Räumliches Schließen in einer kartographischen Datenbasis

Diplomarbeit im Fach Informatik

vorgelegt
von

Richard Jelinek
geb. am 24.11.1970 in Prag

Angefertigt am

Lehrstuhl für künstliche Intelligenz der FAU Erlangen

Betreuer:	Prof. Dr.-Ing. Günther Görz
Beginn der Arbeit:	01.02.1997
Abgabe der Arbeit:	01.08.1997

Ich versichere, daß ich die Arbeit ohne fremde Hilfe und ohne Benutzung anderer als der angegebenen Quellen angefertigt habe und daß die Arbeit in gleicher oder ähnlicher Form noch keiner anderen Prüfungsbehörde vorgelegen hat und von dieser als Teil einer Prüfungsleistung angenommen wurde. Alle Ausführungen, die wörtlich oder sinngemäß übernommen wurden, sind als solche gekennzeichnet.

Fürth, den 31. Juli 1997

Zusammenfassung

Die vorliegende Diplomarbeit entstand im Rahmen des Projektes BEHAIM. Dieses Multimedia-Informationssystem wird am Lehrstuhl für künstliche Intelligenz der Universität Erlangen-Nürnberg, in Zusammenarbeit mit dem Bayerischen Forschungszentrum für Wissensbasierte Systeme, entwickelt. Ziel des Projektes ist ein System zu entwerfen und zu implementieren, welches interessierten Benutzern auf Anfrage umfassende Informationen zu verschiedenen Aspekten des Behaim-Globus liefern kann.

Ziel dieser Arbeit war die Bereitstellung einer Komponente mit der Fähigkeit zum qualitativen Räumlichen Schließen. Hierfür mußte zunächst ein Formalismus entwickelt und spezifiziert werden, der es erlaubt die möglichen Fragestellungen an das System sowie Ihre Bearbeitung zu beschreiben. Als praktischer Teil wurde dann eine Inferenzkomponente zum Räumlichen Schließen für eine kartographische 2D–Datenbasis implementiert. Die darin eingesetzten Algorithmen sind für eine Wiederverwendung in kontemporären Applikationen, in denen qualitatives Räumliches Schließen notwendig ist, ausgelegt. Dies können insbesondere moderne Geoinformationssysteme sein.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Kapitel 1

Einleitung

Maschinelle Verarbeitung von Daten beschränkt sich häufig auf quantitative Auswertungen bzw. Transformationen dieser Daten. Menschliche Verarbeitung ist jedoch zumeist qualitativ orientiert. Diese Diskrepanz sorgt sowohl für Probleme bei der Mensch-Maschine Schnittstelle als auch für schwierige Lösungsansätze, wenn es darum geht, qualitative Verarbeitung von Daten maschinell zu bewerkstelligen.

Allgemein versteht man unter Qualität einer Sache die charakteristischen Eigenschaften bzw. ihre Beschaffenheit. Die Quantität einer Größe ist indes ihre durch Zahlen erfaßbare, mengenmäßige Bestimmtheit. Wie man an diesen Definitionen erkennen kann, eignet sich eine quantitative Repräsentation also hervorragend zur maschinellen Verarbeitung, wohingegen der Umgang mit Daten qualitativer Natur nicht besonders mit der Arbeitsweise eines konventionellen Rechners harmoniert.

Dabei ist die Fähigkeit, qualitative Daten maschinell verarbeiten zu können, essentiell. Ganze Klassen von Problemen bringen diese Anforderung mit sich:

- quantitative Daten nicht bekannt
- quantitative Daten nicht erwünscht (Mensch-Maschine Schnittstelle)
- Effizienzüberlegungen

Natürlich bringt eine qualitativ orientierte Herangehensweise weitere interessante Vorteile. Montag und Struß [MS95] geben eine gute Einführung und Motivation in das Thema „Qualitatives und modellbasiertes Schließen“.

1.1 Ziel der Diplomarbeit

Die zentrale Aufgabe für diese Arbeit bestand darin, einen Formalismus zu entwickeln und zu spezifizieren, der es erlaubt, Fragestellungen bezüglich räumlicher Eigenschaften von Objekten in einer kartographischen Datenbasis sowie ihre Bearbeitung zu beschreiben. Desweiteren sollte eine Komponente zum Räumlichen Schließen entwickelt werden, um die mit der bestehenden Anfragekomponente möglichen Fragestellungen bearbeiten und beantworten zu können. Da die erfolgenden Anfragen qualitativer Natur, die zur Verfügung stehenden

notwendigen Daten jedoch quantitativer Art sind, ist eine Transformation der quantitativen Repräsentation in eine qualitative ebenso notwendig.

