede in topologischen Beziehungen können nur aufgelöst werden, wenn neben dem Inhaltskriterium (leere oder nicht-leere Menge) andere topologische Invarianten hinzugezogen werden. Beispiele für topologische Invarianten, die für die Schnittmenge festgehalten werden können, sind deren Dimension sowie deren Zahl an Separationen.

So wird in (Egenhofer & Franzosa, 1991) gezeigt, dass detailliertere topologische Beziehungen beschrieben werden können, wenn die Dimension der Schnittmengen, an denen ein Rand beteiligt ist, ebenfalls berücksichtigt wird. Beispielsweise können zwei Regionen einen Punkt (nulldimensionale Schnittmenge) oder ein linienförmiges Objekt (eindimensionale Schnittmenge) gemeinsam haben, was zu spezialisierteren Beziehungen wie *0-meet* und *1-meet* führt.

Diese Idee wird auch von (Clementini *et al.*, 1993) aufgegriffen. Sie schlagen die *Dimension Extended Method* vor, die die Dimensionen der Schnittmengen des *4-Intersection Model* berücksichtigt. Demnach kann im zweidimensionalen Raum der Schnitt zweier Punktmengen leer, nulldimensional (Punkt), eindimensional (Linie) oder zweidimensional (Region) sein. Um das Problem der zu hohen Zahl an Konstellationen zu lösen, werden des Weiteren die mit Hilfe des 9-IM modellierbaren Beziehungen zu fünf Prädikaten *touch*, *in*, *cross*, *overlap* und *disjoint* gruppiert. Die fünf Prädikate sind für alle möglichen Kombinationen der Typen *Punkt*, *Linie* und *Region* anwendbar.

Es wird nachgewiesen, dass diese fünf topologischen Beziehungen *sich gegenseitig ausschließen*, d.h. dass niemals mehr als eine Beziehung gelten kann, und dass sie *vollständig* sind, d.h. jede mögliche Konstellation zwischen zwei geometrischen Objekten genau einem der fünf Prädikate entspricht.

Die *Dimension Extended Method* wird in (Clementini & Di Felice, 1995) mit dem *9-Intersection Model* verknüpft und das daraus entstehende Modell als *Dimension Extended 9-Intersection Model* (DE-9IM) bezeichnet. Es basiert auf der Formalisierung topologischer Beziehungen mit Hilfe einer um den Dimensionsoperator erweiterten Matrix:

$$\mathbf{I_{DE}} = \begin{pmatrix} \dim(A^\circ \cap B^\circ) & \dim(A^\circ \cap \partial B) & \dim(A^\circ \cap B^-) \\ \dim(\partial A \cap B^\circ) & \dim(\partial A \cap \partial B) & \dim(\partial A \cap B^-) \\ \dim(A^- \cap B^\circ) & \dim(A^- \cap \partial B) & \dim(A^- \cap B^-) \end{pmatrix} .$$

Der Dimensionsoperator dim(S) ist dabei definiert als:

$$\dim(S) = \begin{cases} - & \text{wenn S} = \varnothing \\ 0 & \text{wenn S mindestens einen Punkt und weder Regionen} \\ & \text{noch Linien beinhaltet} \\ 1 & \text{wenn S mindestens eine Linie und keine Regionen beinhaltet} \\ 2 & \text{wenn S mindestens eine Region beinhaltet} \end{cases}$$

In (Clementini & Di Felice, 1996) wird gezeigt, dass das DE-9IM mit leichten Abwandlungen auch für komplexe räumliche Objekte eingesetzt werden kann (vgl. Abschnitt 7.2.4). McKenney *et al.* greifen diesen Ansatz auf, verwenden jedoch eine spezielle dim(P)-Funktion, die nicht nur die maximale Dimension der resultierenden Punktmenge, sondern die Dimensionen aller ihrer Komponenten zurückgibt (McKenney *et al.*, 2005).

Das DE-9IM wurde vom OpenGIS-Konsortium im Rahmen seiner Standardisierungsbemühungen für die Definition von topologischen Beziehungen übernommen (OpenGIS Consortium (OGC), 1999). Die Matrizen der definierten Prädikate können den Abbildungen 7.26 und 7.27 entnommen werden. Es ist zu beachten, dass im Standard eine gegenüber (Clementini & Di Felice, 1995) leicht veränderte Notation verwendet wird: T (für *true*) steht für die nicht-leere Menge, F (für *false*) steht für die leere Menge, die Zahlen *0, 1* und *2* für die Dimension der Schnittmenge entsprechend der Definition des dim(S)-Operators und der Stern (*) fungiert als Platzhalter. Die Angabe eines Platzhalters bedeutet, dass die betreffende Schnittmenge leer oder nicht-leer sein kann, sie also keinen Einfluss auf die Vergabe des entsprechenden Prädikats hat.

Aus Gründen der Benutzerfreundlichkeit wurden zusätzlich die Prädikate *contains* (als Inverse zu *within*) und *intersects* (mit der Bedeutung *not disjoint*) aufgenommen. Damit gilt allerdings die von Clementini *et al.* geforderte Eigenschaft, dass sich die Prädikate untereinander gegenseitig ausschließen, nur noch bedingt.

Eine eingehende Untersuchung zu topologischen Beziehungen zwischen *komplexen* räumlichen Objekten wird in (Schneider & Behr, 2006) vorgenommen. Sie verwenden wiederum die ursprüngliche Variante der 9-IM, bei der nur die Inhaltsinvariante festgeschrieben wird. Da die Zahl der topologischen Konstellationen für komplexe räumliche Objekte noch weitaus höher ist als für einfache Objekte, schlagen Schneider & Behr ein *Clustering* topologischer Situationen vor. Dabei werden in Analogie zur Notation im OGC-Standard verschiedene topologische Konstellationen zusammengefasst, indem nur für einen Teil der neun Schnittmengen eine Festlegung getroffen wird. Für alle anderen Stellen der 9-IM werden Platzhalter (*) verwendet.

Schneider & Behr weisen darauf hin, dass die Wahl der Clustering-Regeln beliebig ist, solange sich die ergebenden Prädikate gegenseitig ausschließen. Des

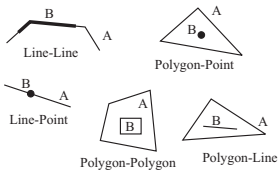
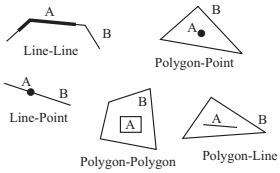
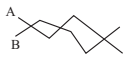
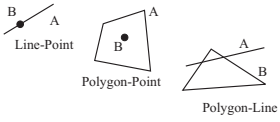
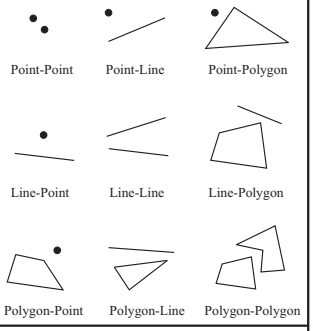
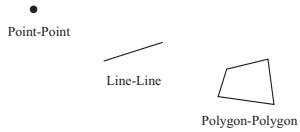
Operator	9-Intersection-Matrix	Operanden		Beispiele
		erster (A)	zweiter (B)	
contains	$\begin{bmatrix} T & * & * \\ * & * & * \\ F & F & * \end{bmatrix}$	Line	Point	
		Line	Line	
		Polygon	Point	
		Polygon	Line	
		Polygon	Polygon	
within	$\begin{bmatrix} T & * & F \\ * & * & F \\ * & * & * \end{bmatrix}$	Point	Line	
		Line	Line	
		Point	Polygon	
		Line	Polygon	
		Polygon	Polygon	
cross	$\begin{bmatrix} 0 & * & * \\ * & * & * \\ * & * & * \end{bmatrix}$	Line	Line	
	$\begin{bmatrix} T & * & T \\ * & * & * \\ * & * & * \end{bmatrix}$	Line	Point	
		Line	Polygon	
		Polygon	Point	
disjoint	$\begin{bmatrix} F & F & * \\ F & F & * \\ * & * & * \end{bmatrix}$	Point	Point	
		Point	Line	
		Point	Polygon	
		Line	Point	
		Line	Line	
		Line	Polygon	
		Polygon	Point	
		Polygon	Line	
		Polygon	Polygon	
equals	$\begin{bmatrix} T & F & F \\ F & T & F \\ F & F & T \end{bmatrix}$	Point	Point	
		Line	Line	
		Polygon	Polygon	

Abbildung 7.26: Im OGC-Standard verankerte topologische Prädikate und ihre formale Definitionen (Teil 1).

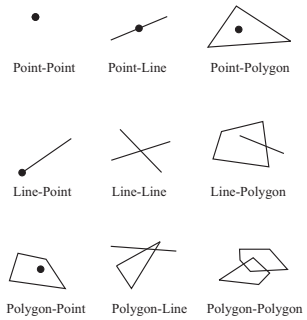
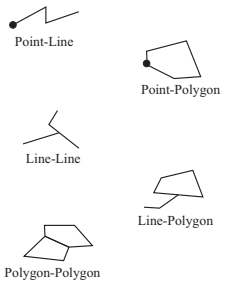
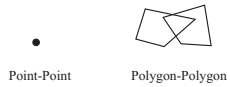
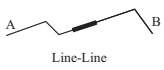
Operator	9-Intersection-Matrix	Operanden		Beispiele
		erster (A)	zweiter (B)	
intersect	$\begin{bmatrix} T & * & * \\ * & * & * \\ * & * & * \end{bmatrix}$ or $\begin{bmatrix} * & * & * \\ * & T & * \\ * & * & * \end{bmatrix}$ or $\begin{bmatrix} * & T & * \\ * & * & * \\ * & * & * \end{bmatrix}$ or $\begin{bmatrix} T & * & * \\ * & * & * \\ * & * & * \end{bmatrix}$	Point	Point	
		Point	Line	
		Point	Polygon	
		Line	Point	
		Line	Line	
		Line	Polygon	
		Polygon	Point	
		Polygon	Line	
		Polygon	Polygon	
touch	$\begin{bmatrix} F & T & * \\ * & * & * \\ * & * & * \end{bmatrix}$ or $\begin{bmatrix} F & * & * \\ T & * & * \\ * & * & * \end{bmatrix}$ or $\begin{bmatrix} F & * & * \\ * & T & * \\ * & * & * \end{bmatrix}$	Point	Line	
		Point	Polygon	
		Line	Line	
		Line	Polygon	
		Polygon	Polygon	
overlaps	$\begin{bmatrix} F & T & * \\ * & * & * \\ * & * & * \end{bmatrix}$	Point	Point	
		Polygon	Polygon	
	$\begin{bmatrix} 1 & * & T \\ * & * & * \\ T & * & * \end{bmatrix}$	Line	Line	

Abbildung 7.27: Im OGC-Standard verankerte topologische Prädikate und ihre formale Definitionen (Teil 2).

	complex point	complex line	complex region
complex point	5	14	7
complex line	14	82	43
complex region	7	43	33

Abbildung 7.28: Zahl der topologischen Konfigurationen zwischen zwei komplexen räumlichen Objekten, die durch die 9-IM beschrieben werden können (Schneider, 2006).

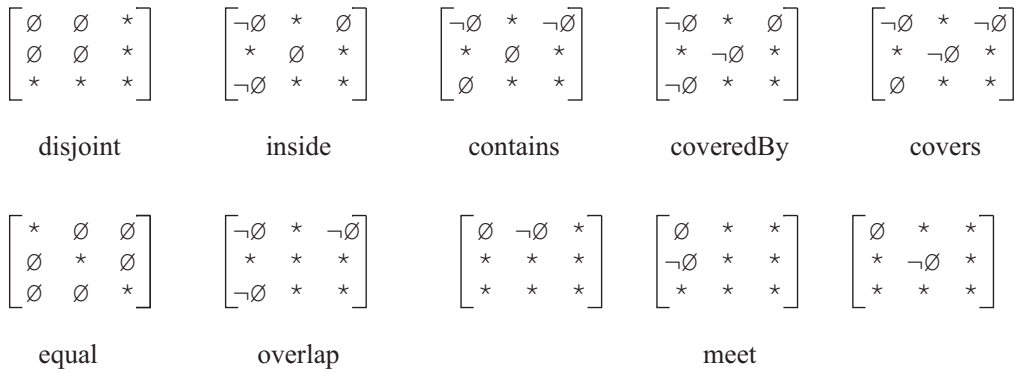


Abbildung 7.29: Die 9-IM-Matrizen für die in (Schneider & Behr, 2006) vorgeschlagenen Cluster-Prädikate.

	Point	Line	Surface	Body
Point	2	3	3	3
Line	3	33	31	19
Surface	3	31	38	19
Body	3	19	19	8

Abbildung 7.30: Die in (Zlatanova, 2000) ermittelte Anzahl der mit Hilfe der 9-IM beschreibbaren topologischen Beziehungen zwischen einfachen Objekten in 3D.

Weiteren sollten sie den Erwartungen des Anwenders und den Anforderungen des Anwendungsgebietes entsprechen. Das Clustering hat auch für die Performanz einer Implementierung Vorteile, da für die Prüfung eines Prädikats weniger Schnittmengen berechnet werden müssen.

Schneider & Behr stellen als Beispiel acht Cluster-Prädikate vor: *disjoint, meet, inside, contains, coveredBy, covers, equal* und *overlap* (Abb. 7.29). Wendet man sie auf einfache Regionen an, sind sie äquivalent zu den Definitionen der Prädikate gleichen Namens in (Egenhofer & Franzosa, 1991).

Das DE-9IM bzw. die ihm zugrunde liegende Kalkül-basierte Methode wird in (van Oosterom *et al.*, 1994) auf dreidimensionale Modelle angewandt. Wei *et al.* verwenden hingegen das einfache 9-IM für die Beschreibung topologischer Beziehungen zwischen dreidimensionalen räumlichen Objekten (Wei *et al.*, 1998). In beiden Fällen bildet die zelluläre Zerlegung die Grundlage der Modellierung der räumlichen Objekte.

In (Zlatanova, 2000) werden die möglichen topologischen Beziehungen zwischen einfachen 0-, 1-, 2- und 3-dimensionalen räumlichen Objekten untersucht. Auf Basis des einfachen *9-Intersection Model* werden systematisch Bedingungen entwickelt, die die in der Realität nicht konstruierbaren Konstellationen ausschließen. Die Ergebnisse der Untersuchungen sind in Abb. 7.30 zusammengefasst.

Da analog zum zweidimensionalen Fall die Verwendung aller dieser Konstellationen als topologische Prädikate nicht sinnvoll ist, wird in (Borrmann *et al.*, 2006) die Anwendung der DE-9IM für die Definition topologischer Beziehungen zwischen komplexen dreidimensionalen Objekten vorgeschlagen. Dies resultiert in den neun topologischen Prädikaten *disjoint, equal, contain, within, overlap, touch, meet, onBoundary* und *cross*. Dabei wird der Dimensionsoperator für die Definition von *overlap* und *cross* verwendet.

Definitionen

Für die im Rahmen dieser Arbeit vorgeschlagene Implementierungsvariante auf Basis von Rastergeometrien (Kap. 8) ist der Dimensionsoperator jedoch nur schwer zu realisieren. Daher wird hier in Anlehnung an (Schneider & Behr, 2006) die Verwendung des „puren“ 9-IM bevorzugt, das sich ausschließlich auf die Inhaltsinvariante stützt und daher als Einträge in der Intersection-Matrix lediglich „leere Menge“ und „nicht-leere Menge“ erlaubt.

Da auch hier die Anzahl der möglichen topologischen Beziehungen zu groß wäre, wird, wie in (Schneider & Behr, 2006) vorgeschlagen, die Methode des *Clustering* angewendet. In den Matrizen werden an den Stellen Platzhalter (*) eingetragen, die keinen Einfluss auf die Vergabe des entsprechenden topologischen Prädikats haben.

In Abb. 7.31 und 7.32 sind die in die Algebra aufgenommenen topologischen Operatoren zusammen mit der 9-Intersection-Matrix aufgeführt. Auf der rechten Seite der Abbildungen befinden sich Piktogramme, die die Semantik des Prädikates für verschiedene Typkombinationen illustrieren. Das System der topologischen Operatoren genügt der Forderung nach Vollständigkeit und gegenseitigem Ausschluss der einzelnen Prädikate. Dies erlaubt die Einführung des Operators *whichTopoPredicate*, der für zwei übergebene räumliche Objekte beliebigen Typs das zutreffende topologische Prädikat zurückliefert.

Die Verwendung der einfachen 9-Intersection-Matrix ohne Dimensionsoperator führt dazu, dass im Gegensatz zu dem in (Borrmann *et al.*, 2006) aufgestellten System eine Unterscheidung zwischen *overlap* und *cross* nicht möglich ist (Abb. 7.33). Ebenso entfallen die dort vorgeschlagenen Verfeinerungen von *touch*: *meet* und *onBoundary*. Die Unterscheidung sah vor, dass für *onBoundary* ein Objekt mit niedriger Dimensionalität vollständig auf dem Rand eines höherdimensionalen Objekts liegen muss, während sich bei *meet* ausschließlich die Ränder der beiden beteiligten Objekte treffen. Zwar ist eine solche Unterscheidung wünschenswert, jedoch nicht mit den in Kapitel 8 vorgestellten Algorithmen realisierbar.

In der Wahl der Clustergruppen gibt es leichte Unterschiede zu den Definitionen von Schneider & Behr. Die Prädikate *coverBy* und *covers* werden nicht übernommen, da es für die hier betrachteten Anwendungsgebiete in der Regel unerheblich ist, ob ein Objekt mit einem anderen lediglich das Innere oder zusätzlich auch den Rand gemeinsam hat. Diese beiden Möglichkeiten werden daher unter *inside*

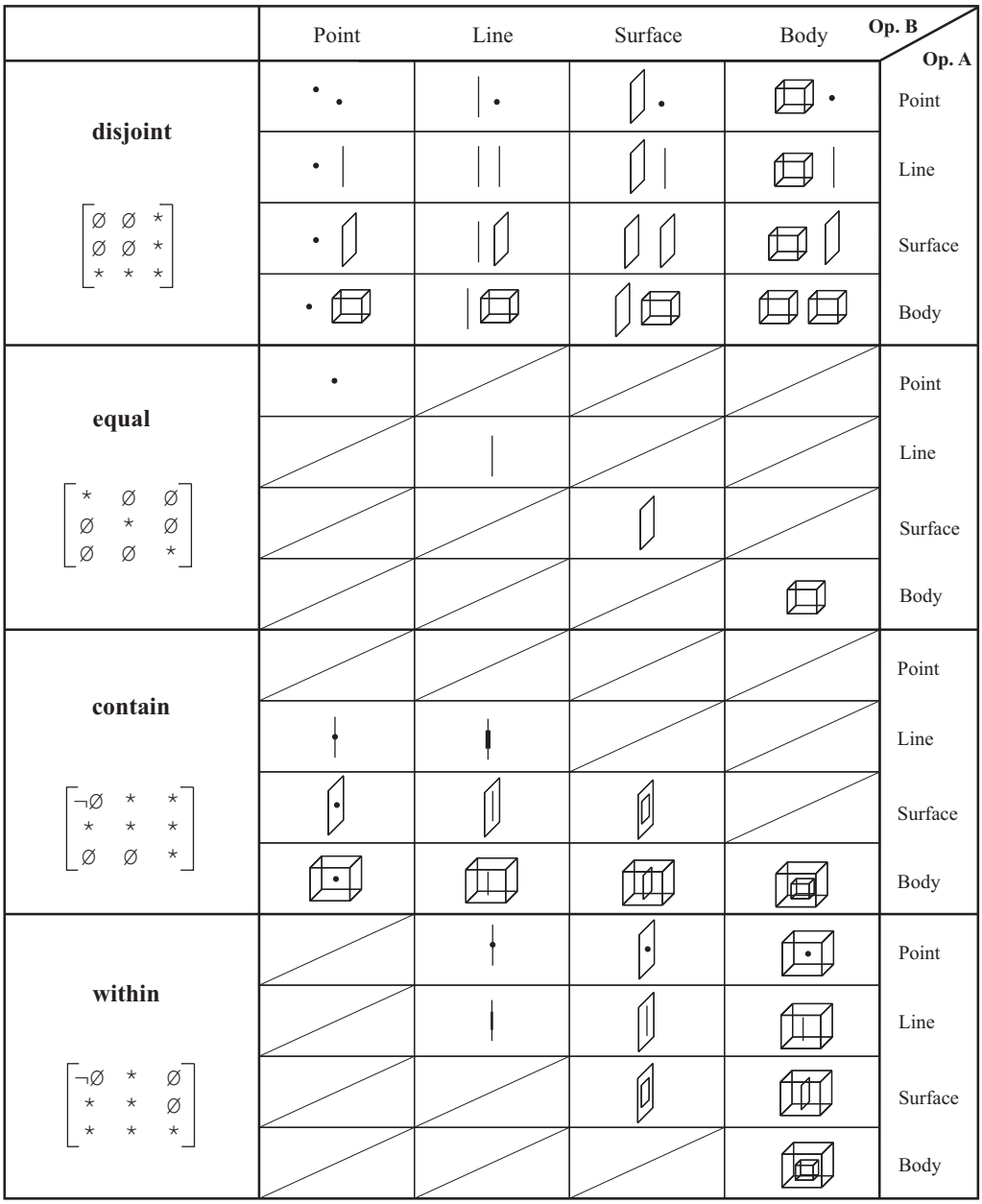


Abbildung 7.31: Die innerhalb der räumlichen Algebra spezifizierten topologischen Beziehungen (Teil 1). Die 9-Intersection-Matrix auf der linken Seite formalisiert die Semantik des entsprechenden Prädikats.

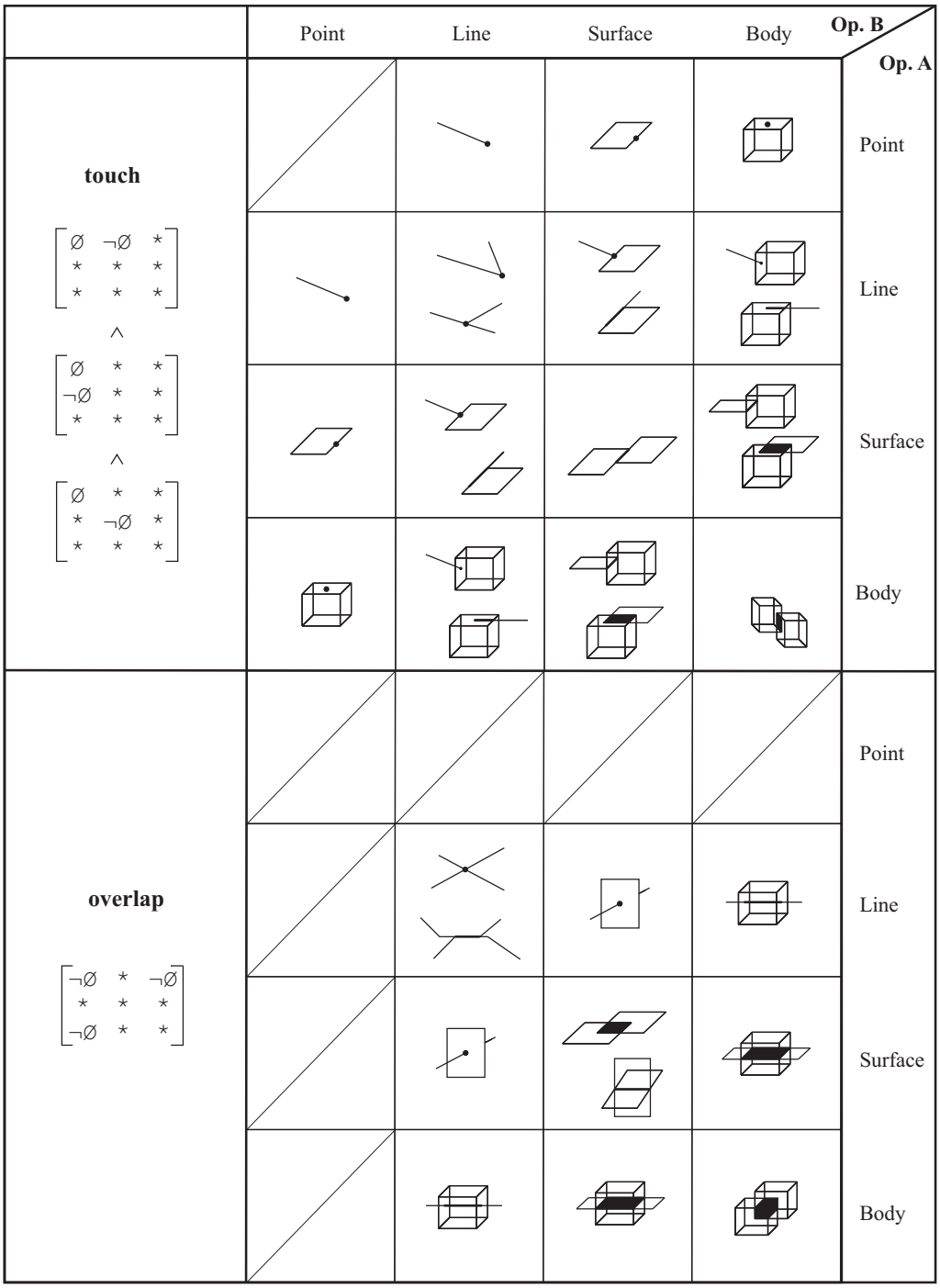


Abbildung 7.32: Die innerhalb der räumlichen Algebra spezifizierten topologischen Beziehungen (Teil 2). Die 9-Intersection-Matrix auf der linken Seite formalisiert die Semantik des entsprechenden Prädikats.

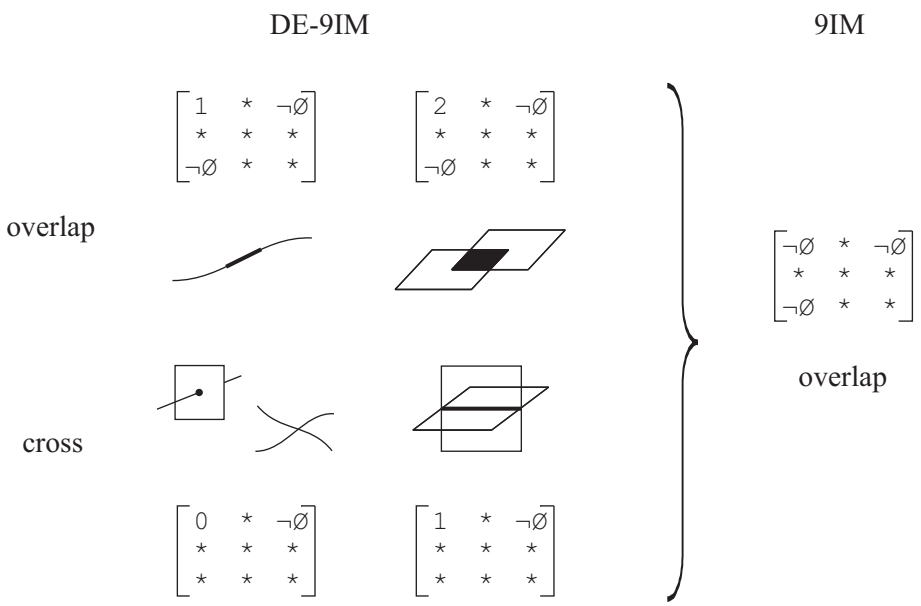


Abbildung 7.33: Eine Unterscheidung zwischen *overlap* und *cross* ist nur mit Hilfe der DE-9IM unter Verwendung des Dimensionsoperators möglich. Im hier vorgeschlagenen 9IM-basierten Modell werden daher beide Konstellationen unter dem Prädikat *overlap* subsumiert.

bzw. *contains* zusammengefasst. Darüber hinaus wird statt der in (Schneider & Behr, 2006) verwendeten Bezeichnung *meet* die Bezeichnung *touch* verwendet, um eine weitestgehende Übereinstimmung mit den im OGC-Standard verwendeten Namen zu erzielen.

Da Objekte vom Typ *Point* keinen Rand besitzen, werden in (Borrmann *et al.*, 2006) gesonderte Matrizen für Relationen aufgestellt, an denen *Point*-Objekte beteiligt sind. Dies ist immer dann notwendig, wenn für den allgemeinen Fall in der *Boundary*-Zeile bzw. -Spalte ein ¬∅-Wert eingetragen wird. Durch das hier vorgenommene *Clustering* der topologischen Beziehungen und den daraus resultierenden Platzhaltern in den 9-Intersection-Matrizen ist eine solche Fallunterscheidung im hier vorgestellten Schema nicht erforderlich.

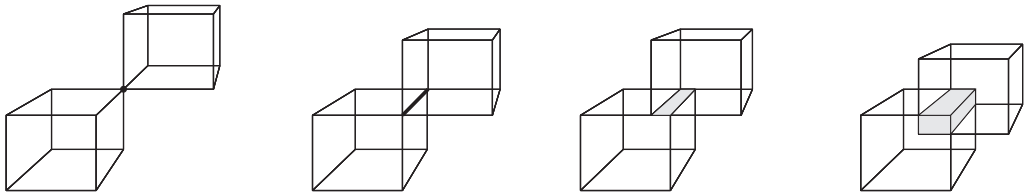


Abbildung 7.34: Der Schnitt zweier *Body*-Objekte kann in einem *Point*-, einem *Line*-, einem *Surface*- oder einem *Body*-Objekt resultieren.

7.4.5 Objekterzeugende und -modifizierende Operatoren

Objekterzeugende Operatoren generieren neue räumliche Objekte, die innerhalb einer Anfrage als temporäre Objekte verwendet werden können. Ein typisches Beispiel ist das Erzeugen einer Pufferzone um ein bestehendes Objekt und seine Verschneidung mit einem anderen Objekt zur Kollisionskontrolle.

Im Zentrum der Betrachtungen dieser Arbeit sollen Operatoren stehen, die als Selektionskriterium in einer deklarativen Anfrage Verwendung finden können. Operatoren, die räumliche Objekte generieren oder bestehende räumliche Objekte verändern, sind diesbezüglich von untergeordneter Bedeutung und werden daher zunächst von der Untersuchung ausgeklammert. Trotzdem soll hier ein kurzer Überblick über die dazu bekannten Forschungstätigkeiten gegeben werden.

Mit Hilfe der booleschen Operationen *Vereinigung*, *Schnitt* und *Differenz* lässt sich aus zwei bestehenden Objekten ein neues generieren. Ihre Semantik entspricht den entsprechenden mengentheoretischen Operationen auf den Punktmengen der Operanden.

Die Integration der booleschen Operationen in ein räumliches Typsystem stellt eine nicht-triviale Aufgabe dar. Die Ursache hierfür liegt darin, dass die Anwendung der booleschen Operationen auf zwei räumliche Objekte in Ergebnisobjekten ganz unterschiedlichen Typs resultieren kann. Der Schnitt zweier *Line*-Objekte kann beispielsweise ein *Line*- oder ein *Point*-Objekt sein und der Schnitt zweier *Body*-Objekte in einem *Point*-, einem *Line*- oder einem *Body*-Objekt resultieren (Abb. 7.34).

Die bekannten Arbeiten schlagen hierzu keine befriedigende Lösung vor. Die Mehrzahl empfiehlt, verschiedene Operationen anzubieten, die sich nach dem zu erwartenden Ergebnistyp unterscheiden (Güting, 1988; Svensson & Huang, 1991; Huang *et al.*, 1992; Güting & Schneider, 1995). Dies setzt jedoch voraus, dass der Anwender bereits weiß, in welcher Konstellation sich die zu verschneidenden Objekte befinden.

Andere vorgeschlagene objekterzeugende Operationen ermitteln einzelne Komponenten von räumlichen Objekten: Die in (Svensson & Huang, 1991; Huang *et al.*, 1992) beschriebene *boundary*-Operation erzeugt beispielsweise den Rand einer *Region* als *Line*-Objekt bzw. die Endpunkte einer *Line* als *Point*-Objekt.

Operationen, die ein bestehendes Objekt transformieren sind beispielsweise *extend* (Chan & Zhu, 1996), *rotate* (Svensson & Huang, 1991) und *translate* (Svensson & Huang, 1991). Die *extend*-Operation führt die Expansion eines räumlichen Objekts um einen gegebenen Abstand durch. Diese Operation ist auch als *buffer zoning* bekannt und wird von vielen GI-Systemen angeboten.

7.5 Zusammenfassung

Die Algebra umfasst neben den atomaren Datentypen *bool*, *real*, *int* und den dazugehörigen booleschen bzw. numerischen Operatoren die räumlichen Typen *Point*, *Line*, *Surface*, *Body* und den Supertyp *Spatial* sowie die in den Tabellen 7.1, 7.2 und 7.3 aufgeführten räumlichen Operatoren.

Metrische Operatoren	
diameter	SpatialObject → real
volume	Body → real
distance	SpatialObject × SpatialObject → real
maxdist	SpatialObject × SpatialObject → real
isCloser	SpatialObject × SpatialObject × real → bool
isFarther	SpatialObject × SpatialObject × real → bool

Tabelle 7.1: Die metrischen Operatoren.

Direktionale Operatoren	
eastOf_proj	SpatialObject × SpatialObject → bool
westOf_proj	SpatialObject × SpatialObject → bool
northOf_proj	SpatialObject × SpatialObject → bool
southOf_proj	SpatialObject × SpatialObject → bool
above_proj	SpatialObject × SpatialObject → bool
beneath_proj	SpatialObject × SpatialObject → bool
eastOf_proj_strict	SpatialObject × SpatialObject → bool
westOf_proj_strict	SpatialObject × SpatialObject → bool
northOf_proj_strict	SpatialObject × SpatialObject → bool
southOf_proj_strict	SpatialObject × SpatialObject → bool
above_proj_strict	SpatialObject × SpatialObject → bool
beneath_proj_strict	SpatialObject × SpatialObject → bool
eastOf_hs	SpatialObject × SpatialObject → bool
westOf_hs	SpatialObject × SpatialObject → bool

Fortgesetzt auf Folgeseite ...

northOf_hs	SpatialObject × SpatialObject → bool
southOf_hs	SpatialObject × SpatialObject → bool
above_hs	SpatialObject × SpatialObject → bool
beneath_hs	SpatialObject × SpatialObject → bool
eastOf_hs_strict	SpatialObject × SpatialObject → bool
westOf_hs_strict	SpatialObject × SpatialObject → bool
northOf_hs_strict	SpatialObject × SpatialObject → bool
southOf_hs_strict	SpatialObject × SpatialObject → bool
above_hs_strict	SpatialObject × SpatialObject → bool
beneath_hs_strict	SpatialObject × SpatialObject → bool

Tabelle 7.2: Die direktionalen Operatoren.

Topologische Operatoren	
whichTopoPredicate	SpatialObject × SpatialObject → int
disjoint	SpatialObject × SpatialObject → bool
equal	Point × Point → bool
	Line × Line → bool
	Surface × Surface → bool
	Body × Body → bool
contain	Line × Point → bool
	Line × Line → bool
	Surface × Point → bool
	Surface × Line → bool
	Surface × Surface → bool
	Body × Point → bool
	Body × Line → bool
	Body × Surface → bool
	Body × Body → bool
within	Point × Line → bool
	Point × Surface → bool
	Point × Body → bool
	Line × Line → bool
	Line × Surface → bool
	Line × Body → bool
	Surface × Surface → bool

Fortgesetzt auf Folgeseite ...

	Surface × Body → bool
	Body × Body → bool
touch	Point × Line → bool
	Point × Surface → bool
	Point × Body → bool
	Line × Line → bool
	Line × Surface → bool
	Line × Body → bool
	Surface × Line → bool
	Surface × Surface → bool
	Surface × Body → bool
	Line × Body → bool
	Surface × Body → bool
	Body × Body → bool
overlap	Line × Line → bool
	Line × Surface → bool
	Line × Body → bool
	Surface × Line → bool
	Surface × Surface → bool
	Surface × Body → bool
	Body × Line → bool
	Body × Surface → bool
	Body × Body → bool

Tabelle 7.3: Die topologischen Operatoren.

Kapitel 8

Oktalbaumbasierte Algorithmen für die Implementierung räumlicher Operatoren

Im Folgenden soll eine Implementierungsmöglichkeit für die im vorangegangen Kapitel definierte Funktionalität der räumlichen Operatoren vorgestellt werden. Diese basiert auf einer Geometriebeschreibung nach dem Normzellenaufzählungsschema, wobei die *Oktalbaumcodierung* zur Vermeidung überproportionalen Speicherbedarfs Anwendung findet. Darüber hinaus bietet der rekursive Charakter der hierarchisch-raumpartitionierenden Datenstruktur Oktalbaum die Möglichkeit der sukzessiven Genauigkeitssteigerung bei der Verarbeitung räumlicher Operatoren. Dies erlaubt dem Anwender, zwischen benötigter Genauigkeit und erforderlicher Bearbeitungszeit für eine räumliche Anfrage abzuwägen. Alternative Implementierungsvarianten auf Basis der klassischen *Computational Geometry* (de Berg *et al.*, 2000) oder der simplizialen Zerlegung (Breunig, 1995) sind möglich, sollen jedoch hier nicht näher betrachtet werden.

8.1 Zellzerlegung

Die Zellzerlegung gehört zur Gruppe der raumpartitionierenden Methoden der Geometriepräsentation. Bei dieser Gruppe wird ein Körper bzw. ein geometrisches Objekt in eine Ansammlung von sich nicht schneidenden Primitiven zerteilt, also Körpern, deren Struktur einfacher als die des Ausgangsobjekts ist. Dabei können sich die verwendeten Primitive nach Form und Größe unterscheiden.

Die Zellzerlegung (siehe auch Abschnitt 7.2.3) ist eine der generellsten Ausprägungen der Raumpartitionierung. Jedes Zellzerlegungssystem definiert einen Satz von primitiven Zellen, die in der Regel parametrisiert sind und ggf. eine ge-

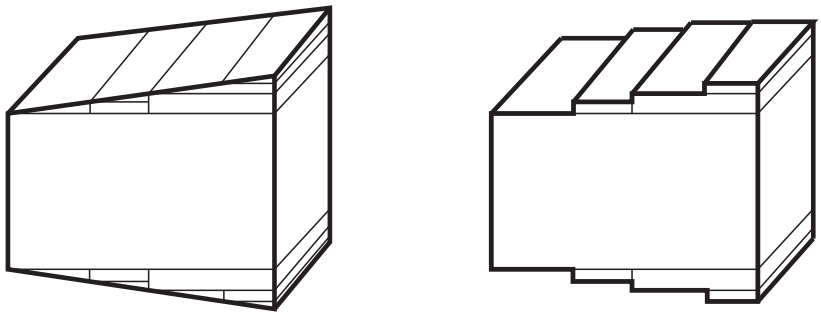


Abbildung 8.1: Die in (Weiss, 2005) zur Implementierung topologischer Analysefunktionalität gewählte Zellzerlegung in achsenparalle Quader.

krümmte Berandung aufweisen. Die Zellzerlegung kommt zum Beispiel bei der Netzgenerierung für Finite-Elemente-Analysen zum Einsatz.

In (Weiß, 2005) wird zur topologischen Analyse von Gebäudemodellen ein Ansatz verfolgt, der auf der Beschreibung der Bauteilgeometrie mit Hilfe einer Zellzerlegungstechnik basiert. Als Primitive dienen hierbei parametrisierte achsenparallele Quader (Abb. 8.1). Der Unterschied zu dem in dieser Arbeit verfolgten Ansatz liegt vor allem in der resultierenden flachen Datenstruktur, die verhindert, dass die Vorteile einer hierarchischen Datenstruktur (hier Oktalbaum) für eine sukzessive Genauigkeitssteigerung genutzt werden können.

Das Normzellenaufzählungsschema (im Englischen als *spatial-occupancy enumeration* bezeichnet) ist ein spezieller Fall der Zellzerlegung. Hierbei wird der Körper in Zellen mit identischer Form und Größe zerteilt, die in einem uniformen Gitter angeordnet sind. Der gebräuchlichste Zelltyp ist dabei der Würfel. In Analogie zum *Pixel* (kurz für *picture element*) in 2D wird eine einzelne würfelförmige Zelle als *Voxel* (kurz für *volume element*) bezeichnet.

Bei der Repräsentation eines Objekts mit dem Normzellenaufzählungsschema wird festgehalten, welche Zellen belegt sind und welche nicht. In der Regel kann die Ursprungsgeometrie mit dieser Technik nur angenähert werden. Beispielsweise sind mit würfelförmigen Primitiven weder gekrümmte, noch schräge Begrenzungsflächen darstellbar. Um die Genauigkeit der Näherung zu erhöhen, muss die Zellgröße entsprechend verkleinert werden. Dies führt allerdings schnell zu einem enormen Speicherplatzbedarf, da im Dreidimensionalen die Zahl der für die Beschreibung des Körpers notwendigen Zellen mit $(\frac{1}{d})^3$ steigt, wobei d die Länge der Würfelkante und damit die erreichbare Genauigkeit beschreibt. Um dies zu umgehen, wurden hierarchische Speicherstrukturen entwickelt, die im Folgenden vorgestellt werden.

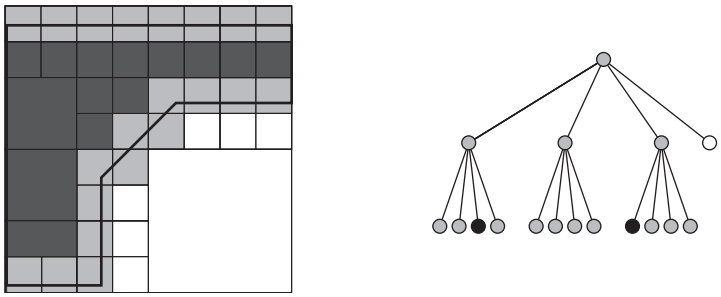


Abbildung 8.2: Quadtree-Repräsentation einer Region in 2D. Der Baum auf der rechten Seite spiegelt die Auflösung bis zur Ebene 2 wider.

8.2 Oktalbaumcodierung

8.2.1 Definition

Der Oktalbaum ist eine spezielle Ausprägung des *Spacetree* für den dreidimensionalen Raum. Beim *Spacetree* handelt es sich um eine raumpartitionierende, hierarchische Datenstruktur, bei der jedes Element (im Weiteren wird ein Element als Zelle bezeichnet) einen Vorgänger (Elternzelle) und entweder *0* oder *2d* Nachfolger (Kindzellen) hat, wobei d für die Anzahl der Raumdimensionen steht (Frank, 2000).

Für $d = 2$ spricht man von einem *Quadtree* (Klinger, 1971), für $d = 3$ von einem *Octree* (Hunter, 1978; Jackins & Tanimoto, 1980; Meagher, 1982). Das Verhältnis der Kantenlänge einer Kindzelle zur Elternzelle ist dabei stets 1:2. Die Vereinigung aller Kindzellen ergibt die Elternzelle. Betrachtet man nun eine Zelle an einer beliebigen Position im Baum isoliert mit ihrer kompletten nachfolgenden Struktur, so erhält man wiederum einen *Spacetree*. Diese Eigenschaft erlaubt die Verwendung rekursiver Algorithmen.

Kern der *Spacetree*-Codierung ist, dass diejenigen Zellen keine Kinder besitzen, die vollständig innerhalb oder vollständig außerhalb des zu repräsentierenden Objekts liegen. Auf diese Weise wird gegenüber der Beschreibung mit dem Normzellenaufzählungsschema Speicherplatz gespart. Je nach Lage werden den Zellen bestimmte Färbungen zugewiesen: *schwarz* für innenliegende, *weiß* für außenliegende und *grau* für teilweise außen und teilweise innenliegende Zellen (Abb. 8.2).

Für eine äquidistante Diskretisierung mit n Zellen je Raumrichtung besitzt das Normzellenaufzählungsschema einen Speicheraufwand von $\mathcal{O}(n^3)$ Zellen zur Darstellung der Geometrie (Foley *et al.*, 1996; Bungartz *et al.*, 2002). Wie in (Frank, 2000) nachgewiesen, lässt sich bei gleicher Qualität der Diskretisierung mit Hilfe der Oktalbaumcodierung ein um eine Dimension geringerer Speicheraufwand von $\mathcal{O}(n^2)$ erreichen. In Einzelfällen kann der Aufwand aufgrund unvorteilhafter – beispielsweise fraktaler – Geometrien die Abschätzung $\mathcal{O}(n^2)$ zwar deutlich überschreiten, jedoch ist er stets durch $\mathcal{O}(n^3)$ Zellen limitiert, was einem vollbesetzten Baum und damit exakt dem Normzellenaufzählungsschema entspricht. Hunter und Meagher kommen zu einem ähnlichen Schluss: Sie argumentieren, dass die

Anzahl an Knoten im Oktalbaum immer proportional zur Größe der Oberfläche des Ausgangsobjekts ist, da eine Verfeinerung nur dort auftritt, wo der Rand des Objekts die entsprechenden Zellen schneidet (Hunter, 1978; Meagher, 1982).

Die Wurzeln der Verwendung von Spacetrees liegen im Bereich der Computergraphik, wo die Quadtree-Codierung vor allem für eine verlustfreie Kompression von digitalen Bildern eingesetzt wird. Auf diese Weise wurden sie auch schon früh in Geographischen Informationssystemen verwendet (Samet, 1989).

Oktalbäume wurden bereits in unterschiedlichsten Bereichen des Computational Engingeering gewinnbringend eingesetzt, u.a. zur Kompression von Simulationsdaten für die Übertragung vom Simulationsrechner zu einer Visualierungs-Workstation (Kühner, 2003; Kühner *et al.*, 2004), zur Erkennung von Kollisionen von Bauteilen während des Planungsprozesses (Mundani, 2005) und für die Bildung von Teilsteifigkeitsmatrizen für eine parallelisierte FE-Berechnung nach dem *Nested Dissection*-Verfahren (Niggl, 2007).

Daneben kommen Oktalbäume häufig bei der Gittergenerierung für numerische Berechnungen zum Einsatz. So wurden Oktalbäume für die Erzeugung von Hexaederelementen für Strukturanalysen (Shephard & Georges, 1991) und für Strömungssimulationen (Tchon *et al.*, 1997) verwendet. Besonders effizienzsteigernd können sie bei der für Strömungssimulationen nach der Lattice-Boltzmann-Methode notwendigen Generierung eines uniformen kartesischen Gitters eingesetzt werden (Jaksch, 2001; Wenisch & Wenisch, 2004). Inzwischen gibt es darüber hinaus erste erfolgversprechende Ansätze dafür, auch die LB-Simulation selbst auf hierarchischen Gittern durchzuführen (Crouse, 2003; Geller *et al.*, 2006).

Des Weiteren eignet sich die raumpartitionierende Struktur von Oktalbäumen ausgezeichnet zur einfachen und schnellen Kollisionsdetektion sowie zur effizienten Bestimmung benachbarter Elemente des zugrunde liegenden Modells. Beispielsweise wird in (Mundani, 2005) eine oktalbaumbasierte Erkennung geometrischer Konflikte zur Sicherung der Konsistenz einer gemeinsam benutzten Datenbasis in einer verteilt kooperativen Arbeitsumgebung vorgestellt. In (Zachmann, 2000) werden u.a. oktalbaumbasierte Aspekte bezüglich der Kollision von Bauteilen im Rahmen des *Digital MockUp* betrachtet, in (Jung & Gupta, 1996) werden mit Hilfe von Oktalbäumen mögliche Kollisionen detektiert, die bei der Pfadplanung zur Bewegung von Industrierobotern auftreten können. Im Bereich von Computerspielen steht insbesondere eine hohe Geschwindigkeit bei der Kollisionserkennung im Vordergrund, die oftmals unter Verwendung einer Oktalbaumstruktur realisiert wird (Ericson, 2004).

Wie in diesem Abschnitt gezeigt wird, eignen sich oktalbaumcodierte Rastergeometrien dank der ihnen inhärenten Verfeinerungshierarchie hervorragend für eine Umsetzung räumlicher Operationen auch bei Objekten mit komplexen Geometrien. Die Besonderheit bei dem dabei verfolgten Ansatz ist, dass die Genauigkeit der durch das System erzielbaren Antwort auf eine Anfrage nicht a priori determiniert ist, sondern zur Laufzeit durch den Nutzer entsprechend seinen Anforderungen festgelegt werden kann. Dies verhindert unvertretbare Wartezeiten bei der Verar-

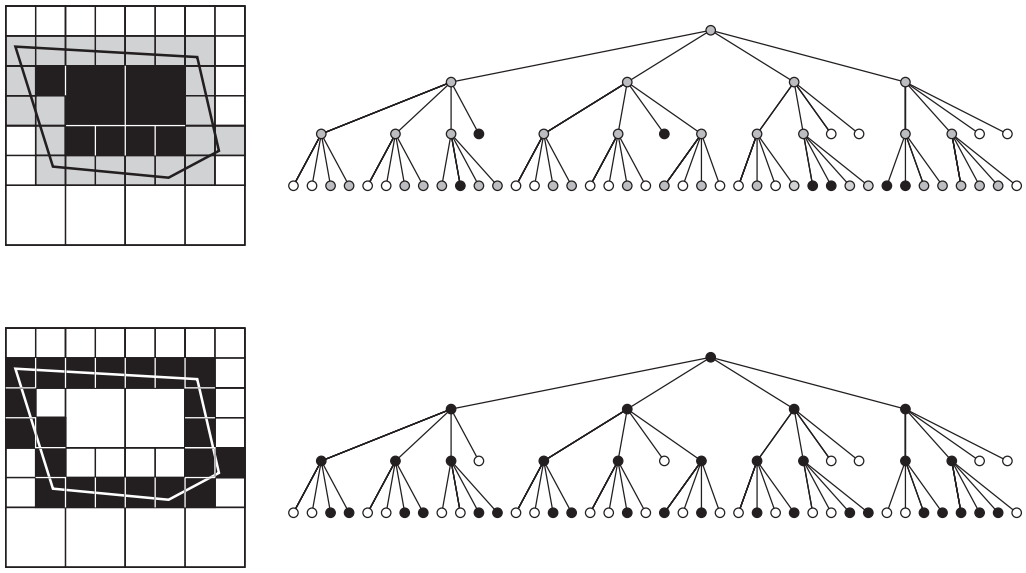


Abbildung 8.3: Zwei- und dreifarbiger Quadtree zum Vergleich. Beim zweifarbigen wird lediglich der Rand aufgelöst, beim dreifarbigen auch das Innere des Objekts.

beitung einer Anfrage und entspricht einer ingenieurgemäßen Herangehensweise, zu der der sinnvolle Umgang mit Fehlern bzw. Ungenauigkeiten bei gleichzeitiger Begrenzung des Aufwands gehört.

8.2.2 Generierung

Für den Aufbau des *Spacetree* gibt es verschiedene Vorgehensweisen, abhängig davon, ob bei einem zu diskretisierenden Element, das die gleiche Dimensionalität wie der umgebende Raum hat (2D: Region, 3D: Körper), lediglich der Rand aufgelöst (Jaksch, 2001; Wenisch & Wenisch, 2004) oder auch das Innere entsprechende markiert werden soll (Tamminen & Samet, 1984; Mundani, 2005). Gemein ist allen Ansätzen, dass beginnend beim Wurzelelement nur diejenigen Kindzellen weiter verfeinert werden, die weder vollständig innerhalb, noch vollständig außerhalb des zu diskretisierenden Gebildes liegen, also seine Berandung schneiden. Wie Abb. 8.3 zeigt, werden im Falle eines zweifarbigen Baums die Randzellen als belegt (=*schwarz*) markiert und alle anderen als leer (=*weiß*). Beim dreifarbigen Baum werden hingegen innen liegende Zellen als *schwarz*, außen liegende als *weiß* und auf dem Rand liegende als *grau* markiert.