Der für diese Arbeit interessante Aspekt qualitativer Datenverarbeitung betrifft insbesondere die Mensch-Maschine Schnittstelle. Es gilt Anfragen von Menschen über Sachverhalte im zweidimensionalem Raum derart zu behandeln, daß diese vom System korrekt bearbeitet und beantwortet werden können. Es geht also darum, diese Form von Interaktion korrekt zu modellieren und zu realisieren.

Eine ideale Schnittstelle würde dem Anfragenden die Möglichkeit bieten, seine Anfragen in der für Ihn gewohnten Form – der natürlichen Sprache – zu stellen. Die Verarbeitung natürlichsprachlicher Anfragen stellt jedoch bis heute eine (größtenteils unbefriedigend gelöste) technische Herausforderung dar. Dies gilt auch für Systeme in denen es um sprachliche Analyse innerhalb einer beschränkten Domäne geht.

Anfragen an die zu realisierende Komponente erfolgen somit nicht in natürlicher Sprache, enthalten jedoch natürlichsprachliche Elemente. So wird nach der Entfernung zwischen den Objekten $\boldsymbol{X}$ und $\boldsymbol{Y}$ nicht in Pixeln sondern mit Formulierungen qualitativer Natur wie *weit*, *nah*, *benachbart* etc. gefragt.

Somit ist es also notwendig, diese Anfragen zu transformieren, so daß ihre Verarbeitung unter Zuhilfenahme der vorhandenen quantitativen Datenbasis erfolgen kann. Weiterhin müssen die aufbereiteten Daten bzw. Antworten – zumindest bei einigen Anfragetypen – ebenfalls in qualitativer Form bereitgestellt werden.

Die mit qualitativ orientierten Inferenzmechanismen erreichbaren Performanzvorteile sind ein ebenso wichtiger und interessanter Aspekt. Insbesondere die Reduktion des Datensatzes auf das Wesentliche, nämlich die Größenordnungen und Verhältnisse in einer zu inferierenden Szene. Diese Einsparungen sollten Algorithmen ermöglichen, welche gegenüber quantitativ orientierten Verfahren hinsichtlich Speicher- und Rechenzeitbedarf ebenfalls im Bereich von Größenordnungen effizienter arbeiten.

1.2 Gliederung der vorliegenden Arbeit

Kapitel 2 beinhaltet eine Übersicht bisheriger Verfahren zum räumlichen Schließen. Hierbei werden auch Abhandlungen betrachtet, in denen für diese Arbeit wichtige Hilfsmittel vorgestellt werden.

Kapitel 3 beschreibt die dieser Arbeit zugrundeliegende Problemstellung sowie die Inferenzkomponente zum qualitativen räumlichen Schließen in 2D.

Kapitel 4 erläutert näher einige Punkte bezüglich der Implementation und dient zugleich als technische Dokumentation.

Kapitel 5 gibt eine Übersicht einiger Anfragen an die Komponente mitsamt der Ergebnisse. Weiterhin werden Laufzeitmessungen einiger Anfragen präsentiert.

Kapitel 6 enthält eine Zusammenfassung der vorliegenden Arbeit und gibt einen Ausblick auf mögliche Erweiterungen und Modifikationen der implementierten Komponente.

Kapitel 2

Ansätze zum qualitativen räumlichen Schließen

In der Literatur wird der Begiff „qualitatives räumliches Schließen“ in zweierlei Hinsicht verwendet. Zum einen werden darunter Verfahren verstanden, mit derer Hilfe es möglich ist, aus einer vorhandenen qualitativen Datenbasis Schlußfolgerungen ziehen zu können, bzw. Konsistenzprüfungen dieser Daten durchzuführen (vgl. Grigni et al. [GPP95]), zum anderen sollen Anfragen qualitativer Art zu Daten quantitativer Natur bearbeitet werden. Dies geschieht entweder durch Transformation der Anfrage oder des Datensatzes.

2.1 Ansätze mit einer qualitativen Datenbasis

2.1.1 Logiken im 2D-Raum

Randell et al. [RCC92] stellen eine räumliche Logik vor, die auf Regionen und Verbindungen zwischen diesen basiert, und es erlaubt, Anfragen auf einer qualitativen Datenbasis zu operationalisieren. Die vorgestellte Logik ist Bestandteil einer allgemeineren Theorie, welche es zusätzlich ermöglicht, Objekte mit Zuständen zu versehen und mittels zeitabhängiger Relationen ganze Prozesse zu beschreiben.