Da bei dimensionsreduzierten Objekten immer auch Zellen auftreten, bei denen zwar Inneres und Äußeres aber nicht der Rand zusammen fallen, ist im Kontext dieser Arbeit die Einführung der vierten Farbe *schwarz-weiß* für Zellen des Oktalbaums notwendig. Ein *schwarz-weißer* Knoten ist ebenso wie der *graue* Knoten kein Blattknoten und hat entsprechend immer acht Kinder. In vierfarbigen Oktalbäumen kann ein *grauer* Knoten neben *schwarzen* und *weißen* auch *schwarz-weiße* Kinder haben. Ein *schwarz-weißer* Knoten kann hingegen keinen grauen Knoten als Kind besitzen. Derartige vierfarbige Oktalbäume werden vor

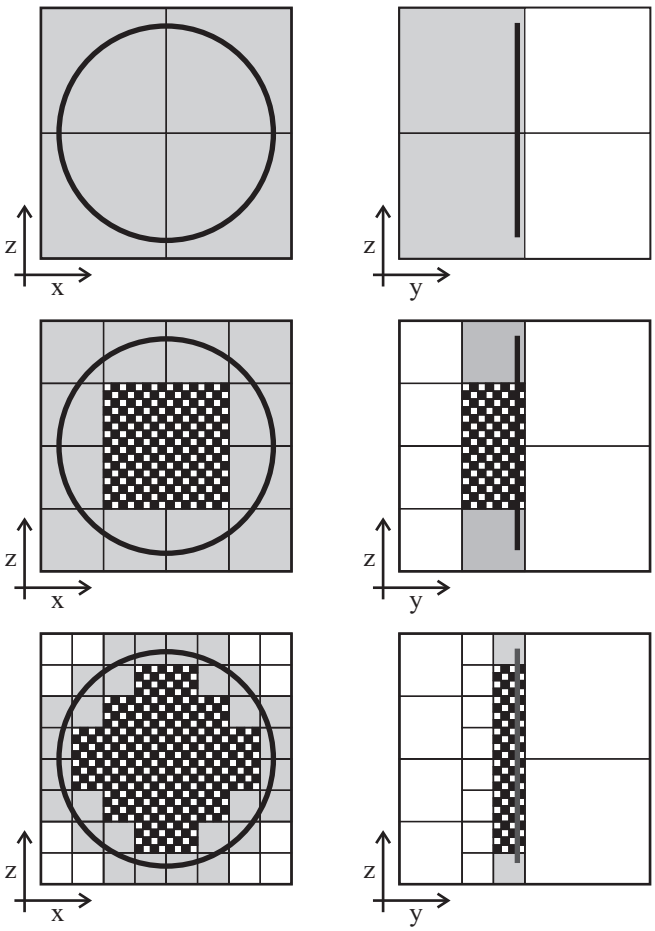


Abbildung 8.4: Oktalbaumcodierung eines zweidimensionalen Objekts (Scheibe) im dreidimensionalen Raum. Gezeigt sind zwei Schnitte durch den Oktalbaum jeweils auf den Ebene 1 bis 3. Die Einführung der vierten Farbe *schwarz-weiß* ist notwendig, um Zellen abzubilden, in denen zwar Inneres und Äußeres des diskretisierten Objekts, nicht aber dessen Rand auftritt.

allem zur Implementierung topologischer Operatoren benötigt, die in Abschnitt 8.5 erläutert wird.

Die vorgestellten Verfahren gehen von einer geradlinigen Berandung des zu diskretisierenden Körpers aus. Krummlinig, d.h. durch Freiformflächen berandete Körper können durch eine Tessellierung bzw. Triangulierung (Piegl & Richard, 1995; Chen & Bishop, 1997) der Oberfläche approximiert werden. Das gleiche gilt für flächenförmige Objekte, die ganz oder teilweise aus Freiformflächen bestehen. Linienförmige Objekte, die sich ganz oder teilweise aus Freiformkurven zusammensetzen, können entsprechend durch eine Polygonalisierung angenähert werden.

Verfahren von Mundani zur Oktalbaumcodierung von Körpern

In (Mundani, 2005) wird eine äußerst effiziente Methode zur Erzeugung von dreifarbigen Oktalbäumen für konvexe Körper vorgeschlagen, deren Oberfläche facet-

tiert vorliegt. Sie basiert auf der Generierung von Oktalbäumen für Halbräume, die durch die einzelnen Facetten des Ausgangsobjekts beschriebenen werden, und anschließender Verschneidung dieser Oktalbäume.

Klassische Ansätze zur Überführung von *BRep*-modellierten Körpern in ein oktalbaumcodiertes Normzellenaufzählungsschema (Ayala *et al.*, 1985; Carlbom *et al.*, 1985; Kela, 1989; Anand & Knott, 1991) beruhen zum großen Teil auf Algorithmen, die ausführliche Berechnungen zur Bestimmung des Schnitts der Facetten mit der Oktalbaumzelle und eine große Zahl an Punktklassifikationen beinhalten. Zwar werden erstere bei dem in (Krishnan *et al.*, 1996) vorgeschlagenen Algorithmus vermieden, dennoch ist auch hier erheblicher Aufwand für die Klassifikationen von Zellen hinsichtlich *Innen* bzw. *Außen* notwendig.

Bei dem in (Mahler, 2003) vorgeschlagenen Verfahren wird im Gegensatz dazu eine *Bottom-Up*-Strategie verfolgt, bei der zunächst auf dem feinsten Auflösungsniveau Zellen für die Knoten der Ausgangsgeometrie gesetzt werden. Daraufhin werden die Kanten zwischen den Knoten mit Hilfe des Bresenham-Algorithmus (Bresenham, 1965) diskretisiert und dann die zwischen den Kanten liegenden Flächen mittels eines *Scan-line*-Algorithmus aufgefüllt. Anschließend werden mit Hilfe von Füllalgorithmen – zum Beispiel (Smith, 1979; Pavlidis, 1981) – die im Inneren liegenden Zellen markiert. Die so erzeugte Beschreibung des Körpers nach dem Normzellen-Aufzählungsschema wird schließlich in einem Kompaktifizierungsschritt in einen Oktalbaum überführt. Für die hier betrachtete Anwendungsklasse ist eine *Bottom-Up*-Strategie jedoch als weniger geeignet einzustufen, da für die Abarbeitung der räumlichen Operatoren die Erzeugung eines vollständigen Oktalbaums in der Regel nicht notwendig ist.

Unter diesem Gesichtspunkt deutlich besser geeignet ist das von Mundani entwickelte *Top-Down*-Verfahren, das auf der Verschneidung von Halbräumen beruht. Dabei wird jede Facette des oberflächenorientierten geometrischen Modells als Teil einer unendlichen Ebene betrachtet, die den gesamten Raum des $\mathbb{R}^3$ in zwei Halbräume unterteilt. Entsprechend der Orientierung der Flächennormale der betrachteten Facette lassen sich die beiden Halbräume mit den Kennzeichnungen *Innen* bzw. *Außen* versehen. Wird anschließend der geometrische Durchschnitt der auf diese Weise erzeugten Halbräume gebildet, so entspricht das Resultat einer volumenorientierten Darstellung der ursprünglichen oberflächenorientierten Geometrie.

Ein wesentliches Effizienzproblem bei der Oktalbaumgenerierung liegt in den aufwändigen Entscheidungskriterien zur Verfeinerung einer Zelle, da diese in der Regel durch eine Berechnung des Schnitts zwischen der Oberfläche des Ausgangsobjekts und der betreffenden Zelle im Oktalbaum gelöst wird. Zwar lassen sich derartige Schnittberechnungen mathematisch effizient beschreiben, eine entsprechende algorithmische Umsetzung benötigt jedoch mehrere Gleitkomma-Operationen. Effiziente Algorithmen müssen aufwändige Schnittberechnungen weitestgehend vermeiden und einfache, billige Entscheidungskriterien formulieren.

Mundani wählt aus diesem Grund einen auf Koordinatentransformationen beruhenden Ansatz, der im Folgenden erläutert werden soll. Schreibt man die Gleichung der Ebene, in der eine einzelnen Facette liegt, in der Form

$$E:\ a_0 + a_1 \cdot x_1 + a_2 \cdot x_2 + a_3 \cdot x_3 = 0 \quad \text{mit} \quad \sum_{i=1}^{3} |a_i| = 1 \quad ,$$

dann lässt sich ein Schnitt der Ebene mit der betreffenden Zelle des Oktalbaums allein durch den Parameter a_0 bestimmen. Durch die Normalisierung von a_1, a_2 und a_3 wird das Koordinatensystem so skaliert und verschoben, dass die betrachtete Oktalbaumzelle nun dem Einheitswürfel $W := [-1;1]^3$ entspricht. Während der Parameter a_0 dabei den Abstand der Ebene vom Koordinatenursprung beschreibt, der im Mittelpunkt des Würfels liegt, bestimmen die Parameter a_1 bis a_3 die Flächennormale $\vec{n}$ und damit die Orientierung und Lage der Ebene im Raum. Für den Fall, dass $|a_0| = 1$ gilt, schneidet die Ebene entweder eine Würfelecke, eine Würfelkante oder fällt mit einer der Würfelseiten zusammen.

Die bei einer Verfeinerung entstandenen Zellen müssen ihrerseits auf einen Schnitt mit der Ebene hin überprüft werden. Um analog dem obigen Schema zu verfahren, bedarf es der Darstellung der Ebene in einem neuen Koordinatensystem, dessen Ursprung im jeweiligen Mittelpunkt der acht neuen Zellen liegt. Bei der dafür notwendigen Koordinatentransformation, ergeben sich außer beim Parameter a_0, der den Abstand der Ebene zum Ursprung beschreibt, keine weiteren Änderungen, da die Flächennormale $\vec{n}$ der Ebene gegenüber dieser Transformation invariant ist und damit a_1, a_2 und a_3 konstant bleiben. Die Koordinatentransformation ergibt sich aus einer Skalierung mit dem Faktor 0,5 und einer Translation um den Faktor ζ:

$$\begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} \mapsto \begin{pmatrix} \gamma_1 \\ \gamma_2 \\ \gamma_3 \end{pmatrix} = \begin{pmatrix} 0,5 \cdot x_1 + \zeta_1 \\ 0,5 \cdot x_2 + \zeta_2 \\ 0,5 \cdot x_3 + \zeta_3 \end{pmatrix} \quad .$$

Damit ergibt sich die Darstellung der Ebene bezüglich des neuen Koordinatensystems zu

$$\widetilde{E}:\ \widetilde{a_0} + a_1 \cdot x_1 + a_2 \cdot x_2 + a_3 \cdot x_3 = 0, \quad \sum_{i=1}^{3} |a_i| = 1$$

mit

$$\widetilde{a_0} = 2 \cdot a_0 \cdot \zeta_1 + 2 \cdot a_1 \cdot \zeta_1 + 2 \cdot a_2 \cdot \zeta_2 + 2 \cdot a_3 \cdot \zeta_3 \ .$$

Da die Mittelpunkte der Kindzellen gegenüber dem Mittelpunkt der Elternzelle jeweils um 0,5 Längeneinheiten verschoben sind und somit $\zeta_{1,2,3} \in \{-0,5; 0,5\}$ gilt, lässt sich die letzte Gleichung auch in der Form

$$\widetilde{a_0} = 2 \cdot a_o \pm a_1 \pm a_2 \pm a_3$$

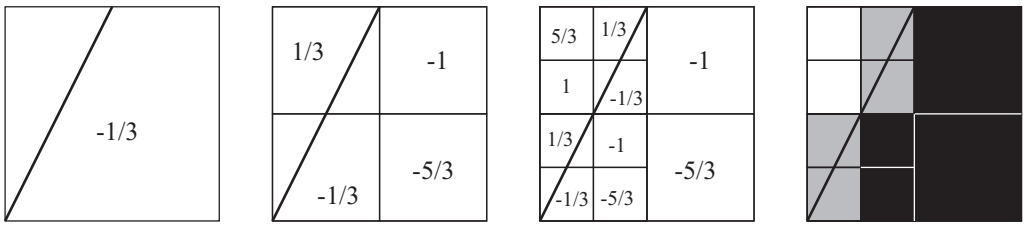


Abbildung 8.5: Quadtree-Generierung nach der von Mundani vorgeschlagenen Methode am Beispiel der Geraden $-1/3 - 2/3 \cdot x_1 + 1/3 \cdot x_2 = 0$. Dargestellt sind die für die jeweiligen Zellen berechneten Werte a_0 und die sich daraus ergebende Belegung auf Ebene 2. Eine Verfeinerung wird nur durchgeführt, wenn a_0 im Intervall $[-1, 1]$ liegt. Beispiel entnommen aus (Mundani, 2005).

schreiben. Das jeweilige Vorzeichen von a_1, a_2 und a_3 hängt hierbei von der gerade betrachteten Zelle ab. Legt man ein Rechtssystem zugrunde, so ergibt sich beispielsweise für die Zelle im rechten, oberen, hinteren Oktanten $+a_1$, $+a_2$ und $-a_3$. Die Vorzeichen für die restlichen sieben Oktanten ergeben sich analog. Daraus ergibt sich der entscheidende Vorteil, dass bei Verfeinerung einer Zelle lediglich der entsprechende Parameter a_0 für alle neuen Zellen berechnet werden muss, um über weitere Verfeinerungen zu entscheiden. Abb. 8.5 illustriert die Vorgehensweise anhand eines Beispiels.

Da die Flächennormale der betrachteten Ebene gegenüber einer Koordinatentransformation invariant ist, lassen sich alle acht möglichen Fälle des Ausdrucks $\pm a_1 \pm a_2 \pm a_3$ im Voraus bestimmen und in den Variablen t^1 bis t^8 speichern. Die Berechnung von $\widetilde{a_0}$ für den Oktanten i reduziert sich dadurch auf eine Gleitkommamultiplikation und eine Gleitkommaaddition. Auf diese Weise wird die Forderung nach einem einfachen und bezüglich des Rechenaufwands billigen Entscheidungskriterium erfüllt.

Die Klassifikation der betrachteten Zelle in *Innen*, *Außen* oder *Rand* kann ebenfalls auf Grundlage des Werts von a_0 erfolgen, da dank der Normierung von a_1, a_2 und a_3 die Richtung der Flächennormalen durch das Vorzeichen von a_0 ausgedrückt wird. Mundani wählt auf dieser Grundlage folgende Enscheidungskriterien:

$$\begin{aligned} a_0 \leq -1 &\Rightarrow \text{Innen} , \\ -1 < a_0 < 1 &\Rightarrow \text{Rand} , \\ 1 \leq a_0 &\Rightarrow \text{Außen} . \end{aligned}$$

Da im Kontext dieser Arbeit die Beschreibung des Randes durch *graue* Zellen im Oktalbaum insbesondere für die Implementierung topologischer Operatoren (Abschnitt 8.5) von herausgehobener Bedeutung ist, wird hier das Entscheidungskriterium leicht abgewandelt:

$$\begin{aligned} a_0 < -1 &\Rightarrow \text{Innen} , \\ -1 \leq a_0 \leq 1 &\Rightarrow \text{Rand} , \\ 1 < a_0 &\Rightarrow \text{Außen} . \end{aligned}$$

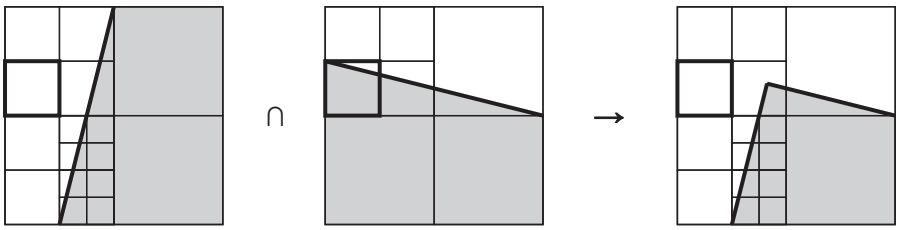


Abbildung 8.6: Verschneidung zweier Oktalbäume während ihres Aufbaus. Durch parallele Abarbeitung der Bäume und direkte Auswertung des Verschneidungsausdrucks können unnötige Verfeinerungen vermieden werden. Als Ergebnis der Verschneidung der umrahmten Zellen in den Eingangsbäumen ergibt sich die Farbbelegung *weiß* ∪ *grau* ⇒ *weiß*. Eine Verfeinerung des zweiten Eingangsbaums ist an dieser Stelle daher nicht notwendig. Abbildung entnommen aus (Mundani, 2005).

Durchschnitt	schwarz	weiß	grau
schwarz	schwarz	weiß	grau
weiß	weiß	weiß	weiß
grau	grau	weiß	grau

Tabelle 8.1: Boolesche Operation *Durchschnitt* zweier Oktalbaumzellen.

Dadurch werden Zellen, die von der Ebene genau auf einer Seitenfläche, einer Kante oder einer Ecke geschnitten werden, als *grau* markiert. Auf diese Weise wird sichergestellt, dass auch Körper, deren Berandung teilweise oder ganz mit den Seitenflächen einer oder mehrerer Zellen zusammenfällt, nach der Überführung in einen Oktalbaum von einer lückenlosen Schicht *grauer* Zellen umgeben sind. Die Konsequenz dieser Vorgehensweise ist allerdings, dass im Fall der Koinzidenz der betrachteten Ebene mit einer Seitenfläche *zwei* benachbarte Zellen, im Fall des Schnitts mit einer Kante *vier* und im Fall des Schnitts mit einem Eckpunkt *acht* benachbarte Zellen als Randzellen (*grau*) markiert und damit potentiell mehr Zellen für die nächsten Ebene des Oktalbaums verfeinert werden müssen.

Die notwendige Verschneidung der auf diese Weise codierten Halbräume wird durch Anwendung der booleschen Operation *Durchschnitt* auf die Oktalbäume realisiert. Die Definition dieser Operation kann Tabelle 8.1 entnommen werden. Zur Steigerung der Effizienz wird von Mundani vorgeschlagen, die Verschneidung bereits während des Erzeugungsprozesses durchzuführen. Auf diese Weise wird überflüssiger Aufwand – bei der Generierung der einzelnen Oktalbäume werden sonst in der Regel sehr viele Verfeinerungen vorgenommen, die im resultierenden Oktalbaum nicht auftauchen – vermieden. Das resultierende Verfahren beruht auf einer parallelen Abarbeitung der beiden Oktalbäume, wobei stets Zellen mit gleicher Positionierung im Raum betrachtet werden (Abb. 8.6).

Die Behandlung *nicht-konvexer* Körper basiert auf einem mathematischen Ausdruck, der den betrachteten Körper als Verknüpfung von Halbräumen mittels

Vereinigung	schwarz	weiß	grau
schwarz	schwarz	schwarz	schwarz
weiß	schwarz	weiß	grau
grau	schwarz	grau	grau

Tabelle 8.2: Boolesche Operation *Vereinigung* zweier Oktalbaumzellen.

Differenz	schwarz	weiß	grau
schwarz	weiß	schwarz	grau
weiß	weiß	weiß	weiß
grau	weiß	grau	grau

Tabelle 8.3: Boolesche Operation *Differenz* zweier Oktalbaumzellen.

boolescher Operationen beschreibt. Im Unterschied zu konvexen Körpern ist neben der Anwendung des booleschen Operators *Durchschnitt* auch die Anwendung von *Vereinigung* und *Differenz* notwendig, wobei die Reihenfolge der Anwendung eine wesentliche Rolle spielt. Zum Aufstellen des booleschen Ausdrucks wird zunächst die konvexe Hülle aller beteiligten Flächenstücke berechnet, alle auf der konvexen Hülle liegenden Flächenstücke markiert und diese in der Menge K zusammengefasst. Die nicht markierten Flächen f_i werden in disjunkten Teilmengen C_k zusammengefasst mit

$$C_k := \{f_i \mid \exists f_i \in C_k : f_i \lhd \rhd^* f_j\} ,$$

wobei $\lhd\rhd^*$ bedeutet, dass die Flächenstücke f_i und f_j jeweils eine Kante teilen oder über andere Flächen $f_t \in C_k$ in Verbindung stehen. Mit Hilfe von K und C_k lässt sich schließlich der folgende boolesche Ausdruck zur Beschreibung des Körpers ableiten:

$$K \setminus (C_1 \cup C_2 \cup \ldots \cup C_m) .$$

Diese Schritte – Berechnung der konvexen Hülle, Bildung von K und C_k – werden rekursiv solange wiederholt, bis sich keine neuen Teilmengen C_k mehr bilden lassen. Dies liefert letztlich die Beschreibung des Körpers als boolesche Verknüpfung der Form

$$F_B := K_0 \setminus \bigcup_{i=1}^{r-1} \left(K_r \setminus \bigcup_{j=r-1}^{s-1} \left(K_s \setminus \bigcup_{k=s+1}^{t-1} (K_t \setminus \ldots) \right) \right) ,$$

wobei K_i durch den Durchschnitt der zu den Flächenstücken zugehörigen Halbräume gewonnen wird. Das Beispiel in Abb. 8.7 illustriert diese Vorgehensweise.

Die Auswertung von F_B wird wie bei konvexen Objekten der Oktalbaumgenerierung vorangestellt, um unnötige Verfeinerungen zu vermeiden. Wie bereits be-

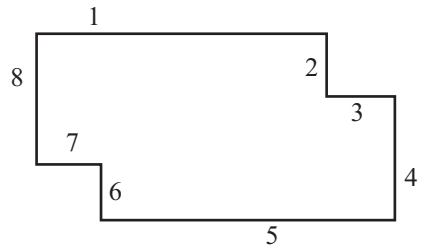


Abbildung 8.7: Beispiel für eine konvexe Zerlegung auf Basis der booleschen Verknüpfung von Halbräumen. Es ergibt sich der Term $F_B = K_0 \setminus (K_1 \cup K_2)$ mit $K_0 = (H_1 \cap H_4 \cap H_5 \cap H_8)$, $K_1 = (H_2 \cap H_3)$ und $K_2 = (H_6 \cap H_7)$.

schrieben, ist die Verfeinerung einer Zelle nur dann notwendig, wenn sich aus der Berechnung von F_B die Belegung *grau* ergibt.

Die Behandlung von Körpern mit Hohlräumen und Löchern soll hier nicht weiter ausgeführt werden. Lösungsansätze hierzu finden sich in (Mundani, 2005).

Oktalbaumcodierung von flächenartigen Objekten

Da dimensionsreduzierte Objekte naturgemäß nicht durch die Verschneidung von Halbräumen darstellbar sind, kann das von Mundani entwickelte Verfahren nicht zur Oktalbaumcodierung derartiger Objekte herangezogen werden. Daher wird hier eine „klassische“ Herangehensweise gewählt, bei der nicht der Schnitt einer Ebene mit der Oktalbaumzelle, sondern der Schnitt der tatsächlichen Facette mit dieser Zelle geprüft wird (Wenisch & Wenisch, 2004).

Der Unterschied liegt vor allem in der notwendigen Klassifikation der Lage der Eckpunkte des Dreiecks bezüglich der Eckpunkte der Oktalbaumzelle. Ein optimierter Algorithmus zum Test des Schnitts zwischen einem Dreieck und einem Quader wird beispielsweise in (Akenine-Möller, 2001) vorgestellt. Eine weitere Beschleunigung kann dadurch erreicht werden, dass ein Oktant alle positiv getesteten Facetten in einer Kandidatenliste speichert und diese an seine Kindoktanten weitergibt. Für diese muss dann nicht mehr der Schnitt mit allen Facetten des Ausgangsobjekts geprüft werden, sondern nur mit den Mitgliedern der Kandidatenliste (Wenisch & Wenisch, 2004).

Für das Setzen des Farbwerts einer Zelle ist zu beachten, dass nach den Definitionen von Abschnitt 7.3.3 Rand und Inneres von flächenförmigen Objekten anders definiert sind als von Körpern. Wie bereits zu Beginn des Abschnitts 8.2.2 besprochen, muss für die Oktalbaumcodierung dimensionsreduzierter Objekte die zusätzliche Farbe *schwarz-weiß* eingeführt werden, um das Auftreten von Innerem und Äußerem ohne Rand abzubilden (vergl. Abb. 8.8 links). Gibt es einen Schnitt zwischen dem betrachteten Dreieck und der Oktalbaumzelle, wird diese zunächst als *schwarz-weiß* markiert.

Anschließend wird der Rand markiert. Dabei gilt für triangulierte Flächen, dass der Rand der Gesamtfläche sich aus allen Dreieckskanten zusammensetzt, die genau ein Dreieck begrenzen. Bei Wahl einer geeigneten Datenstruktur zur Spei-

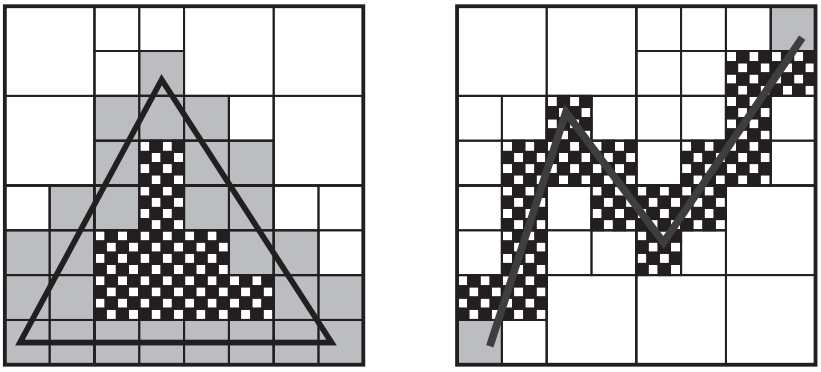


Abbildung 8.8: Oktalbaumgenerierung für ein *Surface*- (**links**) und ein *Line*-Objekt (**rechts**). Bei dimensionsreduzierten Objekten werden *schwarz-weiße* Zellen verwendet, wenn zwar das Innere, nicht aber der Rand die betreffende Zelle schneidet. Gezeigt ist ein Schnitt durch den Oktalbaum.

cherung des Dreiecksnetzes (*vef*-Graph) sind diese Kanten mit Hilfe von graphenanalytischen Verfahren vor Beginn der eigentlichen Oktalbaumgenerierung zu bestimmen. Für die auf diese Weise ermittelten Dreieckskanten wird auf jeder Ebene ein zusätzlicher Schnitttest Kante–Zelle durchgeführt. Sollte dieser positiv ausfallen, wird die betrachtete Zelle *grau* markiert und auf diese Weise der vorher gesetzte Farbwert *schwarz-weiß* überschrieben.

Oktalbaumcodierung von linienartigen Objekten

Bei der Oktalbaumcodierung von linienartigen Objekten wird analog zu der von flächenartigen Objekten vorgegangen. Ausgehend davon, dass das Ausgangsobjekt als (offenes) Polygon[1] vorliegt, werden entsprechend der Definition von Abschnitt 7.3.2 diejenigen Endpunkte der Einzelstrecken als Endpunkte des *Line*-Objekts ermittelt, die nur zu genau einer Teilstrecke gehören. Hierbei ist eine geeignete Datenstruktur zur Verwaltung von *Line*-Objekten hilfreich.

Bei der sich anschließenden Oktalbaumgenerierung wird auf jeder Ebene für alle Teilstrecken der Schnitt mit den Oktalbaumzellen getestet, zum Beispiel mit dem in (Gregory *et al.*, 1999) beschriebenen Verfahren. Liegt ein solcher Schnitt vor, wird die Zelle als *schwarz-weiß* markiert. Anschließend wird für alle Zellen mit Hilfe eines einfachen Koordinatenvergleichs geprüft, ob einer der zuvor ermittelten Endpunkte innerhalb der Zelle liegt. Ist dies der Fall, wird die entsprechende Zelle als *grau* markiert (Abb. 8.8 rechts).

Oktalbaumcodierung von *Point*-Objekten

Die Oktalbaumcodierung von *Point*-Objekten ist vergleichsweise trivial. Immer dann, wenn ein Punkt innerhalb einer Oktalbaumzelle liegt, wird diese als *schwarz-weiß* markiert. Bei komplexen *Point*-Objekten mit einer großen Anzahl an geometrischen Punkten kann gegebenenfalls wiederum die Verwendung einer

[1] im Englischen häufig als *polyline* bezeichnet

Kandidatenliste sinnvoll sein, die an die jeweiligen Kindzellen übergeben wird, um die Menge der jeweils zu testenden Punkte zu begrenzen.

Zeitpunkt der Erzeugung

Hinsichtlich des Zeitpunkts der Erzeugung von Oktalbäumen aus der originären Geometrierepräsentation (in der Regel BRep) gibt es zwei Möglichkeiten. Die erste besteht darin, einen Oktalbaum einer festgelegten Tiefe beim Anlegen eines geometrischen Objekts einmal zu erzeugen und im Weiteren in der Datenbank vorzuhalten. Dies bedeutet jedoch einen nicht zu unterschätzenden Speicheraufwand. Beispielsweise sind zum Erreichen der im Bauwesen üblichen Genauigkeit im Bereich von 1 cm für ein Gesamtgebiet von 50 m × 50 m × 30 m (typische Gebäudeabmessungen) bei Verwendung des reinen Normzellenaufzählungsschemas 75 Milliarden Zellen notwendig. Nach den im Abschnitt 8.2.1 besprochenen Ansätzen ist zwar unter Verwendung der Oktalbaumcodierung eine Reduktion von $\mathcal{O}(n^3)$ auf $\mathcal{O}(n^2)$ anzunehmen. Dies reduziert die Menge an benötigten Zellen (Oktanten) jedoch lediglich auf geschätzte 17,8 Millionen. Bei Annahme von 9 Byte Speicherplatzbedarf pro Oktant (1 Byte für die Farbe, 8 Byte für die Pointer zu den Kindoktanten) ergibt sich ein Speicherplatzbedarf von etwa 152 MByte. Ein weiterer Nachteil ist, dass bei jeder Änderung am originären geometrischen Objekt der zugehörige Oktalbaum neu erzeugt werden muss, d.h. jede *Update*-Operation entsprechend zeitaufwändig ist.

Die geeignetere Variante besteht darin, den Oktalbaum *on-the-fly*, d.h. erst während der Anfragebearbeitung zu erzeugen. Dies führt zwangsläufig zu einer höheren Bearbeitungszeit bei Verwendung räumlicher Operatoren in einer Anfrage, hat aber den Vorteil, dass die Tiefe des Oktalbaums zur Anfragezeit gewählt und damit die Genauigkeit der Berechnung festgelegt werden kann. Zudem entfällt die Generierung des Oktalbaums beim erstmaligen Speichern und bei allen sukzessiven Modifikationen.

Die Entscheidung zugunsten der einen oder anderen Variante hängt stark von der relativen Häufigkeit von Änderungsoperationen (*Update*) und Anfrageoperationen ab. Für hochgradig dynamische Modelle, die häufigen Änderungen unterliegen, wie Bauwerksmodelle während der Planungsphase, ist der *on-the-fly*-Ansatz vorzuziehen.

8.3 Metrische Operatoren

8.3.1 Basisalgorithmus zur Distanzbestimmung

Im Folgenden soll ein Algorithmus vorgestellt werden, mit dessen Hilfe der Euklidische Abstand zweier Objekte bestimmt werden kann, deren Geometrie gerastert und oktalbaumcodiert vorliegt. Wesentlicher Vorteil gegenüber herkömmlichen Algorithmen, die meist auf der Abstandsbestimmung zwischen den Dreiecken der facettierten Oberflächen beruhen, ist die ausgezeichnete Skalierbarkeit des Algorithmus.

Die Bestimmung des Abstands zweier einfacher Punktobjekte ist trivial und sollte daher nicht mit dem hier vorgestellten oktalbaumbasierten Algorithmus umgesetzt werden. Für die Bestimmung des Abstands zweier Punktwolken (komplexe Punktobjekte) ist er jedoch ebenso in Betracht zu ziehen, wie für die Bestimmung des Abstands zwischen einem Punktobjekt und einem Objekt eines höherdimensionalen Typs.

Der Algorithmus arbeitet zwar grundsätzlich mit zwei-, drei- und vierfarbigen Oktalbäumen, für ihn ist jedoch lediglich von Interesse, ob eine Zelle belegt oder nicht belegt ist. Alle anderen Informationen werden ignoriert.

Eingangsgrößen sind die beiden Oktalbäume der Objekte, deren Abstand bestimmt werden soll, sowie die maximale Verfeinerungsstufe. Ausgabewert ist ein Intervall in $\mathbb{R}^+$, in dem der Abstand der Objekte liegt. Dies kann als Angabe des maximalen Fehlers interpretiert werden.

Bei einer naiven Herangehensweise könnte der Abstand zweier Rastergeometrien durch die Bestimmung des Abstands aller Zellen aus A zu allen Zellen aus B und anschließender Minimumbildung berechnet werden. Bei einer sehr feinen Auflösung von beispielsweise $2^{12} \times 2^{12} \times 2^{12}$ Rasterpunkten, wie sie für eine ausreichend genaue Repräsentation der Bauteile eines Gebäude notwendig ist, würde das jedoch im Extremfall dazu führen, dass $2^{36} \times 2^{36}$ Abstandsbestimmungen notwendig wären.

Um diese hohe Zahl an Operationen zu vermeiden, beruht der hier vorgeschlagene Algorithmus *findClosestCells* (Alg. 8.1) auf einer sukzessiven Verfeinerung im Sinne der Ebenen eines Spacetree und dem Prinzip, dass auf einer beliebigen Ebene jeweils diejenigen Zellpaare (ein Paar besteht aus einer Zelle aus Baum A und einer aus Baum B) von einer weiteren Verfeinerung ausgeschlossen werden können, deren Abstand bereits über dem der übrigen Zellpaare liegt. Dazu wird auf jeder Rekursionsebene die Funktion *createCandidateList* (Alg. 8.2) aufgerufen, die diejenigen Zellpaare der nächsten Verfeinerungsstufe herausfiltert, die zu den sich potentiell am nächsten liegenden gehören.

Der Algorithmus berücksichtigt dabei, dass eine als „gefüllt“ markierte Zelle lediglich eine unscharfe Aussage hinsichtlich der tatsächlichen Position der Berandung liefert (Abb. 8.9). Daher wird für jedes Zellpaar jeweils ein oberer und ein unterer Abstandswert bestimmt (Alg. 8.2, Z. 6–14). Diese beiden Werte reflektieren den Bereich, in dem der tatsächliche Abstand liegt. Sie berechnen sich aus den Koordinaten der Zelle im Koordinatensystem der aktuellen Ebene des Spacetree.

Dabei wird in jeder Koordinatenrichtung der minimale und der maximale Abstandswert wie folgt berechnet: Ist der Koordinatenwert der beiden Zellen gleich, so ist der minimale Abstand 0 und der maximale Abstand 1 (Alg. 8.2, Z. 12). In allen anderen Fällen ergibt sich der minimale Abstand aus der Differenz der Koordinatenwerte minus 1 und der maximale Abstand aus der Differenz der Koordinatenwerte plus 1 (Z. 9–10). Aus den drei so erhaltenen minimalen und maximalen Werten berechnen sich der minimale und der maximale Abstandswert als Summe der Quadrate dieser Werte. Die aufwändige Berechnung der Quadratwur-

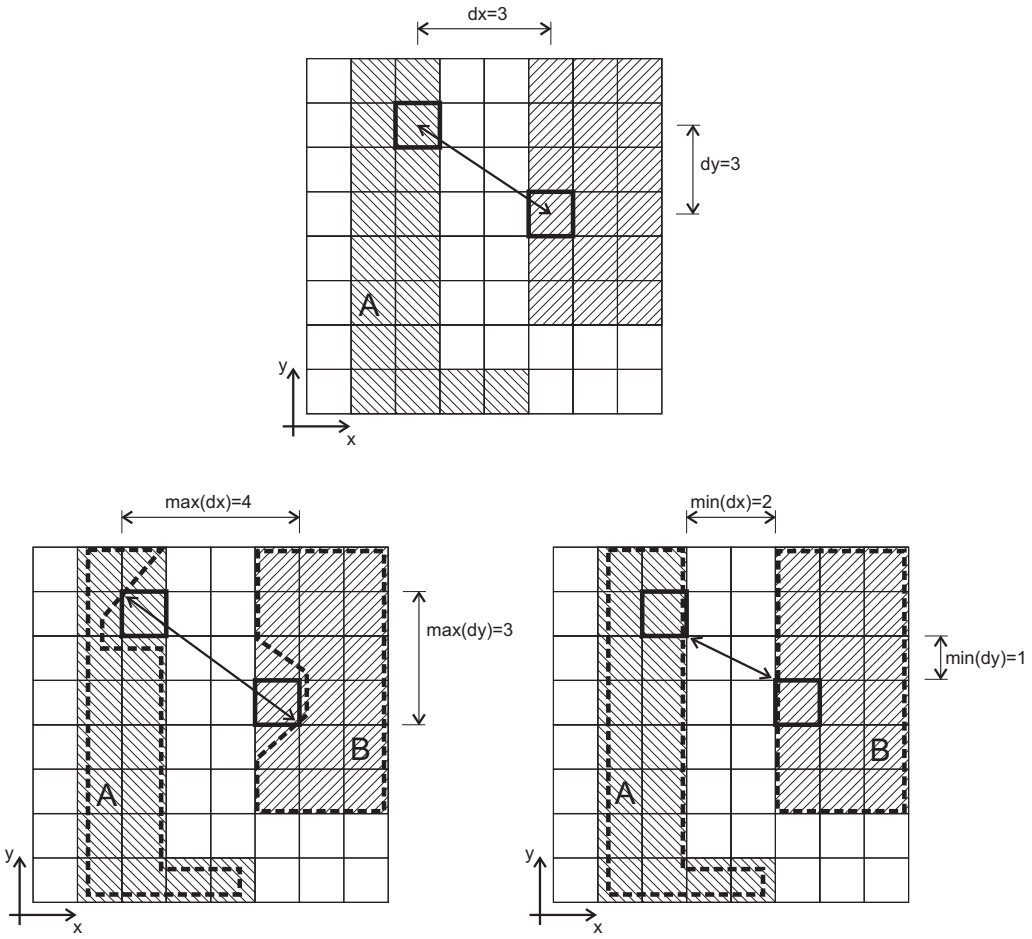


Abbildung 8.9: Da die genaue Geometrie der Objekte (strichliert) nicht bekannt ist, müssen für jedes Zellpaar ausgehend von den richtungsweisen Zellabständen (**oben**), obere Abstandswerte (potentielle Konstellation **unten links**) und untere Abstandswerte (potentielle Konstellation **unten rechts**) bestimmt werden. Für das gezeigte Zellpaar ergibt sich der obere Abstandswert zu $4^2 + 3^2 = 25$ und der untere zu $2^2 + 1^2 = 5$.

zel ist hier nicht notwendig, da es zunächst lediglich festzustellen gilt, in welcher Ordnungsrelation (näher, ferner) die Zellpaare untereinander stehen.

Auf dieser Grundlage werden diejenigen Zellpaare ermittelt, die zu den Kandidaten für das Paar mit dem kürzesten Abstand gehören. Dabei wird zunächst der kleinste obere Abstandswert aller Paare bestimmt (Z. 16). Danach werden alle diejenigen Zellpaare ausgeschlossen, deren unterer Abstandswert größer ist als der kleinste obere Abstandswert (Z. 23).

Alle übrigen Zellpaare sind Kandidaten. Für sie wird der Algorithmus *findClosestCells* rekursiv aufgerufen (Alg. 8.1, Z. 10), d.h. sie werden verfeinert (Z. 3) und für die dabei entstehenden Subzellenpaare werden wiederum Abstandswerte bestimmt, auf deren Grundlage Kandidaten für die nächste Ebene ausgewählt werden. Abbruchkriterium für die Rekursion ist das Erreichen der geforderten Genauigkeit bei der Abstandsberechung bzw. das Erreichen der untersten Ebene der Spacetrees.

Das Intervall, in dem der tatsächliche Abstand liegt, wird nach Beendigung der Rekursion ermittelt. Die untere Intervallgrenze ergibt sich aus der Quadratwurzel des kleinsten unteren Abstandswertes, die obere aus der Quadratwurzel des kleinsten oberen Abstandswertes, jeweils multipliziert mit der Kantenlänge einer Zelle auf der letzten betrachteten Rekursionsebene. Rückgabewert ist entweder ein Tupel zweier reeller Zahlen mit den Grenzen des Abstandsintervalls oder ein einziger Wert, der sich aus dem arithmetischen Mittel der beiden Intervallgrenzen ergibt. Letzteres ist vor allem für den beabsichtigten Einsatz in einer Anfragesprache sinnvoller, da auf diese Weise die Verwendung des Operators in einer Konditionalformulierung erleichtert wird.

Der Anwender bzw. Administrator legt mit der Wahl der maximalen Verfeinerungsstufe sowohl den größtmöglichen Fehler bei der Abstandsbestimmung als auch eine obere Grenze für den Aufwand des Algorithmus fest. Auf diese Weise ist eine Abwägung zwischen dem Ziel einer kurzen Bearbeitungszeit für eine Anfrage und der erzielbaren Genauigkeit möglich.

Mit Hilfe des Basisalgorithmus können bei leichter Abwandlung neben dem Operator *distance* auch *maxdist*, *diameter* sowie *isCloser* und *isFarther* implementiert werden. Gerade bei den letzten beiden Operatoren kommt der iterative Charakter des Algorithmus zum Tragen, da er hier ggf. schon bei vergleichsweise grober Auflösung abbrechen kann.

8.3.2 Beispiel

Die Arbeitsweise des Algorithmus soll anhand eines Beispiels in 2D erläutert werden. Gegeben ist die Quadtree-codierte Rasterrepräsentation zweier Flächen A und B (Abb. 8.10), deren exakte Hülle strichliert dargestellt ist. Zum besseren Verständnis werden in den Abbildungen die Rasterrepräsentationen der beiden Körper auf der jeweiligen Ebene in einem gemeinsamen Raster gezeigt. Dabei werden durch Fläche A belegte Quadranten mit nach rechts unten laufender Schraffur

```
OctantPair[] findClosestCells
                    (OctantPair[] octantPairs, int currentLevel)
 1: currentLevel ← currentLevel + 1
 2: for all octantPairs do
 3:    children ← refine(octantPairs[i])
 4:    allChildrenPairs += children
 5: end for
 6: OctantPair[] closestPairs ← createCandidateList(allChildrenPairs, currentLevel)
 7: if currentLevel = maxLevel then
 8:    return closestPairs
 9: end if
10: OctantPair[] closestChildren ← findClosestCells(closestPairs, currentLevel)
11: return closestChildren
```

Algorithmus 8.1: Die Funktion *findClosestCells* dient der Bestimmung der sich am nächsten liegenden Zellen und ruft sich selbst rekursiv auf.

```
OctantPair[] createCandidateList(OctantPair[] octantPairs)
 1: int lowestUpperBound ← domainSize;
 2: int lowestLowerBound ← lowestUpperBound;
 3: for all octantPairs do
 4:    lowerBound ← 0
 5:    upperBound ← 0
 6:    for i = 0 to 2 do
 7:       calculate dist[i] // 0 ≅ x, 1 ≅ y, 2 ≅ z
 8:       if dist[i] > 0 then
 9:          lowerBound ← lowerBound+(dist[i]−1)²
10:          upperBound ← upperBound+(dist[i]+1)²
11:       else
12:          upperBound ← upperBound+1
13:       end if
14:    end for
15:    if upperBound < lowestUpperBound then
16:       lowestUpperBound ← upperBound;
17:    end if
18:    if lowerBound < lowestLowBound  then
19:       lowestLowerBound ← lowerBound;
20:    end if
21: end for
22: for all octantPairs do
23:    if lowerBound ≤ lowestUpperBound then
24:       candidateList += octantPairs;
25:    end if
26: end for
27: return candidateList
```

Algorithmus 8.2: Die Subroutine *createCandidateList* filtert diejenigen Zellpaare der nächsten Verfeinerungsstufe heraus, die zu den sich potentiell am nächsten liegenden gehören.

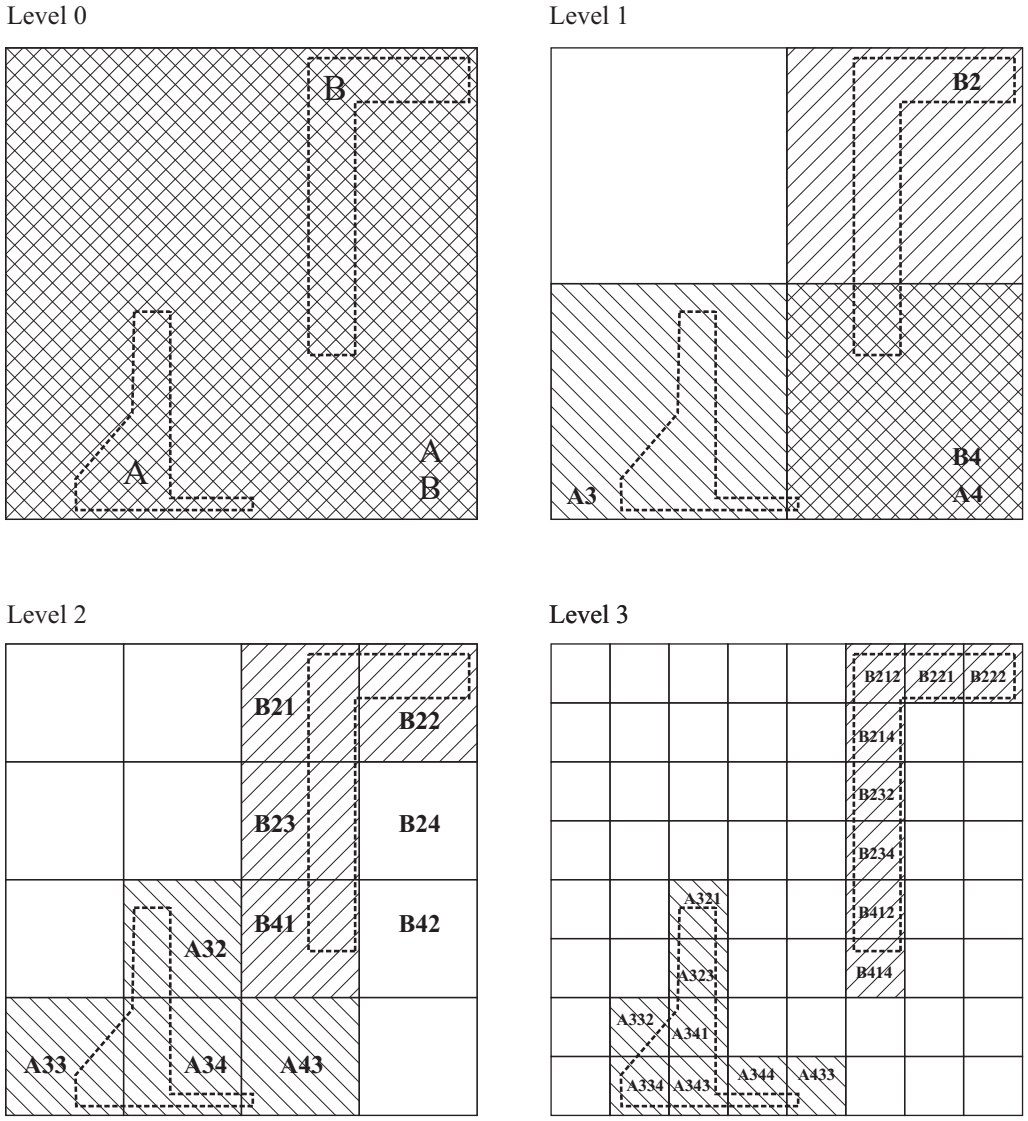


Abbildung 8.10: Darstellung der Ebenen 0 bis 3 der beiden Quadtrees in jeweils einem gemeinsamen Raster. Mit nach schräg unten verlaufender Schraffur sind Zellen gekennzeichnet, die zur Fläche A gehören, mit schräg nach oben verlaufender Schraffur zu Fläche B gehörende. Die verwendete Nummerierung der Zellen ergibt sich aus ihrer „Geschichte“, d.h. aus der Nummerierung der Vater- und Großvaterzellen.

dargestellt, durch Fläche B belegte Quadranten mit nach rechts oben laufender Schraffur.

Der Algorithmus beginnt auf der obersten Ebene der Quadtrees (Ebene 0), auf der es nur eine einzelne Zelle (die Wurzelzelle) gibt. Auf Ebene 0 wird sich immer eine Überlappung der beiden Körper ergeben (Abb. 8.10, links oben). Trotzdem gelten auch hier die oben beschriebenen Regeln zur Berechnung des unteren und des oberen Abstandswertes. Entsprechend kann auf dieser Ebene die Aussage getroffen werden, dass der Abstand der Körper zwischen 0 und $\sqrt{2}$ mal der Länge der Ursprungszelle liegt. Angenommen, die Ursprungszelle hat eine Länge von 8 m, so ergibt sich ein Abstandsbereich von 0 bis 11,3 m. Eine derartig grobe Aussage wird aber in der Regel nicht von Nutzen sein. Daher ist eine weitere Verfeinerung notwendig.

Tabelle 8.4 zeigt die Berechnung der oberen und unteren Abstandswerte für die Ebenen 1 bis 3. Auf Ebene 1 ergeben sich zwei A-Quadranten und zwei B-Quadranten und somit 4 Paare. Der kleinste obere Abstandswert stellt sich beim Paar (*A4, B4*) ein und beträgt 2.

Da es kein Paar gibt, dessen unterer Abstandswert oberhalb dieses Wertes liegt, müssen alle Paare den nächsten Rekursionsschritt durchlaufen, d.h. jedes der Paare wird verfeinert und für die sich ergebenden Kindzellen werden die entsprechenden Abstandswerte bestimmt.

Auf Ebene 2 ergeben sich vier von A und vier von B belegte Quadranten und damit insgesamt 16 Paare. Betrachtet man die sich ergebenden Abstandswerte in Tabelle 1, so ist festzustellen, dass genau ein Paar den Bedingungen für den Ausschluss aus der Rekursion (einer weiteren Verfeinerung) genügt: Das Paar (*A33, B22*) hat einen Abstandswert von 8 und liegt damit über dem kleinsten oberen Abstandswert von 5. Alle anderen Paare erfüllen dieses Kriterium nicht und müssen daher weiter verfeinert werden.

Würde der Algorithmus bei dieser Ebene abbrechen, wäre das Ergebnis, dass der Abstand zwischen A und B im Intervall zwischen 0 (= Wurzel aus kleinstem unteren Abstandswert) und $\sqrt{5}$ (Wurzel aus kleinstem oberen Abstandswert) mal Rasterabstand (= Zellgröße = 2 m) liegt, also zwischen 0 und 4,47 m.

Ebene	Paar		Zellabstand		Unterer Abstandswert			Oberer Abstandswert			Kand.
	A	B	x	y	x	y	gesamt	x	y	gesamt	
0	1	1	0	0	0	0	0	1	1	2	ja
1	3	2	1	1	0	0	0	2	2	8	ja
	3	4	1	0	0	0	0	2	1	5	ja
	4	2	0	1	0	0	0	1	2	5	ja
	4	4	0	0	0	0	0	1	1	**2**	ja
2	32	22	2	2	1	1	2	3	3	18	ja
	32	23	1	1	0	0	0	2	2	8	ja
	32	24	2	1	1	0	1	3	2	13	ja
	32	41	1	0	0	0	0	2	1	**5**	ja
	32	42	2	0	1	0	1	3	1	10	ja
	33	21	2	3	1	2	5	3	4	25	ja
	33	22	3	3	2	2	**8**	4	4	32	**nein**
	33	23	2	2	1	1	2	3	3	18	ja
	33	24	3	2	2	1	5	4	3	25	ja
	33	41	2	1	1	0	1	3	2	13	ja
	33	42	3	1	2	0	4	4	2	20	ja
	34	21	1	3	0	2	4	2	4	20	ja
	34	22	2	3	1	2	5	3	4	25	ja
	34	23	1	2	0	1	1	2	3	13	ja
	34	24	2	2	1	1	2	3	3	18	ja
	34	41	1	1	0	0	0	2	2	8	ja
	34	42	2	1	1	0	1	3	2	13	ja
	43	21	0	3	0	2	4	1	4	17	ja
	43	22	1	3	0	2	4	2	4	20	ja
	43	23	0	2	0	1	1	1	3	10	ja
	43	24	1	2	0	1	1	2	3	13	ja
	43	41	0	1	0	0	0	1	2	5	ja
	43	42	1	1	0	0	0	2	2	8	ja
3	321	212	3	4	2	3	13	4	5	41	ja
	321	214	3	3	2	2	8	4	4	32	ja
	321	221	4	4	3	3	**18**	5	5	50	nein
	321	222	5	4	4	3	**25**	6	5	61	nein
	321	232	3	2	2	1	5	4	3	25	ja
	321	234	3	1	2	0	4	4	2	20	ja
	321	412	3	0	2	0	4	4	1	17	ja
	321	414	3	1	2	0	4	4	2	20	ja
	323	212	3	5	2	4	**20**	4	6	52	nein
	323	214	3	4	2	3	13	4	5	41	ja
	323	221	4	5	3	4	**25**	5	6	61	nein
	323	222	5	5	4	4	**32**	6	6	72	nein
	323	232	3	3	2	2	8	4	4	32	ja
	323	234	3	2	2	1	5	4	3	25	ja
	323	412	3	1	2	0	4	4	2	20	ja
	323	414	3	0	2	0	4	4	1	17	ja
	332	212	4	6	3	5	**34**	5	7	74	nein
	~~332~~	~~221~~									
	~~332~~	~~222~~									
	332	214	4	5	3	4	**25**	5	6	61	nein
	332	232	4	4	3	3	**18**	5	5	50	nein
	332	234	4	3	3	2	13	5	4	41	ja
	332	412	4	2	3	1	10	5	3	34	ja

Fortgesetzt auf Folgeseite . . .

Ebene	Paar		Zellabstand		Unterer Abstandswert			Oberer Abstandswert			Kand.
	A	B	x	y	x	y	gesamt	x	y	gesamt	
	332	414	4	1	3	0	9	5	2	29	ja
	341	212	3	6	2	5	**29**	4	7	65	nein
	341	221	4	6	3	5	**34**	5	7	74	nein
	341	222	5	6	4	5	**41**	6	7	85	nein
	341	214	3	5	2	4	**20**	4	6	52	nein
	341	232	3	4	2	3	13	4	5	41	ja
	341	234	3	3	2	2	8	4	4	32	ja
	341	412	3	2	2	1	5	4	3	25	ja
	341	414	3	1	2	0	4	4	2	20	ja
	334	212	4	7	3	6	**45**	5	8	89	nein
	334	221	5	7	4	6	**52**	6	8	100	nein
	334	222	6	7	5	6	**61**	7	8	113	nein
	334	214	4	6	3	5	**34**	5	7	74	nein
	334	232	4	5	3	4	**25**	5	6	61	nein
	334	234	4	4	3	3	**18**	5	5	50	nein
	334	412	4	3	3	2	13	5	4	41	ja
	334	414	4	2	3	1	10	5	3	34	ja
	343	212	3	7	2	6	**40**	4	8	80	nein
	343	221	4	7	3	6	**45**	5	8	89	nein
	343	222	5	7	4	6	**52**	6	8	100	nein
	343	214	3	6	2	5	**29**	4	7	65	nein
	343	232	3	5	2	4	**20**	4	6	52	nein
	343	234	3	4	2	3	13	4	5	41	ja
	343	412	3	3	2	2	8	4	4	32	ja
	343	414	3	2	2	1	5	4	3	25	ja
	344	212	2	7	1	6	**37**	3	8	73	nein
	344	221	3	7	2	6	**40**	4	8	80	nein
	344	222	4	7	3	6	**45**	5	8	89	nein
	344	214	2	6	1	5	**26**	3	7	58	nein
	344	232	2	5	1	4	**17**	3	6	45	nein
	344	234	2	4	1	3	10	3	5	34	ja
	344	412	2	3	1	2	5	3	4	25	ja
	344	414	2	2	1	1	2	3	3	18	ja
	433	212	1	7	0	6	**36**	2	8	68	nein
	433	221	2	7	1	6	**37**	3	8	73	nein
	433	222	3	7	2	6	**40**	4	8	80	nein
	433	214	1	6	0	5	**25**	2	7	53	nein
	433	232	1	5	0	4	**16**	2	6	40	nein
	433	234	1	4	0	3	9	2	5	29	ja
	433	412	1	3	0	2	4	2	4	20	ja
	433	414	1	2	0	1	1	2	3	**13**	ja

Tabelle 8.4: Berechnung der oberen und unteren Abstandswerte für die Ebenen 1 bis 3 des Beispiels aus Abb. 8.10. Fett gedruckte Zahlen markieren untere Abstandswerte, die über dem kleinsten oberen Abstandswert liegen, der ebenfalls fett gedruckt ist. Die betroffenen Paare müssen nicht verfeinert werden.

Auf Ebene 3 ergeben sich sechs von A und sechs von B belegte Zellen und damit insgesamt 36 Paare. Da die Kinder von *A33* und *B22* jedoch bereits ausgeschlossen wurden, entfallen die Paare (*A332, B221*) und (*A332, B222*) und die Anzahl der zu berechnenden Paare erniedrigt sich auf 34. Wichtig ist, dass nicht alle Kombinationen, in denen einer der beiden Väter *A33* oder *B22* ist, ausgeschlossen werden, sondern nur die Kombinationen, bei denen sowohl *A33* als auch *B22* Väter sind. Der kleinste obere Abstandswert auf Ebene 4 beträgt 13 und tritt beim Paar (*A433, B414*) auf. Es können 33 Paare von der weiteren Verfeinerung ausgeschlossen werden, da ihr größter unterer Abstandswert größer als 5 ist. Lediglich 29 Paare müssen weiter verfeinert werden. Würde der Algorithmus hier

abbrechen, ließe sich der Abstand der Körper A und B als zwischen $1 \cdot$ Zellänge $= 1\,\mathrm{m}$ und $\sqrt{13} \cdot 1\,\mathrm{m} = 3,61\,\mathrm{m}$ gelegen angeben.

8.3.3 Abwandlung des Basisalgorithmus

Der im vorangegangenen Abschnitt beschriebene Algorithmus dient der Bestimmung der Distanz zweier Objekte. Bei leichter Abwandlung können mit seiner Hilfe auch *maxdist*, *diameter* sowie *isCloser* und *isFarther* implementiert werden.

Die Implementierung der Operatoren *isCloser* bzw. *isFarther* ist zum großen Teil identisch mit dem Basisalgorithmus. Sobald das berechnete Distanzintervall vollständig innerhalb des *isCloser* bzw. *isFarther*-Bereichs liegt, kann der Algorithmus abbrechen und *true* zurückgeben. Sobald das Distanzintervall vollständig außerhalb des *isCloser* bzw. *isFarther*-Bereichs liegt, kann der Algorithmus ebenfalls abbrechen, muss in diesem Fall jedoch *false* zurückgeben.

Im Falle von *isCloser* kann der Algorithmus abbrechen und *false* zurückgeben, sobald die untere Grenze des ermittelten Abstandintervalls größer als der übergebene Abstandsparameter ist. Der Algorithmus zur Implementierung von *isFarther* kann hingegen abbrechen und *false* zurückgeben, sobald die obere Grenze des ermittelten Abstandsintervalls kleiner als der übergebene Abstandsparameter ist.

Für die Implementierung von *maxdist* ist eine leichte Abwandlung des Basisalgorithmus notwendig: Statt diejenigen Paare für eine weitere Verfeinerung auszuwählen, deren unterer Abstandswert kleiner als der kleinste obere Abstandswert ist, sind hier diejenigen Paare Kandidaten, deren oberer Abstandswert größer als der größte untere Abstandswert ist. Anders ausgedrückt, können die Paare ausgeschlossen werden, deren oberer Abstandswert kleiner als der größte untere ist.

Mit Hilfe des *maxdist*-Algorithmus kann auch *diameter* implementiert werden. Dazu wird an *maxdist* der Oktalbaum des zu untersuchenden Objekts sowohl als erster und als auch als zweiter Operand übergeben und auf diese Weise die maximale Distanz dieses Objekts zu sich selbst bestimmt, was seiner Ausdehnung entspricht.

Der *maxdist*-Algorithmus lässt sich ebenso zur Feststellung der Durchdringungstiefe zweier Körper verwenden. Dazu muss zunächst der boolesche Schnitt der beiden oktalbaumcodierten Geometrien bestimmt werden, beispielsweise mit dem in (Mundani *et al.*, 2003) beschriebenen Verfahren. Für den resultierenden Oktalbaum wird wiederum der maximale Abstand zu sich selbst bestimmt.

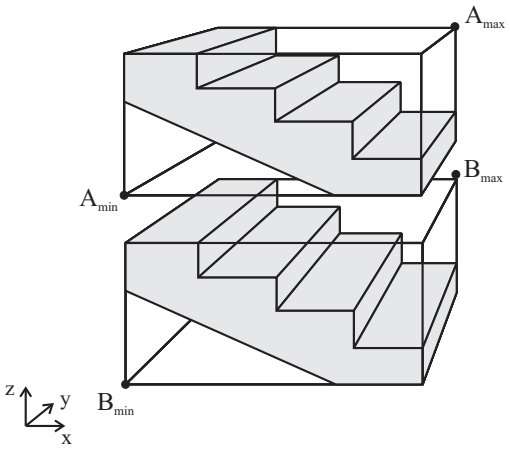


Abbildung 8.11: Die Implementierung der Halbraum-basierten direktionalen Operatoren beruht auf einem Vergleich der Koordinaten der *Bounding Boxes* von Referenz- und Zielobjekt.

8.4 Direktionale Operatoren

8.4.1 Halbraum-basierte direktionale Operatoren

Die Halbraum-basierten direktionalen Operatoren lassen sich einfach und effizient mit Hilfe der *Bounding Boxes* von Referenz- und Zielobjekt implementieren. Geprüft werden muss lediglich, ob sich die Eckpunkte der *Bounding Box* des Zielobjekts im entsprechenden Halbraum bezüglich des Referenzobjekts befinden. Dazu muss ausschließlich die mit der gefragten Richtung assoziierte Koordinate untersucht werden. Für die relaxierten Operatoren genügt es, wenn diese Koordinate bei einem der beiden Eckpunkte der *Bounding Box* größer bzw. kleiner als die Eckpunkte der *Bounding Box* des Referenzobjekts ist (Abb. 8.11). Für die strikten Operatoren müssen beide Eckpunkte in der betreffenden Richtung liegen.

Seien $A_{\min} = (a_{\min,x}, a_{\min,y}, a_{\min,z})$ und $A_{\max} = (a_{\max,x}, a_{\max,y}, a_{\max,z})$ die Eckpunkte der *Bounding Box* des Referenzobjekts und $B_{\min}$ und $B_{\max}$ die Eckpunkte der *Bounding Box* des Zielobjekts. Dann prüft beispielsweise *above_hs*, ob $b_{\max,z} \geq a_{\max,z}$ erfüllt ist. Der strikte Operator *above_hs_strict* hingegen prüft, ob $b_{\min,z} > a_{\max,z}$ erfüllt ist. Bei einfachen Punktobjekten ist $A_{\min} = A_{\max}$ bzw. $B_{\min} = B_{\max}$.

8.4.2 Projektionsbasierte direktionale Operatoren

Die Implementierung der projektionsbasierten Operatoren ist weitaus aufwändiger. Der vorzustellende Algorithmus ist vergleichbar mit der Abstandsbestimmung. Er beruht ebenfalls auf der Hierarchie der Oktalbaumcodierung, verwendet als Basis allerdings dreifarbige Oktalbäume. Der Einsatz zweifarbiger Oktalbäume ist zwar ebenfalls denkbar, in diesem Fall können die rekursiven Algorithmen

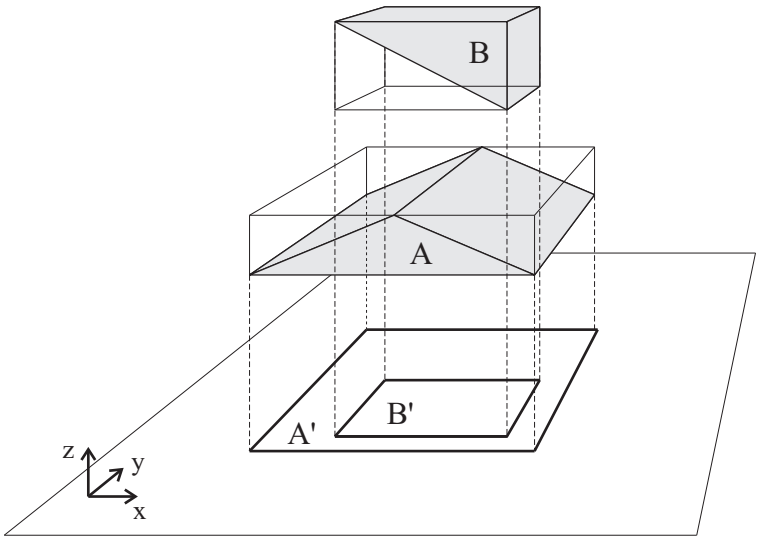


Abbildung 8.12: Bei der Implementierung der projektionsbasierten direktionalen Operatoren wird zunächst geprüft, ob sich die Projektionen der *Bounding Boxes* auf die zur betrachteten Koordinatenachse senkrechte Ebene überlappen.

jedoch nicht vor Erreichen der maximalen Auflösungsstufe abgebrochen werden. Punkt-, linien-, flächen-, und körperförmige Objekte werden gleich behandelt.

Zunächst ist eine Prüfung der relativen Lage der *Bounding Boxes* als grobes Ausschlusskriterium sinnvoll. Die Projektionen der *Bounding Boxes* auf die zur untersuchten Koordinatenachse senkrechten Ebene (x-Achse $\rightarrow$ y-z-Ebene, y-Achse $\rightarrow$ x-z-Ebene, z-Achse $\rightarrow$ x-y-Ebene) müssen sich bei den relaxierten Operatoren überlappen. Bei den strikten Operatoren muss die *Bounding Box* des Zielobjekts vollständig innerhalb der des Referenzobjekts liegen (Abb. 8.12). Ist das jeweilige Kriterium nicht erfüllt, kann der Algorithmus abbrechen und *false* zurückgeben.

Ist der initiale Test bestanden, muss eine intensive Untersuchung auf Basis der genauen Geometrie von Referenz- und Zielobjekt erfolgen. Hierbei wird Gebrauch von einer neu einzuführenden Datenstruktur gemacht, dem so genannten Slot-Baum. Ein Slot-Baum reorganisiert die Zellen des Oktalbaums bezüglich ihrer Lage senkrecht zur betrachteten Koordinatenrichtung. Basis dieser Datenstruktur ist der *Slot*. Betrachtet man beispielsweise die z-Richtung, so beinhaltet ein Slot jeweils übereinander liegende Zellen (Abb. 8.13).

Ein Slot besitzt hierzu eine Liste von Oktanten, die in der Reihenfolge ihres Auftretens entlang der betrachteten Koordinatenrichtung geordnet sind. Dabei können die Oktanten von unterschiedlichen Ebenen des Oktalbaums stammen, also unterschiedlich groß sein (Abb. 8.14). Das bedeutet auch, dass mehrere Slots einer Ebene auf den gleichen Oktanten einer höheren Ebene zeigen können. Ein Slot-Baum ordnet die Slots hierarchisch, ähnlich dem Prinzip des Quadtree bzw. Octree. Ein Slot in 3D hat dabei entweder vier oder keine Kinder, abhängig davon, ob er *graue* Oktanten besitzt.

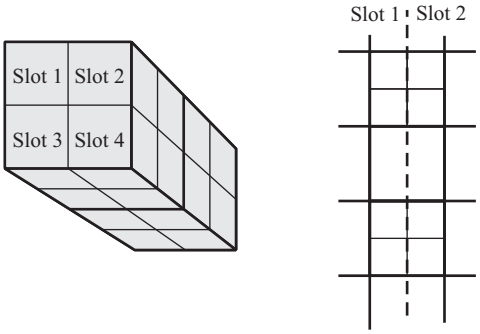


Abbildung 8.13: Slot im drei- und zweidimensionalen Raum. Ein Slot der z-Richtung beinhaltet übereinander liegende Zellen.

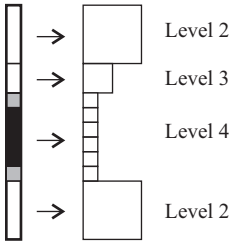


Abbildung 8.14: Ein Slot kann Oktanten von unterschiedlichen Ebenen des Oktalbaums beinhalten. Gezeigt ist hier Slot 1212 aus Abb. 8.15.

Der Slot-Baum kann direkt aus dem Oktalbaum gewonnen werden. Dieser Vorgang ist in Abb. 8.15 illustriert. Der Algorithmus startet beim Ursprungsoktanten und traversiert den Oktalbaum entlang nach unten. Gleichzeitig wird der Slot-Baum aufgebaut, d.h. es werden ggf. Slots erzeugt und als Kinder in den Baum eingefügt. Die Verfeinerung eines Slots ist immer nur dann notwendig, wenn die Zellen des zugrunde liegenden Oktalbaums Kinder besitzen, d.h. also mindestens eine der Zellen des Slots die Farbe *grau* aufweist. Koppelt man die Erzeugung des Slot-Baums an die Abarbeitung des Algorithmus zur Umsetzung des direktionalen Operators, kann darüber hinaus die Verfeinerung an unnötigen Stellen vermieden werden.

Die Einführung der Slots erlaubt die Verwendung einfacher Tests zur Prüfung der direktionalen Prädikate, die auf der Farbe der Zellen und ihrer absoluten Position bezüglich der untersuchten Koordinatenachse beruhen. Die rekursive Struktur erlaubt dabei wieder die schrittweise Verfeinerung der betrachteten Geometrie, was zur einem begrenzten Aufwand führt.

Wegen der differierenden Semantik müssen die strikten Operatoren anders implementiert werden als die relaxierten Operatoren. Die entsprechenden Algorithmen werden daher in den folgenden Abschnitten getrennt vorgestellt. Für eine bessere Verständlichkeit soll dabei ohne Beschränkung der Allgemeinheit nur die Richtung *above* betrachtet werden. Die beschriebenen Algorithmen gelten für alle anderen Richtungen analog.

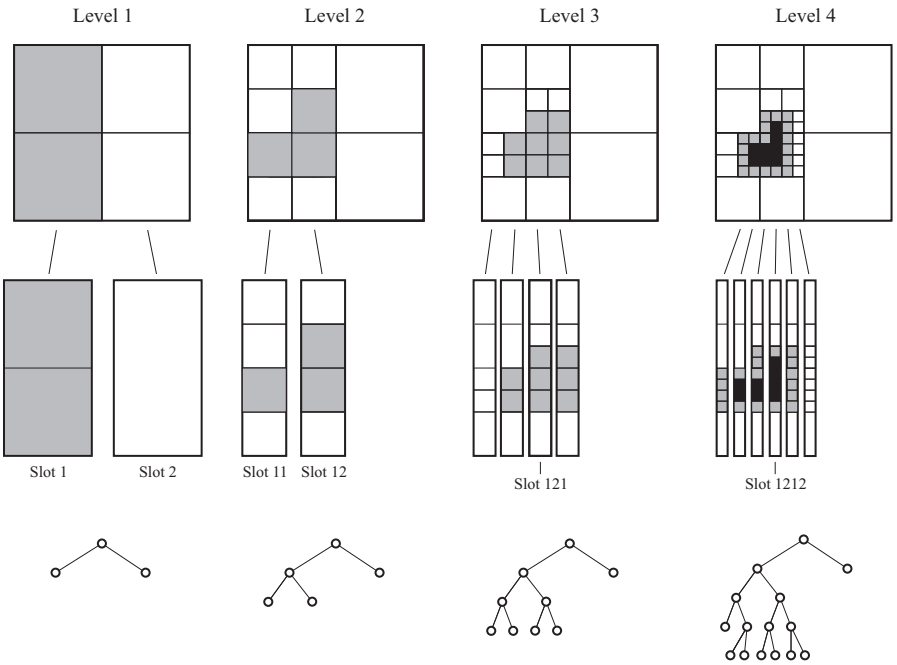


Abbildung 8.15: Aufbau eines Slotbaums bis zur Ebene 4. Ein Slot wird nur verfeinert, wenn er mindestens einen *grauen* Oktanten beinhaltet. Der Slot-Baum wird direkt aus der Oktalbaumdarstellung gewonnen.

Strikte projektionsbasierte Operatoren

Die Steuerung der Rekursion übernimmt die Hauptroutine *isAboveStrict* (Alg. 8.3), der zu Beginn die Ursprungsslots der beiden Objekte A und B als Slotpaar übergeben werden. Es werden immer erst alle Tests auf einer Hierarchieebene durchgeführt und die entsprechenden Paare gebildet, und erst dann wird der Sprung zur nächsten Hierarchieebene durchgeführt (Breitensuche). Auf diese Weise wird erreicht, dass der Algorithmus zum frühestmöglichen Zeitpunkt bereits mit *true* oder *false* abbrechen kann und unnötige Verfeinerungen und Tests vermieden werden.

Die Rekursion bricht ab, wenn entweder einer der Tests für die Kinderpaare ein negatives Ergebnis liefert (Alg. 8.3, Z. 3–5), das maximale Level der Verfeinerung erreicht ist (Z. 9) oder keine Paare für die nächste Verfeinerungsstufe mehr gebildet wurden (Z. 7). Letzteres tritt immer dann auf, wenn alle Slotpaare einer Ebene die Tests vollständig bestehen und daher keine Verfeinerung mehr notwendig ist. In diesem Fall gibt *isAboveStrict true* zurück (Alg. 8.3, Z. 15).

Der Kern des Algorithmus besteht in der slotweisen Überprüfung von Regeln. Dies übernimmt die Subroutine *applyRules_createChildrenPairs* (Alg. 8.4). Sie wird für jedes Slotpaar aufgerufen (Alg. 8.3, Z. 2).

Zunächst werden allgemeine Tests auf der Basis der Farbe des Slots durchgeführt (Alg. 8.4, Z. 3–16). Die Farbe eines Slots ergibt sich aus den Farben der ihm zugeordneten Oktanten. Sobald einer der Oktanten *grau* ist, hat auch der Slot

```
boolean isAboveStrict(SlotPair[] slotpairs, int currentLevel)
 1: for all slotpairs do
 2:    boolean result ← applyRules_createChildrenPairs( slotpairs[i], childrenPairs )
 3:    if result = false then
 4:       return false
 5:    end if
 6: end for
 7: if number of childrenPairs > 0 then
 8:    if currentLevel = maxLevel then
 9:       return true
10:    else
11:       currentLevel ← currentLevel + 1
12:       return isAboveStrict(childrenPairs, currentLevel) // recursive call
13:    end if
14: else
15:    return true
16: end if
```

Algorithmus 8.3: Die sich selbst rekursiv aufrufende Funktion *isAboveStrict*.

die Farbe *grau*. Das gleiche gilt, wenn der Slot sowohl *weiße* als auch *schwarze* Oktanten besitzt. Nur bei ausschließlich *weißen* bzw. ausschließlich *schwarzen* Oktanten erhält er die entsprechende Farbe.

Die farbbasierten Regeln für den strikten direktionalen Operator *above_proj_strict* stellen sich wie folgt dar: Hat der B-Slot die Farbe *schwarz*, kann *false* zurückgegeben werden (Z. 4), da damit Objekt B an dieser Stelle die gesamte Höhe ausfüllt und dadurch nicht mehr gewährleistet ist, dass alle Punkte in B vollständig oberhalb eines A-Punktes liegen. Hat Slot B des Paares die Farbe *weiß*, so muss dieses Slotpaar nicht weiter untersucht werden, d.h. es werden keine Kinderslotpaare gebildet und *true* zurückgegeben (Z. 7).

Ist Slot B *grau* und Slot A *weiß*, bedeutet dies, dass mindestens ein Punkt von B „schwebt“, d.h. sich unter ihm kein A-Punkt befindet. Es muss also *false* zurückgegeben werden (Z. 10). Ist Slot B *grau* und Slot A *schwarz* bedeutet dies hingegen, dass es in diesem Slot *keinen* Punkt in B gibt, der über *allen* A-Punkten mit den gleichen (x, y)-Koordinaten liegt. Entsprechend muss ebenfalls *false* zurückgegeben werden (Z. 13).

Genaue Untersuchungen bezüglich der Lage und Farbe einzelner Oktanten (Z. 17–32) sind nur dann notwendig, wenn beide Slots des Paares die Farbe *grau* aufweisen. Die Subroutine macht hierbei Gebrauch von den Slot-eigenen Methoden *lowestNonWhite, highestNonWhite, highestBlack* und *lowestBlack*, die eine Ganzzahl zurückgeben, sowie *hasBlack*, die *true* oder *false* zurückgibt, je nachdem, ob der Slot eine *schwarze* Zelle besitzt. Die Implementierung dieser Methoden basiert jeweils auf einer Traversierung der dem Slot zugeordneten Oktantenliste.

Die Regeln der genauen Untersuchung sind in Abb. 8.16 dargestellt. Der erste Test überprüft, ob der niedrigste *nicht-weiße* Oktant in B eine geringere Höhe

```
boolean applyRules_createChildrenPairs
                    (SlotPair slotPair, SlotPair[] childrenSlotPairs)
 1: slotA ← slotPair.slotA
 2: slotB ← slotPair.slotB
 3: if slotB.color = black then
 4:   return false
 5: else
 6:   if slotB.color = white then
 7:     return true
 8:   else // slotB.color = grey
 9:     if slotA.color = white then
10:       return false
11:     end if
12:     if slotA.color = black then
13:       return false
14:     end if
15:   end if
16: end if
17: if slotB.lowestNonWhite() < slotA.lowestNonWhite() then
18:   return false // Neg1-Rule
19: end if
20: if slotA.hasBlack then
21:   if slotB.highestBlack() ≥ slotA.lowestNonWhite() then
22:     return false // Neg2-Rule
23:   end if
24:   if slotA.highestNonWhite() < slotB.lowestNonWhite() then
25:     return true // Pos-Rule
26:   end if
27: end if
28: if slotB.hasBlack then
29:   if slotB.lowestBlack() ≤ slotA.highestNonWhite() then
30:     return false // Neg3-Rule
31:   end if
32: end if
33: childrenSlotPairs ← createChildrenSlotPairs(slotA, slotB)
```

Algorithmus 8.4: Die Subroutine *applyRules_createChildrenPairs* übernimmt die slotweise Überprüfung von Regeln und erzeugt Kindzellenpaare.

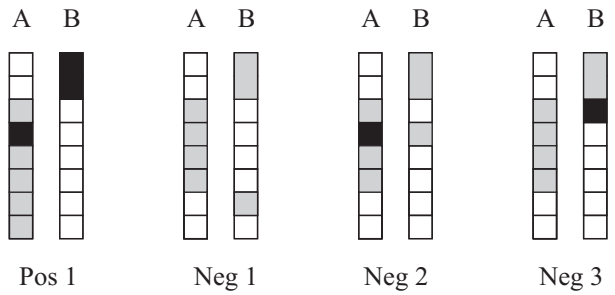


Abbildung 8.16: Drei Beispielkonfigurationen, bei denen die Regeln *Pos, Neg1, Neg2* bzw. *Neg3* im Fall von *isAboveStrict(A,B)* angewendet werden.

aufweist als der niedrigste *nicht-weiße* Oktant in A. Ist dies der Fall, steht das im Widerspruch zur Definition des Prädikats, da es offensichtlich mindestens einen B-Punkt gibt, unter dem sich kein A-Punkt befindet (Regel *Neg1*). Es muss also *false* zurückgegeben werden (Z. 18).

Die nächsten beiden Tests werden nur durchgeführt, wenn A mindestens einen *schwarzen* Oktanten aufweist. Liegt der höchste *schwarze* A-Oktant höher bzw. auf der selben Höhe wie der höchste *schwarze* B-Oktant, so muss *false* zurückgegeben werden (Regel *Neg2*, Z. 22), da nach Definition alle *grauen* oder *schwarzen* B-Zellen oberhalb der obersten *schwarzen* A-Zelle liegen müssen. Liegt hingegen im betrachteten Slot die höchste *nicht-weiße* A-Zelle unter der niedrigsten *nicht-weißen* B-Zelle, ist die Definition vollständig erfüllt und es kann *true* zurückgegeben werden (Regel *Pos*, Z. 25), ohne Kinderslotpaare erzeugen zu müssen.

Der darauf folgende letzte Test wird nur dann durchgeführt, wenn B mindestens einen *schwarzen* Oktanten besitzt. Sollte die niedrigste *schwarze* B-Zelle unterhalb einer *nicht-weißen* A-Zelle liegen, so liegt damit mindestens ein B-Punkt unterhalb eines A-Punktes, d.h. die Definition ist verletzt und es wird *false* zurückgegeben (Regel *Neg3*, Z. 30).

Sollte keiner dieser Tests ein negatives oder positives Ergebnis liefern, so ist für das betrachtete Slotpaar keine eindeutige Aussage möglich. Das heißt, hier ist eine weitere Verfeinerung notwendig. Es werden Paare der jeweiligen Kinderslots gebildet (Z. 33) und diese Liste an die aufrufende Routine zurückgegeben, die sie für die nächste Ebene der Rekursion verwendet.

Relaxierte projektionsbasierte Operatoren

Die Implementierung der relaxierten projektionsbasierten Operatoren arbeitet analog zu der strikter Operatoren. Auch hier werden Slot-Bäume verwendet. Allerdings unterscheiden sich die Regeln, die während der Abarbeitung geprüft werden, und die Vorgehensweise hinsichtlich eines vorzeitigen Abbrechens der Rekursion.

Aus der Definition der relaxierten Operatoren ergibt sich, dass es im Unterschied zu den strikten Operatoren keine Regel geben kann, die den Algorithmus zum

```
boolean isAbove( SlotPair[] slotpairs, int currentLevel )
 1: for all slotpairs do
 2:    boolean result ← applyRules_createChildrenPairs( slotpairs[i], childrenPairs )
 3:    if result = true then
 4:       return true
 5:    end if
 6: end for
 7: if number of childrenPairs > 0 then
 8:    if currentLevel = maxLevel then
 9:       return false
10:    else
11:       currentLevel ← currentLevel + 1
12:       return isAboveStrict(childrenPairs, currentLevel) // recursive call
13:    end if
14: else
15:    return false
16: end if
```

Algorithmus 8.5: Die sich selbst rekursiv aufrufende Funktion *isAbove*. Die Rekursiontiefe entspricht der gerade untersuchten Ebene im Slotbaum.

Abbruch zwingt, weil eine Negativbedingung eingetreten ist. Im Gegenzug kann der Algorithmus dann abbrechen, wenn ein Slot-Paar der Definition genügt.

Dies spiegelt sich in der Prozedur *isAbove* wider (Alg. 8.5), die die rekursive Breitensuche steuert. Die Rekursion wird genau dann abgebrochen, sobald für ein Slot-Paar ein positives Ergebnis von *applyRules_createChildrenPairs* zurückgegeben wird (Z. 3–4). Das gleiche gilt, wenn auf einer Ebene keine untersuchenswerten Kinderslot-Paare erzeugt wurden (Z. 7, 15). Schließlich wird von *isAbove* im Unterschied zu *isAboveStrict* *false* zurückgegeben, wenn auch bei Erreichen der maximalen Verfeinerungsstufe keines der Slot-Paare den Bedingungen genügt (Z. 9).

Die Regeln selbst werden wiederum in der Subroutine *testSlots_createChildrenPairs* geprüft (Alg. 8.6). Auch hier werden zunächst einfache Tests durchgeführt, die auf der Farbe der Slots beruhen. Ist einer der beiden Slots *weiß*, so ist dies kein untersuchenswerter Fall, d.h. es werden keine Kinderslotpaare gebildet und *false* zurückgegeben (Z. 4). Ist keiner der Slots *weiß* und hat mindestens einer die Farbe *schwarz*, bedeutet dies, dass mindestens ein Punkt aus B oberhalb von oder auf der gleichen Position wie ein Punkt aus A liegt (Regel *Pos1*). Damit ist die Definition des relaxierten vollständig Richtungsoperators erfüllt. Es müssen ebenfalls keine Kinderslotpaare erzeugt werden und es kann *true* zurückgegeben werden (Z. 7), was zum Abbruch der Rekursion führt. Ist weder die eine, noch die andere Bedingung erfüllt, sind beide Slots *grau* und es müssen detaillierte Untersuchungen hinsichtlich der Lage und Farbe einzelner Oktanten durchgeführt werden.

Besitzt Slot B einen *schwarzen* Oktanten und liegt dieser auf der selben Höhe oder oberhalb des höchsten *nicht-weißen* Oktanten von Slot A, dann befindet

```
boolean applyRules_createChildrenPairs
                    (SlotPair slotPair, SlotPair[] childrenSlotPairs)
 1: slotA ← slotPair.slotA
 2: slotB ← slotPair.slotB
 3: if slotA.color = white or slotB.color = white then
 4:    return false
 5: else
 6:    if slotA.color = black or slotA.color = black  then
 7:       return true
 8:    end if
 9: end if
10: if slotB.hasBlack and (slotB.highestBlack()≥slotA.lowestNonWhite()) then
11:    return true // Pos1-Rule
12: end if
13: if slotA.hasBlack and (slotB.highestNonWhite()≥slotA.lowestBlack()) then
14:    return true // Pos2-Rule
15: end if
16: childrenSlotPairs ← createChildrenSlotPairs(slotA, slotB)
```

Algorithmus 8.6: Die Subroutine *applyRules_createChildrenPairs* übernimmt die slotweise Überprüfung der Regeln und erzeugt ggf. Kindzellenpaare.

sich damit mindestens ein B-Punkt oberhalb eines A-Punkts, d.h. die Definition ist vollständig erfüllt und es kann *true* zurückgegeben werden (Z. 11), ohne Kinderslotpaare bilden zu müssen (Regel *Pos1*).

Wenn Slot A mindestens einen schwarzen Oktanten besitzt und der höchste nichtweiße Oktant in B auf der selben Höhe oder höher als der niedrigste schwarze Oktant in A liegt, dann befindet sich wiederum mindestens ein B-Punkt oberhalb eines A-Punktes. Damit ist die Definition vollständig erfüllt (Regel *Pos2*) und es kann *true* zurückgegeben werden.

Sollte keiner der Tests ein negatives oder positives Ergebnis liefern, so bedeutet das entweder, dass Slot A und Slot B ausschließlich *graue* Oktanten besitzen oder dass die *schwarzen* Oktanten keine aussagekräftige Position einnehmen. In beiden Fällen ist eine weitere Verfeinerung notwendig. Es werden Paare der jeweiligen Kinderslots gebildet (Z. 16) und diese Liste an die aufrufende Routine *isAbove* zurückgegeben, die sie für die nächste Ebene der Rekursion verwendet.

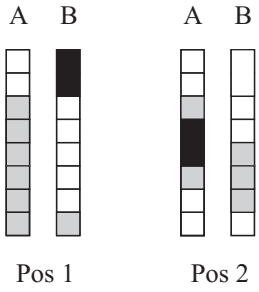


Abbildung 8.17: Beispiele für Konstellationen, bei denen die Regeln *Pos1* bzw. *Pos2* im Fall von *isAbove(A,B)* angewendet werden.

8.5 Topologische Operatoren

Ebenso wie bei der Implementierung der direktionalen und von Teilen der metrischen Operatoren ist auch hier zunächst die Verwendung von *Bounding Box*-Tests als grobes Ausschlusskriterium sinnvoll. Diese fallen je nach Art des topologischen Operators unterschiedlich aus. Zur Erfüllung von *equal* müssen die *Bounding Boxes* der beiden Operanden identisch sein, für *disjoint* dürfen sie sich nicht überlappen, jedoch berühren. Für den Operatoren *touch* gilt, dass sich die *Bounding Boxes* entweder berühren oder überlappen müssen. Für *overlap* hingegen müssen sie sich in jedem Fall überlappen. Um *contain* positiv zu beantworten, muss die *Bounding Box* des zweiten Operators innerhalb der des ersten Operators liegen, bei *within* ist es genau anders herum.

Ist der *Bounding Box*-Test erfolgreich, wird der im Folgenden vorzustellende oktalbaumbasierte Algorithmus für eine detaillierte Betrachtung herangezogen. Das Prinzip des Algorithmus besteht darin, dass bei einer sukzessiven Verfeinerung der Oktalbäume schließlich eine für die Falsifizierung oder Validierung des untersuchten topologischen Prädikats notwendige Menge an Einträgen in der 9IM-Matrix bestimmt werden kann.

Die Implementierung der topologischen Operatoren basiert auf der Verwendung eines vierfarbigen Baumes. Er codiert das Auftreten von Innerem, Rand und Äußerem eines räumlichen Objektes nach den in Abschnitt 7.3 vorgestellten Definitionen. Mit einem dreifarbigen Baum lassen sich zwar alle Varianten für *Body*-Objekte abbilden (*weiß* = nur Außen, *schwarz* = nur Innen, *grau* = Innen, Rand und Außen). Für Objekte mit geringerer Dimensionalität kann jedoch noch zusätzlich die Kombination Innen-Außen auftreten. Diese wird durch eine vierte Farbe (*schwarz-weiß*) repräsentiert (vgl. Abschnitt 8.2.1).

Der Algorithmus ist identisch für alle topologischen Operatoren mit Prädikatcharakter: *disjoint*, *equal*, *within*, *contain*, *touch* und *overlap*. Wie auch die bereits vorgestellten Algorithmen zur Bearbeitung von metrischen und direktionalen Operatoren handelt es sich hierbei um einen rekursiven Algorithmus. Zunächst werden an die Hauptroutine *testTopoPredicate* (Alg. 8.7) die Ursprungsoktanten der beiden Oktalbäume zusammen mit einem Integerwert übergeben, der das zu untersuchende Prädikat beschreibt *(predicate)*, und eine ausschließlich mit Platzhaltern (Sternen) besetzte Matrix *(currentMatrix)*. Diese wird im Laufe der Rekursion mit Werten gefüllt. Zur Codierung der drei möglichen Werte *leere Menge*, *nicht-leere Menge* und *unbestimmt* (Stern) bietet sich die Verwendung der Zahlwerte -1 (unbestimmt), 1 (nicht-leere Menge) und 0 (leere Menge) an. Die Abbildungen 8.27 bis 8.40 illustrieren die Arbeitsweise des Algorithmus für verschiedene topologische Konstellationen und Typkombinationen.

Der Kern des Algorithmus besteht in der Ausführung von Regeln. Das Auftreten bestimmter Farbkombinationen führt dazu, dass in der 9IM-Matrix eine *nicht-leere Menge* eingetragen werden kann (Positivregeln, Abb. 8.18 und 8.19). Auf der anderen Seite kann bei Nicht-Auftreten anderer Kombinationen (Negativregeln,

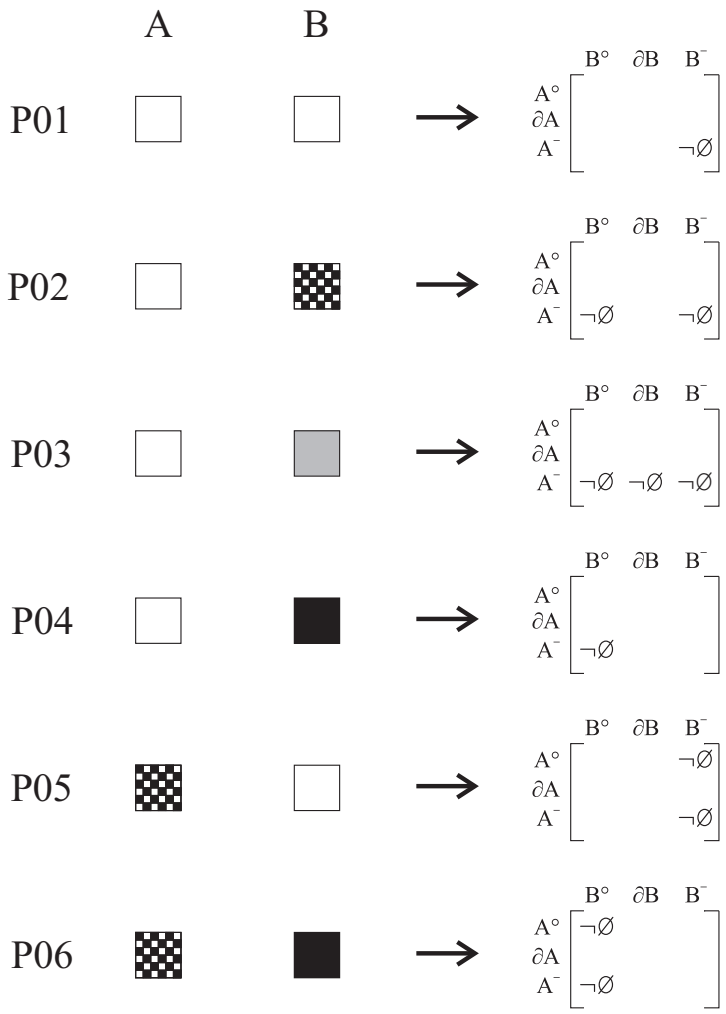


Abbildung 8.18: Positiv-Regeln 1. Wenn die Farbkombination auf der linken Seite auftritt, kann die 9IM-Matrix entsprechend der rechten Seite gefüllt werden. Kombinationen aus gemischtfarbigen Zellen lassen grundsätzlich keine Aussage über die Schnittmengen zu. Die Ursache hierfür liegt darin, dass bei einer Oktalbaumcodierung der genaue Verlauf des Randes unbekannt ist. Aus demselben Grund gibt es des Weiteren keine Kombination, aus der sich $\partial A \cap \partial B = \neg\varnothing$ ableiten lässt.

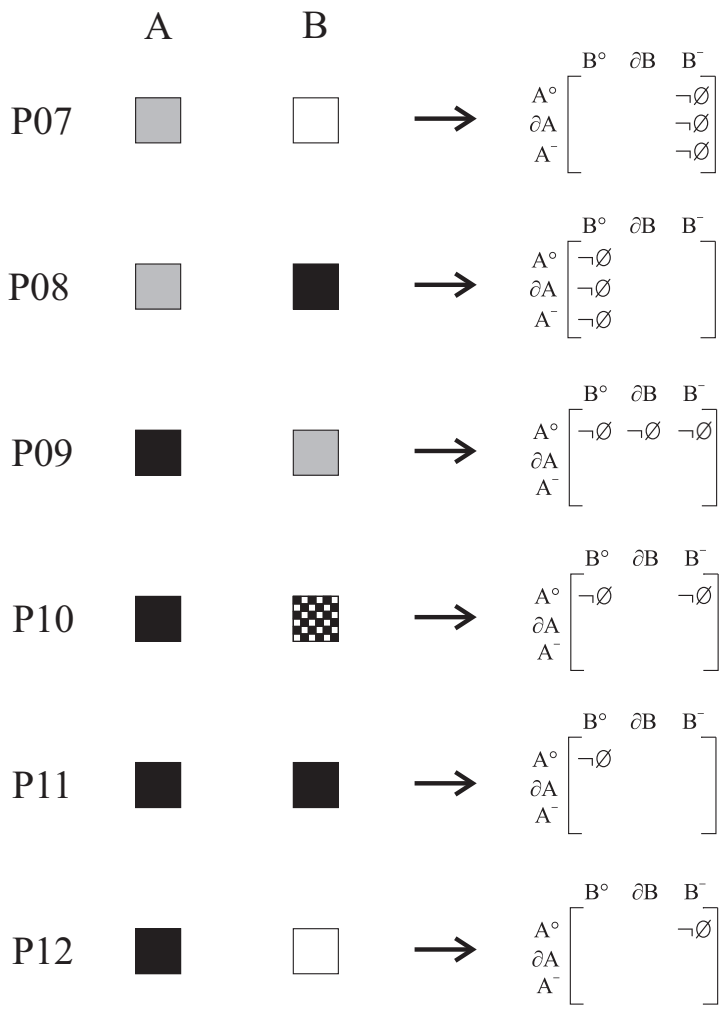


Abbildung 8.19: Positiv-Regeln 2. Wenn die Farbkombination auf der linken Seite auftritt, kann die 9IM-Matrix entsprechend der rechten Seite gefüllt werden. Kombinationen aus gemischtfarbigen Zellen lassen grundsätzlich keine Aussage über die Schnittmengen zu. Die Ursache hierfür liegt darin, dass bei einer Oktalbaumcodierung der genaue Verlauf des Randes unbekannt ist. Aus demselben Grund gibt es des Weiteren keine Kombination, aus der sich $\partial A \cap \partial B = \neg\varnothing$ ableiten lässt.

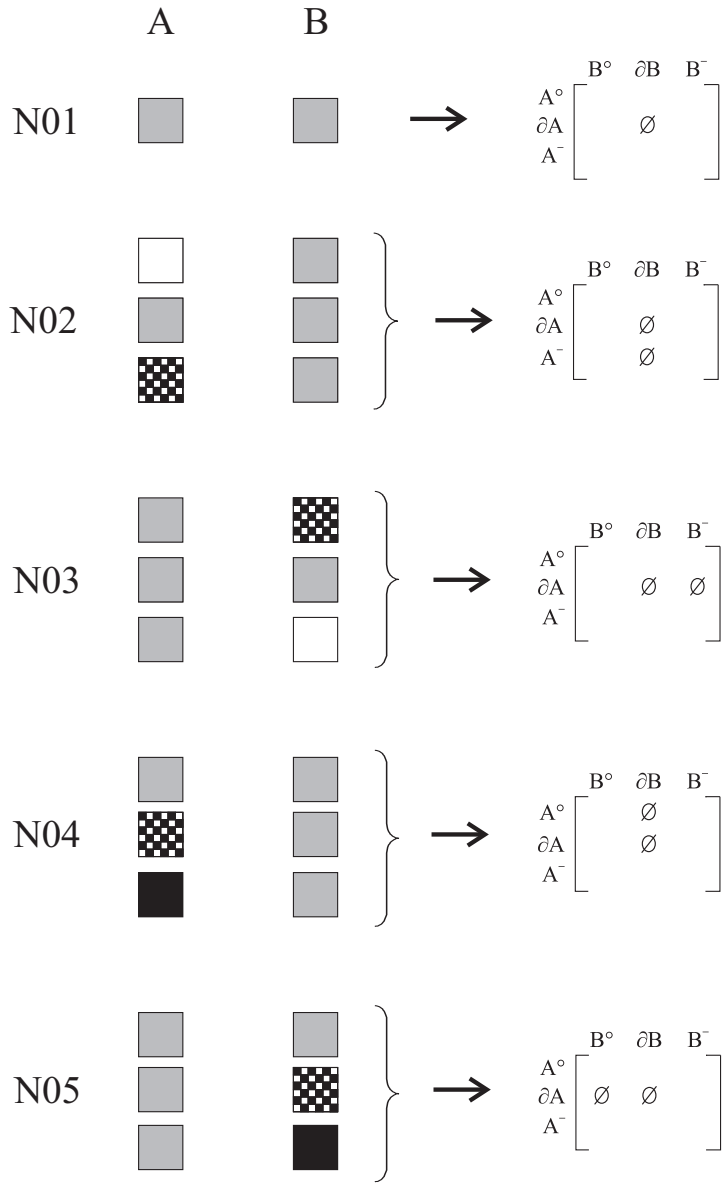


Abbildung 8.20: Negativ-Regeln 1. Wenn die Farbkombinationen auf der linken Seite im gesamten Gebiet *nicht* auftreten, kann die 9IM-Matrix entsprechend der rechten Seite gefüllt werden.

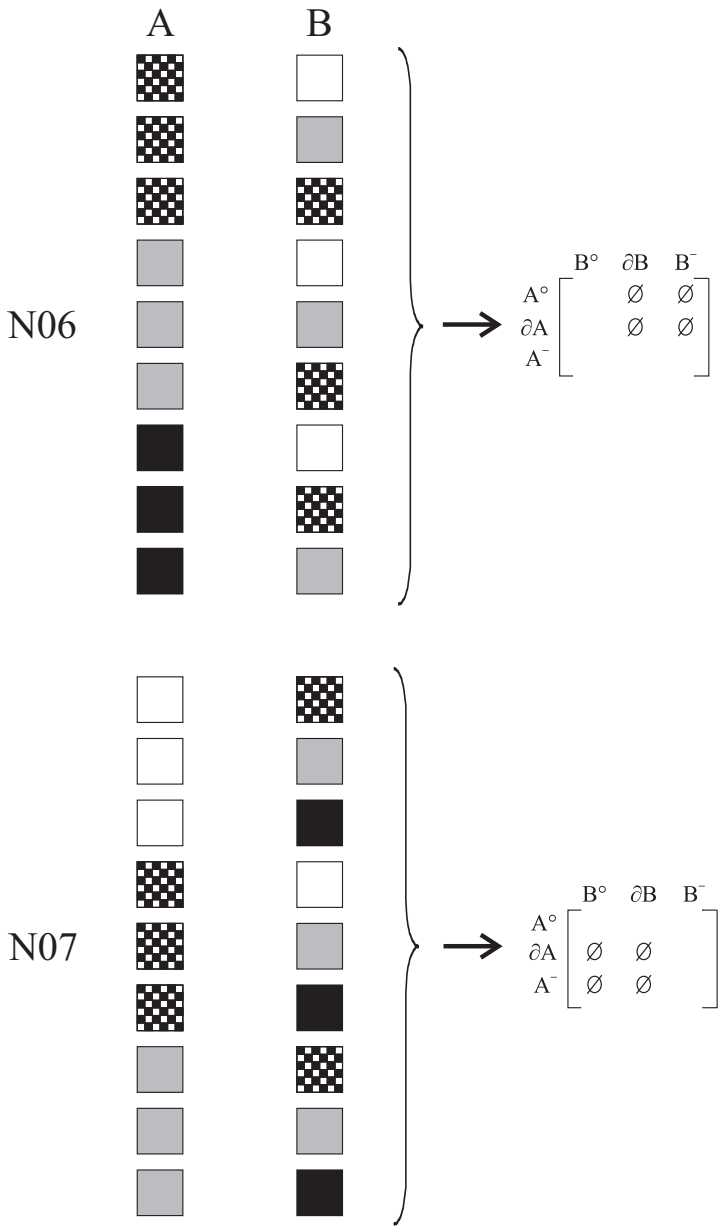


Abbildung 8.21: Negativ-Regeln 2. Wenn die Farbkombinationen auf der linken Seite im gesamten Gebiet *nicht* auftreten, kann die 9IM-Matrix entsprechend der rechten Seite gefüllt werden.