Die vorgestellte räumliche Logik baut auf acht binären topologischen Relationen zwischen Regionen auf (vgl. auch Egenhofer [EF91] und Abb. 2.1). Da sich dieses Konzept allgemeiner Beliebtheit erfreut, soll es hier näher erläutert werden. Grigni et al. [GPP95] stellen dieses Konzept als *high resolution case* dar. Es werden sowohl seine Berechenbarkeitseigenschaften als auch die der daraus abgeleiteten reduzierten Formalismen ermittelt.

Wie Abbildung 2.1 zeigt, können die möglichen Relationen wie folgt interpretiert werden:

a) D: disjunkt(p,q), kommutativ

b) B: berührt(p,q), kommutativ

c) G: gleich(p,q), kommutativ

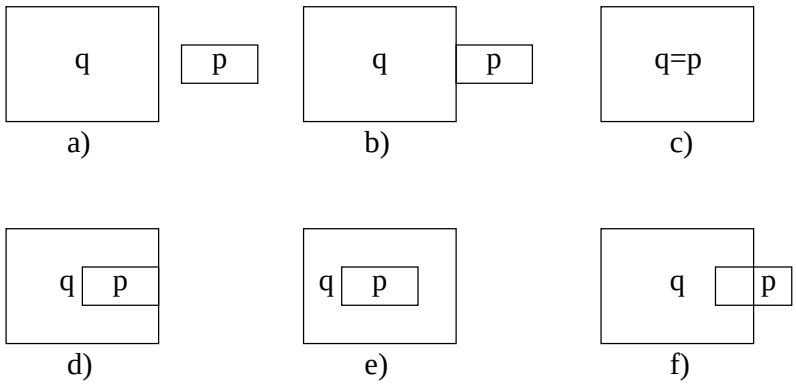

Abbildung 2.1: Die klassischen acht topologischen Relationen zwischen zwei Objekten in der Ebene. (Die Fälle *d)* und *e)* werden als jeweils zwei Relationen angesehen)

d) (2 Fälle) IB und EB: innerhalb_berührtRand(p,q), enthält_berührtRand(q,p)

e) (2 Fälle) I und E: innerhalb(p,q), enthält(q,p)

f) S: schneidet(p,q), kommutativ

Da bei dieser Vorgehensweise jedoch weder Abstand noch Richtung zwischen den beteiligten Objekten betrachtet werden, wären die von uns benötigten Inferenzen basierend auf diesem Formalismus nicht realisierbar.

Führt man diese beiden Konzepte ein, lassen sich die hier vorgestellten acht Relationen auf vier reduzieren (siehe 3.8.2).

2.1.2 Richtung und Entfernung

Frank [Fra92] stellt eine formale Methode für qualitatives räumliches Schließen vor, welche die beiden Konzepte Richtung und Entfernung verwendet. Der Ansatz geht von einer qualitativen Datenbasis aus, in der die aufgeführten Objekte hinsichtlich ihrer Richtungen und Entfernungen relativ zueinander abgelegt wurden.

Es werden verschiedene Modelle für Unterteilungen dieser Größen angegeben, die sich hinsichtlich ihrer Granularität unterscheiden. Die Modelle für Entfernungen entnehme man Tabelle 2.1, die Modelle für Richtungen sind in Tabelle 2.2 zu sehen.

Modelltyp	Wertebereich
zweiwertig	`nah, fern`
dreiwertig	`nah, normal, fern`
mehrwertig	`sehr nah, ..., sehr fern`

Tabelle 2.1: Qualitative Modelle von Abständen

Modelltyp	Wertebereich
4 Richtungen ohne Identität	`N, O, S, W`
4 Richtungen mit Identität	`N, O, S, W,` id
8 Richtungen mit Identität	`N, NO, O, SO, S, SW, W, NW,` id

Tabelle 2.2: Qualitative Modelle von Richtungen

Die Unterteilungen mit ihren verschiedenen Differenzierungsgraden sind ein Versuch, sprachlichen Modellen verschiedener Granularität gerecht zu werden. Die vorgenommene starre Einteilung (man muß sich bei dem verwendeten Formalismus für ein Modell entscheiden) erlaubt jedoch nicht, sich auf einen bestimmten Sachverhalt zu „konzentrieren“. Die Änderung der Granularität, mit der eine bestimmte Szene erfahren wird, ist jedoch unabdingbare Voraussetzung, um lokale Probleme trotz qualitativer Schlußfolgerung hinreichend genau lösen zu können.