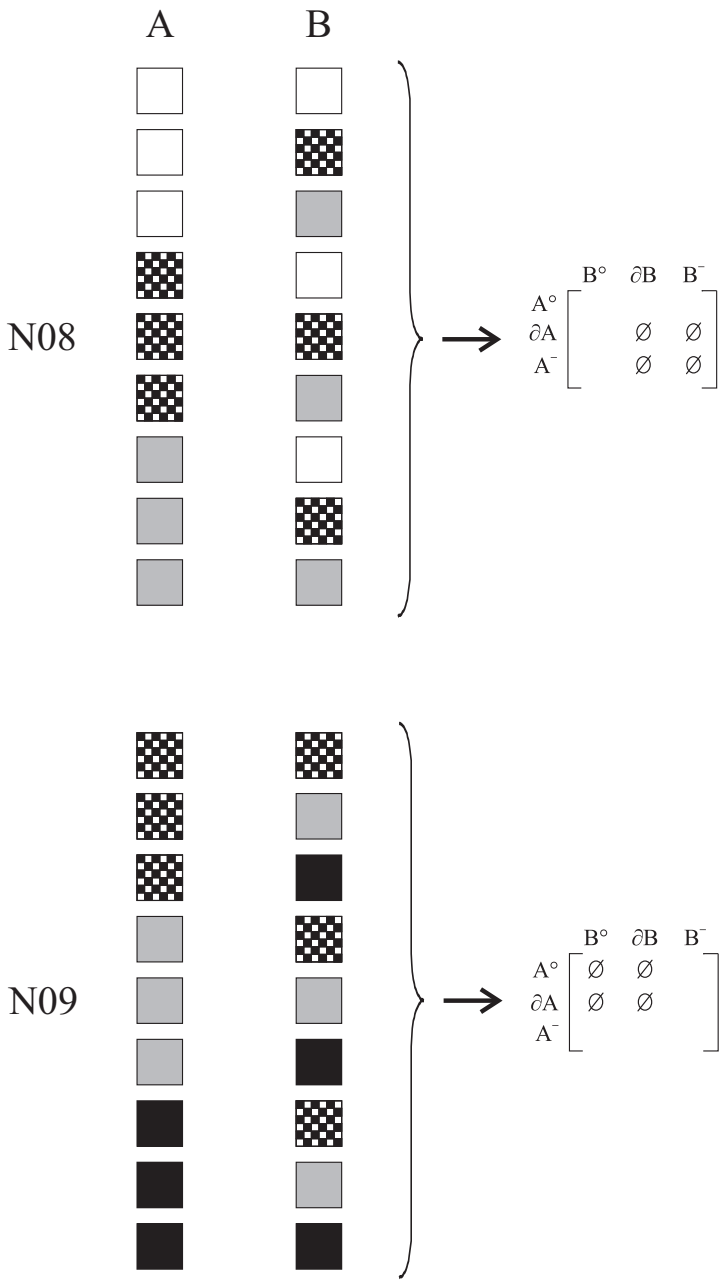


Abbildung 8.22: Negativ-Regeln 3. Wenn die Farbkombinationen auf der linken Seite im gesamten Gebiet *nicht* auftreten, kann die 9IM-Matrix entsprechend der rechten Seite gefüllt werden.

```
boolean testTopoPredicate
    (OctantPair[ ] pairs, int predicate, int[ ] currentMatrix, int currentLevel)
 1: currentLevel ← currentLevel + 1
 2: int[16] color_combinations
 3: for all pairs do
 4:    add 1 to accordant position in color_combinations
 5:    fill currentMatrix according to positive rules
 6:    if currentMatrix dismatches 9IM-matrices[predicate] then
 7:       return false
 8:    end if
 9:    if currentMatrix matches 9IM-matrices[predicate] then
10:       return true
11:    end if
12:    for all predicates i≠predicate  do
13:       if currentMatrix matches 9IM-matrices[i] then
14:          return false
15:       end if
16:    end for
17: end for
18: for all negativ rules do
19:    if rule is applicable then
20:       fill currentMatrix
21:       if currentMatrix matches 9IM-matrices[predicate] then
22:          return true
23:       end if
24:       if currentMatrix dismatches 9IM-matrices[predicate] then
25:          return false
26:       end if
27:       for all  predicates i≠predicate  do
28:          if currentMatrix matches 9IM-matrices[i] then
29:             return false
30:          end if
31:       end for
32:    end if
33: end for
34: if currentLevel = maxLevel then
35:    for all  i < predicate  do
36:       if currentMatrix does not dismatch 9IM-matrices[i] then
37:          return false
38:       end if
39:    end for
40:    return true
41: else
42:    for all pairs do
43:       childrenPairs += createChildrenPairs(pair[i])
44:    end for
45:    return testTopoPredicate(childrenPairs, currentMatrix, currentLevel)
46: end if
```

Algorithmus 8.7: Die Funktion *testTopoPredicate* untersucht das Auftreten einer topologischen Situtation durch sukzessives Füllen der 9IM-Matrix. Sie ruft sich dabei rekursiv selbst auf.

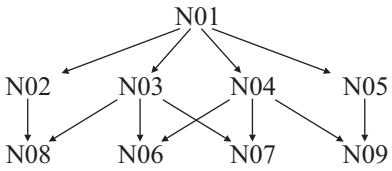


Abbildung 8.23: Abhängigkeiten innerhalb der Negativbedingungen. Wenn die Voraussetzungen eines Vorgängers nicht erfüllt sind, sind sie auch für die Nachfolger nicht erfüllt.

Abb. 8.20, 8.21 und 8.22) an den entsprechenden Stellen der Matrix die *leere Menge* eingetragen werden.

Das Auftreten einer bestimmten Farbkombination wird im Feld *color_combinations* mit 16 booleschen Werten festgehalten (Alg. 8.7, Z. 2). Positivregeln werden sofort bei jedem Zellenpaar angewandt (Z. 5). Anschließend wird geprüft, ob die sich ergebende Matrix bereits eine Validierung oder Falsifizierung des Prädikats zulässt. Ist dies der Fall, wird *true* bzw. *false* zurückgegeben, die Rekursion bricht ab und die Hauptroutine gibt auf der obersten Ebene der Rekursion das entsprechende Ergebnis zurück. Dies ist auch der Fall, wenn ein anderes Prädikat validiert werden konnte (Z. 12–16).

Negativregeln können erst umgesetzt werden, nachdem das Auftreten aller Kombinationen einer Rekursionsstufe festgehalten wurde (Z. 18). Die Negativregeln sind hierarchisch voneinander abhängig (Abb. 8.23): Wenn die Voraussetzungen einer Vorgängerregel nicht erfüllt sind, sind sie automatisch auch für alle Nachfolger nicht erfüllt. Die Negativregeln werden daher in entsprechender Reihenfolge angewendet. Ist eine Regel anwendbar, wird durch einen Vergleich mit der Zielmatrix geprüft, ob die sich ergebende Matrix eine Validierung (Z. 21, 22) oder Falsifizierung (Z. 24, 25) des Prädikats zulässt.

Wenn nach Anwendung aller Regeln die aktuelle Belegung der Matrix keine Validierung oder Falsifizierung zulässt und die maximale Verfeinerungsstufe noch nicht erreicht ist, werden mit Hilfe der Subroutine *createChildrenPairs* (Alg. 8.5) Paare der Kindzellen gebildet (Alg. 8.7, Z. 43). Sollten beide Zellen des Paares keine Kinder haben, so werden keine Kinderzellpaare gebildet. Sollte nur eine der Elternzellen Kinder haben, bilden diese mit der anderen Elternzelle ein Paar. Alle auf diese Weise gebildeten Kinderzellpaare werden durch einen Aufruf von *testTopoPredicate* an die nächste Ebene der Rekursion übergeben (Alg. 8.7, Z. 45).

Ist auch bei Erreichen der maximalen Verfeinerungsstufe das untersuchte Prädikat weder erfüllt noch widerlegt, weil beispielsweise beide Operanden sehr klein bezüglich der Zellgröße der gewählten maximalen Verfeinerungsstufe sind, wird die in Abb. 8.24 gezeigte Hierarchie der topologischen Prädikate angewendet.

Diese Hierarchie beruht auf der Beobachtung, dass bei tatsächlichem Vorliegen einer bestimmten topologischen Beziehung alle in Abb. 8.24 über dem dazugehörigen Prädikat liegenden Beziehungen im Laufe sukzessiver Verfeinerung ausgeschlossen werden, die darunter liegenden jedoch nicht in jedem Fall. Im Sinne

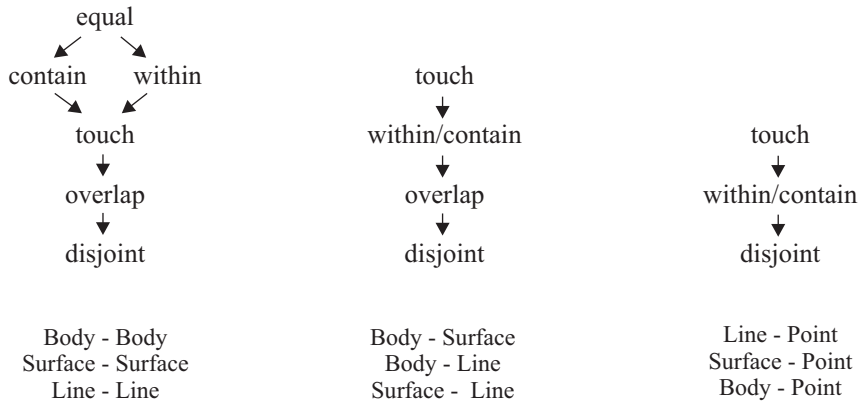


Abbildung 8.24: Die Hierarchien der topologischen Prädikate für unterschiedliche Typkombinationen. Liegt eine bestimmte topologische Beziehung tatsächlich vor, werden alle darüber liegenden Prädikate im Laufe sukzessiver Verfeinerung ausgeschlossen, die darunter liegenden jedoch nicht in jedem Fall.

```
boolean createChildrenPairs(OctantPair pair)
 1: if  (pair.A.color = grey) or (pair.A.color = black-white) then
 2:    if (pair.B.color = grey) or (pair.B.color = black-white) then
 3:       for all A.children do
 4:          for all B.children do
 5:             create pair(A.child, B.child)
 6:             childrenPairs += pair
 7:          end for
 8:       end for
 9:    else // B is white or black
10:       for all A.children do
11:          create pair(A.child, B)
12:          childrenPairs += pair
13:       end for
14:    end if
15: else // A is white or black
16:    if (pair.B.color = grey) or (pair.B.color = black-white) then
17:       for all A.children do
18:          for all B.children do
19:             create pair(A, B.child)
20:             childrenPairs += pair
21:          end for
22:       end for
23:    end if
24: end if
25: return childrenPairs
```

Algorithmus 8.8: Die Subroutine *createChildrenPairs* generiert Kindzellenpaare, die für die Tests auf der nächsten Ebene relevant sind.

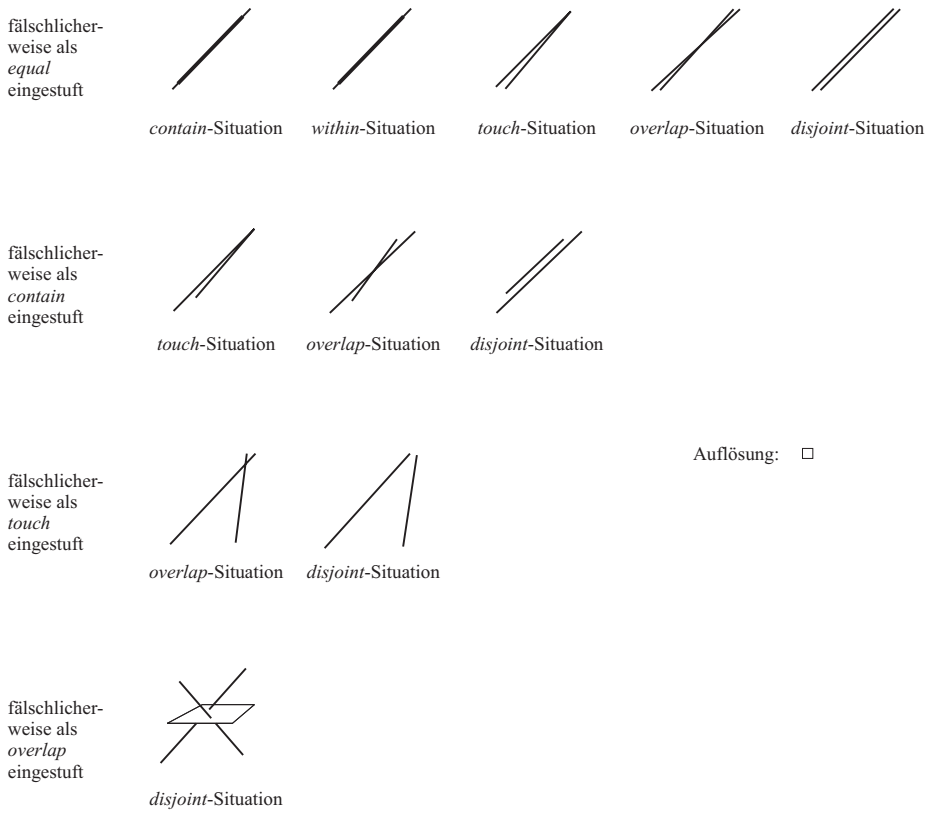


Abbildung 8.25: Fehleinschätzungen bei zu grober Auflösung im Fall von *Line-Line*-Konstellationen. Welche Situationen wie eingeschätzt werden, resultiert direkt aus der Wahl der Hierarchie der topologischen Beziehungen.

eines „positivistischen“ Ansatzes wird nun davon ausgegangen, dass das oberste nicht widerlegte Prädikat gilt.[2]

Wie im Folgenden noch ausführlich diskutiert werden soll, kann es auf diese Weise zu Fehleinschätzungen kommen. Die Hierarchie ist jedoch so gewählt, dass die sich ergebenden Fehleinschätzungen vertretbar sind und mit dem intuitiven Verständnis des Nutzers korellieren. Die Abbildungen 8.25 und 8.26 illustrieren die sich bei zu grober Auflösung ergebenden Fehleinschätzungen bei Anwendung der in Abb. 8.24 gezeigten Hierarchien für *Line-Line*- und *Surface-Surface*-Konstellationen.

Für den Algorithmus *testTopoPredicate* ergibt sich damit folgendes Vorgehen: Ist bei Erreichen der vorgegebenen Genauigkeitsstufe (Alg. 8.7, Z. 34) das untersuch-

[2]Für Konstellationen bei denen beide Operanden vom gleichen Typ sind, sind *contain* und *within* gleichwertig: Für die Validierung eines nachfolgenden Prädikats müssen sowohl *contain* als auch *within* widerlegt sein. Es kann gezeigt werden, dass bei Ausschluss von *equal* gleichzeitig entweder *contain* oder *within* widerlegt werden: *equal* wird durch Anwendung einer oder mehrerer der Regeln P02, P03, P04, P05, P06, P07, P08, P12 ausgeschlossen. Dabei widerlegen die Regeln P02, P03, P04 auch *contain* und P05, P07, P12 *within*. Die Regeln P06, P08, P09, P10, P11 führen zum Ausschluss von beiden Prädikaten. Damit gilt: Sobald *equal* widerlegt ist, sind auch *contain* und/oder *within* widerlegt.

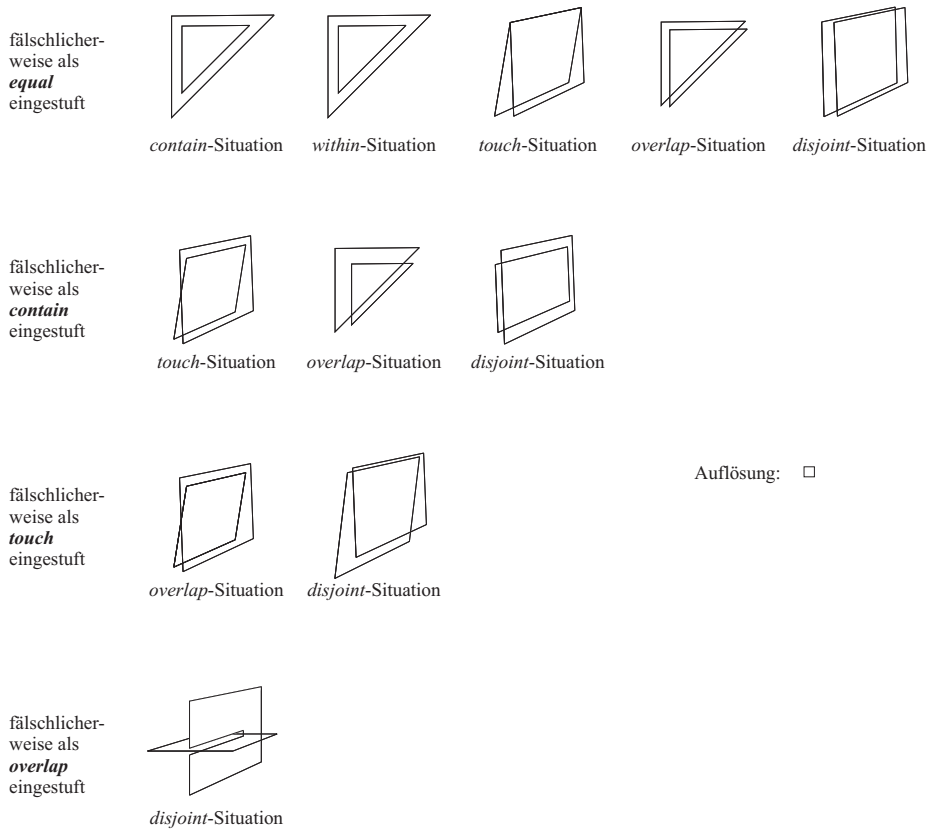


Abbildung 8.26: Fehleinschätzungen bei zu grober Auflösung im Fall von *Surface-Surface*-Konstellationen. Welche Situationen wie eingeschätzt werden, resultiert direkt aus der Wahl der Hierarchie der topologischen Beziehungen.

te Prädikat weder bestätigt noch widerlegt, wird getestet, ob die in der Hierarchie (Abb. 8.24) darüber liegenden Prädikate allesamt widerlegt sind (Z. 35–39). Ist dies der Fall, so wird *true* zurückgegeben (Z. 40), andernfalls *false* (Z. 37). Zu diesem Zweck werden die 9IM-Matrizen in der durch die Hierarchie vorgegebenen Reihenfolge im Feld *9IM-matrices* gespeichert. Der Sonderfall *contain/within* ist im Sinne einer besseren Verständlichkeit im dargestellten Algorithmus nicht berücksichtigt.

Analog dazu verläuft die Bearbeitung des Operators *whichTopoPredicate* (Alg. 8.5): Es wird solange verfeinert, bis eines der Prädikate bestätigt werden kann oder die vorgegebene maximale Verfeinerungsstufe erreicht ist. In letzterem Fall wird das Prädikat zurückgegeben, für das alle in Abb. 8.24 darüber liegenden widerlegt sind.

Auf diese Weise ist die logische Konsistenz und die geforderte *Vollständigkeit* des Systems topologischer Operatoren für beliebige Verfeinerungsstufen gewährleistet, da immer nur von genau einem topologischen Operator *true* und von allen anderen *false* zurückgegeben wird bzw. von *whichTopoPredicate* genau ein Prädikat ermittelt wird.

Dabei wird bewusst das potentielle Auftreten von Fehlern in Kauf genommen. Bei zu grober Auflösung können Objekte beispielsweise als gleich eingestuft werden, die sich bei genauerer Betrachtung überlappen. Hier bleibt es dem Anwender überlassen, durch eine sinnvolle Wahl der Genauigkeit relevante Ergebnisse zu erzielen. Darauf soll im Folgenden noch näher eingegangen werden. Nachteilig ist, dass mit diesem Ansatz für bestimmte Prädikate ein „absolut sicheres“ Ergebnis nicht zu erzielen ist.

Der positivistische Ansatz erlangt vor allem für die topologischen Beziehungen Bedeutung, die auch bei beliebig starker Verfeinerung nicht eindeutig bestätigt werden können. Ist beispielsweise einer der beiden Operanden ein Körper, so ist für die Prädikate *touch* und *equal* auch bei beliebig starker Verfeinerung keine positive Aussage möglich. Die Ursache hierfür liegt darin, dass sich bei tatsächlichem Vorliegen einer dieser topologischen Beziehungen auch bei größtmöglicher Verfeinerung immer eine *graue A*-Zelle mit einer *grauen B*-Zelle überlappt (vgl. Abb. 8.29 und 8.32). Daher ist zum einen die für die Bestätigung des *touch*-Prädikats notwendige Aussage $A^\circ \cap B^\circ = \varnothing$ nicht treffbar und zum anderen keine Negativregel anwendbar, die zur Generierung der $\varnothing$-Einträge in der *equal*-Matrix notwendig wäre.

Bei näherer Betrachtung ist festzustellen, dass *touch* mit den beiden Prädikaten *overlap* und *disjoint* konkurriert. Bereits eine minimale Verschiebung der Objekte zueinander kann zur Erfüllung eines der beiden anderen Prädikate führen. Bei genügend starker Verfeinerung können jedoch sowohl *overlap* als auch *disjoint* bestätigt werden.

Daraus ließe sich schlussfolgern, dass ein Algorithmus, der das Vorliegen einer *touch*-Beziehung zweifelsfrei bestätigen soll, so weit verfeinern könne, bis *disjoint* und *overlap* widerlegt sind. Allerdings ist *overlap* nicht widerlegbar, solange sich

```
int whichTopoPredicate
      (OctantPair[ ] pairs, int[ ] currentMatrix, int currentLevel)
 1: currentLevel ← currentLevel + 1
 2: int[16] color_combinations
 3: for all pairs do
 4:    add 1 to accordant position in color_combinations
 5:    fill currentMatrix according to positive rules
 6:    for all predicates i do
 7:       if currentMatrix matches 9IM-matrices[i] then
 8:          return i
 9:       end if
10:    end for
11: end for
12: for all negativ rules do
13:    if rule is applicable then
14:       fill currentMatrix
15:       for all predicates i do
16:          if currentMatrix matches 9IM-matrices[i] then
17:             return i
18:          end if
19:       end for
20:    end if
21: end for
22: if currentLevel = maxLevel then
23:    for  i = 0 to 5  do
24:       if currentMatrix does not dismatch 9IM-matrices[i] then
25:          return i
26:       end if
27:    end for
28: else
29:    for all pairs do
30:       childrenPairs += createChildrenPairs(pair[i])
31:    end for
32:    return whichTopoPredicate(childrenPairs, currentMatrix, currentLevel) // recur-
       sive call
33: end if
```

Algorithmus 8.9: Die Funktion *whichTopoPredicate* ermittelt, welche topologische Situation für die gegebenen räumlichen Objekte vorliegt.

graue Zellen treffen, was bei tatsächlichem Vorliegen einer *touch*-Beziehung immer der Fall ist. Daher gilt im Sinne des positivistischen Ansatzes *touch* als erfüllt, wenn es bei Erreichen einer zuvor festgelegten Verfeinerungsstufe nicht falsifiziert wurde und weder *disjoint* noch *overlap* erfüllt sind.

Wie bereits erwähnt, werden bei dieser Vorgehensweise Fehler in Kauf genommen. Zwei Objekte, die sich „knapp“ überlappen bzw. zwischen denen ein „kleiner“ Spalt existiert, können irrtümlich als sich berührend eingestuft werden. Die Größe dieser nicht erkannten Überlappung bzw. des nicht erkannten Spalts lässt sich jedoch beschränken: Sie entspricht der Diagonale einer Zelle auf der untersten Rekursionsebene. Es bleibt damit dem Anwender bzw. Systemadministrator überlassen, durch eine sinnvolle Wahl der Verfeinerungstiefe die erforderliche Genauigkeit zu gewährleisten.

Das Problem der Nicht-Erfüllbarkeit tritt auch für das Prädikat *equal* auf, hier für Kombinationen von beliebig-dimensionalen Operanden. Da auch hier bei tatsächlich vorliegender Gleichheit der beiden Operanden bei beliebiger Verfeinerung immer Paare mit *grauen* bzw. *schwarz-weißen* Zellen existieren, können negative Regeln nicht angewendet werden, die zu der notwendigen Besetzung der 9IM-Matrix führen würden. *equal* konkurriert hierbei mit *overlap*, *inside* und *contain*, die sich jeweils von der *equal*-Situation nur durch eine beliebig kleine Verschiebung bzw. Skalierung unterscheiden. Entsprechend dem oben beschriebenen positivistischen Ansatz gilt daher *equal* als erfüllt, wenn bei Erreichen der maximalen Verfeinerungsstufe keines der anderen Prädikate erfüllt ist.

Sind die übergebenen Operanden ausschließlich niedrigdimensionale Objekte (Punkte, Linien, Flächen), so kann bis auf *disjoint* keines der Prädikate auch bei noch so starker Verfeinerung bestätigt werden. Ein Grund hierfür liegt darin, dass das Innere dimensionsreduzierter Objekte auch bei beliebiger Verfeinerung immer durch *schwarz-weiße* Zellen repräsentiert wird. Die dadurch entstehende Überlappungen von *schwarz-weißen* Zellen, wie sie bei tatsächlichem Vorliegen einer *equal*, *contain*, *within*, *touch* oder *overlap*-Situation auftritt, lässt keine Aussage darüber zu, ob die Schnittmenge $A^\circ \cap B^\circ$ leer oder nicht-leer ist. Ebenso sind alle übrigen $\varnothing$-Einträge in der 9IM-Matrix von *equal*, *contain* und *within* nicht zu realisieren, solange sich *graue* bzw. *schwarz-weiße* Zellen treffen, was bei tatsächlichem Vorliegen eines der Prädikate der Fall ist.

Es lässt sich daher auch bei beliebiger Verfeinerung bis auf *disjoint* keines der Prädikate bestätigen, da weder die betreffende 9IM-Matrix erfüllt werden kann, noch alle anderen widerlegt werden können. Daher ist die Anwendung des positivistischen Prinzips in Kombination mit der in Abb. 8.24 gezeigten Hierarchie von essentieller Bedeutung. Liegt beispielsweise tatsächlich eine *touch*-Situation vor, werden bei genügend großer Verfeinerung *equal*, *contain*, *within* und *contain* widerlegt, *disjoint* und *overlap* hingegen nicht. Im Sinne des positivistischen Ansatz geht man in diesem Fall davon aus, dass das „präzisere“ Prädikat erfüllt ist, also *touch* anstelle von *disjoint* oder *overlap*.

Letzteres bedeutet, dass bei Vorliegen zweier niedrigdimensionaler Operanden die Algorithmen *testTopoPredicate* und *whichTopoPredicate* lediglich im Fall einer *disjoint*-Beziehung vor Erreichen der maximalen Genauigkeit abbrechen können.

Strikte topologische Operatoren

Für bestimmte Anwendungsszenarien ist es notwendig, dass der Nutzer absolute Sicherheit über das Vorliegen einer topologischen Situation erlangen kann. Aus diesem Grund werden *strikte* topologische Operatoren eingeführt, die nur dann *true* zurückgeben, wenn das zugrundeliegende topologische Prädikat bis zum Erreichen der vorgegebenen maximalen Verfeinerungsstufe eindeutig bestätigt werden kann, und nur dann *false* zurückgeben, wenn dieses Prädikat eindeutig widerlegt ist. In allen Fällen von Uneindeutigkeit geben die strikten topologischen Operatoren den Wert *unknown* zurück.

Aus den Darstellungen am Ende des vorangegangenen Abschnitts wird klar, dass im Fall zweier Operanden vom Typ *Body* lediglich die Prädikate *overlap, disjoint* und *contain/within* sowie im Fall von zwei dimensionsreduzierten Operanden ausschließlich *disjoint* als strikte topologische Operatoren implementierbar sind, die *true* zurückgeben können. Bei allen anderen Prädikaten wird vom zugehörigen strikten Operator entweder *false* oder *unknown* zurückgegeben.

Bei den strikten topologischen Operatoren wird die Forderung nach Vollständigkeit aufgegeben, nach der für jede auftretende Konstellation zweier räumlicher Objekte mindestens eines der topologischen Prädikate bestätigt wird. Entsprechend gibt die strikte Implementierung von *whichTopoPredicate* den Wert *unknown* zurück, wenn eine eindeutige Zuordnung eines topologischen Prädikats nicht möglich ist. Dies ist zum Beispiel bei tatsächlichem Vorliegen einer *touch*-Konstellation der Fall.

Die Konsequenzen dieser Festlegungen sollen anhand der Beispielsituationen der Abb. 8.27 bis 8.40 erläutert werden: Wenn ein zu testendes Prädikat bei Erreichen der maximalen Verfeinerungsstufe einen Stern aufweist, wird von dem betreffenden strikten Operator *unknown* zurückgegeben.

Bei Vorliegen einer *disjoint*- oder *overlap*-Situation entsprechend Abb. 8.27 bzw. 8.28 wird beispielsweise vom strikten *touch*-Operator anders als vom nicht-strikten Äquivalent nicht *true* zurückgegeben, sondern *unknown*. Das Verhalten von *equal*, *contain* und *within* ändert sich hingegen nicht. Sie geben in der strikten Implementierungsvariante ebenso wie in der nicht-strikten *false* zurück. Eine Anfrage von *whichTopoPredicate* wird erst ab einer maximalen Verfeinerungsstufe von 4 *disjoint* liefern, zuvor jedoch *unknown*.

Die *touch*-Situation von Abb. 8.29 kann von einer oktalbaumbasierten Implementierung nie ganz aufgelöst werden. Entsprechend gibt die strikte Implementierung des *touch*-Operators bei beliebiger maximaler Verfeinerung *unknown* zurück. Die Anwendung des *whichTopoPredicate*-Operators liefert ebenfalls unabhängig von der gewählten maximalen Verfeinerung den Wert *unknown*. Die Operatoren *equal*,

contain und *within* werden jedoch bereits ab einer maximalen Verfeinerungsstufe von 1 *false* zurückgeben.

Im Fall des Vorliegens der *contain* bzw. *within*-Konstellation von Abb. 8.30 bzw. 8.31 gibt die strikte Variante des *contain* bzw. *within*-Operators erst bei einer maximalen Verfeinerungsstufe von 4 den Wert *true* zurück. Auch hier zeigt sich ein Unterschied zur Vorgehensweise bei der nicht-strikten Implementierung der topologischen Prädikate, bei der durch Anwendung des positivistischen Prinzips und der gewählten Prädikatenhierarchie bereits vor Erreichen von Ebene 4 *contain* bzw. *within* gelten. Ebenso wie *contain* und *within* gibt die strikte Variante von *whichTopoPredicate* bei einer maximalen Verfeinerungstufe kleiner als 4 *unknown* zurück.

equal-Situationen (Abb. 8.32) können ebenso wie *touch*-Situationen von einer oktalbaumbasierten Implementierung nie ganz aufgelöst werden. Die strikte Variante des *equal*-Operators wird daher ebenso wie *whichTopoPredicate* unabhängig von der gewählten maximalen Verfeinerungsstufe immer *unknown* zurückgeben.

Diese Betrachtungen gelten analog für topologische Beziehungen zwischen dimensionsreduzierten Objekten (Abb. 8.36 bis 8.40). Allerdings ist hier die Situation dahingehend verschärft, dass bis auf *disjoint* keines der Prädikate eindeutig bestätigt werden kann. Von allen anderen strikten Operatoren wird daher in der Regel *unknown* und nur in seltenen Fällen *false* zurückgegeben. Ähnliches gilt für *whichTopoPredicate*, das nur bei *disjoint*-Situationen eine eindeutige Antwort liefert und sonst *unknown* zurückgibt.

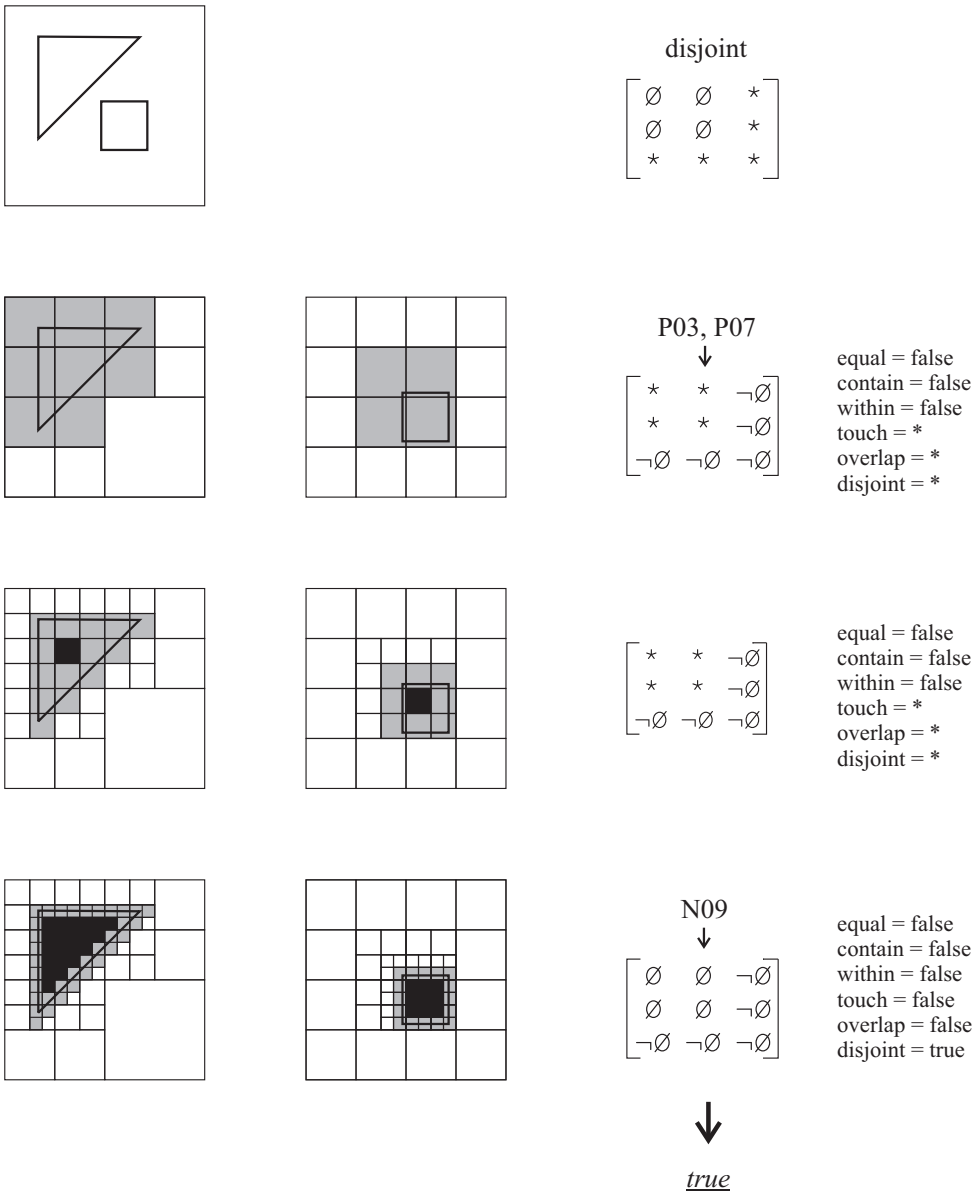


Abbildung 8.27: 2D-Darstellung der Untersuchung einer *disjoint*-Situation für nicht-dimensionsreduzierte Objekte auf den Ebenen 2 bis 4. Die *disjoint*-Situation für 2D-Regionen wird auf Ebene 4 korrekt aufgelöst. Entsprechend der Prädikatenhierarchie gilt auf den vorhergehenden Ebenen das Prädikat *touch*. Das Beispiel gilt analog im 3D-Raum für *Body*-Objekte.

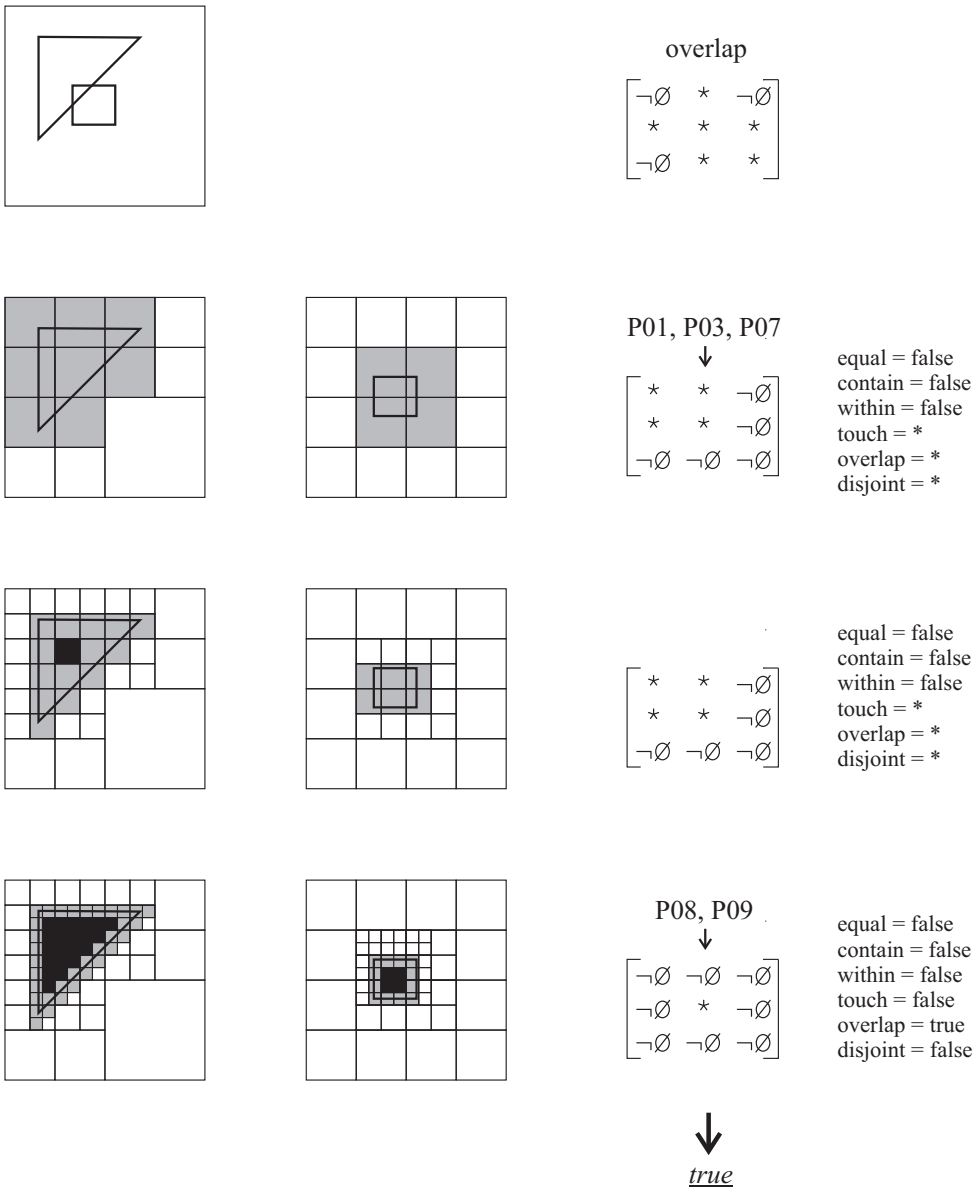


Abbildung 8.28: 2D-Darstellung der Untersuchung einer *overlap*-Situation für nicht-dimensionsreduzierte Objekte auf den Ebenen 2 bis 4. Die *overlap*-Situation für 2D-Regionen wird auf Ebene 4 korrekt aufgelöst. Entsprechend der Prädikatenhierarchie gilt auf den vorhergehenden Ebenen das Prädikat *touch*. Das Beispiel gilt analog im 3D-Raum für *Body*-Objekte.

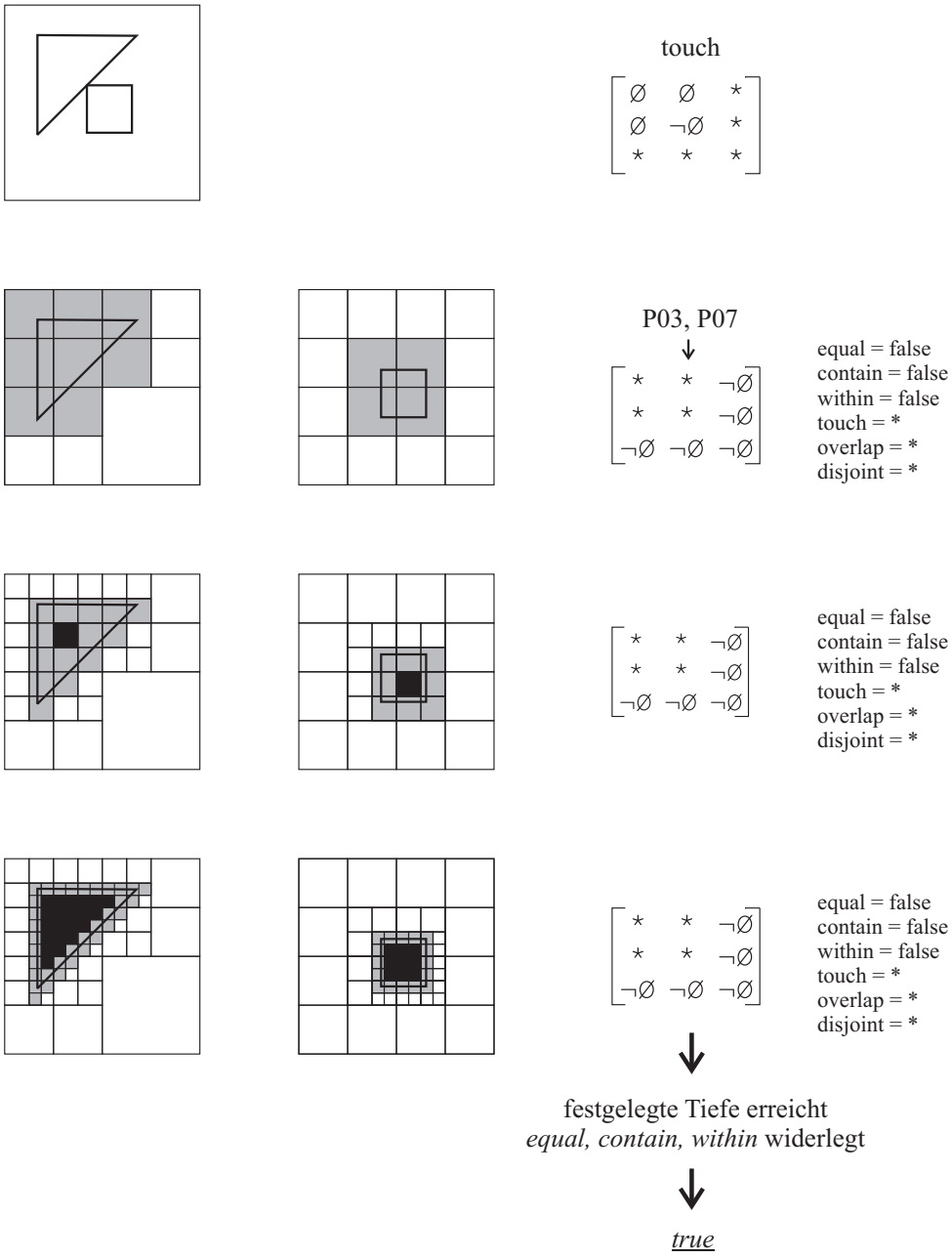


Abbildung 8.29: 2D-Darstellung der Untersuchung einer *touch*-Situation für nicht-dimensionsreduzierte Objekte auf den Ebenen 2 bis 4. Auch bei beliebiger Verfeinerung werden *touch*-Situationen nie ganz aufgelöst, da sie mit *overlap* und *disjoint* konkurrieren. Daher gilt *touch* bei Erreichen der maximalen Verfeinerung als erfüllt, wenn die in der Hierarchie darüber liegenden Prädikate *equal*, *contain* und *within* widerlegt werden konnten. Dies gilt analog im 3D-Raum für *Body*-Objekte.

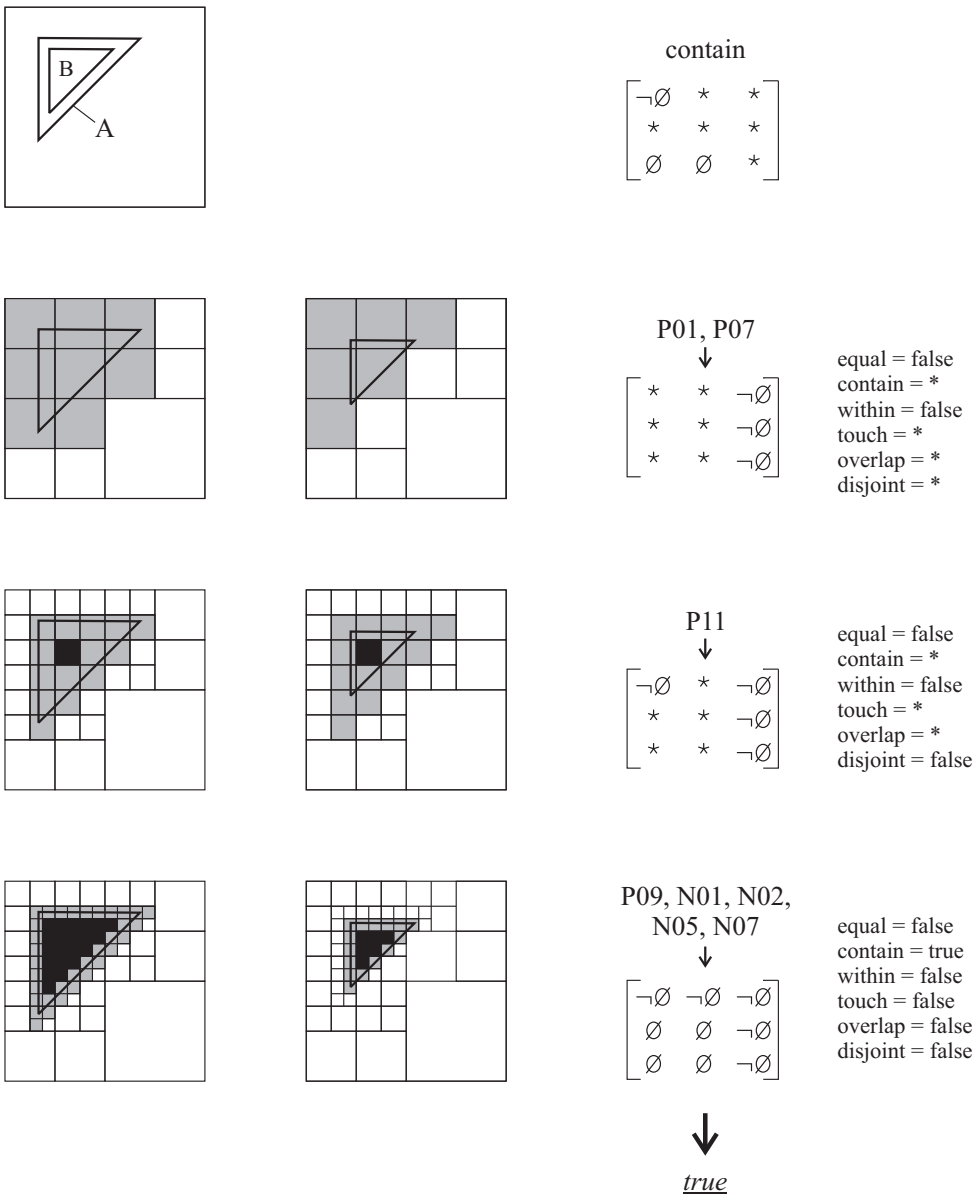


Abbildung 8.30: 2D-Darstellung der Untersuchung einer *contain*-Situation für nicht-dimensionsreduzierte Objekte auf den Ebenen 2 bis 4. Die *contain*-Situation für 2D-Regionen wird auf Ebene 4 korrekt aufgelöst. Dank des positivistischen Ansatzes in Verbindung mit der Prädikatenhierarchie gilt *contain* jedoch bereits auf den vorhergehenden Ebenen. Das Beispiel gilt analog im 3D-Raum für *Body*-Objekte.

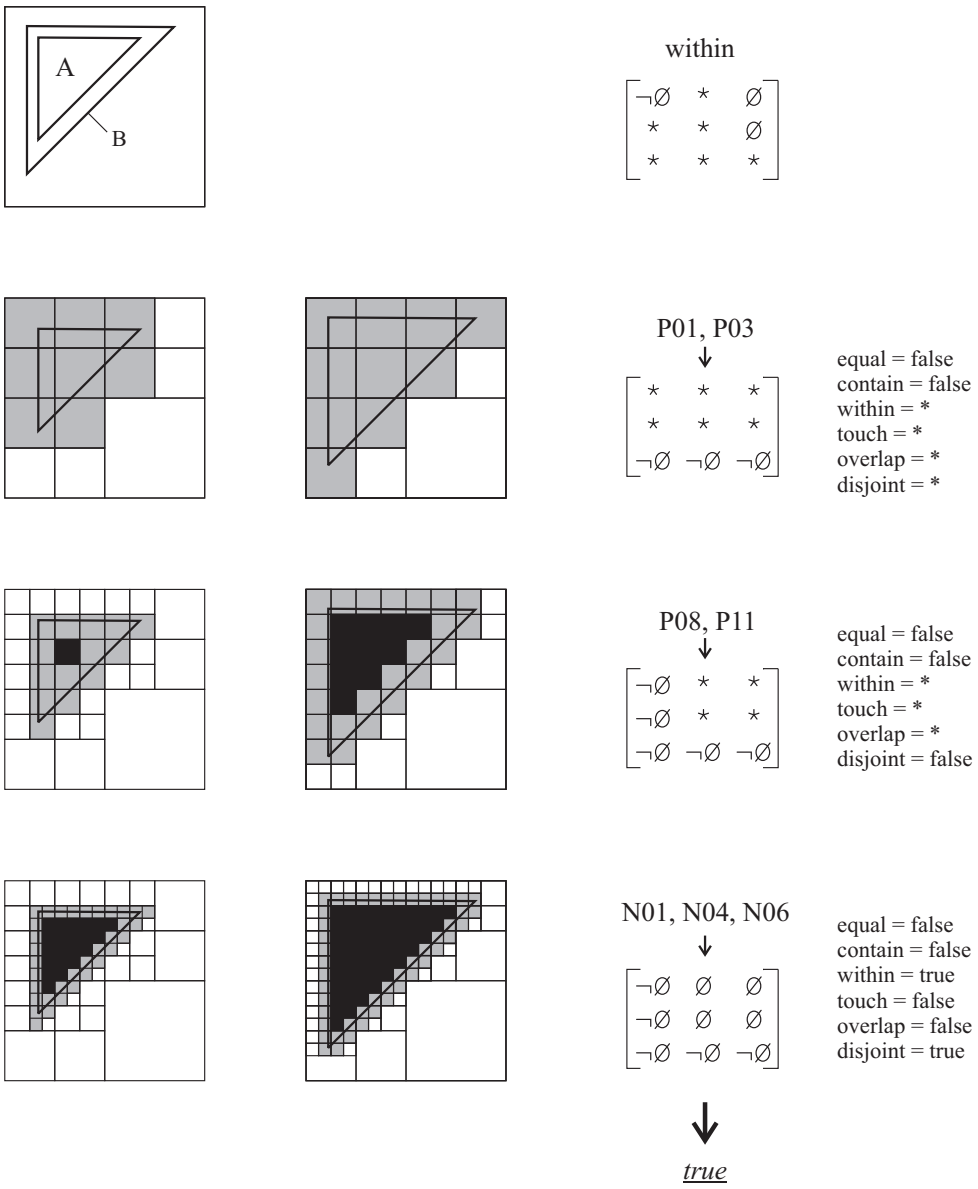


Abbildung 8.31: 2D-Darstellung der Untersuchung einer *within*-Situation für nicht-dimensionsreduzierte Objekte auf den Ebenen 2 bis 4. Die *within*-Situation für 2D-Regionen wird auf Ebene 4 korrekt aufgelöst. Dank des positivistischen Ansatzes in Verbindung mit der Prädikatenhierarchie gilt *within* jedoch bereits auf den vorhergehenden Ebenen. Das Beispiel gilt analog im 3D-Raum für *Body*-Objekte.

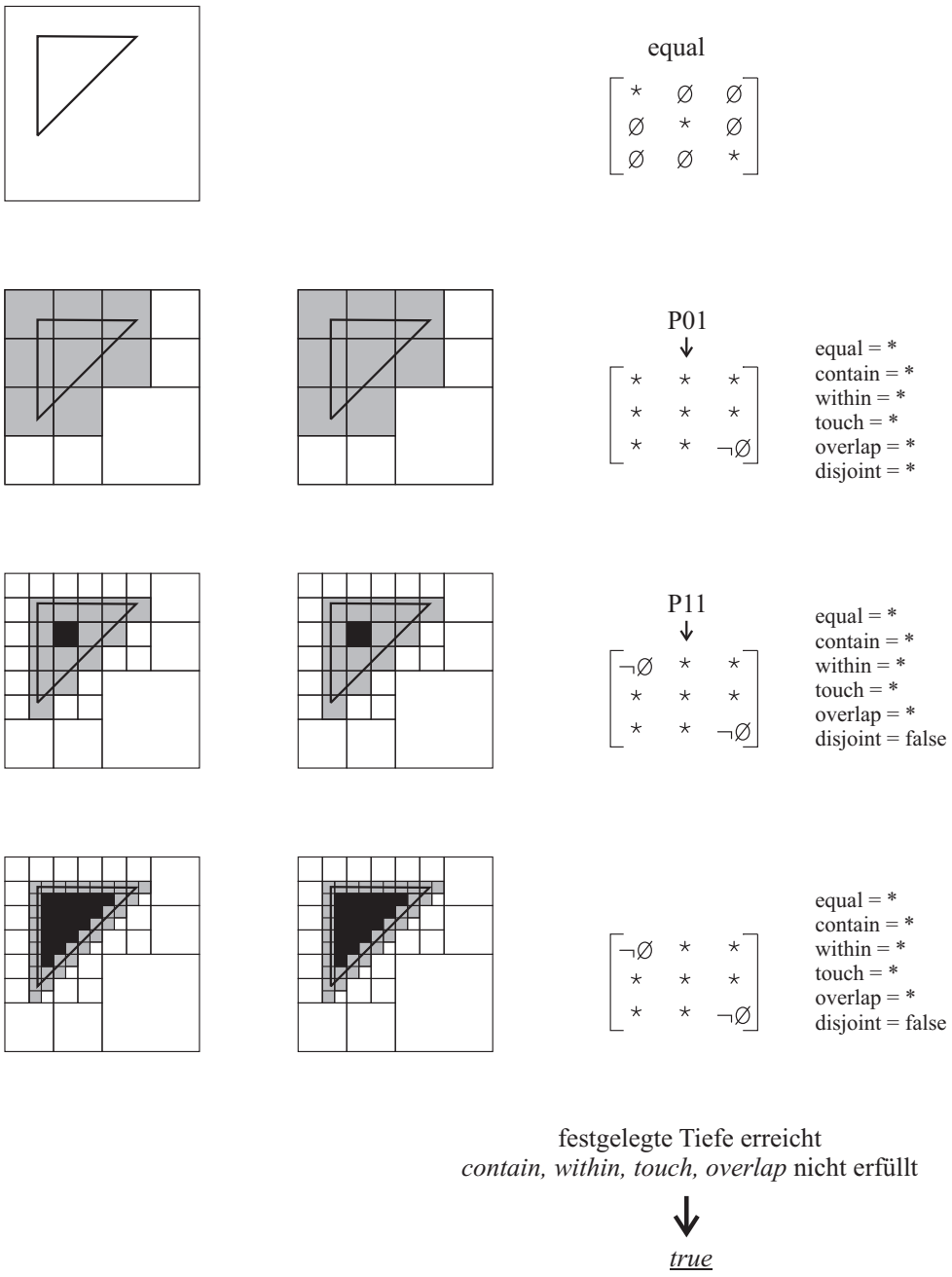


Abbildung 8.32: 2D-Darstellung der Untersuchung einer *equal*-Situation für nicht-dimensionsreduzierte Objekte auf den Ebenen 2 bis 4. Auch bei beliebiger Verfeinerung werden *equal*-Situationen nie ganz aufgelöst, da sie mit *contain*, *within*, *touch* und *overlap* konkurrieren. Daher gilt *equal* bei Erreichen der maximalen Verfeinerung als erfüllt, wenn kein anderes Prädikat bestätigt werden kann. Dies gilt analog im 3D-Raum für *Body*-Objekte.

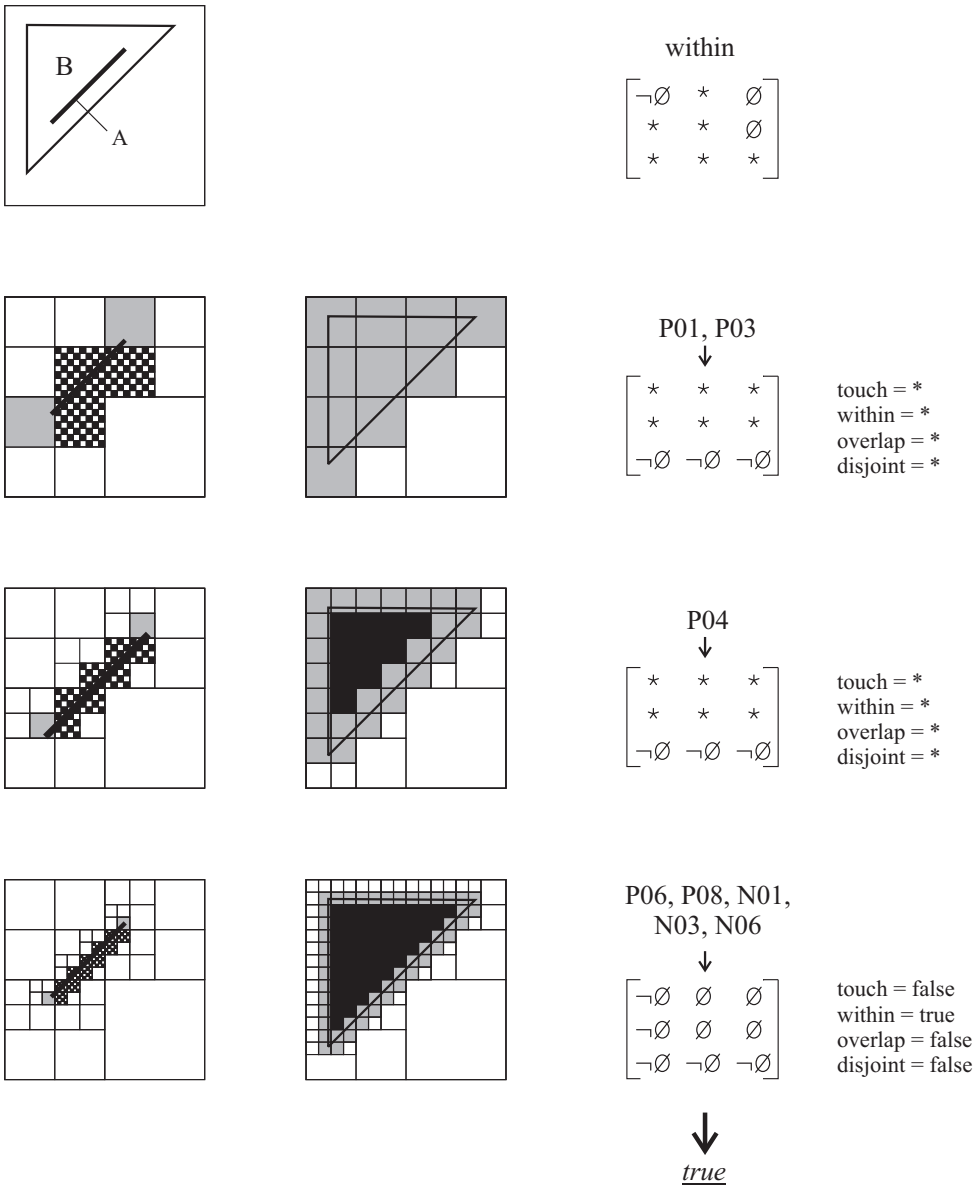


Abbildung 8.33: 2D-Darstellung der Untersuchung einer *within*-Situation zwischen einem dimensionsreduzierten und einem nicht-dimensionsreduzierten Objekt. Die *within*-Situation wird auf Ebene 4 korrekt aufgelöst. Entsprechend der Prädikatenhierarchie gilt auf den vorhergehenden Ebenen das Prädikat *touch*. Das Beispiel gilt analog im 3D-Raum für die Beziehungen zwischen *Surface-, Line-* oder *Point-* und *Body*-Objekten.

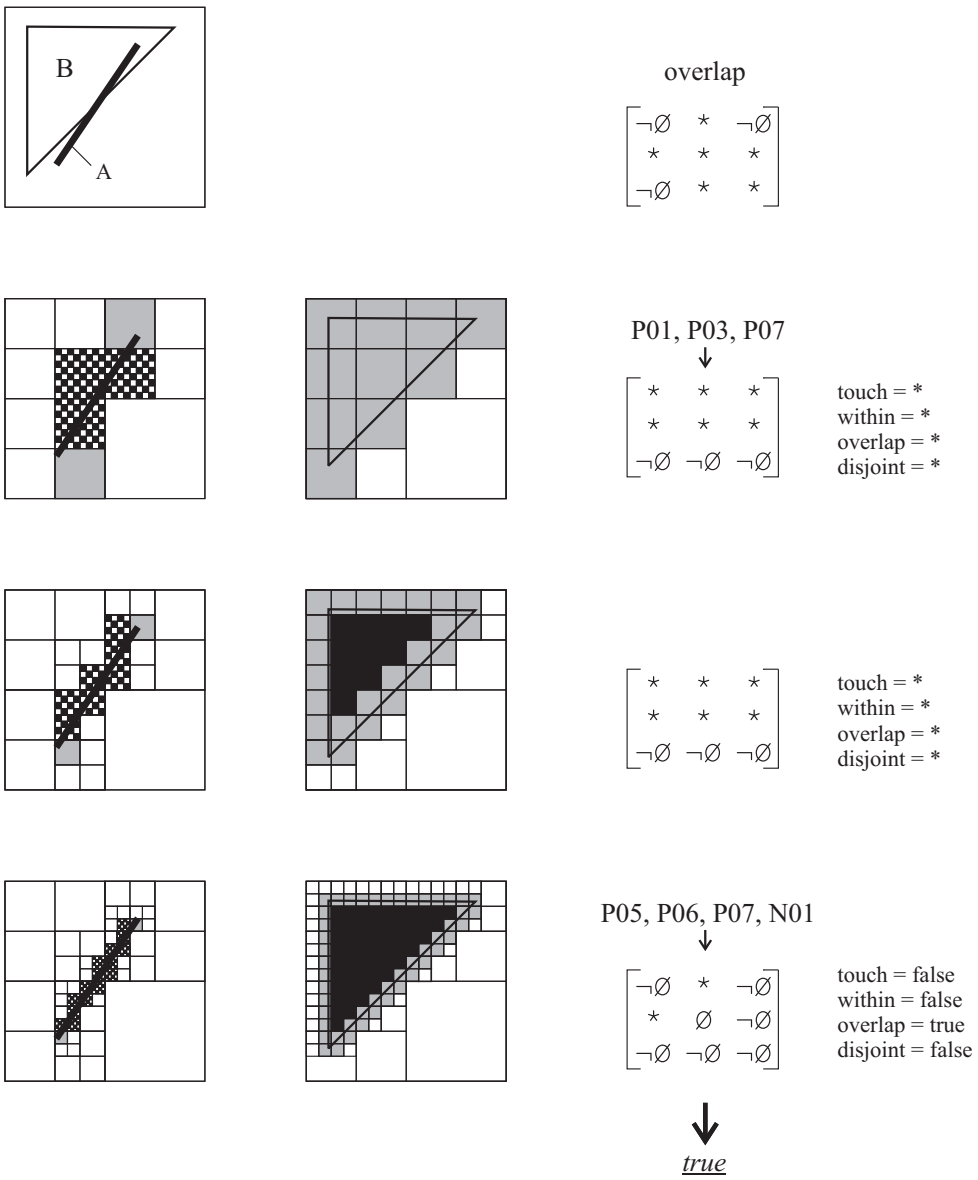


Abbildung 8.34: 2D-Darstellung der Untersuchung einer *overlap*-Situation zwischen einem dimensionsreduzierten und einem nicht-dimensionsreduzierten Objekt. Die *overlap*-Situation wird auf Ebene 4 korrekt aufgelöst. Entsprechend der Prädikatenhierarchie gilt auf den vorhergehenden Ebenen das Prädikat *touch*. Das Beispiel gilt analog im 3D-Raum für die Beziehungen zwischen *Surface*- oder *Line*- und *Body*-Objekten.

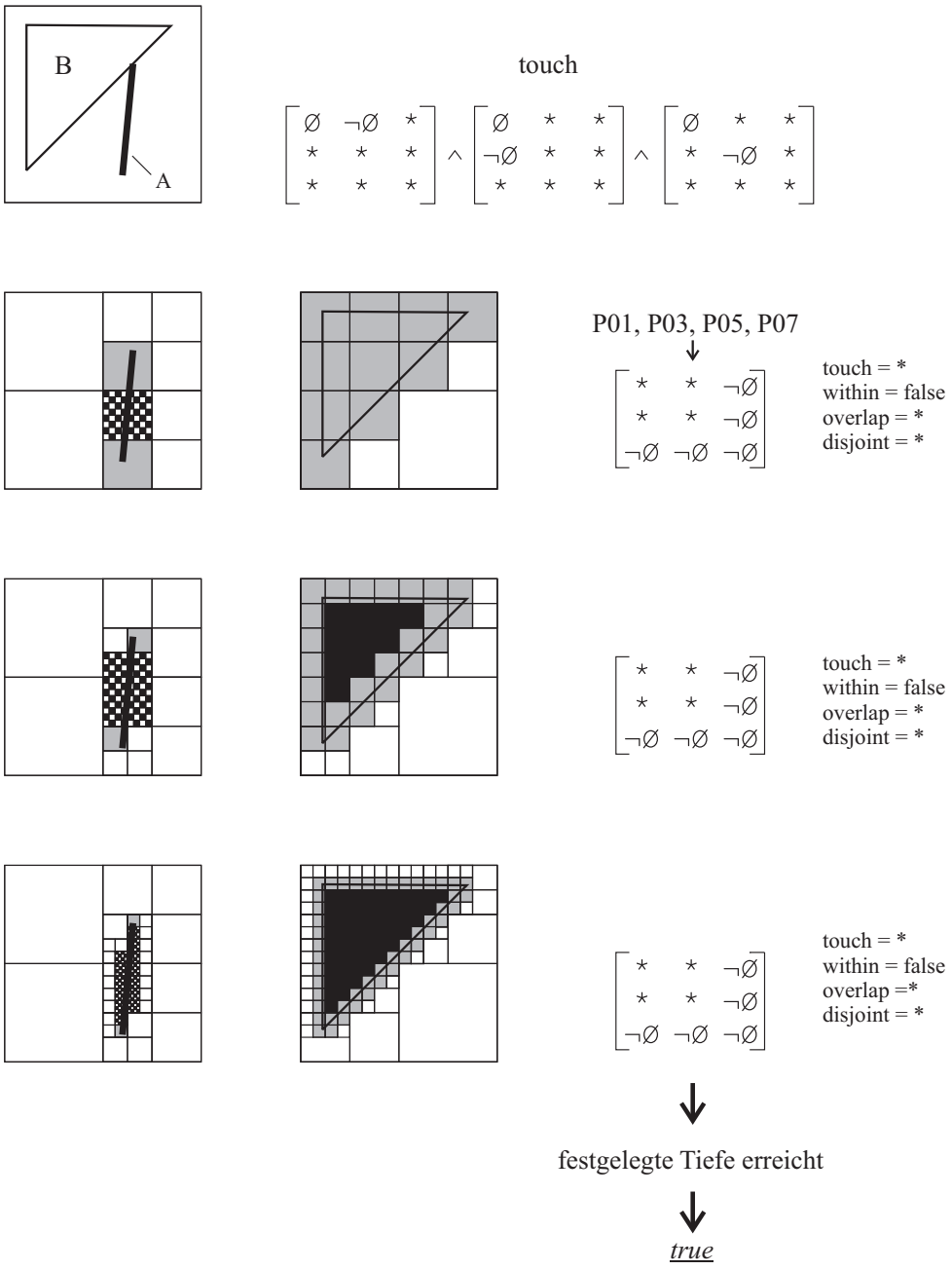


Abbildung 8.35: 2D-Darstellung der Untersuchung einer *touch*-Situation zwischen einem dimensionsreduzierten und einem nicht-dimensionsreduzierten Objekt. Auch bei beliebiger Verfeinerung werden *touch*-Situationen nie ganz aufgelöst, da sie mit *overlap* bzw. *within* und *disjoint* konkurrieren. Daher gilt *touch* bei Erreichen der maximalen Verfeinerung als erfüllt, wenn keines der anderen Prädikate bestätigt werden kann. Dies gilt analog im 3D-Raum für die Beziehungen zwischen *Surface-, Line-* oder *Point-* und *Body*-Objekten.

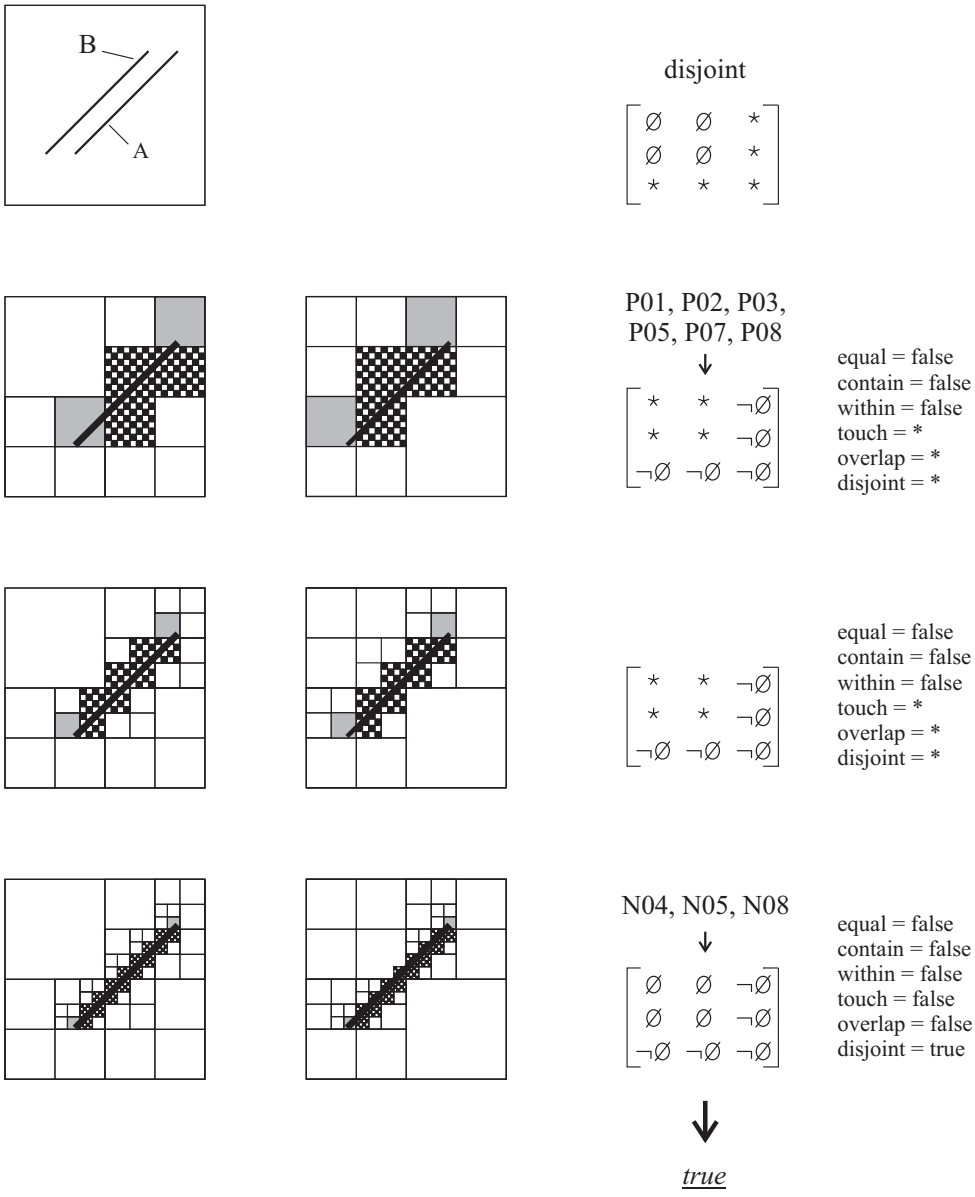


Abbildung 8.36: 2D-Darstellung der Untersuchung einer *disjoint*-Situation für dimensionsreduzierte Objekte auf den Ebenen 2 bis 4. Die *disjoint*-Situation wird auf Ebene 4 korrekt aufgelöst. Entsprechend der Prädikatenhierarchie gilt auf den vorhergehenden Ebenen das Prädikat *touch*. Das Beispiel gilt analog im 3D-Raum für beliebige Kombinationen aus *Surface-, Line-* und *Point*-Objekten.

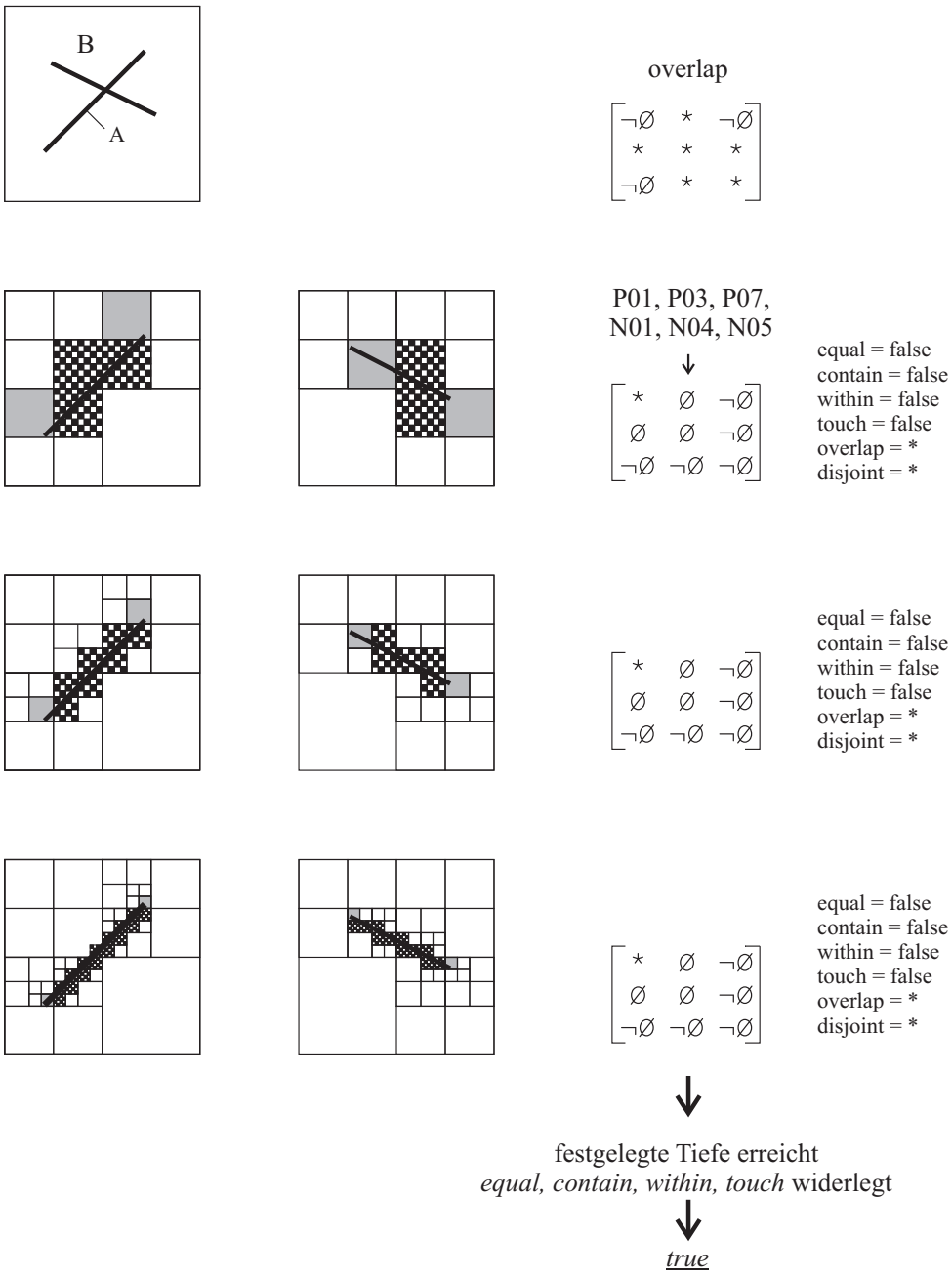


Abbildung 8.37: 2D-Darstellung der Untersuchung einer *overlap*-Situation für dimensionsreduzierte Objekte auf den Ebenen 2 bis 4. *overlap*-Situationen dimensionsreduzierter Objekte werden nie aufgelöst. Entsprechend wird der positivistische Ansatz in Verbindung mit der Hierarchie der topologischen Prädikate von Abb. 8.23 angewandt: Bei Erreichen der maximalen Verfeinerungsstufe gilt *overlap* als erfüllt, wenn *equal*, *contain*, *within* und *touch* widerlegt sind. Das Beispiel gilt analog im 3D-Raum für beliebige Kombinationen aus *Surface-*, *Line-* und *Point*-Objekten.

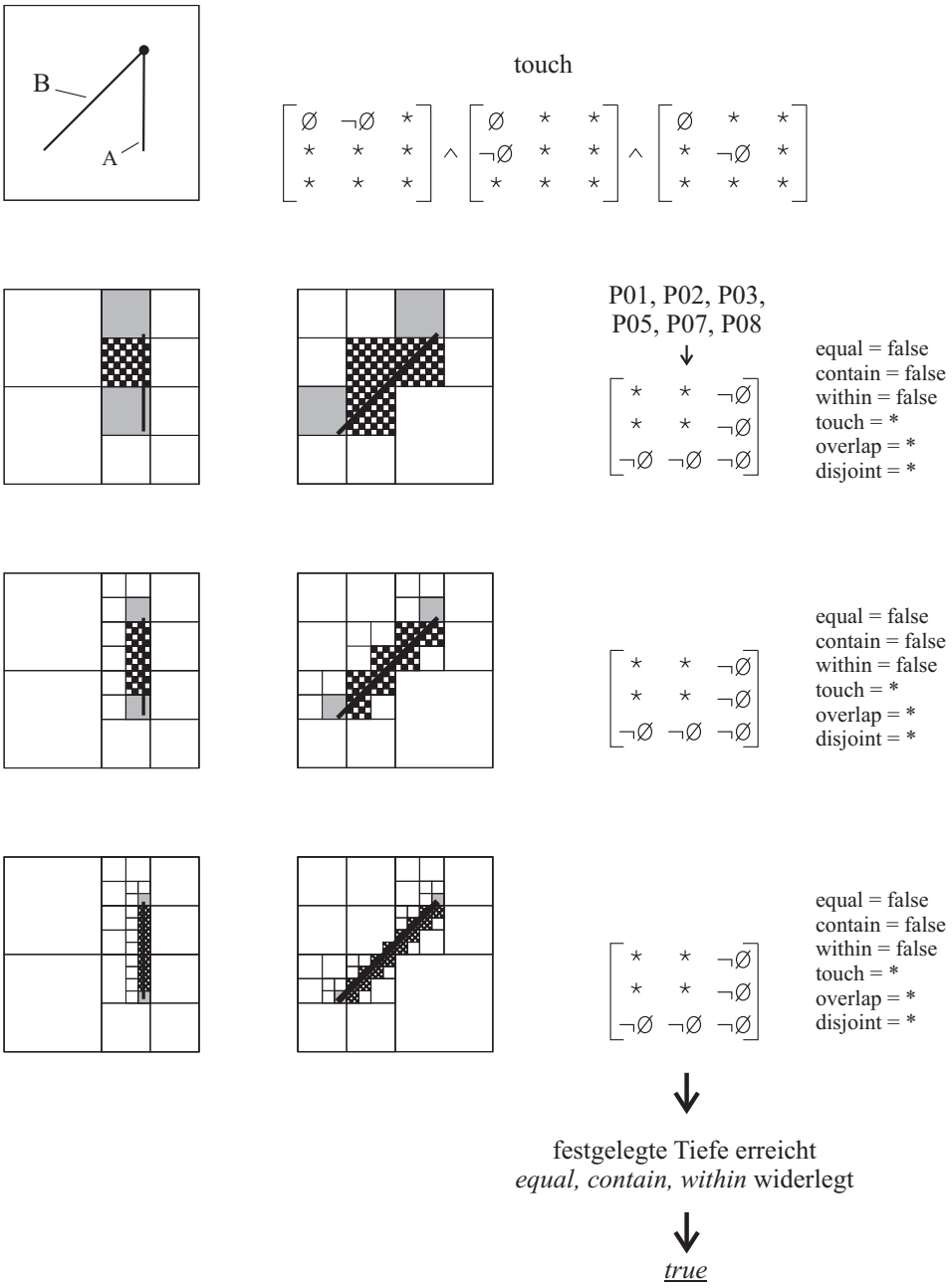


Abbildung 8.38: 2D-Darstellung der Untersuchung einer *touch*-Situation für dimensionsreduzierte Objekte auf den Ebenen 2 bis 4. *touch*-Situationen dimensionsreduzierter Objekte werden nie aufgelöst. Entsprechend wird der positivistische Ansatz in Verbindung mit der Hierarchie der topologischen Prädikate von Abb. 8.23 angewandt: Bei Erreichen der maximalen Verfeinerungsstufe gilt *touch* als erfüllt, wenn *equal*, *contain* und *within* widerlegt sind. Dies gilt analog im 3D-Raum für beliebige Kombinationen aus *Surface-, Line-* und *Point*-Objekten.

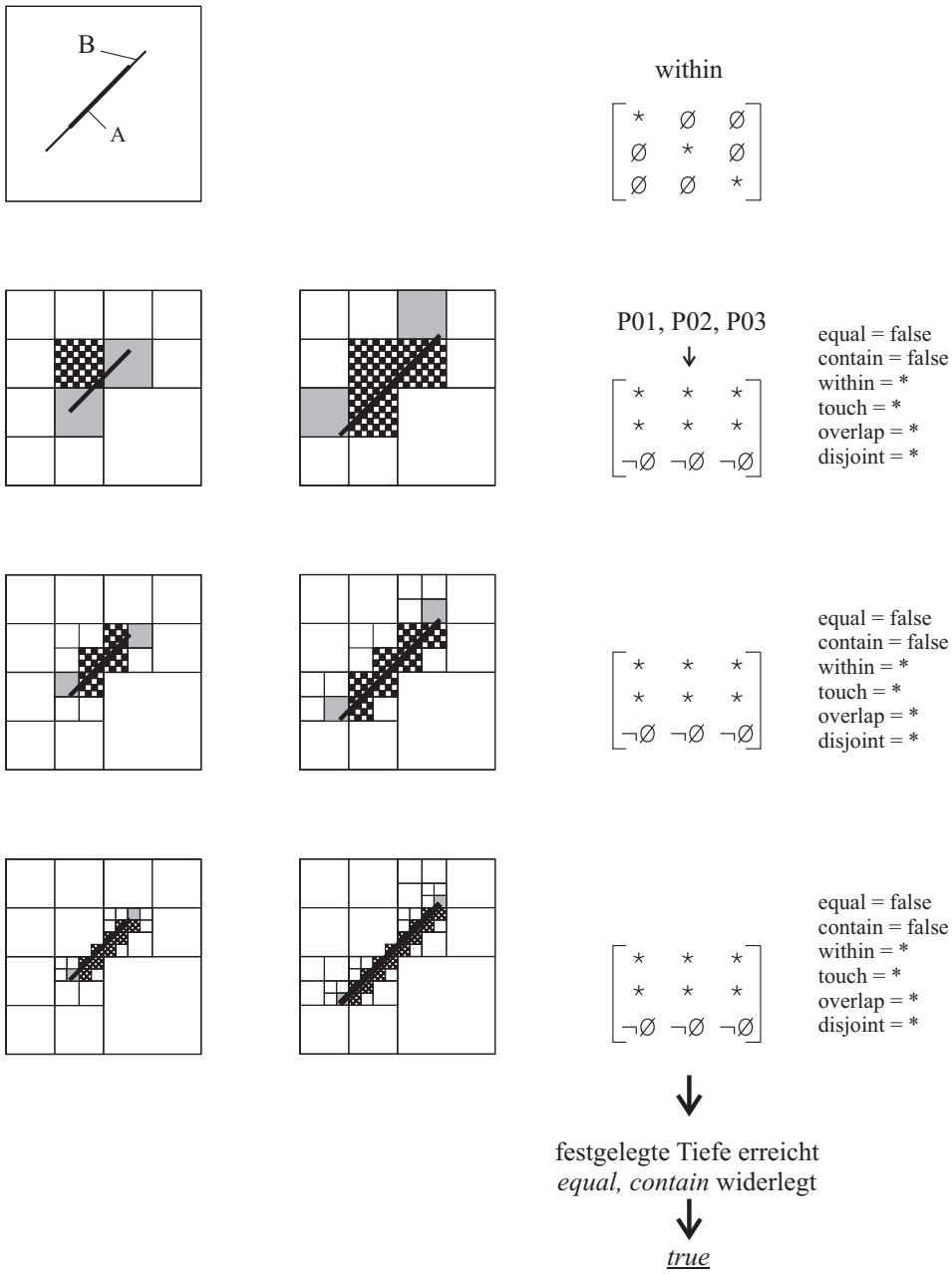


Abbildung 8.39: 2D-Darstellung der Untersuchung einer *within*-Situation für dimensionsreduzierte Objekte auf den Ebenen 2 bis 4. *within*-Situationen dimensionsreduzierter Objekte werden nie aufgelöst. Entsprechend wird der positivistische Ansatz in Verbindung mit der Hierarchie der topologischen Prädikate von Abb. 8.23 angewandt: Bei Erreichen der maximalen Verfeinerungsstufe gilt *within* als erfüllt, wenn *equal* und *contain* widerlegt sind. Dies gilt analog im 3D-Raum für beliebige Kombinationen aus *Surface-, Line-* und *Point-*Objekten.

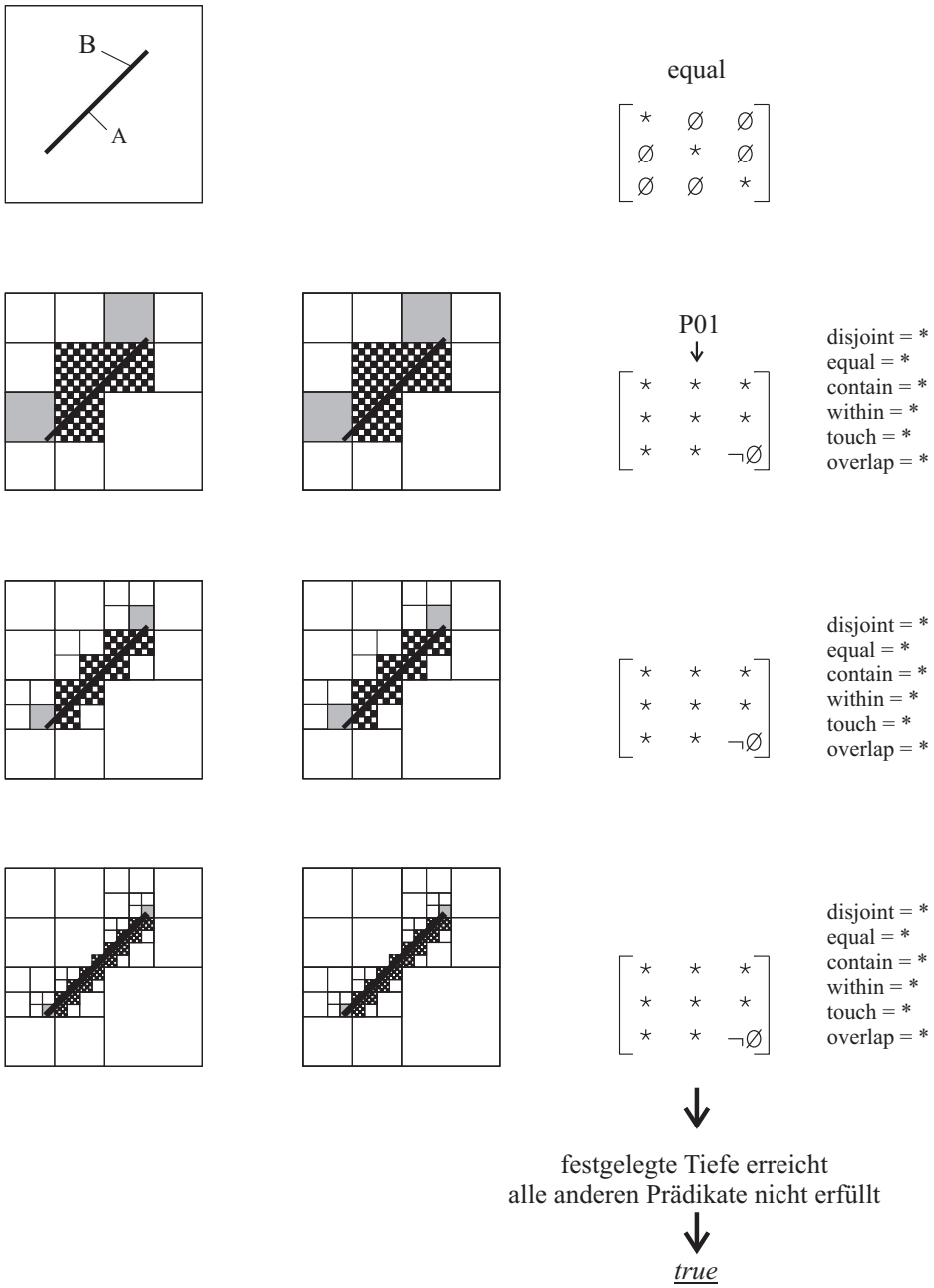


Abbildung 8.40: 2D-Darstellung der Untersuchung einer *equal*-Situation für dimensionsreduzierte Objekte auf den Ebenen 2 bis 4. *equal*-Situationen dimensionsreduzierter Objekte werden nie aufgelöst. Entsprechend wird der positivistische Ansatz in Verbindung mit der Hierarchie der topologischen Prädikate von Abb. 8.23 angewandt: Bei Erreichen der maximalen Verfeinerungsstufe gilt *within* als erfüllt, wenn keines der beiden Prädikate *equal* und *contain* verifiziert werden konnte. Dies gilt analog im 3D-Raum für beliebige Kombinationen aus *Surface-, Line-* und *Point*-Objekten.

Kapitel 9

Objekt-relationale Datenbanken als Basis räumlicher Analysefunktionalität

In diesem Kapitel wird gezeigt, wie mit Hilfe von Datenbanktechnologie eine räumliche Anfragesprache für digitale Bauwerksmodelle umgesetzt werden kann. Datenbanksysteme sind dank der Bereitstellung generischer Services wie Transaktionen, Nebenläufigkeitskontrolle, Integritätssicherung und Rechteverwaltung ein äußerst geeignetes Werkzeug zur Verwaltung von Gebäudemodellen im Kontext verteilt-kooperativen Engineerings.

Wie in diesem Kapitel durch einen ausführlichen Vergleich der verschiedenen Datenbanktechniken gezeigt wird, sind im besonderen *objekt-relationale* Datenbanksysteme durch die von ihnen zur Verfügung gestellte Erweiterbarkeit des Typsystems, die Unterstützung von serverseitigem Programmcode und die standardisierte und weit verbreiteten Anfragesprache SQL für die Umsetzung der zuvor beschriebenen räumlichen Algebra geeignet.

9.1 Datenbankverwaltungssysteme

9.1.1 Definition

Datenbankverwaltungssysteme (engl. *Database Management Systems*, DBMS) bilden eine gute Grundlage für die Unterstützung von asynchronem bzw. reziprokem kollaborativen Engineering. Sie können große Mengen von Daten verwalten und bieten dabei entscheidende Vorteile gegenüber dateibasierten Lösungen: Sie stellen logische und physische Datenunabhängigkeit, Transaktionen, Nebenläufigkeitskontrolle, Integritätsprüfung, Wiederherstellung, Sicherheit und Verteilung zur Verfügung (Date, 2003). Daneben bieten sie mit einer deklarativen Anfrage-

sprache ein mächtiges Instrument zur Definition von Teilmengen bzw. zur effizienten Suche in großen Datenbeständen.

Für eine Definition des Begriffs Datenbank (DB) ist es notwendig, sorgfältig zwischen den zu verwaltenden Daten (= Datenbasis) und dem System zu unterscheiden, welches diese Daten verwaltet (= Datenbankverwaltungssystem). Eine Datenbasis ist eine Sammlung von Datensätzen, die in einem Computer auf systematische Weise gespeichert wird, so dass ein Computerprogramm sie konsultieren kann, um Anfragen zu beantworten. Das Computerprogramm, das diese Datenbasis verwaltet und entsprechende Anfragen beantwortet, wird als Datenbankverwaltungssystem bezeichnet. Häufig wird, wenn der Kontext unzweideutig ist, der Begriff Datenbank (engl. *data base*) im Sinne der einen oder anderen Bedeutung verwendet.

Typischerweise hält das Datenbanksystem eine strukturelle Beschreibung der Datenbasis vor, das sogenannte *Schema*. Das Schema beschreibt die Struktur der Daten in der Datenbank und ihre Beziehungen untereinander. Das Datenmodell einer Datenbank legt fest, auf welche Weise die Struktur der Datenbasis im Schema modelliert werden kann.

Obwohl in der Fachwelt keine vollkommene Einigkeit diesbezüglich herrscht, soll im Rahmen dieser Arbeit davon ausgegangen werden, dass ein Datenbanksystem folgende zusätzliche Eigenschaften aufweist:

- es gewährleistet die Integrität und Qualität der Daten,
- es erlaubt den konkurrierenden Zugriff von mehreren Nutzern,
- es regelt Zugriffsrechte,
- es besitzt ein Schema, das die Struktur der Daten festlegt,
- es stellt eine Anfragesprache zu Verfügung.

Erst durch diese Merkmale heben sich DBMS deutlich von dateibasierten Lösungen zur Datenverwaltung ab. Die sich daraus ergebenden Vorteile sollen im nächsten Abschnitt diskutiert werden.

9.1.2 Vorteile gegenüber dateibasierter Datenverwaltung

Im Folgenden sollen die wesentlichen Vorteile erläutert werden, die sich bei Verwendung eines DBMS gegenüber einer dateibasierten Lösung ergeben:

Sicherung der Konsistenz, Vermeidung von Redundanz. Wenn Daten in isolierten Dateien vorgehalten werden, muss oft dieselbe Information mehrfach gespeichert werden, d.h. sie liegt redundant vor. Inkonsistenzen können nun genau dann auftreten, wenn die gespeicherte Information in einer Datei geändert, in der anderen aber beibehalten wird.

Einheitliches Datenmodell. Informationen aus einer Datei mit logisch verknüpften Informationen aus anderen Dateien zusammenzuführen ist nur unter großem Aufwand möglich. In einer Datenbank ist der gesamte Datensatz homo-

gen in einem gemeinsamen Datenmodell modelliert. Dies erlaubt dem Nutzer, auf sehr flexible Art logisch verknüpfte Daten zu extrahieren.

Nebenläufigkeitskontrolle. Derzeit verfügbare Dateisysteme bieten nur wenig oder gar keine Kontrollmechanismen für den Mehrbenutzerbetrieb. Wenn der gleichzeitige Zugriff mehrerer Nutzer nicht kontrolliert wird, kann es jedoch zu unerwünschten Effekten kommen. Ein einfaches Beispiel ist die gleichzeitige Modifikation eines Datensatzes durch zwei Nutzer. Ohne Nebenläufigkeitskontrolle werden die Änderungen des Nutzers, der zuerst die Daten zurückschreibt, von denen des zweiten überschrieben. Datenbanken verfügen über leistungsfähige Mechanismen zur Nebenläufigkeitskontrolle, um solche Anomalien zu verhindern.

Schutz vor Datenverlust. Bei Speicherung der Daten in isolierten Dateien ist es schwierig, nach einem Systemausfall einen gültigen Zustand der Gesamtinformationsmenge wiederherzustellen. Dateisysteme bieten lediglich die Möglichkeit eines periodischen Backups aller Dateien. Wenn es jedoch während der Bearbeitung einer Datei zu einem Systemausfall kommt, können Daten verloren gehen. Datenbankverwaltungssysteme besitzen hingegen eine leistungsfähige Wiederherstellungskomponente, die die Daten vor allen vorhersehbaren Systemausfällen schützt.

Sicherung der Integrität. In jedem Anwendungsgebiet gibt es eine Vielzahl von Integritätsbedingungen, die sich über mehrere Informationseinheiten erstrecken. Ein Beispiel aus dem universitären Bereich ist, dass Studenten alle Zulassungsvoraussetzungen (Pflichtpraktika etc.) erfüllt haben müssen, bevor sie zur Prüfung zugelassen werden. Wenn die Daten über mehrere Dateien verstreut vorliegen, ist es schwierig und aufwändig, ihre Integrität zu prüfen. Weiterhin sollte das System Modifikationen, die gegen eine der Integritätsregeln verstoßen, ablehnen. Während Dateisysteme keinerlei derartige Funktionalität aufweisen, werden in einem DBMS sogenannte Transaktionen (das sind aus Nutzersicht atomare Verarbeitungsvorgänge) nur dann ausgeführt, wenn sie die Datenbasis in einem konsistenten Zustand halten.

Zugangskontrolle und Rechteverwaltung. In der Regel gibt es in jedem Anwendungsgebiet eine Reihe sensitiver Daten zu denen der Zugang eingeschränkt sein sollte. Üblicherweise ist einer bestimmten Gruppe von Nutzern gestattet, diese Daten lediglich zu lesen, während einer anderen das Recht zur Modifikation eingeräumt wird.

Zwar bieten Dateisysteme üblicherweise eine Rechteverwaltung. Die zugewiesenen Rechte gelten jedoch immer für ganze Dateien und nicht für einzelne Daten innerhalb einer Datei, was für die meisten Anwendungsfälle als zu grobgranular einzuschätzen ist. Datenbanken besitzen hingegen eine sehr feingranulare Rechteverwaltung, bei denen verschiedenartigste Zugriffsrechte bis auf die Ebene einzelner Datensätze vergeben werden können.

9.2 Datenbankmodelle und ihre Eignung für die Umsetzung räumlicher Anfragen

Datenbankverwaltungssystem werden nach ihrer internen Struktur und dem zugrundeliegenden Ansatz zur Abbildung (Modellierung) einer Anwendungsdomäne klassifiziert. Das Ende der 1970er Jahre entwickelte *hierarchische Datenmodell* und das *Netzwerkdatenmodell* haben mittlerweile nur noch historische Bedeutung. Heute basiert die Mehrzahl der eingesetzten Datenbanken auf dem *relationalen Datenmodell*, dem *objektorientierten Datenmodell* oder dem *objekt-relationalen Datenmodell*. Diese drei Datenmodelle sollen im Folgenden vorgestellt und auf ihre Eignung für die Umsetzung einer räumlichen Anfragesprache untersucht werden.

9.2.1 Relationale Datenbanken

Das *Relationale Modell* wurde 1970 von Codd (Codd, 1970) eingeführt, um Datenbanksysteme unabhängig von einem bestimmten Einsatzgebiet zu machen. Inzwischen hat es sich zu dem bei weitem erfolgreichsten Datenbankmodell entwickelt – relationale Datenbanken beherrschen mehr als 80 Prozent des Datenbanken-Markts. Seinen Erfolg verdankt dieses Modell vor allem der sowohl mächtigen als auch vergleichsweise einfach erlernbaren Anfragesprache SQL, die auf der *Relationalen Algebra* beruht.

Die Relationale Algebra

In der Relationalen Algebra sind Operatoren definiert, die sich auf eine Menge von Relationen anwenden lassen, darunter solche zum Verknüpfen, Filtern und Umbenennen von Relationen. Die Ergebnisse aller Operationen sind ebenfalls Relationen. Aus diesem Grund bezeichnet man die Relationenalgebra als abgeschlossen.

Das grundlegende Element der Relationenalgebra ist die *Relation*, die wie folgt definiert ist: Gegeben seien n *Wertebereiche* (auch *Domänen* genannt) $D_1, D_2 \ldots, D_n$. Diese Domänen dürfen nur atomare Werte enthalten, die nicht strukturiert sein dürfen. Gültige Domänen sind beispielsweise Zahlen oder Zeichenketten. Eine Relation R ist definiert als eine Teilmenge des kartesischen Produkts (Kreuzprodukts) der n Domänen:

$$R \subseteq D_1 \times \cdots \times D_n \quad .$$

Ein Element der Menge R wird als *Tupel*, die einzelnen *Stellen* eines Tupels als *Attribute* bezeichnet. Allen Attributen einer Relation sind eindeutige Namen zugewiesen. Als Beispiel sollen im Folgenden die Relation *Wände* mit den Attributen *WandID, Dicke, Höhe, Breite*, die Relation *Decken* mit den Attributen *ID, Dicke, Länge, Breite*, die Relation *Material* mit den Attributen *MaterialID, Name, Festigkeit* sowie die Relation *bestehen_aus* mit den Attributen *MaterialID* und *WandID* dienen. Beispiele für Ausprägungen dieser Relationen zeigt Abb. 9.1.

Wände			
WandID	Dicke	Höhe	Breite
W247	24	250	310
W248	24	250	270
W249	12	220	240
W250	12	250	280
W251	24	250	300

Materialien		
MaterialID	Name	Festigkeit
M07	B15	15
M08	B25	25
M09	B35	35
M10	B45	45
M11	B55	55

Decken			
ID	Dicke	Länge	Breite
D24	25	250	310
D25	25	250	270
D26	20	220	240
D27	20	250	280
D28	23	250	300

bestehen_aus	
WandID	MaterialID
W247	M09
W248	M09
W249	M08
W250	M08
W251	M09

Abbildung 9.1: Beispiele für Ausprägungen der Relationen *Wände, Decken, Material* und *bestehen_aus*.

Die wesentlichen auf Relationen anwendbaren Operationen sind die *Selektion*, die *Projektion*, die *Vereinigung*, die *Differenz* und das *Kartesische Produkt*.