2.1.3 Granularität

Um Beziehungen zwischen verschiedenen Auflösungsstufen einer Szene darstellen zu können, stellt Euzenat [Euz95] zwei Operatoren vor, mit denen zwischen Stufen von Granularitäten konvertiert werden kann. Beide Operatoren werden eingesetzt, um qualitative zeitliche Beziehungen zu bearbeiten. Es wird besprochen, wie man diese Operatoren auch auf räumliche Sachverhalte anwenden kann.

Die gewünschten Eigenschaften reduzieren die Menge möglicher Operatoren bis zur Eindeutigkeit. $\boldsymbol{r}$ sei hierbei eine Relation zwischen zwei Zeitintervallen und $g \to g'$ ein Operator auf diesen Relationen. Mit $g \to g'$ ist allgemein sowohl der vergröbernde $g \uparrow g'$, als auch der verfeinernde $g' \downarrow g$ Operator gemeint, falls die Eigenschaft für beide zutreffend ist.

selbstkonservierend
: Die Relation muß in ihrer eigenen Konversion enthalten sein. Das Wissen um die Beziehungen mag minder präzise sein, muß jedoch für gleiche Operandenpaare auch[1] gleiche Ergebnisse liefern.

$$\boldsymbol{r} \in g \to g' \boldsymbol{r} \tag{2.1}$$

strukturerhaltend
: Diese Eigenschaft besagt, daß bei einer Konversion der Granularität dennoch konzeptuelle Nachbarschaft der Relationen erhalten bleibt. Konzeptuelle Nachbarschaft zwischen Relationen wurde in Freksa [Fre92] (S. 204) wie folgt definiert:

> Two relations between pairs of events are (conceptual) neighbors, if they can be directly transformed into one another by continuously deforming (i.e., shortening, lenghtening, moving) the events (in a topological sense).

[1] Damit ist die konvertierte Relation tatsächlich unschärfer. $>$ wird z.B. zu $> \vee =$.

Die formale Definition sieht eine konzeptuelle Nachbarschaft als binäre Relation N_Γ^X, wobei Γ eine Menge von Relationen zwischen Objekten der Menge X ist. Es soll gelten: $r_1, r_2 \in \Gamma; o_1, o_2 \in X; N_\Gamma^X(r_1, r_2)$, falls die Transformation von $o_1 r_1 o_2$ in $o_1 r_2 o_2$ ohne Transition durch eine andere Relation erfolgen kann. Für einen auflösungskonvertierenden Operator $g \to g'$ lautet die Forderung:

$$\begin{aligned} &\forall r, \forall r', r'' \in g \to g' r, \exists r_1, \ldots, r_n \in g \to g', \text{so daß} \\ &r_1 = r', r_n = r'' \text{ und } \forall i \in [1, n-1] \quad N_\Gamma^X(r_i, r_{i+1}) \end{aligned} \tag{2.2}$$

symmetrisch

Erweitert man die Konversion auf Mengen von Relationen

$$g \to g' \rho = \bigcup_{r \in \rho} g \to g' r$$

so ist eine weitere gewünschte bzw. notwendige Eigenschaft die Symmetrie der Konversion:

$$g \to g' \rho^{-1} = (g \to g' \rho)^{-1} \tag{2.3}$$

inverskompatibel

Diese - etwas schwer zu formulierende - Eigenschaft besagt, daß eine Relation zwischen zwei Objekten auf einer Granularitätsstufe, die als andere Relation auf einer anderen Granularitätsstufe angesehen werden kann, aus der letzteren durch Anwenden des inversen Operators ermittelbar ist.

$$r \in \bigcap_{r' \in g \uparrow g' r} g' \downarrow g r' \text{ und } r \in \bigcap_{r' \in g \downarrow g' r} g' \uparrow g r' \tag{2.4}$$

Faktisch bedeutet dies die Kenntnis über mögliche Verhältnisse zwischen Objekten auf einer anderen Granularitätsstufe wenn die Verhältnisse bei gegebener Granularität bekannt sind.

idempotent

Ein großer Unterschied zwischen dem Umgang mit quantitativen und qualitativen Daten besteht in der Forderung nach Idempotenz.

$$\uparrow \cdot \uparrow = \uparrow \text{ und } \downarrow \cdot \downarrow = \downarrow \tag{2.5}$$

Dies garantiert gleiche Ergebnisse sobald Änderungen des Granularitäts-Fokus in verschiedenen Schritten vorgenommen werden.

repräsentationsunabhängig

Die Operatoren sollen die Eigenschaft besitzen unabhängig von der gewählten Repräsentation zu sein.

$$g \to g' \rho = \Leftarrow g \to g' \Rightarrow \rho \text{ and } g \to g' \rho = \Rightarrow g \to g' \Leftarrow \rho \tag{2.6}$$

Hiermit wird zunächst nur auf die Repräsentation von zeitlichen Objekten (Intervall oder Menge von Begrenzungspunkten) Bezug genommen. Natürlich gilt diese Forderung auch für Objekte im 2D-Raum.

2.1.4 Konsistenzprüfung einer qualitativen Datenbasis

Grigni et al. [GPP95] untersuchen die Berechenbarkeitseigenschaften von Systemen, bei denen aus gegebenen primitiven topologischen Relationen wie „A innerhalb B“, „B berührt C“ komplexere Sachverhalte inferiert werden sollen. Hierbei sind die Objekte wie in unserem Fall einfach verbundene Regionen in der Ebene. Die betrachteten Systeme unterscheiden sich hinsichtlich ihrer Auflösung. So werden hoch, mittel und niedrig auflösende Fälle betrachtet. Tabelle 2.3 gibt einen Überblick über die verwendeten topologischen Relationen in diesen Systemen. Die verwendeten Abkürzungen entsprechen den in Abschnitt 2.1.1 (Abbildung 2.1) genannten.

Systemauflösung	verwendete Relationen
hoch	`D,B,G,I,E,S,IB,EB`
mittel	`D,G,I,E,S`
niedrig	`D,S`

Tabelle 2.3: Systeme topologischer Relationen mit unterschiedlicher Auflösung.

Drei Regionen können nun nicht in einer beliebigen Relation zueinander stehen. So müssen in obigem Beispiel A und C disjunkt sein. Solche Einschritt-Inferenzen sind für die in Abbildung 2.1 gezeigten Relationen in [Ege91] und [GPP95] vollständig tabellarisch aufgelistet worden. Grigni et al. geben auch eine Übersicht von Einschritt-Inferenzen der Fälle mittlerer und niedriger Auflösung.

Bei Betrachtung der Berechenbarkeitseigenschaften sind drei Größen zu beachten:

Darstellung der Ausdrücke topologischer Relationen
: Die Darstellung topologischer Relationen kann entweder explizit (alle möglichen Paare werden genannt), konjunktiv (Relationen durch eine UND-Verknüpfung verbunden) oder uneingeschränkt (alle aussagelogischen Verknüpfungen gestattet) erfolgen.

Systemauflösung
: Bezeichnet die Anzahl der topologischen Relationen die in der Repräsentation zugelassen sind. Hier sind außer den drei oben erwähnten Systemauflösungen weitere Spezialfälle denkbar. So auch die von den Autoren betrachtete „medium resolution“, in der die topologische Relation *schneidet* durch *berührt* ersetzt wurde.

Art der Erfüllbarkeit
: Laut den Autoren gibt es zwei Grade von Konsistenz. Zum einen die *relational consistency*, welche besagt, daß es für eine gegebene Darstellung topologischer Relationen eine widerspruchsfreie Belegung der möglichen Werte aus der Tabelle der Einschritt-Inferenzen gibt. Die Tabelle muß natürlich hinsichtlich ihrer Auflösung übereinstimmen.

  Diese Eigenschaft ist jedoch bei zweidimensionalen Problemen notwendig aber nicht hinreichend. Im Unterschied zum eindimensionalen Schließen, wo relationale Konsistenz ausreichend ist, muß hier ermittelt werden, ob

der beschriebene Sachverhalt ein planares Modell besitzt oder nicht. Die Autoren geben ein Beispiel an, bei dem zwar relationale Konsistenz gegeben ist, welches jedoch in der Ebene nicht realisierbar ist.[2]

Die Komplexität der untersuchten 24 Fälle wurde bis auf einen Sachverhalt (planare Realisierbarkeit, konjunktive Darstellung und mittlere Auflösung mit „berührt") bestimmt. Die Ergebnisse weisen die *planare Erfüllbarkeit* als NP-vollständig in nahezu allen Fällen aus, wohingegen *relationale Konsistenz* für konjunktive und explizite Darstellungen in polynomialer Zeit lösbar ist.