Bei der *Selektion* werden diejenigen Tupel einer Relation ausgewählt, die das sogenannte *Selektionsprädikat* erfüllen. Die Selektion wird durch das Symbol σ dargestellt mit dem Selektionsprädikat als Subskript. Ein Beispiel für die Selektion wäre

$$\sigma_{\text{Dicke}>24}(\text{Wände}),$$

das aus der Relation *Wände* diejenigen Tupel auswählt, bei denen das Attribut *Dicke* einen Wert größer als 24 aufweist.

Allgemein ist das Selektionsprädikat eine Formel F, die aufgebaut ist aus

- Attributnamen der Argumentrelation R oder Konstanten als Operanden,
- den arithmetischen Vergleichsoperatoren $=, <, \leq, >, \geq, \neq$ und
- den logischen Operatoren $\wedge$, $\vee$ und $\neg$.

Das Ergebnis der Selektion $\sigma_F(R)$ besteht dann aus allen Tupeln $t \in R$, für die die Formel F erfüllt ist.

Bei der *Projektion* werden einzelne Attribute aus der Argumentrelation extrahiert. Die Projektion wird mit dem Operatorsymbol Π dargestellt, das die Menge der Attributnamen im Subskript beinhaltet. Als Beispiel sei hier

$$\Pi_{\{\text{Dicke,Höhe}\}}(\text{Wände})$$

aufgeführt, das als Ergebnis die *Dicke*- und *Höhe*-Werte aller Tupel der *Wände*-Relation liefert.

Zwei Relationen mit gleichen Attributnamen und Attributtypen (Domänen) kann man durch die *Vereinigung* zu einer Relation zusammenfassen. Als Beispiel soll

$$\Pi_{\{\text{Dicke,Höhe}\}}(\text{Wände}) \cup \Pi_{\{\text{Dicke,Höhe}\}}(\text{Decken})$$

betrachtet werden. Bei dieser Operation werden zunächst die beiden Relationen *Wände* und *Decke* auf die gleiche Attributmenge reduziert. Danach wird die Vereinigung auf der Basis dieser temporären Relationen durchgeführt.

Die *Differenz* $R \setminus S$ zweier Relationen R und S ist definiert als die Menge der Tupel, die in R, aber nicht in S vorkommen.

Das *Kreuzprodukt* $R \times S$ enthält alle $|R| \cdot |S|$ möglichen Paare von Tupeln aus R und S. Das Schema der Ergebnisrelation ist die Vereinigung der Attribute aus R und S. Als Beispiel soll das Kreuzprodukt der Relationen *Wände* und *bestehen_aus* betrachtet werden. Die Ergebnisrelation besitzt die Attribute *WandID, Dicke, Höhe, Breite, MaterialID* und *WandID* und besteht aus $5 \cdot 5$ Tupeln.

Schließlich ist die *Umbenennung* als ein weiterer wichtiger Operator der Relationalen Algebra zu nennen. Mit seiner Hilfe kann einem Attribut oder einer Relation ein anderer Name zugewiesen werden. Die Umbenennung von Relationen ist u.a. dann notwendig, wenn dieselbe Relation mehrfach in einer Anfrage verwendet wird. Für die Umbenennung wird das Symbol ρ verwendet, wobei im Subskript der neue Name der Relation angegeben wird. Beispielsweise kann die Umbennung der Relation *Material* in *mat* wie folgt aufgeschrieben werden:

$$\rho_{\text{mat}}(\text{Material}) \quad .$$

Der ρ-Operator wird auch zur Umbenennung von Attributen einer Relation verwendet. Dazu werden im Subskript der neue Name, ein Pfeil und der Originalname angegeben, zum Beispiel

$$\rho_{\text{matID} \leftarrow \text{MaterialID}}(\text{bestehen\_aus}) \quad .$$

Mit den vorgestellten Operationen *Selektion, Projektion, Vereinigung, Differenz, Kreuzprodukt* und *Umbennung* ist die minimale Menge von Operationen gegeben, d.h. die Menge von Operationen, die notwendig ist, um alle Ausdrücke der Relationalen Algebra bilden zu können. Darüber hinaus gibt es weitere Operatoren der Relationalen Algebra, die zwar die Ausdruckskraft nicht weiter erhöhen, jedoch für eine kompaktere Darstellbarkeit sorgen. Einer der wichtigsten dieser Operatoren ist der *Verbund*.

Ein *Verbund* (engl. Join) bezeichnet die beiden hintereinander ausgeführten Operationen *Kartesisches Produkt* und *Selektion*. Er filtert die in der Regel irrelevanten Tupel des Kartesischen Produkts zweier Relationen heraus. Die Selektionsbedingung ist dabei üblicherweise ein Vergleich von Attributen A Θ B, wobei Θ ein passender Vergleichsoperator ist. Man bezeichnet den allgemeinen Verbund daher auch als Θ-Verbund („*Theta-Verbund*“). Spezialfälle des allgemeinen Verbundes sind der *Natürliche Verbund*, der *Equi-Join* und der *Semi-Join*.

Wände ⋈ bestehen_aus ⋈ Materialien						
WandID	Dicke	Höhe	Breite	MaterialID	Name	Festigkeit
W247	24	250	310	M07	B15	15
W248	24	250	270	M08	B25	25
W249	12	220	240	M09	B35	35
W250	12	250	280	M10	B45	45
W251	24	250	300	M11	B55	55

Abbildung 9.2: Das Ergebnis des Natürlichen Verbunds der drei Relationen *Wände*, *bestehen_aus* und *Materialien*.

Beim *Natürlichen Verbund* zweier Relationen wird zunächst das Kreuzprodukt gebildet, dann werden diejenigen Tupel selektiert, deren Attributwerte für gleich benannte Attribute der beiden Ausgangsrelationen gleich sind. Das Join-Prädikat Θ legt in diesem Fall also fest, dass gleich benannte Attribute den gleichen Wert aufweisen müssen. Wenn k die Anzahl der gleich benannten Attribute in R und S ist, R insgesamt $m + k$ Attribute $A_1, \ldots, A_m, B_1, \ldots, B_k$ und S insgesamt $n + k$ Attribute $B_1, \ldots, B_m, C_1, \ldots, C_n$ hat, dann hat der Verbund $R \bowtie S$ die Stelligkeit $m + n + k$ und ergibt sich wie folgt:

$$R \bowtie S = \Pi_{A_1,\ldots,A_m,R.B_1,\ldots,R.B_k,C_1,\ldots,C_k}(\sigma_{R.B_1=S.B1\wedge\ldots\wedge R.B_k=S.B_k}(R \times S)) \quad .$$

Durch die abschließende Projektion werden die gleich benannten Attribute nur einmal in die Ergebnisrelation übernommen.

Ein anschauliches Beispiel ist der Verbund der drei Relationen *Wände*, *Materialien* und *bestehen_aus*:

$$\text{Wände} \bowtie \text{bestehen\_aus} \bowtie \text{Materialien} \quad .$$

Die Ergebnisrelation beinhaltet alle Wände und die ihnen zugeordneten Materialien (Abb. 9.2).

Vereinfachtes Modell

Die meisten relationalen Datenbanken implementieren ein vereinfachtes Modell, das lediglich eine Annäherung an die Relationale Algebra darstellt. Beispielsweise ist die zugrundeliegende Datenstruktur anstelle der Relation die Tabelle. Der wesentliche Unterschied besteht darin, dass eine Tabelle mehrere identische Zeilen besitzen kann und dass sowohl Zeilen als auch Spalten eine Ordnung besitzen.

Vereinfacht betrachtet, besteht eine relationale Datenbasis also aus mehreren Tabellen. Jede Zeile einer Tabelle entspricht dabei einem Datensatz (Tupel), wobei die Spalten den Namen und den Datentyp der einzelnen Komponenten (Attribute) festlegen. Beziehungen zwischen Datensätzen werden nicht explizit definiert, sondern es werden sogenannte Schlüssel verwendet, um eine Verknüpfung zwischen den Zeilen verschiedener Tabellen zu modellieren. Ein solcher Schlüssel setzt sich aus dem Inhalt einer oder mehrerer Spalten einer Tabelle zusammen. Er muss identisch mit dem Schlüssel in der verknüpften Zeile einer anderen Tabelle sein.

```
SELECT Liste von Attributen
FROM Liste von Quellrelationen
WHERE Selektionsprädikat
```

Code 9.1: Struktur einer SQL-Anfrage.

SQL

Die Anfragesprache relationaler Datenbanken ist die deklarative Sprache SQL. Deklarativ bedeutet, dass der Nutzer lediglich angibt, welche Daten ihn interessieren, nicht aber, wie die Suche nach diesen Daten vorgenommen wird. Die oft sehr komplexen, für eine zügige Auswertung nötigen Entscheidungen werden von dem in das Datenbanksystem integrierten Anfrageoptimierer übernommen. Neben einer vereinfachten Bedienbarkeit hat dies den Vorteil, dass die physische Datenunabhängigkeit gewährleistet werden kann.

Auf Basis des relationalen Datenmodells entwickelte IBM 1974 die Datenbanksprache SEQUEL (kurz für Structured English QUEry Language), die heute als Urvater von SQL gilt. 1976 entstand mit *System R* der erste Prototyp eines relationalen Datenbanksystems, das SEQUEL bzw. dessen Nachfolger SEQUEL2 implementierte (Chamberlin *et al.*, 1976). Die Anbieter der ersten kommerziell verfügbaren relationalen Datenbanksysteme setzten auf einer Untermenge von SEQUEL2 auf und entwickelten daraus ihre eigenen Dialekte unter dem Akronym SQL (Structured Query Language).

Das *American National Standards Institute* (ANSI) verabschiedete 1986 die erste genormte Version von SQL, die auch als SQL-86 bezeichnet wird. Eine revidierte bzw. ergänzte Version dieser Norm wurde 1989 von der *International Standards Organization* (ISO) herausgegegeben und später SQL-89 getauft. Bei der Entwicklung der ersten gemeinsamen ISO/ANSI-Norm aus dem Jahr 1992, die unter dem Namen SQL-92 bzw. SQL2 bekannt wurde, besann man sich der theoretischen Ursprünge von SQL und achtete verstärkt auf Konformität mit der Theorie der Relationalen Algebra. Aufgrund der Popularität objektorientierter Programmiersprachen, wurden in die 1999 verabschiedete Version SQL:1999 objektorientierte Konzepte integriert. Dadurch wurde die Ära objekt-relationaler Datenbanksysteme eingeleitet (siehe Abschnitt 9.2.4).

Leider führte das frühe Entstehen verschiedener Dialekte zu einer Divergenz in den SQL-Implementierungen, die auch heute noch vorzufinden ist. Dies erschwert die Portabilität der für eine bestimmte Datenbank entwickelten Anwendungen und die Interoperabilität zwischen den Datenbanken verschiedener Hersteller. Der SQL-Standard beschreibt in diesem Zusammenhang lediglich einen minimalen Funktionsumfang, den viele der Hersteller vollständig unterstützen.

SQL hat eine relativ einfache Syntax, die an die englische Umgangssprache angelehnt ist. Eine SQL-Anfrage besteht immer aus einem SELECT-, einem FROM- und einem WHERE-Teil (Code 9.1).

```
SELECT w.id
FROM Wall w, Storey s
WHERE w.height > 220 AND w.storey_id=s.id AND s.number = 1
```

Code 9.2: Beispiel einer SQL-Anfrage, die das Kreuzprodukt zweier Relationen verwendet.

```
SELECT w.id
FROM Waende NATURAL JOIN bestehen_aus NATURAL JOIN Materialien
```

Code 9.3: Beispiel einer Anfrage in SQL-92, die den natürlichen Verbund zweier Relationen herstellt.

Im SELECT-Teil wird eine Liste mit den Namen der auszugebenden Attribute der Ergebnisrelation angegeben. Seine Funktion entspricht damit der *Projektion* der Relationalen Algebra. Der FROM-Teil gibt die für die Berechnung des Ergebnisses benötigten Tabellen an. Im WHERE-Teil wird schließlich eine Bedingung angegeben (Selektionsprädikat), die jedes Tupel der Ergebnisrelation erfüllen muss. Das entspricht der Anwendung einer *Selektion* in der Relationalen Algebra.

Ein *Kartesisches Produkt* von Relationen wird in SQL über die Angabe mehrerer Tabellen im FROM-Teil realisiert, wie Code 9.2 illustriert. Das Beispiel zeigt auch die Verwendung sogenannter *Tupelvariablen* (w und s), die als Platzhalter für ein Tupel der bezeichneten Relation fungieren. In SQL-92 wurde die Möglichkeit geschaffen, eine spezielle *Join*-Operation explizit zu spezifizieren. Dies geschieht durch Angabe eines der Schlüsselwörter `cross join`, `natural join` oder `inner join` bzw. `left`, `right` oder `full outer join`. Wenn notwendig, wird die *Join*-Bedingung hinter dem Schlüsslwort `on` angegeben. Code-Beispiel 9.3 illustriert eine Anfrage, die zu dem in Abb. 9.2 gezeigten Ergebnis führt.

SQL verfügt neben Befehlen zur Anfrage an eine Datenbasis auch über Befehle zur Festlegung des Schemas (Data Definition Language, DDL), zum Einfügen, Verändern bzw. Löschen von Datensätzen (Data Manipulation Language, DML) sowie zur Rechteverwaltung (Data Control Language, DCL). SQL stellt damit eine vollständige Schnittstelle für alle Datenbankoperationen zur Verfügung.

Bewertung

Während das relationale Modell gut geeignet ist für die Speicherung und Analyse gering strukturierter Daten wie Adressdatensätze u.ä., birgt es vor allem Defizite bei komplexen Datenstrukturen, wie sie beispielsweise verstärkt im Bereich ingenieurwissenschaftlicher Anwendungen auftreten. Diese Defizite resultieren vor allem aus der Beschränkung der Attribute eines Tupels auf *atomare* Datentypen.

Die fehlende Erweiterbarkeit des Typsystems führte zur Entwicklung objektorientierter und objekt-relationaler Datenbanken, die im Folgenden näher betrachtet werden sollen.

9.2.2 Objektorientierte Datenbanken

Mitte der 1980er Jahre wurden objektorientierte Programmiersprachen entwickelt, die durch die Charakteristika *Kapselung*, *Vererbung* und *Polymorphie* eine stärkere Modularisierung und dadurch eine bessere Wiederverwendbarkeit von Softwarebausteinen unterstützen (Khoshafian & Abnous, 1990). Durch die damit verbundene Verminderung der Komplexität umfassender Softwareprojekte erlangten sie im Folgenden eine große Popularität und haben sich mittlerweile in weiten Bereichen der Softwareentwicklung als Standard etabliert.

Die Anwendung des objektorientierten Paradigmas auf Datenbankmodelle führte Anfang der 1990er Jahre zur Entwicklung von objektorientierten Datenbanken (OODBMS). Derzeit kommerziell verfügbare OODBMS sind beispielsweise Versant[1], ObjectStore[2] und Objectivity[3].

Das objektorientierte Datenmodell besitzt eine ganze Reihe von Vorteilen gegenüber dem relationalen Modell (Kemper & Moerkotte, 1994). Wesentliches Merkmal ist die Einführung von komplexen nutzerdefinierten Typen (sog. Klassen), deren Attribute entweder atomare Datentypen oder wiederum nutzerdefinierte Typen sein können.

Damit verbunden ist auch die einfachere Abbildbarkeit von Beziehungen[4] zwischen den einzelnen Entitäten einer Anwendungsdomäne. Während Beziehungen im relationalen Schema durch die Verwendung von Fremdschlüsseln realisiert werden müssen, bietet das objektorientierte Schema mit dem Modellierungskonzept *Assoziation* eine Möglichkeit der direkten Umsetzung von Beziehungen. Darüber hinaus können in OODBMS Vererbungshierarchien zwischen nutzerdefinierten Typen abgebildet werden. Die abgeleiteten Klassen erben dabei die Eigenschaften (Attribute) der Elternklasse.

Diese mächtigen Modellierungsmittel sorgen dafür, dass OODBMS vor allem für Anwendungen mit komplexen Datenstrukturen geeignet sind. Ein gutes Beispiel für eine hochkomplexe Datenstruktur ist das IFC-Schema (siehe Abschnitt 2.2.3) mit seiner umfangreichen Klassenhierarchie und einer Vielzahl von Beziehungen zwischen den Klassen.

Da das Datenbankschema einer OODBMS in der Regel identisch zur Klassenstruktur der Anwendung ist, ist eine nahtlose Integration der Datenbankfunktionalität in das Anwendungsprogramm möglich. Dies bedeutet, dass der Mehraufwand bei der programmtechnischen Kopplung von objektorientierter Anwendung und relationaler Datenbank (auch als *impedance mismatch* bezeichnet), der aus dem Mapping der unterschiedlichen Modellierungskonzepte resultiert, weitestgehend vermieden wird.

[1] http://www.versant.com
[2] http://www.progress.com/realtime/products/objectstore
[3] http://www.objectivity.com
[4] Beziehungen werden in der Unified Modeling Language (UML) als *Assoziation*, im Entity-Relationship-Modell als *Relationship* bezeichnet.

```
SELECT w.id
FROM Wall w
WHERE w.height > 220 AND w.storey.number = 1
```

Code 9.4: Beispiel für die Anfragesprache objektorientierter Datenbanken *Object Query Language* (OQL). Da in objektorientierten Datenbankschemata Assoziationen zwischen Klassen direkt modelliert werden können, ist eine Navigation entlang dieser Assoziationen mit Hilfe nacheinander geschalteter Punktoperatoren möglich. In SQL-92 ist hierzu die Verwendung des Kreuzprodukts notwendig (Code 9.3).

Trotz dieser Vorteile bei der Modellierung der Anwendungsdomäne und der Anbindung von Anwendungssoftware konnte sich die Verwendung von OODBMS nicht durchsetzen. Jüngere Untersuchungen haben ergeben, dass der Marktanteil von relationalen DBMS bei mehr als 80 Prozent liegt, während objektorientierte DBMS nur etwa ein Prozent des Marktes abdecken (Leavitt, 2000).

Die Gründe für den ausbleibenden Erfolg von OODBMS sind vor allem in unzureichenden bzw. schlecht unterstützten Standards zu suchen. Obwohl sich früh die *Object Data Management Group* (ODMG) formierte und entsprechende Standards verabschiedete (Cattell & Barry, 2000), fehlt es noch immer an vollständigen Implementierungen dieser Standards in kommerziellen Produkten.

Ein anderer Grund für die geringe Verbreitung von OODBMS liegt in der fehlenden Reife der kommerziell verfügbaren Produkte, die grundlegende Datenbankfunktionalitäten wie Anfrageoptimierung, Transaktionsverarbeitung und Zugriffskontrolle nicht bzw. nur sehr rudimentär unterstützen. In der Folge erzielen OODBMS bei Standardapplikationen häufig geringere Leistungswerte als RDBMS.

Objektorientierte Datenbanken bieten mit der *Object Query Language* (OQL) eine eng mit SQL verwandte Anfragesprache. Eine OQL-Anfrage besteht ebenso wie eine SQL-Anfrage aus einem SELECT-, einem FROM- und einem WHERE-Teil (Code 9.4).

Zwar unterstützt OQL ebenso wie SQL:1999 im WHERE-Teil die Traversierung entlang von Assoziationen, jedoch keine Methodenaufrufe. Der Grund hierfür liegt darin, dass die Methodenimplementierung nicht auf der Datenbank-, sondern auf der Anwendungsseite lokalisiert ist. Eine Umsetzung von räumlichen Operatoren ist mit objektorientierten Datenbanken daher nicht möglich.

Wegen ihrer hohen Modelliermächtigkeit und starken Popularität wurden objektorientierte Konzepte von den Herstellern relationaler Datenbanken aufgenommen und sukzessive in die eigenen Produkte integriert. Dies führte zur Schaffung einer Zwittergattung, den objekt-relationalen Datenbanken, die die Vorteile beider Modellierwelten vereinen und im Abschnitt 9.2.4 näher besprochen werden.

9.2.3 Produktmodellserver

Bei Produktmodellservern handelt es sich um eine spezielle Ausprägung von OODBMS, die ausschließlich objektorientierte Modelle speichern können, die mit Hilfe der Datenbeschreibungssprache EXPRESS definiert wurden. Kommerzielle Implementationen von Produktmodellservern sind beispielsweise der *EuroSTEP Model Server*[5] und der *EXPRESS Data Manager Server* von EPM[6].

EXPRESS ist Teil des STEP-Standards (Teil 11) und dient zur Modellierung von Datenstrukturen in allen basis- und anwendungsspezifischen Protokollen. Da der IFC-Standard auf den STEP-Protokollen aufsetzt, wird EXPRESS auch dort zur Beschreibung von Datenstrukturen verwendet.

Teil 21 des STEP-Standards definiert ein Dateiformat für den Austausch von EXPRESS-Modellen in Textdateien. Die meisten Produktmodellserver können derartige *Part-21*-Dateien lesen, um auf diese Weise das Schema der zugrundeliegenden Datenbank festzulegen. Auch der Import und Export der eigentlichen Produktdaten kann über *Part-21*-Dateien realisiert werden.

Darüber hinaus wird in Teil 22 die generische Schnittstelle *Standard Data Access Interface* (SDAI) zum Zugriff auf EXPRESS-Daten, die im Haupt- oder Sekundärspeicher eines Rechners vorliegen, spezifiziert. Andere Teile des Standards legen wiederum fest, wie dieses generische Interface auf spezifische Programmiersprachen wie IDL, C++ oder Java abgebildet wird. Diese Abbildung wird auch als *Language Binding* bezeichnet und erlaubt eine nahtlose Integration des Datenbankzugriffs in die anzubindenden Anwendungsprogramme. Auf ein STEP- bzw. IFC-Modell, das von einem Produktmodellserver vorgehalten wird, wird daher in der Regel über ein *Language Binding* zugegriffen.

Darüber hinaus bieten Produktmodellserver aber auch die Möglichkeit, mit Hilfe in EXPRESS geschriebener Prozeduren auf die Daten im Produktmodellserver zuzugreifen bzw. diese zu manipulieren. Dabei kann auch der QUERY-Operator verwendet werden, der zur Formulierung von deklarativen Anfragen dient (Code 9.5).

Der QUERY-Operator erlaubt jedoch lediglich die Anwendung einfacher numerischer Vergleichsoperatoren ($<, >, =$) und boolescher Verknüpfungen (AND, OR, NOT). Der Aufruf von nutzerdefinierten Funktionen oder Operatoren ist hingegen nicht möglich. Dies und die fehlende Unterstützung nutzerdefinierter Indexstrukturen führen dazu, dass EXPRESS als wenig geeignet zur Umsetzung einer deklarativen räumlichen Anfragesprache zu beurteilen ist.

[5] Eurostep Ltd.: The Eurostep Model Server for IFC.
`http://www.eurostep.com/prodserv/ems/ems.html`

[6] Jotne EPM Technology: EDMserver.
`http://www.epmtech.jotne.com/products/edmserver.html`

```
LOCAL
  persons  : set of person;
  male_adults : bag of person;
END_LOCAL;
  male_adults :=
    QUERY(p <* persons | (p.sex = MALE) AND (p.age >=18));
```

Code 9.5: Beispiel für eine mittels EXPRESS formulierte Anfrage an einen Produktmodellserver. Nach der Definition temporärer Variablen wird der QUERY-Operator eingesetzt, dem die zu erfüllenden Bedingungen übergeben werden. Er liefert im Beispiel alle männlichen Personen, die älter als 18 Jahre sind.

```
<pmql>
  <select type="entity" match="IfcWall" action="get">
    <where>
      <expr value="Label LIKE 'Wall#%'"/>
      <expr value="calcWallVolume &lt; 3.0"/>
    </where>
  </select>
</pmql>
```

Code 9.6: Beispiel für die auf XML beruhende *Partial Model Query Language*. Im Beispiel werden alle Wände selektiert, deren Volumen größer als 3.0 ist. Dieser Wert muss jedoch für alle Wände vorher ermittelt und in ein entsprechendes Attribut geschrieben worden sein. Ein Methodenaufruf in der Anfrage ist nicht möglich.

Neben der Verwendung von EXPRESS gibt es erste Ansätze, eine auf XML[7] basierende Anfragesprache in Produktmodellserver zu integrieren. Dazu gehört die *Partial Model Query Language* (PMQL)[8], die in den von Secom Co. Ltd. und VTT entwickelten Prototypen IMSVR integriert wurde (Code 9.6). Eng verwandt damit ist die *Product Model Query Language*, die der kommerziell verfügbare Produktmodellserver der Firma Eurostep bereitstellt.

Allerdings gibt es auch beim XML-basierten Ansatz einer Anfragesprache für Produktmodellserver keine Möglichkeit, Methoden- bzw. Funktionsaufrufe in die Anfrage zu integrieren. Der damit einhergehende Mangel an Erweiterbarkeit verhindert die Verwendung einer derartigen Sprache für die Umsetzung einer räumlichen Anfragesprache.

[7] Extensible Markup Language, Standard zur Modellierung von halbstrukturierten Daten in Textdokumenten mittels Baumstruktur, der vom World Wide Web Consortium (W3C) definiert wurde. `http://www.w3.org/XML`

[8] Overview of Partial Model Query Language. Technical Report VTT-TEC-ADA-12. VTT. `http://cic.vtt.fi/projects/ifcsvr/tec/VTT-TEC-ADA-12.pdf`

9.2.4 Objekt-relationale Datenbanken

Mit SQL:1999 wurde ein ISO-Standard[9] verabschiedet, der das relationale Modell um objektorientierte Aspekte erweitert. Die wesentliche Erweiterung gegenüber SQL-92 liegt in der Möglichkeit der Definition komplexer Datentypen mit dazugehörigen Methoden. Datenbanksysteme, die diesen Standard unterstützen, werden als objekt-relationale Datenbanksysteme (ORDBMS) bezeichnet.

Als kommerziell verfügbare ORDBMS sind u.a. Oracle, IBM DB2 und IBM Informix sowie die OpenSource-Datenbank Postgres zu nennen. Einen guten Überblick über diese Produkte und einen Vergleich der implementierten objekt-relationalen Features bietet (Türker, 2003).

Objekt-relationale Datenbankmanagementsysteme vereinigen die Vorteile einer relationalen Datenbank, wie eine hohe Verarbeitungsgeschwindigkeit und eine standardisierte Anfragesprache, mit denen der objektorientierten Modellierung, wie Abstraktion durch Kapselung und Vererbung (Türker, 2003; Melton, 2003). Auf diese Weise wird ein höheres Maß an Anwendungssemantik direkt in die Datenbank integriert, was zu einer einfacheren und fehlerfreieren Anbindung von Applikationen führt.

Die Einführung nutzerdefinierter Datentypen (User Defined Datatypes, UDT bzw. Abstract Data Types, ADT) erlaubt die Erweiterbarkeit des Typsystems. Durch die geschachtelte Anwendung von Typkonstruktoren können dabei beliebig komplexe Datentypen entstehen. Damit wird zum einen eine präzisere Abbildung der Realweltobjekte und ihrer Beziehungen auf Datenbankobjekte unterstützt und zum anderen die Wiederverwendbarkeit dieser Datentypen sichergestellt.

Man spricht von derartigen nutzerdefinierten Typen auch als Objekttypen, da sie die Eigenschaften (Attribute) und das Verhalten (Methoden) von gleichartigen Datenbankobjekten beschreiben und ihnen folgende Prinzipien der Objektorientierung zugrundeliegen (Khoshafian & Abnous, 1990):

- **Strukturierung bzw. Modularisierung**: Komplexe Datenstrukturen bilden mit den assoziierten Methoden semantische Einheiten, was zu einer besseren Strukturierung und Wartbarkeit der Datenbank führt.

- **Kapselung**: Der Zugriff auf ein Objekt erfolgt über die Methoden seiner Schnittstelle. Die Implementierung bleibt nach außen hin verborgen. Die klare Trennung von Schnittstelle und Implementierung schafft die Basis für eine transparente Evolution der Objektfunktionalität.

- **Objektidentifikation**: Die Entkopplung der Identifikation der Objekte von den zugrundeliegenden Attributwerten ermöglicht zum einen eine eindeutige, unveränderliche Objektreferenz (OID) und zum anderen die Unterscheidung zwischen identischen und gleichen Datenbankobjekten.

[9] ANSI/ISO/IEC 9075-1:99. ISO International Standard: Database Language SQL.

```
SELECT w2.uid
FROM   Wall w1, Wall w2
WHERE  w2.closerThan(w1, 10.0) AND
       w1.uid = HJDSA32asdfh
```

Code 9.7: Beispiel für eine räumliche Anfrage auf Basis von SQL:1999. Der Operator *closerThan* ist hier als Methode des nutzerdefinierten Typs *Wall* umgesetzt.

- **Vererbung**: Ein Subtyp erbt die Attribute und Methoden eines Supertyps. Dabei können die ererbten Methoden überschrieben werden. Das späte Binden der Methoden ermöglicht die dynamische, objektspezifische Auswahl der richtigen Methodenimplementierung zur Laufzeit.
- **Substituierbarkeit**: Ein Objekt eines Subtyps kann überall dort verwendet werden, wo ein Objekt des zugehörigen Supertyps erwartet wird.

Mit der Erweiterung des Typsystems kann eine gewöhnliche Relation (Tabelle) nun Spalten enthalten, die auf konstruierten, komplexen Datentypen beruhen.

Eine Spalte kann

- atomar,
- tupelwertig,
- kollektionswertig,
- objektwertig oder
- referenzwertig

sein. Eine Tupeltabelle ist damit nicht mehr auf atomare Spalten beschränkt.

Darüber hinaus erlauben objekt-relationale Datenbanken die Einrichtung und Verwendung von *Objekttabellen*. Das sind Tabellen, bei denen in jeder Zeile ein Objekt des zugehörigen strukturierten Typs gespeichert werden. Die erste Spalte einer solchen Tabelle beinhaltet dabei immer einen eindeutigen, systemgenerierten Objektidentifikator (OIS), über welche die Objekte mittels eines streng typisierten Referenztyps referenzierbar werden.

Die Methoden dieser Klassen werden als sogenannte *Stored Procedures* implementiert, d.h. sie werden in der Datenbank gespeichert und ausgeführt. Letzteres ist ein wesentlicher Aspekt bei der Realisierung der räumlichen Operatoren, da sie als Selektionskriterium eingesetzt werden sollen und daher vor der Übermittlung der Ergebnismenge an das Anwendungsprogramm angewendet werden müssen.

Für die Umsetzung der räumlichen Anfragesprache ergibt sich mit diesen Erweiterungen eine einfache Möglichkeit, die in Kapitel 7 definierten räumliche Typen als nutzerdefinierte Typen (Klassen) in SQL zu integrieren und die räumlichen Operatoren als Methoden dieser Typen umzusetzen. Ein Beispiel für eine räumliche Anfrage auf Basis von SQL:1999 zeigt Code 9.7.

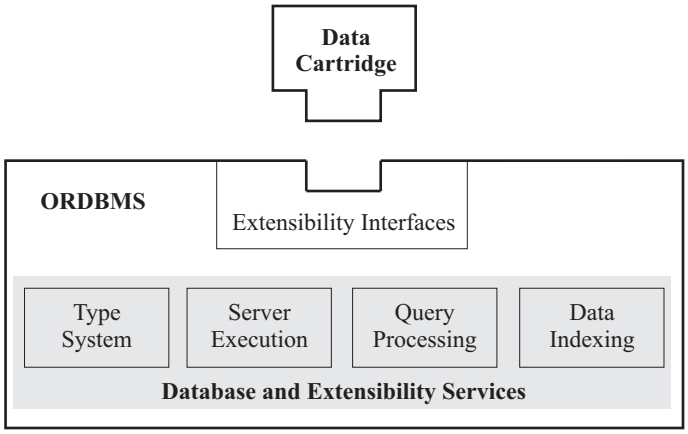


Abbildung 9.3: Erweiterungsschnittstellen einer objekt-relationalen Datenbank.

Die Verwendung objekt-relationaler Datenbankfunktionalität wurde auch bei der Entwicklung von räumlich-topologischen Anfragesprachen für geographische Informationssysteme (GIS) verfolgt, hier allerdings vorwiegend für zweidimensionale räumliche Objekte (Scheugenpflug, 2005). Die entsprechenden Erweiterungen von SQL um räumliche Datentypen und Operatoren wurden durch das OpenGIS-Konsortium standardisiert[10]. Die Implementierung dieses Standards in den kommerziellen Produkten *Oracle Spatial, IBM Spatial Extender for DB2, Spatial Datablade for IBM Informix* und *PostGIS* sind Beispiele für modulare Erweiterungen von Datenbanken auf Basis objekt-relationaler Konzepte.

Zur nahtlosen Integration von anwendungsspezifischen Objekttypen verfügen objekt-relationale Datenbanken über verschiedene Erweiterungsschnittstellen (Abb. 9.3). Dazu gehört u.a. erweiterbares Indexing und erweiterbares Optimizing.

Erweiterbares Indexing erlaubt Anwendungsentwicklern, sekundäre Zugriffsmethoden für die zusätzlich definierten Typen beim Datenbankserver anzumelden. Ein objekt-relationaler Indextyp kapselt dabei Methoden zum Erzeugen und Verwerfen einer anwendungsspezifischen Indexstruktur sowie zum Öffnen und Schließen von Indexscans (Tab. 9.1). Auf diese Weise vervollständigt der Indextyp die funktionelle Implementierung von nutzerdefinierten Operatoren.

Wenn der Anfrageoptimierer entscheidet, den angemeldeten Index für die Verarbeitung einer deklarativen Anfrage zu verwenden, werden die entsprechenden Methoden des Indextyps von der Anfrageverarbeitungseinheit der Datenbank aufgerufen. Dadurch wird sowohl die Pflege des Indexes als auch der Zugriff darauf vollkommen vor dem Anwender versteckt, wodurch dem Prinzip der Datenunabhängigkeit genügt wird.

Üblicherweise unterstützen objekt-relationale Datenbanken sowohl regel- als auch kostenbasierte Anfrageoptimierung. Die regelbasierte Anfrageoptimierung beruht

[10] OpenGIS Konsortium. Simple Features Specification for SQL. `http://www.opengis.org/docs/99-049.pdf`

Methode	Funktion
index_create()	erzeugt einen anwendungsspezifischen Index
index_drop()	verwirft einen anwendungsspezifischen Index
index_open()	öffnet einen anwendungsspezifischen Index
index_close()	schließt einen anwendungsspezifischen Index
index_fetch()	holt den nächsten Eintrag vom Index, der dem Anfrageprädikat genügt
index_insert()	fügt einen neuen Eintrag zum Index hinzu
index_delete()	löscht einen Eintrag vom Index
index_update()	aktualisiert einen Eintrag im Index

Tabelle 9.1: Methoden der Schnittstelle für erweiterbares Indexing.

Methode	Funktion
stats_collect()	sammelt Statistiken zu einem anwendungsspez. Index
stats_delete()	löscht Statistiken zu einem anwendungsspez. Index
predicate_sel()	schätzt die Selektivität eines anwendungsspez. Indexes mit Hilfe der gesammelten Statistiken
index_cpu_cost()	schätzt die CPU-Kosten
index_cpu_cost()	schätzt die I/O-Kosten

Tabelle 9.2: Methoden der Schnittstelle für erweiterbares Optimizing.

auf einer Vereinfachung der Anfrage durch syntaktische Anfragetransformation. Da hierbei die Äquivalenzregeln der Relationenalgebra zur Anwendung kommen, wird auch von algebraischer Optimierung gesprochen. Grundlage der kostenbasierten Optimierung sind Kostenfaktoren, also eine Abschätzung der bei der Anfragebearbeitung entstehenden Kosten für physische Operationen, darunter I/O-Kosten, CPU-Kosten und Speicherkosten. Wegen der besseren Vorhersagegenauigkeit ist der kostenbasierte Ansatz grundsätzlich dem regelbasierten vorzuziehen. Zur Realisierung einer kostenbasierten Anfrageoptimierung enthält die Erweiterungsschnittstelle Methoden, mit deren Hilfe der Anfrageoptimierer Informationen über die Charakteristika eines anwendungsspezifischen Indexes erhalten kann (Tab. 9.2).

Anwendungsspezifische Datentypen, Indexstrukturen und Stored Procedures werden sinnvollerweise in einem Modul gekapselt. Ein solches Modul wird je nach zugrundeliegendem Datenbankprodukt als *Data Cartridge* (Oracle), *DataBlade* (IBM Informix) oder *Extender* (IBM DB2) bezeichnet.

Das ORDBMS Oracle

Die Firma Oracle Inc.[11] ist Marktführer im Bereich von Datenbankprodukten. Hauptprodukt und technologischer Kern der gesamten Produktpalette ist das ORDBMS *Oracle*, das zurzeit (April 2007) in der Version 10g vorliegt. Besonders erwähnenswert ist die ausgezeichnete Unterstützung der objektorientierten Features von SQL:1999, darunter auch die für das hier beschriebene Konzept unerlässliche Unterstützung von Methodenaufrufen in SQL-Anfragen (Türker, 2003).

Darüber hinaus ist eine große Palette von Aufsätzen verfügbar, die zum Teil Bestandteil des Datenbankprodukts sind und zum Teil als gesonderte Produkte vertrieben werden. Von Bedeutung im Rahmen dieser Arbeit sind vor allem folgende Module bzw. Produkte:

- **Oracle Workspace Manager.** Der Workspace Manager stellt eine virtuelle Umgebung bereit, in der mehrere Versionen der Daten einer Tabelle erzeugt bzw. Änderungen an den Daten einer Tabelle isoliert werden können und dabei eine persistente „History of Changes“ vorgehalten wird. Diese Funktionalitäten werden vor allem in Hinblick auf die in Planungsprozessen üblichen Langzeit-Transaktionen benötigt (Firmenich, 2002). So kann mit Hilfe des Workspace Managers der konkurrierende Zugriff mehrerer Planer auf das gespeicherte Bauwerksmodell mit automatischer Konflikterkennung realisiert werden. Der entscheidende Aspekt hierbei ist, dass alle Planer Lese- und Schreibrechte behalten, während Änderungen zunächst isoliert gehalten werden. Der Workspace Manager hilft dabei, diese Änderungen zu organisieren und zu gruppieren, bevor sie mit dem aktuellen Modell zu einer neuen Version zusammengeführt werden. Oracle Workspace Manager ist integraler Bestandteil der Oracle Datenbank.

- **Oracle Workflow.** Oracle Workflow ist ein Workflow-Management-System und erlaubt als solches die Modellierung und Automatisierung von Geschäftsprozessen. Letzteres bedeutet im Wesentlichen, dass beim Eintreten bestimmter Ereignisse (zum Beispiel Fertigstellung der Werkplanung) Informationen an die entsprechenden Projektbeteiligten weitergeleitet werden. Die Schnittstelle zu diesem Benachrichtigungssystem bilden entweder E-Mail-Programme oder Web-Browser.

  Die Bedeutung von Workflow-Management-Systemen für den Bauplanungsprozess wurde in einer Reihe von Veröffentlichungen ausführlich diskutiert, u.a. in (van der Aalst, 2002), (Rüppel & Klauer, 2004) und (Greb & Klauer, 2004).

- **Oracle Product Lifecycle Management.** Kern der PLM-Anwendungen von Oracle ist der „Advanced Product Catalog“, der die gesamten Daten eines Produkts zentral vorhält und allen internen und externen Projektbe-

[11] `http:\\www.oracle.com`

teiligten zugänglich macht. Darüber hinaus gehören folgende Anwendungen zur PLM-Produktpalette von Oracle:

- **Oracle CADView-3D.** Mit Hilfe dieser Technologie ist es möglich, in der Datenbank gespeicherte 2D- und 3D-CAD-Modelle unterschiedlichen Formats zu visualisieren bzw. durch sie hindurch zu navigieren. Weiterhin besteht die Möglichkeit der Kennzeichnung von einzelnen Teilen des CAD-Modells mit textartigen Notizen (sog. *Mark-Up*). Da diese Funktionalitäten auch für mehrere Teilnehmer gleichzeitig zur Verfügung stehen, kann dieses Modul Phasen der synchronen Zusammenarbeit unterstützen. Es kann jedoch lediglich zur Absprache zwischen den Projektbeteiligten genutzt werden, direkte Änderungen am Modell sind nicht möglich.
- **Oracle Project Management.** Dieses Modul ermöglicht sowohl Projektplanung, -überwachung und -budgetierung als auch eine Vielzahl von Prognosen über den gesamten Lebenszyklus eines Produkts hinweg. Dazu werden alle verfügbaren Projektinformationen in einem gemeinsamen Repository vorgehalten: Arbeitspläne, Fortschritts-, Problem- und Änderungsberichte sowie Kosten und Budgets.
- **Oracle Project Collaboration.** Erlaubt den Projektbeteiligten, über den aktuellen Projektstatus zu berichten bzw. diesen nachzuverfolgen. Projektfortschritte werden an alle Projektbeteiligten versandt, inklusive Zulieferer und anderer externer Partner. Auf diese Weise können auch Änderungen am Projekt verwaltet werden.
- **Oracle Collaboration Suite.** Neben der integrierten Verwaltung von E-Mail, Fax und Voicemail (sog. Unified Messaging) und gemeinsam nutzbaren elektronischen Kalendern, stellt die Collaboration Suite mit *Oracle Files* eine Möglichkeit zur Verfügung, dezentrale Fileserver zu einem gemeinsamen skalierbaren Dateiservice zusammenzuschließen. Mit *Oracle Web Conferencing* haben Teams die Möglichkeit, Online-Meetings abzuhalten, bei denen einer verteilten Arbeitsgruppe Präsentationen, Anwendungen oder auch der gesamte Desktop eines Benutzers angezeigt werden können.
- **Oracle Sourcing.** Oracle Sourcing ist ein Web-basiertes Vergabesystem und unterstützt den Auftraggeber dabei, Anforderungen für Zulieferprodukte festzulegen, potentielle Zulieferer zu finden sowie entsprechende Verhandlungen bei Vertragsabschluss zu führen. Im Bauwesen ist diese Funktionalität vor allem im Bereich *Ausschreibung–Vergabe–Abrechnung* (AVA) einsetzbar.

Dieses umfangreiche Portfolio an Softwarewerkzeugen zur Unterstützung des (kooperativen) Engineerings von komplexen Produkten untermauert die Eignung dieses kommerziellen Produkts als Grundlage für die Implementierung von räumlichen Analysefunktionalitäten in 3D.

Oracle Spatial. Der Datenbankaufsatz Oracle Spatial stellt räumliche Datentypen und räumliche Funktionen (metrische und topologische) in der Datenbank zur Verfügung und implementiert dabei einen Teil des OGIS-Standards. Zwar werden mehrdimensionale räumliche Objekte grundsätzlich unterstützt, allerdings werden bei den angebotenen räumlichen Analysefunktionen nur die ersten beiden Dimensionen berücksichtigt. Oracle Spatial eignet sich daher lediglich zur Speicherung und Abfrage von 2D-GIS-Daten (Gröger *et al.*, 2004).

Mit der im April 2006 angekündigten Initiative *Collaborative Building Information Management* (CBIM) will die Oracle Corporation in Zukunft speziell an die Bedürfnisse der Bauindustrie angepasste Produkte entwickeln, um sich auch auf diesem bislang noch wenig entwickelten Markt zu etablieren (Oracle Corporation, 2006).

9.3 Gesamtkonzept einer räumlichen Datenbank für digitale Bauwerksmodelle

9.3.1 Räumliche Datenbanken

Als räumliche Datenbanken (engl. Spatial Databases) werden Datenbanksysteme bezeichnet, die räumliche Datentypen und räumliche Indizierungstechniken zur Verfügung stellen (Güting, 1994). Der Begriff stammt ursprünglich aus dem 2D-GIS-Bereich, lässt sich aber ebenso auf die Verwaltung von dreidimensionalen geometrischen Objekten anwenden. Die Verknüpfung von räumlicher Anfrage- bzw. Analysefunktionalität mit Datenbanktechnologien birgt eine Reihen von Vorteilen, darunter die Verwendbarkeit einer ausgereiften Anfragesprache. Mit dem Einsatz eines DBMS stehen zudem Mechanismen wie die Behandlung von Transaktionen, Nutzerverwaltung und leistungsfähiges *Load Balancing* zur Verfügung, die die Grundlage für eine durchgängige Unterstützung verteilt-kooperativen Engineerings bilden.

Die Definition einer räumlichen Datenbank legt nicht die Art des zugrundeliegenden Datenbankmodells fest. Tatsächlich gab bzw. gibt es sowohl im akademischen als auch im kommerziellen GIS-Bereich Vertreter jeden Typs: Räumliche RDBMS (Lorie & Meier, 1984; Ooi *et al.*, 1989), räumliche OODBMS (Aschwanden *et al.*, 1995; Chen *et al.*, 2000) und räumliche ORDBMS (Gröger *et al.*, 2004). Wie in den vorangegangenen Abschnitten gezeigt werden konnte, erlauben jedoch vor allem die Erweiterungsmechanismen einer objekt-relationalen Datenbank eine besonders einfache und konzeptionell stringente Integration von sowohl räumlichen Datentypen als auch räumlichen Indizierungstechniken. Das hier vorzustellende Konzept sieht daher die Umsetzung einer räumlichen Datenbank für digitale Bauwerksmodelle auf Basis eines ORDBMS vor.

9.3.2 Verwandte Arbeiten außerhalb des GIS-Bereichs

In (Ozel, 2000) wird das Potential der Anwendung von Geographischen Informationssystemen für die Analyse und die Simulation von dynamischen Prozessen in

Gebäuden untersucht. Dabei werden ausschließlich Funktionalitäten im zweidimensionalen Raum verwendet und Gebäudestrukturen in Form von Grundrissen im GIS abgebildet. Ozel weist darauf hin, dass bauteilorientierte CAD-Systeme zwar über ausgereifte Möglichkeiten zur Geometriemodellierung verfügen, aber nicht über die Fähigkeiten räumlicher Analyse von GI-Systemen. Eine Abbildung von Bauwerksmodellen in GI-Systemen sei vor allem in dieser Hinsicht lohnenswert und müsse weiter untersucht werden.

Kriegel *et al.* haben für dreidimensionale CAD-Modelle des Maschinen- bzw. Fahrzeubaus das auf dem objekt-relationalen DBMS Oracle basierende System DIVE (Database Integration for Virtual Engineering) entwickelt, das räumliche Anfragen verarbeiten kann (Pötke, 2001; Kriegel *et al.*, 2003; Pfeifle, 2004).

DIVE sieht die Kopplung eines *Engineering Data Management*-Systems (EDMS)[12] mit einer räumlichen Datenbank vor. Die CAD-Dateien werden dabei vom EDM verwaltet und nicht in der Datenbank vorgehalten. In der Datenbank werden lediglich Approximationen der CAD-Geometrie in Form von Voxel-Feldern gespeichert. Zur Überführung der Ursprungsgeometrie wird diese zunächst tesseliert, danach voxelisiert und schließlich daraus ein sogenannter *Relation Interval Tree* generiert. Dieser dient der Indizierung der Bauteile in der Datenbank und bildet die Grundlage für die effiziente Verarbeitung von räumlichen Anfragen.

Von DIVE werden drei Arten von räumlichen Anfragen unterstützt:

- Volumenanfrage: Finde alle räumlichen Objekte, die einen gegebenen achsenparallelen Quader schneiden.
- Kollisionsanfrage: Finde alle Objekte, die eine beliebige Region schneiden.
- Abstandsanfrage: Finde alle räumlichen Objekte innerhalb eines gegebenen Abstands von einem gegebenen Punkt.

Das Ergebnis einer Anfrage ist dabei jedoch nicht das geometrische Objekt selbst, sondern lediglich eine Liste von URLs, die auf die CAD-Dateien der selektierten Einzelteile verweisen.

Die bei DIVE vorliegende Redundanz der geometrischen Informationen in einem verteilten und vor allem heterogenen System ist als problematisch einzuschätzen. Zudem wird SQL ohne Erweiterungen verwendet, d.h. es werden keine gesonderten räumlichen Operatoren zur Verfügung gestellt.

9.3.3 SQL als Grundlage einer räumlichen Anfragesprache

Der hier verfolgte Ansatz zur Umsetzung räumlicher Anfragefunktionalität beruht auf der Speicherung des gesamten Bauwerksmodells inklusive einer expliziten

[12] EDM-Systeme sind im Bereich des Maschinen- und Fahrzeugbaus verbreitete Dokumentenmanagementsysteme, die alle Abläufe und Daten vorhalten, die im Laufe des Lebenszyklus eines Produkts anfallen. Dazu gehören Produktkonfigurationen (zum Beispiel Stücklisten), CAD-Modelle und -Zeichnungen, beliebige Arten von Dokumenten sowie Projekt- und Arbeitspläne.

Beschreibung seiner Geometrie in einem objekt-relationalen Datenbankmanagementsystem (ORDBMS).

Auf diese Weise kann SQL:1999 als Grundlage für die Umsetzung der räumlichen Anfragesprache genutzt werden. Die Nutzung dieser weit verbreiteten und standardisierten Anfragesprache hat gegenüber der Entwicklung einer properitären Sprache eine Reihe von Vorteilen, darunter

- der Entwurf einer Grammatik für eine properitäre Sprache ist nicht notwendig,
- der Anschluss von Anwendungssoftware ist über Standardschnittstellen möglich,
- die Verwendbarkeit des in das DBMS integrierten, äußerst effizienten SQL-Parsers sowie des Anfrageoptimierers ist gegeben.

Die Verwendung von SQL als Grundlage einer räumlichen Anfragesprache wurde ebenfalls von der GIS-Community präferiert und führte zunächst im Forschungsbereich zur Entwicklung von *Spatial SQL* (Egenhofer, 1994) und später zu einer entsprechenden Standardisierung durch das OpenGIS-Konsortium[13]. Vor- und Nachteile der Verwendung von SQL als Basis für eine 2D räumliche Anfragesprache sowie alternative Ansätze werden ausführlich in (Egenhofer, 1992) diskutiert.

Dass SQL sich sehr gut für die Integration räumlicher Aspekte in eine Anfragesprache eignet, wird ebenso in (Lee & Musick, 2004) gezeigt. Darin wird MeshSQL beschrieben, eine auf SQL basierende Sprache, die für Anfragen auf Simulationsnetzen verwendet werden kann. MeshSQL erlaubt Anfragen mit zeitlicher, räumlicher (2D), statistischer und Ähnlichkeitssemantik. Dreidimensionale Objekte und deren topologische Beziehungen werden jedoch nicht berücksichtigt. Für die Umsetzung von MeshSQL wurde ebenfalls ein objekt-relationales Datenbanksystem verwendet.

9.3.4 Speicherung von räumlichen Objekten

Das hier verfolgte Konzept einer räumlichen Datenbank für digitale Bauwerksmodelle sieht vor, dass einzelne geometrische Objekte des Modells persistent in der Datenbank abgelegt und später wieder daraus abgerufen werden können. Nach dem späteren Abrufen dieser Objekte durch eine Datenbank-Anfrage soll eine direkte Weiterverwendung im CAD-Programm möglich sein.

Um eine nahtlose Anbindung der für eine Bearbeitung von Bauwerksmodellen üblichen CAD-Programme an die räumliche Datenbank zu ermöglichen, muss gewährleistet sein, dass keinerlei Informationsverluste bei der Speicherung der geometrischen Informationen auftreten und nur geringer Aufwand bei der Konvertierung des Geometriemodells entsteht. Die Wahl des Datenbank-Schemas zur Speicherung der Geometrie der räumlichen Objekte ist dabei von entscheidender Bedeutung.

[13] OpenGIS Konsortium. Simple Features Specification for SQL. `http://www.opengis.org/docs/99-049.pdf`

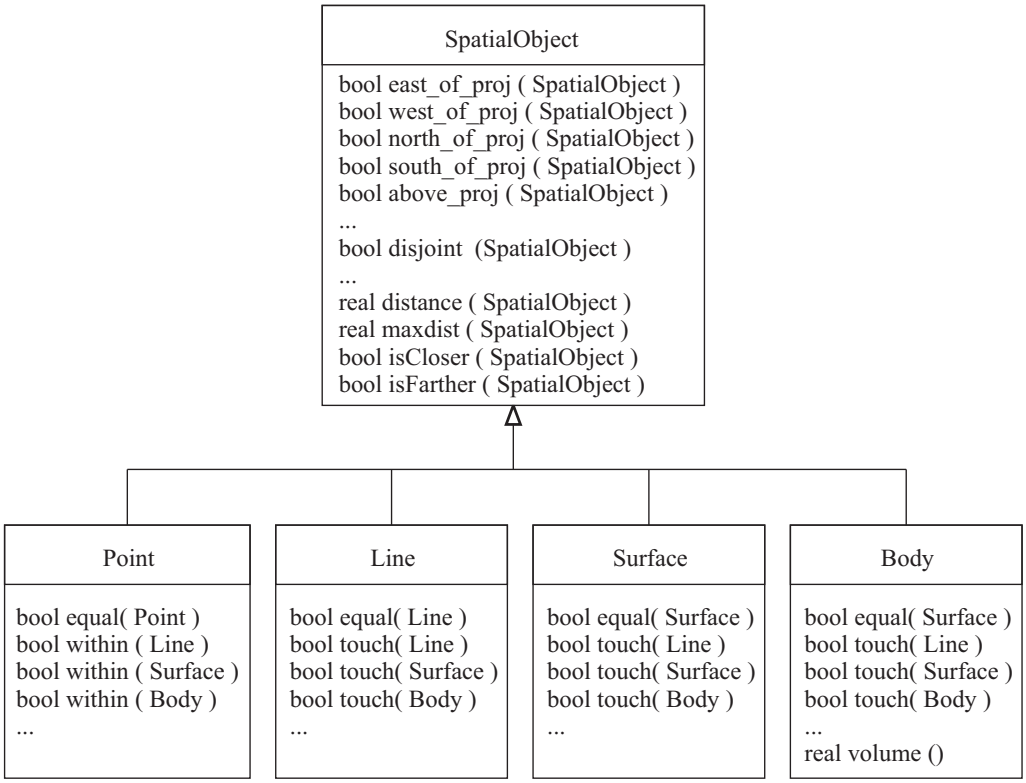


Abbildung 9.4: UML-Diagramm des objekt-relationalen Datenbankschemas: Die räumlichen Typen werden als Klassen abgebildet, die räumlichen Operatoren als Methoden dieser Klassen.

Im Unterschied zu dem von Kriegel *et al.* verfolgten Ansatz (Kriegel *et al.*, 2003) wird die Geometrie der Bauteile direkt in der Datenbank abgelegt und nicht in externen Dateien vorgehalten. Da auf diese Weise der Zugriff auf die Bauwerksdaten ausschließlich mittels der Datenbankschnittstellen möglich ist, wird der Integrationsaspekt der zentralen Datenbank gestärkt und die in Abschnitt 9.1.2 angesprochenen Vorteile der Verwendung eines Datenbanksystems gegenüber dateibasierten Lösungen können auch für geometrische Modelle nutzbar gemacht werden. Datenbank-Features wie Nebenläufigkeitskontrolle und Transaktionssteuerung eröffnen die Möglichkeit, die verteilt-kooperative Arbeit der an der Planung beteiligten Ingenieure fehlerfrei koordinieren zu können.

Um die Funktionalität einer räumlichen Anfragesprache in einem objekt-relationalen Datenbanksystem zur Verfügung stellen zu können, werden zunächst die in Kapitel 7.3 eingeführten räumlichen Typen auf entsprechende Objekt-Typen abgebildet. Dabei entstehen die Klassen *Point*, *Line*, *Surface* und *Body*. Die für sie definierten räumlichen Operatoren werden als Methoden dieser Klassen modelliert (Abb.9.4).

Zur Abbildung der eigentlichen Geometrie dieser räumlichen Objekte können grundsätzlich verschiedenartigste geometrische Modelle[14] zum Einsatz kommen, wie beispielsweise das Modell der *Constructive Solid Geometry* (CSG), verschiedene Zerlegungsmethoden oder oberflächenorientierte Modelle (engl. *Boundary Representation*, kurz BRep). Eine ausführliche Darstellung der unterschiedlichen Geometriemodelle ist beispielsweise (Mäntylä, 1988) zu entnehmen. Dank der nahezu unbeschränkten Fähigkeiten zur Darstellung beliebiger geometrischer Formen hat sich im CAD-Bereich die Verwendung oberflächenorientierter Modelle durchgesetzt.

Um CAD-Objekte ohne Informationsverlust in der Datenbank ablegen zu können und den Aufwand bei der Konvertierung des Geometriemodells zwischen Datenbank und Anwendungsprogramm möglichst gering zu halten, ist es sinnvoll, das Datenbankschema weitestgehend identisch zu dem im Anwendungsprogramm verwendeten Modell zu gestalten. Daher sind hier oberflächenorientierte Geometriemodelle grundsätzlich jenen Modellen vorzuziehen, die auf einer simplizialen Zerlegung oder dem CSG-Ansatz beruhen.

Wesentlicher Aspekt bei Oberflächenmodellen ist die Modellierung der Topologie eines räumlichen Objekts, also der Nachbarschaftsbeziehungen zwischen Knoten, Kanten und Flächen. Hierbei ist zu beachten, dass sich diese Art von topologischen Informationen immer nur auf *ein* räumliches Objekt beziehen und damit gedanklich strikt von den in Abschnitt 7.4.4 behandelten topologischen Beziehungen *zwischen verschiedenen* räumlichen Objekten getrennt werden müssen.

Eine BRep-Datenstruktur bildet die Adjazenzbeziehungen zwischen Knoten v (engl. vertices), Kanten e (engl. edges) und Begrenzungsflächen f (engl. faces) eines Körpers in Form eines sogenannten *vef*-Graphen ab. Die Datenstruktur beschreibt dabei die Relationen zwischen den Elementen des Graphen. Da sich jedoch bestimmte Relationen implizit aus anderen ergeben, ist es nicht notwendig, alle explizit zu speichern. Bei der Auswahl der explizit zu speichernden Relationen muss zwischen einem erhöhten Speicherplatzbedarf infolge redundant vorgehaltener Information und der erforderlichen Rechenzeit für die Ermittlung nicht gespeicherter Relationen abgewogen werden. Darin unterscheiden sich im Wesentlichen die im Folgenden vorzustellenden Varianten einer BRep-Datenstruktur.

- **Einfaches Kantenmodell.** Beim einfachen Kantenmodell werden lediglich die Relationen *ev* und *ef* explizit gespeichert. Mit dieser Datenstruktur kann zwar ein optimaler Speicherplatzbedarf erreicht werden, allerdings ist der Aufwand zur Berechnung nicht gespeicherter Relationen, wie beispielsweise zur Ermittlung der an einen Knoten angrenzenden Flächen, sehr hoch.
- **Winged-Edge-Modell.** Bei der von (Baumgart, 1975) vorgeschlagenen Winged-Edge-Datenstruktur wird zusätzlich jede Kante mit den beiden Flächen assoziiert, die sie begrenzt. Die beiden Flächen bilden gewisser-

[14] In diesem Zusammenhang wird auch von geometrischen *Metamodellen* gesprochen (Mundani, 2005), da es sich hierbei um ein Modell zur Beschreibung eines Modells handelt.

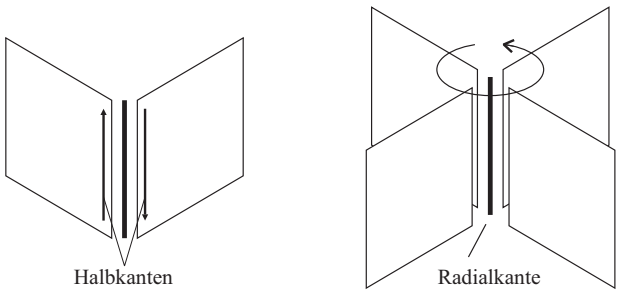


Abbildung 9.5: Links: Eine Halbkante wird jeweils einer der beiden aufeinanderstoßenden Flächen zugeordnet. Rechts: Bei Verwendung von Radialkanten können auch mehr als zwei Kanten aufeinanderstoßen, wodurch nicht-mannigfaltige Körper dargestellt werden können.

maßen die Flügel dieser Kante, woraus sich der Name des Modells ableitet. Darüber hinaus wird in diesem Modell explizit gespeichert, welche Kanten dieselbe Fläche begrenzen. Dazu wird eine zusätzliche Relation $R_{\text{ee'}} \subset E \times E$ eingeführt, die anzeigt, dass zwei Kanten benachbart sind, d.h. einen gemeinsamen Knoten besitzen. Wesentlicher Vorteil gegenüber dem einfachen Kantenmodell ist, dass der Zeitaufwand zur Berechnung nicht gespeicherter Relationen lediglich von lokalen Größen abhängig ist (Bungartz *et al.*, 2002). Es ist also möglich, in konstanter Zeit die mit einer gegebenen Kante assoziierten Ecken und Facetten zu ermitteln.

- **Half-Edge-Modell.** Bei diesem in (Mäntylä, 1988) vorgestellten Modell besteht jede Kante aus zwei sogenannten *Halbkanten*. Jede dieser Halbkanten wird einer der an dieser Kante zusammenstoßenden Flächen zugewiesen (Abb. 9.5). Auf diese Weise erhöht sich zwar die Speicherkomplexität wiederum, allerdings sind nun bestimmte Beziehungen (wie Kante – angrenzende Kante, Eckpunkt – Kanten und Eckpunkt – Flächen) nochmals schneller zu ermitteln als bei Einsatz des Winged-Edge-Modells. Da pro Kante nur zwei benachbarte Flächenstücke zugelassen sind, sind nicht-mannigfaltige Körper nicht darstellbar.

- **Radial-Edge-Modell.** Bei dem in (Weiler, 1988) vorgestellten Modell wird die Halbkante durch eine *Radialkante* ersetzt. An einer solchen Radialkante können nicht nur zwei, sondern beliebig viele Flächenstücke anschließen (Abb. 9.5). Auf diese Weise können auch nicht-mannigfaltige Körper dargestellt werden.

Bei einem Körper mit Mannigfaltigkeit 2 gilt für jeden Punkt auf seiner Oberfläche, dass er eine Umgebung besitzt – in der alle Punkte ebenfalls zur Oberfläche bzw. zum Rand des Körpers gehören (siehe Abschnitt 7.2.2) – die homöomorph zu einer zweidimensionalen Scheibe ist. Zu den „nicht-mannigfaltigen“ Körpern gehören demnach zum einen solche mit „baumelnden“ Flächen oder Linien (Abb. 9.6 oben) und zum anderen solche, bei denen sich mehr als zwei Flächen in einer gemeinsamen Kante oder einem gemeinsamen Punkt treffen (Abb. 9.6 unten). Während erstere durch die Definitionen von Abschnitt 7.3.4

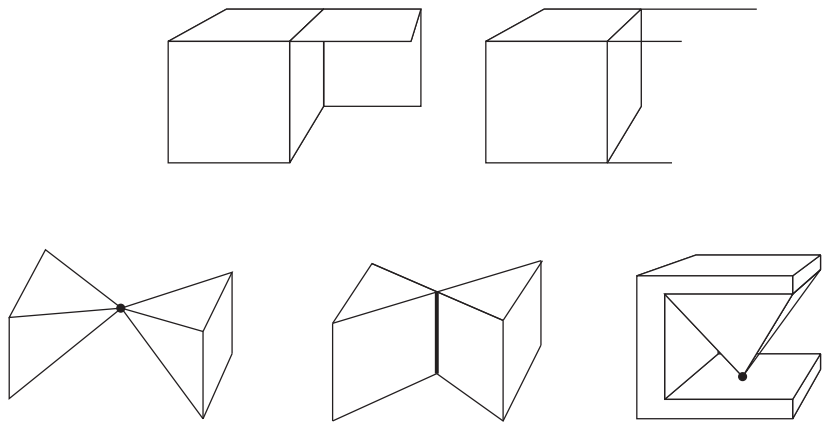


Abbildung 9.6: Nicht-mannigfaltige Körper. **Oben:** Körper mit „baumelnden“ Flächen oder Linien. **Unten:** Körper bei denen sich mehr als zwei Flächen in einer gemeinsamen Kante treffen bzw. bei denen sich Flächen nicht an einer Kante, sondern nur in einem gemeinsamen Punkt treffen.

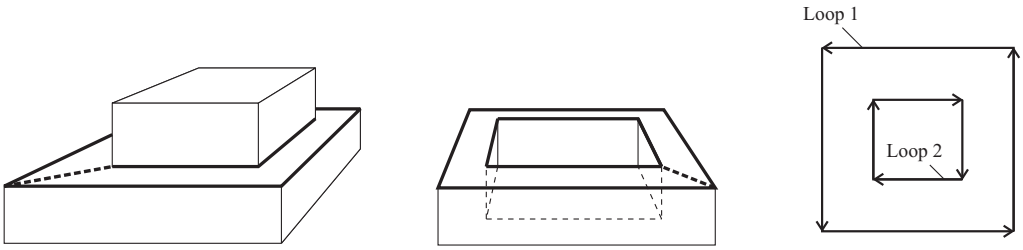


Abbildung 9.7: Die links und in der Mitte dargestellten Körper benötigen für ihre Modellierung mehrere Kantenzüge pro Fläche. Dies kann durch die Verwendung von sogenannten Brückenkanten abgebildet werden oder durch Einführung von *Loop*-Elementen, die einzelne Kantenzüge repräsentieren.

ausgeschlossen sind, sind letztere durchaus erlaubt[15] und müssen durch das Geometriemodell abgebildet werden können. Es empfiehlt sich daher der Einsatz des Radial-Edge-Modells.

Bei Verwendung des einfachen *vef*-Graphs besitzt eine Fläche immer genau einen geschlossenen Polygonzug als Berandung. Möchte man Körper darstellen, bei denen eine Begrenzungsfläche mehr als einen Polygonzug als Berandung besitzt, wie beispielsweise in Abbildung 9.7 gezeigt, und dabei die Einführung von sogenannten Brückenkanten vermeiden, muss der *vef*-Graph bzw. die BRep-Datenstruktur um Elemente vom Typ *Loop* erweitert werden. Ein *Loop* beschreibt dabei immer einen geschlossenen Kantenzug mit einer Orientierung, eine *Face* kann mehrere *Loops* besitzen.

Ähnliches gilt für die Darstellbarkeit von Hohlräumen in Körpern (Abb. 9.8a). Möchte man hier wiederum die Verwendung von Brückenflächen vermeiden, muss ein weiteres Element in die Datenstruktur eingeführt werden. Dieses wird u.a. in

[15] Sie können beispielsweise durch die Anwendung einer booleschen Operation auf zwei mannigfaltige Körper entstehen.

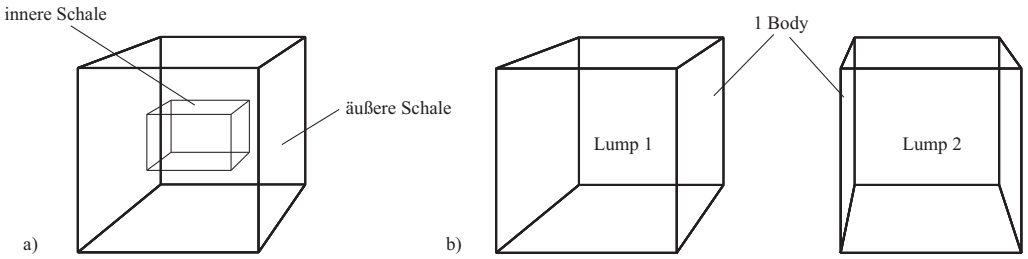


Abbildung 9.8: **(a)** Zur Darstellung von Hohlräumen in Körpern ist die Einführung von Schalen (engl. *shells*) in die Geometrie-Datenstruktur notwendig. **(b)** Sollen Körper auch aus mehreren nicht-zusammenhängenden Teilkörpern bestehen können, ist ein weiterer Datentyp notwendig. Für den ACIS-Modellierkern wurde die engl. Bezeichnung *lump* für einen zusammenhängenden Teilkörper gewählt.

(Weiler, 1988) als Schale (engl. shell) bezeichnet. Ein Körper besitzt demnach in jedem Fall eine Außenschale und optional eine oder mehrere Innenschalen.

Schließlich ist noch von Bedeutung, ob ein Körper auch aus mehreren unzusammenhängenden Teilkörpern bestehen können soll. Ein solcher Teilkörper wird häufig als *Lump* (dt. *Stück*, *Brocken*) bezeichnet. Ein *Body* kann entsprechend mehrere *Lumps* besitzen. Wie in Abschnitt 7.2.4 besprochen, ist dies zur Gewährleistung der Geschlossenheit des Typsystems bei Anwendung boolescher Operatoren notwendig. Das wird auch durch die entsprechenden Definitionen des räumlichen Typs *Body* in Abschnitt 7.3.4 reflektiert.

Fasst man die Erweiterungen der Datenstruktur um die Elemente *Loop*, *Shell* und *Lump* zusammen, ergibt sich das in Abb. 9.9 gezeigte Schema, das dem des kommerziellen Geometriekerns ACIS der Firma *Spatial*[16] entspricht. Geometriekerne stellen Funktionalitäten zum Erzeugen, Verwalten und Verarbeiten von BRep-Modellen zur Verfügung. ACIS beruht auf einer *Radial-Edge*-Datenstruktur und ist somit in der Lage, auch nicht-mannigfaltige Körper darzustellen.

ACIS bildet den Kern einer ganzen Reihe von professionellen CAD-Produkten, darunter AutoCAD von *AutoDesk*, ABAQUS/CAE von *Hibbitt, Karlsson & Sorensen*, CADKEY von *Baystate Technologies*, Grade/CUBE II von *Hitachi Zosen Information Systems*, GSCAD der *Intergraph Corporation*, IronCAD von *Visionary Design Systems*, MegaCAD der *Megatech Software GmbH* und *TurboCAD Solid Modeler* von IMSI.

Andere weit verbreitete 3D-Geometrie-Kernel sind der *Parasolid*-Kernel von *Unigraphics Solution* sowie der Open-Source-Kernel *OpenCascade*. Daneben gibt es eine Reihe von Eigenentwicklungen großer CAD-Hersteller, wie beispielsweise den *A-Modeler* von Autodesk. Dabei handelt es sich um einen auf der einfacheren Winged-Edge-Datenstruktur aufbauenden Geometriemodellierer für Körper mit ungekrümmten Oberflächen, der zusätzlich zu ACIS in AutoCAD integriert ist. Weitere Informationen zum *A-Modeler* sind der Arbeit von Neuberg (Neuberg, 2003) zu entnehmen.

[16] Spatial Corporation, USA. `http://www.spatial.com`

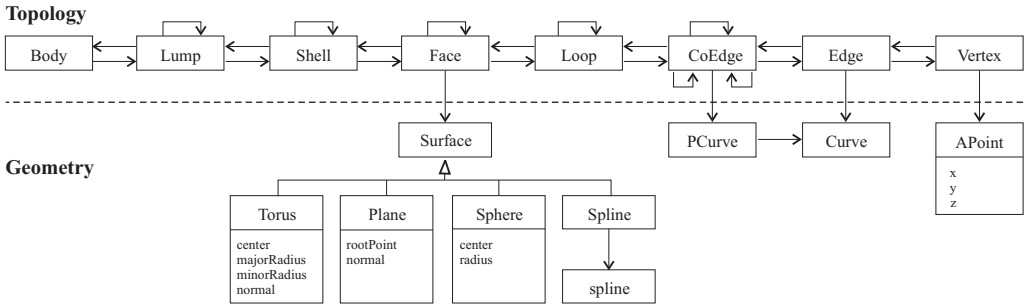


Abbildung 9.9: Das vom Geometrie-Kernel ACIS verwendete Klassenmodell zur Abbildung dreidimensionaler Objekte. Der obere Teil reflektiert vorwiegend topologische Zusammenhänge, während der untere geometrische Informationen beinhaltet.

Im Unterschied zum *A-Modeler* kann der ACIS-Modellierkern auch Körper mit gekrümmten Oberflächen und Flächen mit gekrümmter Berandung darstellen. Dazu werden neben den Punktkoordinaten zusätzliche geometrische Informationen vorgehalten. Wie Abb. 9.9 zeigt, wird dies im ACIS-Kernel über gesonderte Klassen realisiert: Das einem *Face*- zugeordnete *Surface*-Objekt modelliert die Geometrie der betreffenden Begrenzungsfläche und das einem *Edge*- zugeordnete *Curve*-Objekt beschreibt die Geometrie der betreffenden Kante.

Wegen der hohen Modelliermächtigkeit (Körper mit gekrümmten Oberflächen, nicht-mannigfaltige Körper), der weiten Verbreitung des ACIS-Kernels und der Existenz von weitreichenden Vorarbeiten der Arbeitsgruppe (van Treeck, 2004; Romberg, 2005) und der damit einhergehenden Expertise ist eine Entscheidung zugunsten einer am ACIS-Kern orientierten Datenstruktur für das Datenbankschema sinnvoll.

Im Rahmen dieser Arbeit soll sich allerdings auf geradlinige (ungekrümmte) räumliche Entitäten beschränkt werden. Aus diesem Grund kann ein vereinfachtes Schema für die Datenbank verwendet werden, in dem nur für die Knoten geometrische Informationen vorgehalten werden, für Kanten und Flächen hingegen nicht. Des Weiteren kann die Anzahl der Assoziationen zwischen den verschiedenen Elementen der BRep-Datenstruktur stark reduziert werden, da sich, wie bereits erwähnt, viele dieser Beziehungen aus anderen ableiten lassen und für das Datenbankschema die Vermeidung von Redundanz gegenüber dem schnellen Zugriff auf benachbarte Elemente Priorität hat. Führt man das Geometriemodell mit den räumlichen Datentypen von Abb. 9.4 zusammen, ergibt sich das in Abb. 9.10 dargestellte Datenbankschema.

Bei der Bearbeitung einer Anfrage, die räumliche Operatoren beinhaltet, wird mit Hilfe der in Abschnitt 8.2.2 beschriebenen Verfahren aus dem in der Datenbank vorgehaltenen oberflächenorientierten Modell ad-hoc eine oktalbaumbasierte Darstellung generiert. Für flächenförmige Objekte *(Surfaces)* muss zuvor noch ein Triangulationsschritt eingeschoben werden, da die in Abschnitt 8.2.2 beschriebenen Verfahren auf Dreiecken basieren.

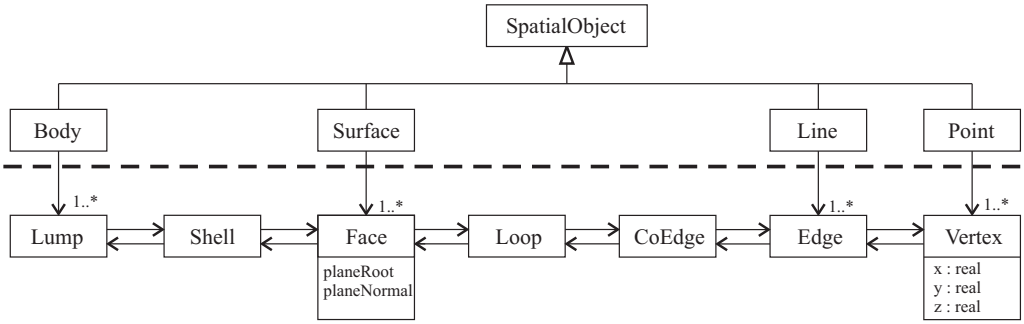


Abbildung 9.10: UML-Darstellung des Datenbank-Schemas, das ein an den Modellierkern ACIS angelegtes BRep-Datenmodell zur Speicherung der Geometrie der räumlichen Typen verwendet, sich aber auf ungekrümmte Geometrie beschränkt.

Verwandte Arbeiten im 3D-GIS-Bereich

Einige der aus dem GIS-Bereich bekannten Ansätze zur Speicherung dreidimensionaler Geometrie in einer Datenbank beruhen auf simplizialen Zerlegungstechniken, wie beispielsweise (Breunig, 1995; de la Losa & Cervelle, 1999) oder das in (Penninga *et al.*, 2006) vorgeschlagene *Tetrahedronized Irregular Network* (TEN). Darüber hinaus gibt es jedoch eine ganze Reihe von Arbeiten, die oberflächenorientierte Modelle benutzen: Die *3D Formal Data Structure* (Molenaar, 1990) basiert auf den vier Primitiven *node*, *arc*, *face* und *edge*, mit denen letztlich ein *vef*-Graph abgebildet wird. Die *Simplified Spatial Structure* (Zlatanova, 2000) verzichtet auf die Elemente *arc* und *edge*, was zwar zu einer Vereinfachung, aber auch zu einer geringeren Darstellungsmächtigkeit führt. Ähnliches gilt für das *Urban Data Model* von (Coors, 2003), bei dem die Oberflächen eines Körpers mit Hilfe von Dreiecken modelliert werden. Eine Datenstruktur mit weitaus höherer Darstellungsmächtigkeit wird in (Arens *et al.*, 2005) vorgestellt. Hier wird als grundlegendes Element das Polyeder eingesetzt. Weitere Details zu verschiedenen Geometrierepräsentationen in 3D-GIS sind (Zlatanova *et al.*, 2004) und (Zlatanova, 2006) zu entnehmen.

Kompatibilität mit dem IFC-Produktmodell

Für die Verwaltung und Speicherung von Gebäudemodellen des Hochbaus konnte sich in den letzten Jahren das Schema der *Industry Foundation Classes* (IFC) weitestgehend durchsetzen (siehe Abschnitt 2.2.3). Darin wird die Geometrie eines Bauteils in der Regel mit Hilfe seiner semantischen Attribute beschrieben. Ein vereinfachtes Beispiel für dieses Prinzip der *attribute-driven geometry* wäre eine Klasse *Wand* mit den Attributen *Dicke*, *Höhe* und *Länge*, aus denen die Kantenlängen des Hexaeders abgeleitet werden, der eine *Wand*-Instanz geometrisch repräsentiert. Methoden zur Gewinnung eines oberflächenbasierten Geometriemodells aus einer IFC-Produktmodellbeschreibung werden beispielsweise in den Arbeiten von Romberg (Romberg, 2005) und van Treeck (van Treeck, 2004) vorgestellt und sollen hier nicht weiter ausgeführt werden. Die Gewährleistung der Konsistenz zwischen geometriebezogenen semantischen Attributen und dem explizit gespeicherten BRep-Modell ist eine offene forschungsrelevante Frage.

9.3.5 Integration von geometrischem und semantischem Datenmodell

Ein wesentlicher Aspekt bei der Entwicklung und Umsetzung einer Anfragesprache für Bauwerksmodelle ist, dass Anfragen neben topologischen, metrischen und direktionalen auch nicht-geometrische Bedingungen enthalten dürfen.

Die in Kapitel 7 definierte räumliche Algebra beinhaltet zunächst nur Operatoren für räumliche Objekte. Damit lassen sich Anfragen formulieren wie beispielsweise:

- „Finde alle Körper im Umkreis von 10 m um Punkt X."
- „Berühren sich Körper A und Körper B?"
- „Befindet sich Körper A oberhalb von Punkt X?"

Durch die Definition eines nicht-geometrischen (semantischen) Datenmodells und einer entsprechenden Verknüpfung mit den vordefinierten räumlichen Typen sind semantisch höherwertige Anfragen möglich wie:

- „Finde alle Wände zwischen Decke 1 und Decke 2."
- „Finde alle Objekte in Raum 107."
- „Finde alle Rohrleitungen im Abstand von 15 cm von der Linie A–B."
- „Welche Stützen berühren Decke 1?"

Die Funktionalität der räumlichen Anfragesprache ist dabei unabhängig vom Datenmodell zur Beschreibung nicht-geometrischer (semantischer) Informationen[17]. Dies wird durch eine klare Trennung der räumlichen Typen und ihrer Operatoren von den Typen (Klassen) der Anwendungsdomäne erreicht (Abb. 9.11). Die Flexibilität seitens des nicht-geometrischen Datenmodells erhöht den Grad der Modularisierung und erleichtert damit die Wiederverwendbarkeit der bereitgestellten räumlichen Anfragefunktionalität in den unterschiedlichsten Bereichen, sowohl innerhalb des Bauwesens (Hochbau, Kraftwerksbau, Tunnelbau, Brückenbau) als auch außerhalb (Fahrzeugbau, Maschinenbau).

Da derzeit verfügbare Bauwerksmodelle in der Regel mit Hilfe der Datenbeschreibungssprache EXPRESS modelliert werden, ist eine Abbildung von EXPRESS auf ein objekt-relationales Schema notwendig. Ein solches Mapping wird beispielsweise in (Urban *et al.*, 2000) beschrieben. Dabei wird vor allem auch auf die Verwendung spezieller Funktionalitäten objekt-relationaler DBMS wie Memberfunktionen, Trigger und Stored Procedures zur Umsetzung von in EXPRESS definierten Constraints eingegangen. Urban *et al.* kommen zu dem Schluss, dass sich objekt-relationale Systeme sehr gut für die Verwaltung von den in Ingenieuranwendungen auftretenden komplexen Modellen eignen.

[17] im GIS-Bereich auch als *Sachdaten* bezeichnet

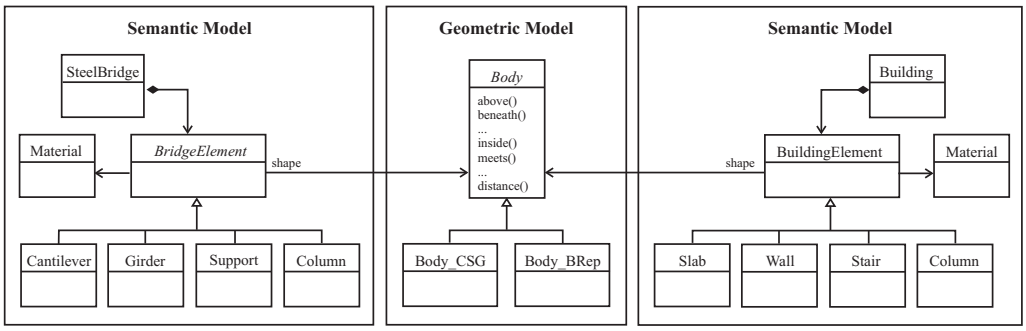


Abbildung 9.11: Durch die Trennung von semantischem und geometrischem Modell bleibt die Funktionalität der räumlichen Anfragesprache unabhängig von der Anwendungsdomäne. Links und rechts: Zwei unterschiedliche semantische Modelle. Mitte: Die Wahl des abstrakten Typs Body erlaubt die transparente Verwendung unterschiedlicher Geometrierepräsentationen.

9.4 Software-Prototyp

Zur Abschätzung typischer Anfragebearbeitungszeiten erfolgte eine prototypische Implementierung der im Kapitel 8 vorgestellten oktalbaumbasierten Algorithmen in der Programmiersprache Java und ihre Integration in ein Oracle-Datenbanksystem auf Basis der *Stored Procedures*-Technologie.

Die Abb. 9.12 und 9.13 zeigen beispielhaft die Visualisierung des Ergebnisses zweier Anfragen, die für das Strukturmodell eines Gebäudes durchgeführt wurden. Hinsichtlich der vergleichsweise langen Bearbeitungsdauer von 6.4 s bzw. 7.2 s ist darauf hinzuweisen, dass im Rahmen der prototypischen Implementation keinerlei zusätzlich beschleunigende Mechanismen wie räumliche Indizierung oder *Bounding Box*-basierte Tests zum Einsatz kommen. Wird beispielsweise die in Abb. 9.12 gezeigte Anfrage nicht innerhalb einer Datenbank ausgeführt, sondern direkt im Hauptspeicher der Visualisierungskomponente, benötigt ihre Bearbeitung lediglich 141 ms. Um vergleichbare Werte auch bei der Anfragebearbeitung durch das Datenbanksystem zu erreichen, sind weitere Entwicklungen notwendig, die vor allem auf eine reduzierte Anzahl von Zugriffen auf den Sekundärspeicher fokussieren müssen.

9.5 Zusammenfassung

In diesem Kapitel wurde aufgezeigt, wie auf Basis von objekt-relationalen Datenbanken eine räumliche Anfragesprache für digitale Bauwerksmodelle umgesetzt werden kann. Objekt-relationale Datenbanken erlauben die Erweiterbarkeit des Typsystems und die serverseitige Ausführung von anwendungsspezifischen Prozeduren bzw. Methoden. Sie bilden daher eine geeignete Grundlage für die Integration von räumlichen Datentypen und räumlichen Analysefunktionen. Darüber hinaus bieten sie mit SQL eine weit verbreitete und intuitiv verständliche Anfragesprache, die mit der Relationalen Algebra über solide mathematische Grundlagen verfügt.

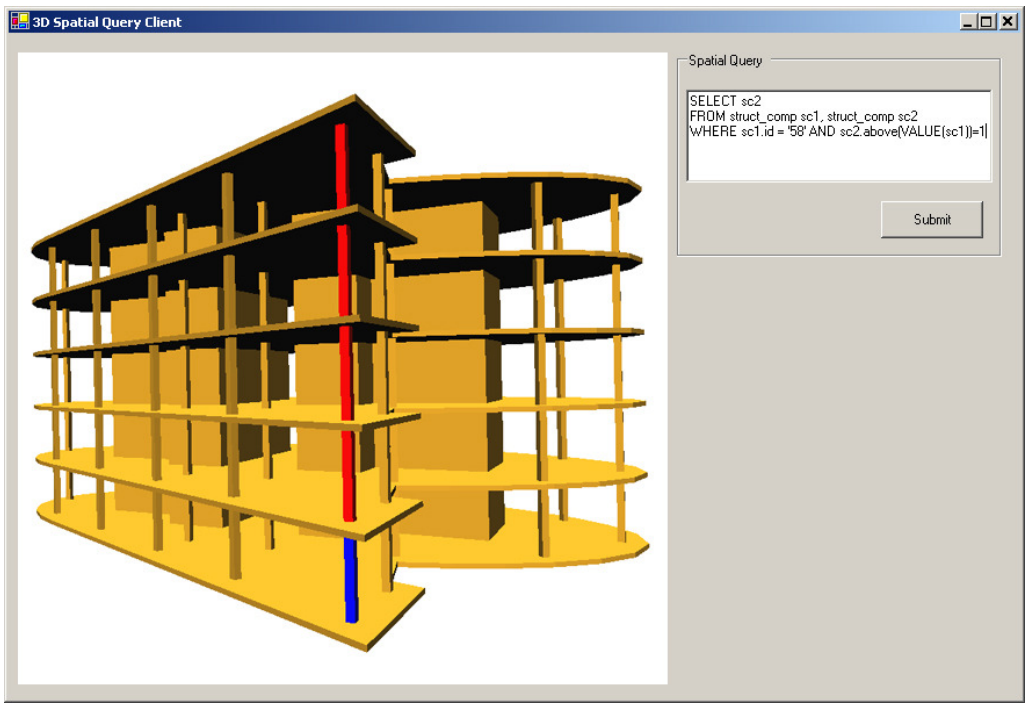


Abbildung 9.12: Das Tragwerksmodell eines Gebäudes mit 216 Bauteilen. Die Ergebnismenge der Anfrage zur Ermittlung aller über Bauteil 58 (blau markiert) liegenden Objekte ist rot gekennzeichnet. Zum besseren Verständnis sind sowohl das in der Datenbank gespeicherte Gesamtmodell als auch das durch die Anfrage erzeugte Partialmodell dargestellt. Die Anfrage wird von der Datenbank in 6.4 s bearbeitet. Dabei kommt der in Abschnitt 8.4.2 vorgestellte Algorithmus *above_proj_strict* als *Java Stored Procedure* in einem Oracle-Datenbanksystem zum Einsatz. Da keine *Bounding Box*-basierten Tests vorgeschaltet sind, müssen alle 216 Objekte mit Hilfe des oktalbaumbasierten Algorithmus untersucht werden. Bei der Laufzeitmessung wurde das Oracle-DBMS auf einem Pentium 4-Rechner mit 3.0 GHz Taktfrequenz ausgeführt, die Datenübertragung zwischen Datenbank-Server und -Client erfolgte mittels 100 MBit-Ethernet.

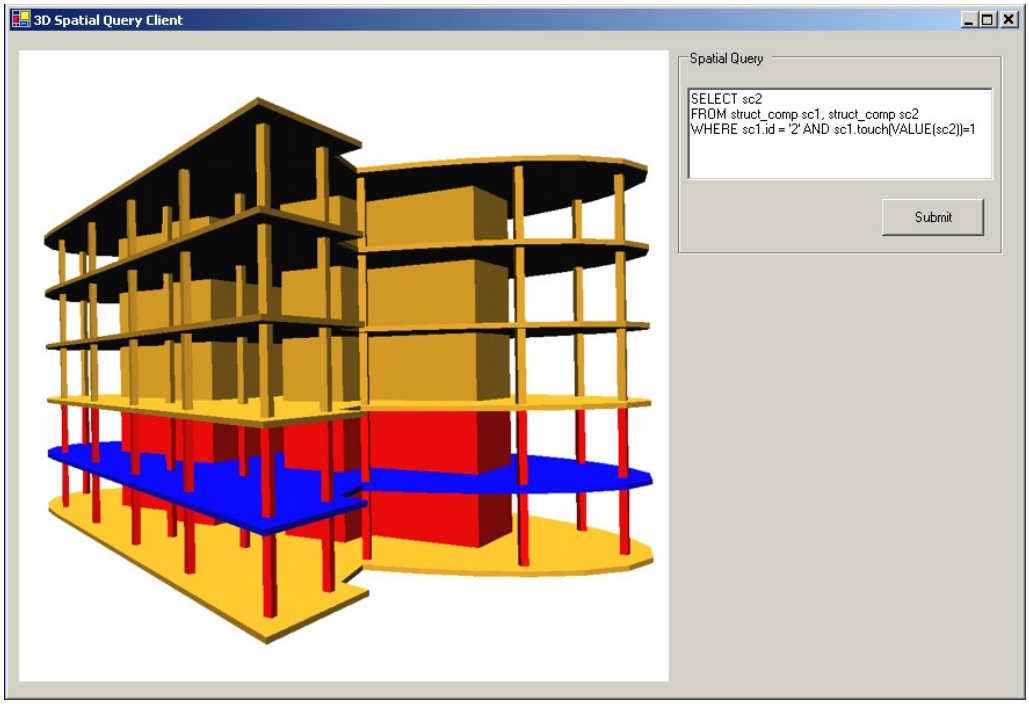


Abbildung 9.13: Die Anwendung des Operators *touch* für das blau markierte Deckenbauteil (Bauteil 2) liefert die rot markierte Ergebnismenge. Die Anfrage wird von der Datenbank in 7.2 s bearbeitet, wobei der der in Abschnitt 8.5 beschriebene Algorithmus zum Einsatz kommt, wiederum ohne vorgeschaltete *Bounding Box*-Tests. Das Hardware-Setup ist identisch zu dem von Abb. 9.13.

Für die Speicherung der Geometrie der räumlichen Objekte in der Datenbank empfiehlt sich die Verwendung eines BRep-Modells, da auf diese Weise die Anbindung an bereits bestehende Anwendungssoftware wie CAD- oder FEM-Programme vereinfacht und ein erhöhter Aufwand bei der Konvertierung der Geometriemodelle vermieden werden kann. Wegen seiner weiten Verbreitung und umfassenden Modelliermächtigkeit vor allem in Hinblick auf nicht-mannigfaltige und gekrümmte Geometrien wurde eine an den ACIS-Modellierkern angelehnte Datenstruktur gewählt.

Die Integration der räumlichen Funktionalitäten mit einem anwendungsspezifischen semantischen Datenmodell erlaubt die Verknüpfung von geometrischen mit nicht-geometrischen Bedingungen und dadurch semantisch höherwertige Anfragen.

Kapitel 10

Zusammenfassung und Ausblick

Diese Arbeit hat zwei Formen der computergestützten Zusammenarbeit bei der Planung von Bauwerken intensiv untersucht: Die *synchrone* Kooperation auf Basis eines dreidimensionalen Mehrbenutzereditors, bei der die Beteiligten verschiedener Fachdisziplinen in einer verteilten Sitzung zusammenkommen können, um gemeinsam Planungsentscheidungen unter Zuhilfenahme interaktiver Simulationen zu treffen, und die *asynchrone* Kooperation auf Grundlage einer räumlichen Datenbank, die die Dekomposition eines Bauwerkmodells anhand geometrisch-topologischer Kriterien erlaubt und damit die Voraussetzung für weitgehend konfliktfreies paralleles Arbeiten schafft.

10.1 Collaborative Computational Steering

In Teil I dieser Arbeit wurden Methoden und Techniken zur Umsetzung eines *Collaborative Computational Steering*-Systems vorgestellt. Ein solches System dient der Unterstützung der synchronen Zusammenarbeit zwischen Ingenieuren der gleichen oder unterschiedlicher Fachdisziplinen.

Unter *Computational Steering* versteht man die interaktive Steuerung von Simulationen, d.h. die Veränderung von Randbedingungen und Parametern zur Laufzeit der Simulation. Bei Verwendung derartiger steuerbarer Simulationen entfallen zeitaufwändige manuelle Konvertierungs- und Datenaufbereitungsschritte, was zur drastischen Verkürzung des gesamten Optimierungszyklus führt. Bei der hier betrachteten Klasse von *Computational Steering*-Systemen steht vor allem die Veränderbarkeit von geometrischen Randbedingungen zur Laufzeit der Simulation im Mittelpunkt.

Bei der vorgestellten *Collaborative Computational Steering*-Plattform CoCoS handelt es sich um ein verteiltes Softwaresystem mit einem zentralen Kollaborationsserver, der u.a. die Verwaltung des gemeinsam verwendeten Modells übernimmt, einer beliebigen Anzahl von Simulationsservern, die die Simulationsdaten

zur Verfügung stellen, und einer ebenfalls beliebigen Anzahl von Clients, die als Frontend für die Teilnehmer der kollaborativen Session dienen. Die Kommunikation zwischen den verteilten Komponenten der CoCoS-Plattform ist auf Basis des CORBA-Standards umgesetzt.

Ein wesentliches Merkmal von CoCoS ist das gemeinsam genutzte Modell, das ein für alle Beteiligte identisches geometrisches Modell mit domänenspezifischen semantischen Modellen verknüpft. Dabei können die verwendeten semantischen Modelle flexibel an die Bedürfnisse der jeweiligen Domäne bzw. Simulationsanwendung angepasst werden. Hierzu wurde ein explizit verfügbares Metamodell in die Schnittstellen des Kollaborationsservers integriert.

Die Arbeitsweise der Kollaborationsplattform wurde anhand des Anwendungsszenarios *Interaktive Komfortanalyse / TGA-Planung* erläutert. Der für eine interaktive Strömungssimulation notwendige Simulationsserver basiert auf der Gitter-Boltzmann-Methode, die dank der Verwendung eines kartesischen Gitters eine vollständig automatisierte Diskretisierung der Geometrie von Strömungshindernissen erlaubt. Darüber hinaus zeigt die Gitter-Boltzmann-Methode einige für die Parallelisierung günstige algorithmische Eigenschaften, die das Laufzeitverhalten des Simulationsservers auf einem Höchstleistungsrechner positiv beeinflusst. Die dadurch ermöglichte Steigerung der Rechenleistung erlaubt eine weitere Verkürzung der Reaktionszeiten der interaktiven Simulation bzw. eine Steigerung der mit ihr erzielbaren Genauigkeit.

10.2 Spatial Query Language

In Teil II der vorliegenden Arbeit wurde das Konzept einer räumlichen Anfragesprache zur Unterstützung asynchroner Phasen kooperativer Arbeit im Planungsprozess vorgestellt. Im Rahmen des verfolgten Integrationsansatzes ist es mit Hilfe dieser deklarativen Sprache möglich, Teile eines Bauwerkmodells aus einem Gesamtmodell zu extrahieren, die bestimmten, durch den Nutzer spezifizierbaren geometrisch-topologischen Bedingungen genügen. Dieses Teilmodell kann innerhalb eines kooperativen Szenarios einzelnen Planungsbeteiligten exklusiv zur Verfügung gestellt werden, womit eine präzise Separation von Zuständigkeiten auf die Modellebene abgebildet wird. Es kann aber ebenso als Datenquelle für Berechnungs- und Simulationsprogramme dienen.

Zur Spezifikation der Ergebnismenge stellt die Anfragesprache topologische, direktionale und metrische Operatoren bereit, für die im Rahmen einer abstrakten Algebra Definitionen auf Grundlage von Punktmengentheorie und Punktmengentopologie gegeben wurden. Darüber hinaus wurde eine mögliche Implementierung dieser Operatoren auf Basis der hierarchischen raumpartitionierenden Datenstruktur Oktalbaum aufgezeigt. Nahezu alle der vorgestellten Algorithmen besitzen rekursiv-iterativen Charakter und eröffnen durch eine sukzessiv verfeinerte Auflösung der Geometrie die Möglichkeit einer Abwägung zwischen dem benötigten Rechenaufwand und der erzielbaren Genauigkeit.

Schließlich wurde die Verwendung von objekt-relationalen Datenbanken zur Umsetzung räumlicher Anfragefunktionalität motiviert. Die besondere Eignung dieses Datenbank-Typs stützt sich vor allem auf die Integrierbarkeit räumlicher Datentypen in die Standardanfragesprache SQL durch Objekt-Typen, deren serverseitig ausgeführte Methoden zur Auswertung räumlicher Operatoren dienen können. Ein Schlüsselaspekt für eine nahtlose Integration räumlicher Anfragefunktionalität in bestehende Arbeitsabläufe ist die Verwendung eines adäquaten Geometriemodells und dessen Abbildung in der Datenbank. Das interne Schema des ACIS-Modellierkerns stellt dank seiner ausgeprägten Modelliermächtigkeit und weiten Verbreitung in kommerziellen CAD-Systemen eine besonders geeignete Option dar.

10.3 Ausblick

Für beide kooperationsunterstützende Ansätze gibt es weiteren Entwicklungsbedarf. Hinsichtlich des in Teil I vorgestellten Systems ist es wünschenswert, weitere interaktive Simulationstypen in das System zu integrieren, um die Tragfähigkeit des vorgeschlagenen Konzepts auch für eine große Zahl unterschiedlicher Simulationsserver zu prüfen. Darüber hinaus gilt es, die Interaktivität des bereits entwickelten interaktiven Strömungssimulators durch eine weitere Erhöhung der Performanz zu verbessern. Für das Anwendungsszenario *Komfortanalyse* ist zudem die Integration eines physikalischen Modells zur Abbildung von Wärmeströmungen vonnöten.

Auch für das in Teil II vorgestellte System ist eine Steigerung der Performanz wünschenswert, um zum einen die Anfrageverarbeitung zu beschleunigen und zum anderen Anfragen auch für große Modelle zu ermöglichen. Um die durch Datenbanksysteme bereitgestellten Möglichkeiten der Anfrageoptimierung zu nutzen, ist die Entwicklung von Kostenmodellen für die in Kapitel 8 vorgestellten Algorithmen notwendig. Für eine effiziente Anfragebearbeitung ist des Weiteren die Integration von räumlichen Indizierungsstrukturen wie Oktal- oder R-Bäume unbedingt erforderlich.

Schließlich ist eine Erweiterung der oktalbaumbasierten Algorithmen zur Implementierung räumlicher Operatoren auf gekrümmte Oberflächen denkbar. Dies kann beispielsweise über einen Facettierungszwischenschritt oder über eine direkte Voxelisierung der Oberfläche mit anschließendem Oktalbaumaufbau realisiert werden. Die für die Abbildung gekrümmter Geometrie notwendige Erweiterbarkeit des Datenbankschemas ist durch die Wahl einer am ACIS-Modellierkern orientierten Datenstruktur gegeben.

Eine wesentliche Herausforderung besteht in der Integration des semantischen Bauwerkmodells mit seiner expliziten Geometrierepräsentation in einer gemeinsamen räumlichen Datenbank. Besonderes Augenmerk muss hierbei auf die Konsistenzsicherung zwischen den geometriebeschreibenden Attributen eines Bauteil-Objekts und der tatsächlichen geometrischen Ausprägung des Objekts gelegt

werden. Dies kann beispielsweise über die durch die Datenbank bereitgestellte Trigger-Technologie gewährleistet werden.

Schließlich lassen sich die am Lehrstuhl für Bauinformatik der TU München entwickelten Techniken zur automatisierten Extraktion von Raumluftvolumen (van Treeck & Rank, 2004, 2007) in die räumliche Datenbank integrieren, was die Verwendbarkeit dieser Volumen als Objekte in räumlichen Anfragen ermöglicht. Auch die Bereitstellung räumlicher Operatoren in alternativen Anfragesprachen wie die im Kontext des *Semantic Web* entstandene Sprache SPARQL zur Abfrage von RDF-Ontologien birgt enormes Forschungspotential (Beetz *et al.*, 2007).

Literaturverzeichnis

Abdelmoty, A. (1995). *Modelling and Reasoning in Spatial Databases: A Deductive Object-Oriented Approach.* Dissertation, Department of Computing and Electrical Engineering, Heriot-Watt University, Edinburgh, Scotland.

Adachi, Y. (2002). Overview of IFC Model Server Framework. In: *Proc. of the 4th Europ. Conf. on Product and Process Modeling.*

Akenine-Möller, T. (2001). Fast 3D triangle-box overlap testing. *Journal of Graphics Tools* 6(1), S. 29–33.

Allen, J. (1983). Maintaining Knowledge about Temporal Intervals. *Communications of the ACM* 26(11), S. 832–843.

Anand, S. & Knott, K. (1991). An algorithm for converting the boundary representation of a CAD model to its octree representation. *Computers & Industrial Engineering* 21, S. 343–347.

Anumba, C. J., Cutting-Decelle, A. F., Baldwin, A. N., Dufau, J., Mommessin, M. & Boughlaghem, N. M. (1998). Integration of Product and Process Models as a Keystone of Concurrent Engineering in Construction. In: *Proc. of the 2nd Europ. Conf. on Product and Process Modeling.*

Anupam, V., Bajaj, C., Bernardini, F., Cutchin, S., Chen, J., Schikore, D., Xu, G., Zhang, P. & Zhang, W. (1994). Scientific problem solving in a distributed and collaborative multimedia environment. *Mathematics and Computers in Simulation* 36(4-6), S. 433–442.

Arens, C., Stoter, J. & van Oosterom, P. (2005). Modelling 3D spatial objects in a geo-DBMS using a 3D primitive. *Computers & Geosciences* 31(2), S. 165–177.

Armstrong, M. A. (1983). *Basic Topology.* Springer Verlag.

Aschwanden, P., Ohler, T., Pajarola, R. & Szabo, K. (1995). Experiences with embedding a spatial access structure into an object-oriented DBMS. Forschungsbericht, Institute of Theoretical Computer Science, ETH Zürich.

Augenbroe, G. (1995). An overview of the COMBINE project. In: *Proc. of the 1st Europ. Conf. on Product and Process Modelling in the Building Industry.*

Ayala, D., Brunet, P., Juan, R. & Navazo, I. (1985). Object representation by means of nonminimal division quadtrees and octrees. *ACM Transactions on Graphics* 4(1), S. 41–59.

Baecker, R., Nastos, D., Posner, I. & Mawby, K. (1993). The user-centered iterative design of collaborative writing software. In: *Proc. of the SIGCHI Conf. on Human factors in computing systems*, S. 399–405.

Bakkeren, W. J. C. & Tolman, F. P. (1995). Integrating structural synthesis and evaluation using product models. In: *Proc. of the 6th Int. Conf. on Computing in Civil and Building Engineering (ICCCBE).*

Balovnev, O., Bode, T., Breunig, M., Cremers, A., Müller, W., Pogodaev, G., Shumilov, S., Siebeck, J., Siehl, A. & Thomson, A. (2004). The story of the GeoToolKit - An object-oriented Geodatabase kernel system. *GeoInformatica* 8(1), S. 5–47.

Baumgart, B. (1975). Winged-Edge Polyhedron Representation for Computer Vision. In: *Proc. of the National Computer Conf.*

Beetz, J., de Vries, B. & van Leeuwen, J. (2007). RDF-based distributed functional part specifications for the facilitation of service-based architectures. In: *Proc. of the 24th CIB-W78 Conf. on Information Technology in Construction.*

Beetz, K., Junge, R. & Steinmann, R. (1998). The O.P.E.N. Platform. In: *Proc. of the 2nd Europ. Conf. on Product and Process Modelling in the Building Industry (ECPPM).*

Behr, T. & Güting, R. (2005). Fuzzy Spatial Objects: An Algebra Implementation in SECONDO. In: *Proc. of the 21st Int. Conf. on Data Engineering.*

Berkhahn, V., Klinger, A., Rüppel, U., Meißner, U. F., Greb, S. & Wagenknecht, A. (2005). Processes Modelling in Civil Engineering based on Hierarchical Petri Nets. In: *Proc. of the 22nd CIB-W78 Conf.*

Bhatnagar, P., Gross, E. & Krook, M. (1954). A model for collision processes in gases. *Physical Review* 94(3), S. 511–525.

Björk, B.-C. (1995). Requirements and information structures for building product data models. Forschungsbericht, VTT Technical Research Centre of Finland.

Björk, B.-C. (2001). Document management - a key technology for the construction industry. In: *Information and Communications Technology in the Practice of Building and Civil Engineering - Proc. of the 2nd worldwide ECCE Symposium.*

Bode, A. (2006). Multicore-Architekturen. *Informatik-Spektrum* 29(5), S. 349–352.

Booch, G. (1994). *Object-oriented Analysis and Design with Applications.* Benjamin-Cummings Publishing.

Borghoff, U. & Schlichter, J. (2000). *Computer-supported cooperative work: Introduction to distributed applications.* Springer.

Borrmann, A., Hauschild, T. & Hübler, R. (2004). Integration of Constraints into Digital Building Models for Cooperative Planning Processes. In: *Proc. of the 10th Int. Conf. on Computing in Civil and Building Engineering.*

Borrmann, A., van Treeck, C. & Rank, E. (2006). Towards a 3D Spatial Query Language for Building Information Models. In: *Proc. of the Joint Int. Conf. for Computing and Decision Making in Civil and Building Engineering.*

Boulanger, P., Garcia, M. J., Badke, C. & Ryan, J. (2006). An Advanced Collaborative Infrastructure for the Real-Time Computational Steering of Large CFD Simulations. In: *Proc. of the ECCOMAS CFD 2006 Conf.*

Bresenham, J. E. (1965). Algorithm for Computer Control of a Digital Plotter. *IBM Systems Journal* 4, S. 1.

Bretschneider, D. (1998). *Modellierung rechnerunterstützter, kooperativer Arbeit in der Tragwerksplanung.* Dissertation, Ruhr-Universität Bochum.

Breunig, M. (1995). *Integration räumlicher Informationen für Geo-Informationssysteme.* Dissertation, Friedrich-Wilhelms-Universität Bonn.

Breunig, M., Bode, T. & Cremers, A. (1994). Implementation of Elementary Geometric Database Operations for a 3D-GIS. In: *Proc. of the 6th Int. Symp. on Spatial Data Handling.*

Breunig, M., Cremers, A., Müller, W. & Siebeck, J. (2001). New methods for topological clustering and spatial access in object-oriented 3D databases. In: *Proc. of the 9th ACM Int. Symp. on Advances in Geographic Information Systems.*

Brodlie, K., Wood, J., Duce, D. & Sagar, M. (2004). gViz: Visualization and Computational Steering on the Grid. In: *Proc. of the UK e-Science All Hands Meeting 2004.*

Brooke, J., Eickermann, T. & Woessner, U. (2003). Application Steering in a Collaborative Environment. In: *Proc. of the 2003 ACM/IEEE Conf. on Supercomputing (SC '03).*

Brooke, J. M., Coveney, P. V., Harting, J., Jha, S., Pickles, S. M., Pinning, R. L. & Porter, A. R. (2003). Computational Steering in RealityGrid. In: *Proc. of the UK e-Science All Hands Meeting.*

Bryson, S. & Levit, C. (1992). The Virtual Wind Tunnel. *IEEE Computer Graphics and Applications* 12(4), S. 25–34.

Bungartz, H.-J., Griebel, M. & Zenger, C. (2002). *Einführung in die Computergraphik: Grundlagen, Geometrische Modellierung, Algorithmen.* Vieweg Verlag.

Capin, T., Pandzic, I., Thalmann, N. & Thalmann, D. (1998). Realistic Avatars and Autonomous Virtual Humans in VLNET Networked Virtual Environments. In: R. Earnshaw & J. Vince (Hrsg.), *Virtual Worlds in the Internet*, S. 157–173. IEEE Computer Society Press.

Carlbom, I., Cakravarty, I. & Vanderschel, D. (1985). A hierarchical data structure for representing the spatial decomposition of 3D objects. *IEEE Computer Graphics and Applications* 5(4), S. 24–31.

Cattell, R. G. G. & Barry, D. K. (2000). *The Object Data Standard. ODMG 3.0.* Morgan Kaufmann Publishers.

Chamberlin, D. D., Astrahan, M. M., Eswaran, K. P., Griffiths, P., Lorie, R. A., Mehl, J., Resiner, P. & Wade, B. W. (1976). SEQUEL2: A unified approach to Data Definition, Manipulation and Control. *IBM Journal Research and Development* 20(6), S. 560–575.

Chan, E. & Zhu, R. (1996). QL/G: A Query Language for Geometric Databases. In: *Proc. of the Int. Conf. on GIS in Urban, Environmental and Regional Planning.*

Chapman, S. & Cowling, T. (1939). *The mathematical theory of non-uniform gases.* Cambridge University Press.

Chen, H. & Bishop, J. (1997). Delaunay Triangulation for Curved Surfaces. In: *Proc. of the 6th Int. Meshing Roundtable.*

Chen, H., Wang, F., J.S. & Yang, S. (2000). GeoSQL: A spatial query language of objectoriented GIS. In: *Proc. of the 2nd Int. Workshop on Computer Science and Information Technology.*

Chin, J., Harting, J., Jha, S., Coveney, P. V., Porter, A. R. & Pickles, S. M. (2003). Steering in computational science: mesoscale modelling and simulation. *Contemporary Physics* 44(5), S. 417–434.

Christiansson, P., Da Dalto, L., Skjaerbaek, J., Soubra, S. & Marache, M. (2002). Virtual Environments for the AEC sector - The Divercity experience. In: *Proc. of the 4th Europ. Conf. on Product and Process Modelling.*

Churchill, E., Snowdon, D. & Munro, A. (2002). *Collaborative Virtual Environments: Digital Places and Spaces for Interaction.* Springer.

Clarke, J. (2001). *Energy Simulation in Building Design.* Butterworth-Heinemann.

Clementini, E. & Di Felice, P. (1995). A Comparison of Methods for Representing Topological Relationships. *Information Sciences - Applications* 3(3), S. 149–178.

Clementini, E. & Di Felice, P. (1996). A model for representing topological relationships between complex geometric features in spatial databases. *Information Sciences* 90(1-4), S. 121–136.

Clementini, E., Di Felice, P. & van Oosterom, P. (1993). A Small Set of Formal Topological Relationships Suitable for End-User Interaction. In: *Proc. of the 3rd Int. Symposium on Advances in Spatial Databases*, S. 277–295.

Codd, E. (1970). A Relational Model of Data for Large Shared Data Banks. *Commun. ACM* 13(6), S. 377–387.

Coors, V. (2003). 3D-GIS in Networking Environments. *Computers, Environment and Urban Systems* 27(4), S. 345–357.

Crouse, B. (2003). *Lattice-Boltzmann Strömungssimulationen auf Baumdatenstrukturen*. Dissertation, Lehrstuhl für Bauinformatik, Technische Universität München.

Date, C. (2003). *An Introduction to Database Systems* (8th Aufl.). Addison-Wesley.

de Berg, M., van Kreveld, M., Overmars, M. & Schwarzkopf, O. (2000). *Computational Geometry*. Springer.

de la Losa, A. & Cervelle, B. (1999). 3D Topological modeling and visualisation for 3D GIS. *Computers & Graphics* 23(4), S. 469–478.

Debras, P., Monceyron, J., Bauer, F. & Ballesta, P. (1998). An Information Server for the Building Industry: Interest of a STEP/IFC Based Approach. In: *Proc. of the 2nd Europ. Conf. on product and process modelling in the building industry*.

Denis, A., Perez, C. & Ribes, A. (2003). Parallel CORBA objects for programming computational grids. *Distributed Systems Online* 4(2).

Dewan, P. & Shen, H. (1998). Controlling access in multiuser interfaces. *ACM Transactions on Computer-Human Interaction (TOCHI)* 5(1), S. 34–62.

d'Humières, D., Ginzburg, I., Krafczyk, M., Lallemand, P. & Luo, L. (2002). Multiple-relaxation-time lattice Boltzmann models in three dimensions. *Philosophical Transactions: Mathematical, Physical and Engineering Sciences* 360(1792), S. 437–451.

Dieker, S. & Güting, R. (2000). Plug and Play with Query Algebras: SECONDO. A Generic DBMS Development Environment. In: *Proc. of the Int. Database Engineering and Applications Symp.*

Döllner, G., Kellner, P. & Tegel, O. (2000). Digital Mock-Up and Rapid Prototyping in Automotive Product Development. *J. of Integrated Design and Process Science* 4(1), S. 55–66.

Dourish, P. & Bellotti, V. (1992). Awareness and coordination in shared workspaces. In: *Proc. of the ACM Conf. on Computer-Supported Cooperative Work*, S. 107–114.

Düster, A., Demkowicz, L. & Rank, E. (2006). High-order finite elements applied to the discrete Boltzmann equation. *Int. Journal for Numerical Methods in Engineering* 67, S. 1094–1121.

Eastman, C. (1999). *Building Product Models: Computer Environments Supporting Design and Construction*. CRC Press.

Egenhofer, M. (1987). An extended SQL syntax to treat spatial objects. In: *Proc. of the 2nd Int. Seminar on Trends and Concerns of Spatial Sciences*.

Egenhofer, M. (1991). Reasoning about Binary Topological Relations. In: *Proc. of the 2nd Symp. on Advances in Spatial Databases (SSD'91)*.

Egenhofer, M. (1992). Why not SQL! *Journal of Geographical Information Systems* 6(2), S. 71–85.

Egenhofer, M. (1994). Spatial SQL: A Query and Presentation Language. *IEEE Trans. Knowl. Data Eng.* 6(1), S. 86–95.

Egenhofer, M., Frank, A. & Jackson, J. P. (1989). A Topological Data Model for Spatial Databases. In: *Proc. of the 1st Int. Symp. on the Design and Implementation of Large Spatial Databases*.

Egenhofer, M. & Franzosa, R. (1991). Point-Set Topological Spatial Relations. *Int. Journal of Geographical Information Systems* 5(2), S. 161–174.

Egenhofer, M. & Herring, J. (1990). A Mathematical Framework for the Definition of Topological Relationships. In: *Proc. of the 4th Int. Symp. on Spatial Data Handling*.

Egenhofer, M. & Herring, J. (1992). Categorizing Binary Topological Relations Between Regions, Lines, and Points in Geographic Databases. Forschungsbericht, Department of Surveying Engineering, University of Maine.

Ellis, C. A., Gibbs, S. J. & Rein, G. (1991). Groupware: Some Issues and Experiences. *Communications of the ACM* 34(1), S. 39–58.

Elmasri, R. & Navathe, S. B. (2003). *Fundamentals of Database Systems, Fourth Edition*. Boston, MA, USA: Addison-Wesley Longman.

Ericson, C. (2004). *Real Time Collision Detection*. Morgan Kaufmann.

Esnard, A., Richart, N. & Coulaud, O. (2006). A Steering Environment for Online Parallel Visualization of Legacy Parallel Simulations. In: *Proc. of the 10th IEEE Int. Symp. on Distributed Simulation and Real-Time Applications (DS-RT'06)*, Washington, DC, USA, S. 7–14. IEEE Computer Society.

Fahrig, T., Krafczyk, M., Nachtwey, B. & Tölke, J. (2006). Enhanced Computational HVAC Simulations using Software Agent Technology. In: *Proc. of the Joint Int. Conf. on Computing and Decision Making in Civil and Building Engineering*.

Ferziger, J. H. & Peric, M. (2001). *Computational Methods for Fluid Dynamics*. Springer.

Firmenich, B. (2002). *CAD im Bauplanungsprozess: Verteilte Bearbeitung einer strukturierten Menge von Objektversionen*. Dissertation, Bauhaus-Universität Weimar.

Fischer, M. (2000). Benefits of four-dimensional (4D) models for facility owners and AEC service providers. In: *Proc. of the 6th Construction Congress*, S. 20–22.

Foley, J., van Dam, A., Feiner, S. & Hughes, J. (1996). *Computer graphics (2nd ed. in C): principles and practice*. Addison-Wesley Longman Publishing Co., Inc.

Forkert, T., Kersken, H.-P., Schreiber, A., Strietzel, M. & Wolf, K. (2000). The Distributed Engineering Framework TENT. In: *Proc. of the 4th Int. Conf. on Vector and Parallel Processing (VECPAR 2000)*.

Foster, I. & Kesselman, C. (1999). *The Grid: Blueprint for a new computing infrastructure*. Morgan Kaufmann Publishers Inc.

Frank, A. C. (2000). *Organisationsprinzipien zur Integration von geometrischer Modellierung, numerischer Simulation und Visualisierung*. Dissertation, Fakultät Informatik, Technische Universität München.

Frank, A. U. (1992). Qualitative Reasoning about Distances and Directions in Geographic Space. *Journal of Visual Languages and Computing* 3(4), S. 343–371.

Frank, A. U. (1996). Qualitative Spatial Reasoning: Cardinal Directions as an Example. *Int. Journal of Geographic Information Systems* 10(3), S. 269–290.

Fuhr, T., Socher, G., Scheering, C. & Sagerer, G. (1998). A three-dimensional Spatial Model for the Interpretation of Image Data. In: P. Olivier & K. Gapp (Hrsg.), *Representation and Processing of Spatial Expressions*, S. 103–118. Mahwah, NJ, USA: Lawrence Erlbaum Associates.

Gaal, S. (1964). *Point Set Topology*. Academic Press.

Gabrielaitis, L. & Baušys, R. (2006). Electronic Document Management in Building Design. *J. of Civil Engineering and Management* 12, S. 103–108.

Gargano, M., Nardelli, E. & Talamo, M. (1991). Abstract Data Types for the Logical Modeling of Complex Data. *Information Systems* 16(5), S. 1991.

Geller, S., Tölke, J. & Kafzcyk, M. (2006). Lattice-Boltzmann Method on quadtree type grids for Fluid-Structure-Interaction. In: H.-J. Bungartz & M. Schäfer (Hrsg.), *Fluid-Structure Interaction: Modelling, Simulation, Optimisation*. Springer-Verlag.

Ginzburg, I. & d'Humières, D. (1996). Local second-order boundary methods for lattice Boltzmann models. *J. Stat. Phys.* 84, S. 927–971.

Goos (1997). *Vorlesungen über Informatik. Band 1: Grundlagen und funktionales Programmieren*. Springer-Verlag.

Goyal, R. (2000). *Similarity Assessment for Cardinal Directions between Extended Spatial Objects*. Dissertation, University of Maine.

Greb, S. (2005). *Petri-Netz-basierte Modellierung und Analyse von Bauplanungsprozessen*. Dissertation, Institut für Numerische Methoden und Informatik im Bauwesen, Technische Universität Darmstadt.

Greb, S. & Klauer, T. (2004). Prozessmodellierung und Workflow-Management im Bauwesen. In: *16. Forum Bauinformatik*.

Greenberg, S. & Marwood, D. (1994). Real time groupware as a distributed system: concurrency control and its effect on the interface. In: *Proc. of the ACM Conf. on Computer supported cooperative work*, S. 207–217.

Greenberg, S., Roseman, M., Webster, D. & Bohnet, R. (1992). Human and technical factors of distributed group drawing tools. *Interact. Comput.* 4(3), S. 364–392.

Greenhalgh, C. & Benford, S. (1995). MASSIVE: A collaborative virtual environment for teleconferencing. *ACM Transactions on Computer-Human Interaction (TOCHI)* 2(3), S. 239–261.

Gregory, A., Lin, M. C., Gottschalk, S. & Tylor, R. (1999). H-Collide: A framework for Fast and Accurate Collision Detection for Haptic Interaction. In: *Proc. of Virtual Reality Conf. 1999*.

Gröger, G., Reuter, M. & Plümer, L. (2004). Representation of a 3-D City Model in Spatial Object-Relational Databases. In: *Proc. of the 20th ISPRS Congress*.

Güting, R. (1994). An Introduction to Spatial Database Systems. *VLDB Journal* 3(4), S. 357–399.

Güting, R. & Schneider, M. (1995). Realm-Based Spatial Data Types: The ROSE Algebra. *VLDB Journal* 4(2), S. 100–143.

Güting, R. H. (1988). Geo-Relational Algebra: A Model and Query Language for Geometric Database Systems. In: *Proc. of the Int. Conf. on Extending Database Technology*.

Güting, R. H., Almeida, V., Ansorge, D., Behr, T., Ding, Z., Höse, T., Hoffmann, F. & Spiekermann, M. (2005). SECONDO: An Extensible DBMS Platform for Research Prototyping and Teaching. In: *Proc. of the 21st Conf. on Data Engineering*.

Guddat, H. & Straube, A. M. (2003). Digitale Produktentwicklung in verteilten immersiven Virtual-Environment-Engineering-Konferenzen. In: J. Gausemeier (Hrsg.), *Augmented und Virtual Reality in der Produktentstehung: Grundlagen, Methoden und Werkzeuge.*, S. 277–289. Heinz Nixdorf Institut, Paderborn.

Guesgen, H. (1989). Spatial Reasoning Based on Allen's Temporal Logic. Forschungsbericht, Int. Computer Science Institue, Berkley, CA, USA.

Haase, H., Dai, F., Strassner, J. & Gbel, M. (1997). Immersive Investigation of Scientific Data. In: G. Nielson, H. Hagen, & H. Müller (Hrsg.), *Scientific Visualization, Overviews, Methodologies, and Techniques*, S. 35–58. Washington, DC, USA: IEEE Computer Society.

Hagsand, O. (1996). Interactive MultiUser VEs in the DIVE System. *IEEE Multimedia Magazine* 3(1), S. 30–39.

Harlow, F. H. & Welch, J. E. (1965). Numerical calculation of time-dependent viscous incompressible flow of fluid with a free surface. *The Physics of Fluids* 8, S. 2182–2189.

Hartmann, D. (2000). *DFG-Schwerpunktprogramm 694: Objektorientierte Modellierung in Planung und Konstruktion, Abschlußbericht über das Forschungsprogramm 1991-1998.* Wiley-VCH, Weinheim.

Hatcher, A. (2001). *Algebraic Topology.* Cambridge University Press.

Hauschild, T. (2003). *Computer Supported Cooperative Work - Applikationen in der Bauwerksplanung auf Basis einer integrierten Bauwerksmodellverwaltung.* Dissertation, Bauhaus-Universität Weimar.

Hauschild, T., Borrmann, A. & Hübler, R. (2003). Techniken der Verwaltung dynamischer digitaler Bauwerksmodelle. In: *Tagungsband des Int. Kolloquiums über Anwendungen der Informatik und Mathematik in Architektur und Bauwesen (IKM).*

Henning, M. & Vinoski, S. (1999). *Advanced CORBA programming with C++.* Addison-Wesley.

Herring, J., Larsen, R. & Shivakumar, J. (1988). Extensions to the SQL language to support spatial analysis in a topological data base. In: *Proc. of GIS/LIS '88.*

Hjelt, M. & Björk, B.-C. (2006). Experiences of EDM usage in Construction Processes. *ITcon* 11, S. 113–125.

Hong, J. (1994). *Qualitative Distance and Direction Reasoning in Geographic Space.* Dissertation, Department of Surveying Engineering, University of Maine, Orono, ME, USA.

Hou, S., Sterling, J., Chen, S. & Doolen, G. (1996). A lattice-Boltzmann subgrid model for high Reynolds number flows. *Fields Inst. Comm.* 6, S. 151–165.

Hovestadt, V. & Hovestadt, L. (1999). The Armilla Project. *Automation in Construction* 8(3), S. 325–337.

Huang, Z., Svensson, P. & Hauska, H. (1992). Solving Spatial Analysis Problems with GeoSAL, a Spatial Query Language. In: *Proc. of the 6th Int. Working Conf. on Scientific and Statistical Database Management.*

Huhnt, W., Racky, P. & Holzer, S. M. (2005). Logic of Processes in Civil Engineering. In: *Proc. of the 22nd CIB-W78 Int. Conf. on Information Technology in Construction.*

Hunter, G. (1978). *Efficient computation and data structures for graphics.* Phd thesis, Princeton University.

IAI - International Alliance for Interoperability (2004). *Industry Foundation Classes.*

Ingram, K. & Phillips, W. (1987). Geographic information processing using a SQL-based query language. In: *Proc. of the 8th Int. Symp. on Computer-Assisted Cartography.*

Jackins, C. L. & Tanimoto, S. L. (1980). Oct-trees and their use in representing three-dimensional objects. *Computational Graphics and Image Processing* 14(3), S. 249–270.

Jaksch, S. (2001). Facettierung dreidimensionaler Gebiete und Gittergenerierung unter Verwendung von Octree-Datenstrukturen. Diplomarbeit, Lehrstuhl für Bauinformatik, Technische Universität München.

Johansen, R. (1988). *Groupware: Computer Support for Business Teams.* Groupware: Computer Support for Business Teams.

Jung, D. & Gupta, K. K. (1996). Octree-Based Hierarchical Distance Maps for Collision Detection. In: *Proc. of the 1996 IEEE Conf. on Robotics and Automation.*

Junge, R., Köthe, M., Schulz, K., Zarli, A. & Bakkeren, W. (1997). The Vega Platform - IT for the Virtual Enterprise. In: *Proc. of CAAD Futures.*

Katranuschkov, P. (1994). COMBI: Integrated Product Model. In: *Proc. of the 1st Europ. Conf. on Product and Process Modelling in the Building Industry.*

Katranuschkov, P. & Hyvärinen, J. (1998). Product Data Server for Concurrent Engineering in AEC. In: *Proc. of 2nd Europ. Conf. on Product and Process Modeling in the Building Industry.*

Keahey, K. & Gannon, D. (1997). PARDIS: A Parallel Approach to CORBA. In: *Proc. of the 6th IEEE Int. Symposium on High Performance Distributed Computing*, S. 31–39.

Kela, A. (1989). Hierarchical octree approximations for boundary representation-based geometric models. *Computer-Aided Design* 21(6), S. 355–362.

Kemper, A. & Moerkotte, G. (1994). *Object-Oriented Database Management: Applications in Engineering and Computer Science*. Englewood Cliffs, NJ, USA: Prentice Hall.

Khemlani, L. & Kalay, Y. E. (1997). An Integrated Computing Environment for Collaborative, Multi-Disciplinary Building Design. In: *Proc. of the 7th Int. Conf. on Computer Aided Architectural Design Futures*.

Kühner, S. (2003). *Virtual Reality - basierte Analyse und interaktive Steuerung von Strömungssimulationen im Bauingenieurwesen*. Dissertation, Lehrstuhl für Bauinformatik, Technische Universität München.

Kühner, S., Crouse, B., Rank, E., Tölke, J. & Krafczyk, M. (2004). From a product model to visualization: Simulation of Indoor Flows with Lattice-Boltzmann Methods. *Computer-Aided Civil and Infrastructure Engineering* 19(6), S. 411–420.

Khoshafian, S. & Abnous, R. (1990). *Object Orientation: Concepts, Languages, Databases, User Interfaces*. John Wiley & Sons.

Klauer, T. (2005). *Eine prozessorientierte Kooperationsplattform für Bauprojekte auf Basis eines internetbasierten Workflow-Managements*. Dissertation, Technische Universität Darmstadt.

Klimetzek, F. (2001). Virtual Intuitive Simulation Testbed VISiT. Forschungsbericht, Daimler-Chrysler AG, Research and Development.

Klinger, A. (1971). Patterns and search statistics. In: J. S. Rustagi (Hrsg.), *Optimizing Methods in Statistics*, S. 303–307. Academic Press, New York, USA.

König, M. (2003). *Ein Prozessmodell für die kooperative Gebäudeplanung*. Dissertation, Institut für Bauinformatik, Universität Hannover.

Kohl, J. A., Wilde, T. & Bernholdt, D. E. (2006). Cumulvs: Interacting with High-Performance Scientific Simulations, for Visualization, Steering and Fault Tolerance. *Int. Journal of High Performance Computing Applications* 20(2), S. 255–285.

Koo, B. & Fischer, M. (2006). Feasibility Study of 4D CAD in Commercial Construction. *Journal of Construction Engineering and Management* 126(4), S. 251–260.

Kopyssov, S., Krasnopyorov, I., Novikov, A. & Rytchkov, V. (2003). Parallel Distributed Object-Oriented Framework for Domain Decomposition. In: *Proc. of the 15th Int. Conf. on Domain Decomposition Methods*.

Kovalevsky, V. A. (1989). Finite Topology as applied to Image Analysis. *Computer Vision, Graphics and Image Processing* 46, S. 141–161.

Kowalczyk, W. (1997). *Ein interaktiver Modellierer für evolutionäre Produktmodelle*. Dissertation, Lehrstuhl für Bauinformatik, Technische Universität München.

Krafczyk, M. (1995). *Simulation von Strömungen mit Gittergasmethoden.* Dissertation, Universität Dortmund.

Krafczyk, M. (2001). *Gitter-Boltzmann-Methoden: Von der Theorie zur Anwendung.* Habilitation, Lehrstuhl für Bauinformatik, Technische Universität München.

Kreylos, O., Tesdall, A., Hamann, B., Hunter, J. & Joy, K. (2002). Interactive visualization and steering of CFD simulations. In: *Proc. of the Symposium on Data Visualisation (VISSYM)*, S. 25–34.

Kriegel, H.-P., Pfeifle, M., Pötke, M., Renz, M. & Seidl, T. (2003). Spatial Data Management for Virtual Product Development. *Lecture Notes in Computer Science* 2598, S. 216–230.

Krishnan, R., Das, A. & Gurmoorthy, B. (1996). Octree encoding of B-Rep based objects. *Computers & Graphics* 20(1), S. 107–114.

Lam, K., Mahdavi, A., Gupta, S., Wong, N., Brahme, R. & Kang, Z. (2002). Integrated and distributed computational support for building performance evaluation. *Advanced Engineering Software* 33(4), S. 199–206.

Leavitt, N. (2000). Whatever Happened to Object-Oriented Databases? *Computer* 33(8), S. 16–19.

Lee, B. & Musick, R. (2004). MeshSQL: the query language for simulation mesh data. *Information Sciences* 159(3-4), S. 177–202.

Lee, K. Y., Price, M. & Armstrong, C. (2004). ViSiCADE - a virtual simulation environment for seamless integration of CAD/CAE into VR. In: *Proc. of the Irish Signals and Systems Conf.*

Liere, R., Mulder, J. & Wijk, J. (1997). Computational Steering. *Future Generation Computer Systems* 12(5), S. 441–450.

Linebarger, J., Janneck, C. & Kessler, G. (2003). Shared Simple Virtual Environment: An object-oriented framework for highly-interactive group collaboration. In: *Proc. of the 7th IEEE Int. Symposium on Distributed Simulation and Real Time Applications*, S. 170–180.

Lorie, R. & Meier, A. (1984). Using relational DBMS for geographical databases. *Geo-Processing* 2, S. 243–257.

Luksch, P., Rathmayer, S. & Sunderam, V. (2000). Internet-based Collaborative Simulation in Computational Prototyping and Scientific research. In: *Proc. of the Int. Conf. on Parallel and Distributed Processing Techniques and Applications.*

Macedonia, M. R., Zyda, M. J., Pratt, D. R., Barham, P. T. & Zeswitz, S. (1994). NPSNET: A Network Software Architecture for Large-Scale Virtual Environments. *Presence: Teleoperators and Virtual Environments* 3(3), S. 265–287.

Mahdavi, A., Ilal, M., Mathew, P., Ries, R., Suter, G. & Brahme, R. (1999). The Architecture of S2. In: *Proc. of Building Simulation '99, 6th Int. IBPSA Conf.*

Mahler, S. (2003). Erzeugung und Evaluierung von Oktalbaumstrukturen als Schnittstelle zu CAD-Programmen. Diplomarbeit, Fakultät Informatik, Elektrotechnik und Informationstechnik, Universität Stuttgart.

Matheu, N. F. (2005). *Life cycle document management system for construction.* Dissertation, Universitat Polytècnica de Catalunya.

Mavriplis, D. J. (2004). Multigrid Solution of the Steady-State Lattice Boltzmann Equation. In: *Proc of Int. Conf. for Mesoscopic Methods in Engineering and Science (ICMMES).*

McKenney, M., Pauly, A., Praing, R. & Schneider, M. (2005). Dimension-Refined Topological Predicates. In: *Proc. of the 13th annual ACM Int. Worksh. on Geographic Information Systems.*

McKinney, K., Kim, J., Fischer, M. & Howard, C. (1996). Interactive 4D-CAD. In: *Proc. of the 3rd Congress on Computing in Civil Engineering.*

Meagher, D. (1982). Geometric modeling using octree encoding. *IEEE Computer Graphics and Image Processing* 19(2), S. 129–147.

Meissner, U., Peters, F. & Rüppel, U. (1995). Graphically interactive object-oriented product modeling of structures. In: *Proc. of the 6th Int. Conf. on Computing in Civil and Building Engineering (ICCCBE).*

Melton, J. (2003). *Advanced SQL:1999. Understanding Object-Relational and Other Advanced Features.* Morgan Kaufmann, San Francisco, USA.

Mäntylä, M. (1988). *An Introduction to Solid Modelling.* Computer Science Press.

Molenaar, M. (1990). A formal data structure for 3D vector maps. In: *Proc. of the 1st Europ. Conf. on Geographical Information Systems (EGIS).*

Moran, T., McCall, K., van Melle, B., Peddersen, E. & Halasz, F. (1995). Some design principles of sharing in Tivoli, a whiteboard meeting support tool. In: S. S. Greenberg & R. Rada (Hrsg.), *Groupware for Real-time Drawing: A Designer's guide*, S. 24–36. McGraw-Hill.

Mortenson, M. (1985). *Geometric Modeling.* Wiley.

Mulder, J. D. & van Wijk, J. J. (1995). 3D computational steering with parametrized geometric objects. In: *Proc. of the IEEE Conf. on Visualization.*

Mundani, R.-P. (2005). *Hierarchische Geometriemodelle zur Einbettung verteilter Simulationsaufgaben.* Dissertation, Universität Stuttgart.

Mundani, R.-P., Bungartz, H.-J., Rank, E., Romberg, R. & Niggl, A. (2003). Efficient Algorithms for Octree-Based Geometric Modelling. In: *Proc. of the 9th Int. Conf. on Civil and Structural Engineering Computing.*

Mungee, S., Surendran, N. & Schmidt, D. (1999). The design and performance of a CORBA audio/video streaming service. In: *Proc. of the 32nd Annual Hawaii Int. Conf. on System Sciences.*

Muralidhar, R. & Parashar, M. (2000). An interactive Object Infrastructure for Computational Steering of Distributed Simulations. In: *Proc. of the 9th IEEE Int. Symposium on High Performance Distributed Computing.*

Neuberg, F. (2003). *Ein Softwarekonzept zur Internet-basierten Simulation des Ressourcenbedarfs von Bauwerken.* Dissertation, Lehrstuhl für Bauinformatik, Technische Universität München.

Newman-Wolfe, R. & Pelimuhandiram, H. (1991). MACE: a fine grained concurrent editor. *ACM SIGOIS Bulletin* 12(2-3), S. 240–254.

Niggl, A. (2007). *Tragwerksplanung am volumenorientierten Gesamtmodell - Ein Ansatz zur Verbesserung der computergestützten Zusammenarbeit im konstruktiven Ingenieurbau.* Dissertation, Lehrstuhl für Bauinformatik, Technische Universität München.

Object Management Group (2004). *The Common Object Request Broker: Architecture and Specification. Revision 3.0.* OMG Document formal/2004-03-12.

Ooi, B., Sacks-Davis, R. & McDonell, K. (1989). Extending a DBMS for geographic applications. In: *Proc. of the IEEE 5th Int. Conf. on Data Engineering.*

OpenGIS Consortium (OGC) (1999). OGC Abstract Specification.

Oracle Corporation (2006). Oracle announces innovative Collaborative Building Information Management initiative for the construction industry. Press Release.

Orenstein, J. A. (1990). An Object-Oriented Approach to Spatial Data Processing. In: *Proc. of the 4th Int. Symp. on Spatial Data Handling.*

Ozel, F. (2000). Spatial Databases and the Analysis of Dynamic Processes in Buildings. In: *Proc. of the 5th Conf. on Computer Aided Architectural Design Research in Asia.*

Pacheco, P. S. (1996). *Parallel programming with MPI.* San Francisco, CA, USA: Morgan Kaufmann Publishers Inc.

Papadias, D., Sellis, T., Theodoridis, Y. & Egenhofer, M. (1995). Topological Relations in the World of Minimum Bounding Rectangles: A Study with R-Trees. In: *Proc. of the 1995 ACM SIGMOD Int. Conf. on Management of Data.*

Paul, N. & Bradley, P. E. (2003). Topological Houses. In: *Proc. of the 16th Int. Conf. of Computer Science and Mathematics in Architecture and Civil Engineering (IKM 2003).*

Paul, N. & Bradley, P. E. (2005). Relationale Datenbanken für die Topologie architektonischer Räume. In: *17. Forum Bauinformatik*.

Pavlidis, T. (1981). Contour Filling in Raster Graphics. In: *Proc. of the 8th annual Conf. on Computer graphics and interactive techniques*.

Pekkola, S., Robinson, M., Korhonen, J., Hujala, S., Toivonen, T. & Saarinen, M.-J. (2000). An architecture for virtual reality, audio, video, text, and document handling in applications supporting multi-person interactions. In: *Proc. of the 26th Euromicro Conf.*, S. 150–157.

Penninga, F., van Oosterom, P. & Kazar, B. M. (2006). A TEN-based DBMS approach for 3D Topographic Data Modelling. In: *Proc. of the 12th Int. Symposium on Spatial Data Handling*.

Peuquet, D. & Zhan, C.-X. (1987). An Algorithm to Determine the Directional Relationship Between Arbitrarily-Shaped Polygons in the Plane. *Pattern Recognition* 20(1), S. 65–74.

Pfaffinger, M., van Treeck, C., Borrmann, A., Wenisch, P. & Rank, E. (2007). An interactive thermal fluid simulator for the design of HVAC systems. In: *Proc. of the ASCE Int. Workshop on Computing in Civil Engineering*.

Pfeifle, M. (2004). *Spatial Database Support for Virtual Engineering*. Dissertation, Ludwig-Maximilians-Universität München.

Picard, S. L. D., Degrande, S. & Gransart, C. (2001). A CORBA based platform as communication support for synchronous Collaborative Virtual Environment. In: *Proc. of the Int. Workshop on Multimedia Middleware*.

Pickles, S., Haines, R., Pinning, R. & Porter, A. (2005). A practical toolkit for computational steering. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences* 363(1833), S. 1843–1853.

Piegl, L. & Richard, A. (1995). Tesselating trimmed NURBS surfaces. *Computer Aided Design* 27(1), S. 16–26.

Plale, B., Elling, V., Eisenhauer, G., Schwan, K., King, D. & Martin, V. (1999). Realizing Distributed Computational Laboratories. *Int. Journal of Parallel and Distributed Systems and Networks* 2(3), S. 180–190.

Priol, T. (2002). Programming the Grid with Distributed Objects. In: *Proc. of the Workshop on Performance Analysis and Distributed Computing*.

Pötke, M. (2001). *Spatial Indexing for Object-Relational Databases*. Dissertation, Fakultät Mathematik und Informatik, Ludwig-Maximilians-Universität München.

Pullar, D. (1988). Data Definition and Operators on a Spatial Data Model. In: *Proc. of the ACSM-ASPRS Annual Convention*.

Pullar, D. & Egenhofer, M. (1988). Towards Formal Definitions of Topological Relations Among Spatial Objects. In: *Proc. of the 3rd Int. Symposium on Spatial Data Handling*.

Rank, E., Borrmann, A., Düster, A., Niggl, A., Nübel, V., Romberg, R., Scholz, D., van Treeck, C. & Wenisch, P. (2005). From Adaptivity to Computational Steering: The Long Way of Integrating Numerical Simulation into Engineering Design Processes. In: *Proc. of the Int. Conf. on Adaptive Modeling and Simulation (ADMOS)*.

Rantzau, D. & Lang, U. (1998). A scalable virtual environment for large scale scientific data analysis. *Future Generation Computer Systems* 14(3-4), S. 215–222.

Retz-Schmidt, G. (1988). Various Views on Spatial Prepositions. *AI Magazine* 9(2), S. 95–105.

Romberg, R. (2005). *Gebäudemodell-basierte Strukturanalyse im Bauwesen*. Dissertation, Lehrstuhl für Bauinformatik, Technische Universität München.

Romberg, R., Niggl, A., van Treeck, C. & Rank, E. (2004). Structural Analysis based on the Product Model Standard IFC. In: *Proc. of the 10th Int. Conf. on Comp. in Civil and Building Engineering (ICCCBE-X)*.

Rosenman, M. & Gero, J. (1996). Modelling multiple views of design objects in a collaborative CAD environment. *Computer-Aided Design* 28(3), S. 193–205.

Rosenthal, A., Heiler, S. & Manola, F. (1984). An Example of Knowledge-Based Query Processing in a CAD/CAM DBMS. In: *Proc. of the 10th Int. Conf. on Very Large Data Bases (VLDB)*, S. 363–370.

Roussopoulos, N., Faloutsos, C. & Sellis, T. (1988). An Efficient Pictorial Database System for PSQL. *IEEE Transactions on Software Engineering* 14(5), S. 639–650.

Rüppel, U. (Hrsg.) (2007). *Vernetzt-kooperative Planungsprozesse im Konstruktiven Ingenieurbau*. Springer.

Rüppel, U. & Klauer, T. (2004). Internet-based Workflow Management for Civil Engineering Projects. In: *Proc. of the 10th Int. Conf. on Computing in Civil and Building Engineering*.

Samet, H. (1989). *Applications of Spatial Data Structures: Computer Graphics, Image Processing and GIS*. Addison-Wesley.

Sauter, C., Morger, O., Mühlherr, T., Hutchison, A. & Teufel, S. (1995). CSCW for Strategic Management in Swiss Enterprises: an Empirical Study. In: *Proc. of the 4th Europ. Conf. on Computer Supported Cooperative Work (ECSCW)*, S. 115–130.

Scherer, R. J. (1995). EU-project COMBI - Objectives and overview. In: *Proc. of the 1st Europ. Conf. on Product and Process Modeling in the Building Industry*.

Scheugenpflug, S. (2005). *Relationale und Objektrelationale Datenbankkonzepte in Geoinformationssystemen.* Dissertation, Institut für Geodäsie, Fachgebiet Geoinformationssysteme, Technische Universität München.

Schneider, M. (1997). *Spatial Data Types for Database Systems: Finite Resolution Geometry for Geographic Information Systems*, Volume 1288 of *Lecture Notes in Computer Science.* Springer Verlag.

Schneider, M. & Behr, T. (2006). Topological relationships between complex spatial objects. *ACM Transactions on Database Systems* 31(1), S. 39–81.

Schneider, M. & Weinrich, B. (2004). An abstract model of three-dimensional spatial data types. In: *Proc. of the 12th annual ACM Int. Workshop on Geographic Information Systems (GIS'04).*

Scholz, D., Düster, A. & Rank, E. (2004). Fully three-dimensional modelling of fluidstructure interaction problems by using high order finite elements. In: *Proc. of PVP.*

Schubert, H. (1968). *Topologie.* Teubner.

Shan, X. & He, X. (1998). Discretization of the velocity space in solution of the Boltzmann equation. *Physical Review Letters* 80, S. 65–68.

Sharma, J. (1996). *Integrated Spatial Reasoning in Geographic Information Systems: Combining Topology and Direction.* Dissertation, Department of Spatial Information science and Engineering, University of Maine, Orono, ME, USA.

Shekhar, S. & Chawla, S. (2003). *Spatial Databases: A Tour.* Pearson Eduction.

Shekhar, S., Liu, X. & Chawla, S. (1999). An Object Model of Direction and Its Implications. *GeoInformatica* 3(4), S. 357–379.

Shephard, M., Beall, M., O'Bara, R. & Webster, B. (2004). Toward simulation-based design. *Finite Elements in Analysis and Design* 40(12), S. 1575–1598.

Shephard, M. S. & Georges, M. K. (1991). Automatic Three-Dimensional Mesh Generation by the Finite Octree Technique. *Int. Journal for Numerical Methods in Engineering* 32, S. 209–749.

Sherman, W. R. & Craig, A. B. (2002). *Understanding Virtual Reality: Interface, Application and Design.* Morgan Kaufmann Publishers.

Shi, W., Yang, B. & Li, Q. (2003). An object-oriented data model for complex objects in three-dimensional geographical information systems. *Int J. of Geographical Information Science* 17(5), S. 411–430.

Singhal, S. & Zyda, M. (1999). *Networked Virtual Environments: Design and implementation.* Addison Wesley Longman.

Smith, A. R. (1979). Tint Fill. In: *Proc. of the 6th annual Conf. on Computer graphics and interactive techniques.*

Stannkey (2002). *Handbook of Virtual Environments*. Lawrence Erlbaum Associates.

Steinmann, F. (1997). *Modellbildung und computergestütztes Modellieren in frühen Phasen des architektonischen Designs*. Dissertation, Bauhaus-Universität Weimar.

Sunderam, V., Cheung, S. Y., Hirsch, M., Chodrow, S., Grigni, M., Krantz, A., Rhee, I., Gray, P., Olesen, S., Hutto, P. & Sult, J. (1998). CCF: Collaborative Computing Frameworks. In: *Proc. of the 1998 ACM/IEEE Conf. on Supercomputing*, Washington, DC, USA, S. 1–6. IEEE Computer Society.

Svensson, P. & Huang, Z. (1991). GeoSAL: A Query Language for Spatial Data Analysis. In: *Proc. of the 2nd Int. Symp. on Advances in Spatial Databases*.

Tamminen, M. & Samet, H. (1984). Efficient octree conversion by connectivity labeling. In: *SIGGRAPH '84: Proc. of the 11th annual Conf. on Computer graphics and interactive techniques*, New York, NY, USA, S. 43–51. ACM Press.

Tanenbaum, A. & van Stean, M. (2002). *Distributed Systems. Principles and Paradigms*. Prentice Hall.

Tchon, K.-F., Hirsch, C. & Schneiders, R. (1997). Octreebased Hexahedral Mesh Generation for Viscous Flow Simulations. In: *Proc. of the 13th AIAA Computational Fluid Dynamics Conf.*.

Teufel, S., Sauter, C., Mühlherr, T. & Bauknecht, K. (1995). *Computerunterstützung für die Gruppenarbeit*. Addison-Wesley, Bonn.

Tilove, R. B. (1980). Set Membership Classification: A Unified Approach to Geometric Intersection Problems. *IEEE Transactions on Computers* C-29, S. 874–883.

Tölke, J. (2001). *Die Lattice-Boltzmann-Methode für Mehrphasenströmungen*. Dissertation, Lehrstuhl für Bauinformatik, Technische Universität München.

Tolman, F. & Poyet, P. (1995). The ATLAS models. In: *Proc. of the 1st Europ. Conf. on Product and Process Modelling in the Building Industry*.

Türker, C. (2003). *SQL:1999 & SQL2000. Objekt-relationales SQL, SQLJ & SQL/XML*. dPunkt Verlag.

Turk, Z., Dolenc, M., Nabrzyski, J., Katranuschkov, P., Balaton, E., Balder, R. & Hannus, M. (2004). Towards engineering on the grid. In: *Proc.of the 5th Europ. Conf. on Product and Process Modeling in the Building and Construction Industry (ECPPM)*.

Urban, S., Tjahjadi, M. & Shah, J. (2000). A Case Study in Mapping Conceptual Designs to Object-Relational Schemas. *Concurrency: Practice and Experience* 12(9), S. 863–907.

van der Aalst, W. M. P. (2002). Workflow Management in Construction: Opportunities for the Future. In: *Proc. of the 19th CIB-W78 Conf.*

van Oosterom, P., Vertegaal, W., van Hekken, M. & Vijlbrief, T. (1994). Integrated 3D modelling within a GIS. In: *Proc. of the Workshop on Advanced Geographic Data Modelling.*

van Treeck, C. (2004). *Gebäudemodell-basierte Simulation von Raumluftströmungen.* Dissertation, Lehrstuhl für Bauinformatik, Technische Universität München.

van Treeck, C. & Rank, E. (2004). Analysis of Building Structure and Topology Based on Graph Theory. In: *Proc. of the 10th Int. Conf. on Computing in Civil and Building Engineering (ICCCBE-X).*

van Treeck, C. & Rank, E. (2007). Dimensional reduction of 3D building models using graph theory and its application in building energy simulation. *Engineering with Computers* 23(2), S. 109–122.

Vijlbrief, T. & van Oosterom, P. (1992). The GEO++ System: An extensible GIS. In: *Proc. of the 5th Int. Symp. on Spatial Data Handling.*

Weatherill, N. P. & Hassan, O. (1994). Efficient three-dimensional Delaunay triangulation with automatic point creation and imposed boundary constraints. *Int. J. for Numerical Methods in Engineering* 37, S. 2005–2039.

Wei, G., Ping, Z. & Jun, C. (1998). Topological data modelling for 3D GIS. In: *Proc.of ISPRS Commission IV Symp. on GIS.*

Weiß, T. (2005). Theoretische und praktische Untersuchung dreidimensionaler Analysefunktionalität. Diplomarbeit, Informations- und Wissensverarbeitung, Bauhaus-Universität Weimar.

Weiler, K. (1988). The radial-edge structure: a topological representation for non-manifold geometric boundary conditions. In: M. Wozny, H. McLaughlin, & J. Encarnacao (Hrsg.), *Geometric Modeling for CAD Applications*, S. 3–36. Elsevier Science Publishers.

Wenisch, P. (2008). *Computational Steering of CFD Simulations on Terraflop-Supercomputers.* Dissertation, Lehrstuhl für Bauinformatik, Technische Universität München.

Wenisch, P., van Treeck, C., Borrmann, A., Rank, E. & Wenisch, O. (2007). Computational Steering on Distributed Systems: Indoor Comfort Simulations as a Case Study of Interactive CFD on Supercomputers. *Int. Journal of Parallel, Emergent and Distributed Systems* 22(4), S. 275–291.

Wenisch, P. & Wenisch, O. (2004). Fast octree-based Voxelization of 3D Boundary Representation-Objects. Forschungsbericht, Lehrstuhl für Bauinformatik, Technische Universität München.

Wenisch, P., Wenisch, O. & Rank, E. (2005a). Harnessing High-Performance Computers for Computational Steering. In: *Proc. of the 12th Europ. Parallel Virtual Machine and Message Passing Interface Conf.*

Wenisch, P., Wenisch, O. & Rank, E. (2005b). Optimizing an interactive CFD simulation on a supercomputer for Computational Steering in a Virtual Reality Environment. In: A. Bode & F. Durst (Hrsg.), *High Performance Computing in Science and Engineering*, S. 83–93. Springer.

Wierse, A. (1995). Performance of the COVISE visualization system under different conditions. In: *Proc. of the SPIE '95 Conf. on Visual Exploration and Analysis.*

Wilde, T., Kohl, J. A. & Flanery, R. E. (2003). Immersive and 3D viewers for CUMULVS: VTK/CAVE and AVS/Express. *Future Generation Computer Systems* 19(5), S. 701–719.

Willenbacher, H. (2002). *Interaktive verknüpfungsbasierte Bauwerksmodellierung als Integrationsplattform für den Bauwerkslebenszyklus.* Dissertation, Bauhaus-Universität Weimar.

Wilson, S., Sayers, H. & McNeill, M. D. J. (2001). Using CORBA middleware to support the development of distributed virtual environment applications. In: *Proc. of the 9th Int. Conf. in Central Europe on Computer Graphics, Visualization and Computer Vision.*

Winter, S. (1995). Topological Relations between Regions in Raster. In: *Proc. of the 4th Int. Symp. on Advances in Spatial Databases.*

Wolf-Gladrow, D. (2000). *Lattice-Gas Cellular Automata and Lattice Boltzmann Models.* Springer Verlag.

Worboys, M. F. & Bofakos, P. (1993). A Canonical Model for a Class of Areal Spatial Objects. In: *Proc. of the 3rd Int. Symp. on Advances in Spatial Databases (SSD'93).*

Wu, D., Hou, Y., Zhu, W., Zhang, Y.-Q. & Peha, J. (2001). Streaming video over the Internet: approaches and directions. *IEEE Transactions on Circuits and Systems for Video Technology* 11(3), S. 282–300.

Zachmann, G. (2000). *Virtual Reality in Assembly Simulation - Collision Detection, Simulation Algorithms, and Interaction Techniques.* Dissertation, Fachbereich Informatik, Technische Universität Darmstadt.

Zanetti, G. & McNamara, G. R. (1988). Use of the Boltzmann equation to simulate lattice-gas automata. *Phys. Rev. Lett.* 61, S. 2332–2335.

Zienkiewicz, O. C. & Taylor, R. L. (2005). *The Finite Element Method* (6 Aufl.). Butterworth Heinemann.

Zimmermann, K. & Freksa, C. (1996). Qualitative Spatial Reasoning Using Orientation, Distance, and Path Knowledge. *Applied Intelligence* 6(1), S. 49–58.

Zlatanova, S. (2000). On 3D Topological Relationships. In: *Proc. of the 11th Int. Workshop on Database and Expert Systems Applications*.

Zlatanova, S. (2006). 3D geometries in spatial DBMS. In: *Proc. of the Int. Worksh. on 3D Geoinformation 2006*.

Zlatanova, S., Rahman, A. & Shi, W. (2004). Topological models and frameworks for 3D spatial objects. *Journal of Computers & Geosciences* 30(4), S. 419–428.