Die vorgestellte Methode besitzt keine Kenntnis über die Form oder Größe der inferierten Objekte. Man könnte aus diesen Daten jedoch weitere Informationen ableiten. Die Autoren geben mit Ihrer Vermutung, daß durch Heuristiken auch die NP-vollständigen Subprobleme handhabbar werden, jedenfalls Hoffnung.

2.2 Ansätze mit einer quantitativen Datenbasis

Die oben besprochenen Ansätze gehen von einer bereits vorhandenen qualitativen Datenbasis aus. Dies ist jedoch oft nicht gegeben, da zum Beispiel sämtliche digitalen Aufnahmen eine Quantisierung realer Sachverhalte darstellen. Dies gilt auch für die hieraus mit Mitteln der Mustererkennung gewonnenen Daten wie Kanten- und Gradientenbild etc. Chang und Jungert [CJ96] beschreiben mit der *Theory of Symbolic Projections* eine Theorie räumlicher Relationen, die auf der Repräsentation symbolischer Bilder mittels 2D-Strings beruht.

Unter einem 2D-String versteht man ein Paar symbolischer Projektionen, mit denen Objekte/Symbole in einem gegebenen zweidimensionalen Bild entlang der x und y-Achse mit Hilfe einer Ordnungsrelation beschrieben werden.

Die Repräsentation in dieser Form ist qualitativer Natur und wird durch mehrere Stufen aus einem digitalisierten Rasterbild ermittelt. Operationen wie die Suche nach Teilbildern und konzeptionell ähnlichen Bildern werden auf eine Substring-Suche reduziert. Die Autoren sind jedoch der Ansicht, daß eine Beschränkung der Inferenzmethoden zum räumlichen Schließen nicht auf qualitative Aspekte beschränkt werden kann (S. 181):

> It may not be clear from this discussion on qualitative spatial reasoning that creating methods for solving problems on a high level abstraction is not sufficient. This is due to the fact that sooner or later any method, no matter which level of abstraction it works on, must use basic data in full resolution. In other words, the "bits and pieces" must never be forgotten. Hence, when the methods are developed not only the qualitative aspects should be of concern but the quantitative ones as well.

[2] Das Beispiel orientiert sich am Problem des Vollständigkeitsgraphen für fünf Punkte in der Ebene.

Kapitel 3

Beschreibung der Inferenzkomponente

3.1 Problemstellung

Ziel ist die Konstruktion einer Inferenzkomponente, die aufgabenunabhängig alle relevanten Fragestellungen in der Domäne „Objekte im 2D-Raum“ bearbeiten kann. Hierzu ist es zunächst notwendig die als relevant anzusehenden Fragentypen zu ermitteln und zu kategorisieren.

Weiterhin ist die Repräsentation der Daten ein entscheidender Faktor, da hiervon Mächtigkeit und Laufzeitverhalten der Inferenzkomponente abhängt.

Ein Problem, welches sich an vielen Stellen des Lösungsansatzes zeigt, ist die nicht eindeutige Zuordnung natürlicher Sprachkonstrukte (insbesondere Präpositionen) zu den hier vorgestellten Relationen zwischen Objekten. Bei der Konzeption der Inferenzkomponente wurde besonders darauf geachtet, daß diese Komponente unabhängig von Aufgabenstellung und den Möglichkeiten natürlicher Sprache in dieser Domäne arbeiten kann. So sollte das o.g. Zuordnungsproblem durch eine externe Wrapper-Komponente, welche die auftretenden Ambiguitäten auflöst, behoben werden können.

3.2 Fragestellungen

Unter der Voraussetzung, daß dem Fragesteller die kartographischen Daten visuell dargelegt werden, und mit Hinblick auf natürlichsprachliche Anforderungen gibt es in o.g. Domäne zwei Sorten von Anfragen:

1. Die angefragten Objekte sind bekannt. Der Fragesteller erkundigt sich nur nach bestimmten Eigenschaften des Objektes, das er durch dessen Position referenziert. Diesen Anfragetyp werden wir als *deiktische Frage* bezeichnen.

2. Die angefragten Objekte sind a priori unbekannt. Der Fragesteller erkundigt sich nach Objekten, die bestimmte Eigenschaften aufweisen. Diesen Anfragetyp werden wir als *iterative Frage* bezeichnen.

	deiktische Frage	iterative Frage
1 Objekt	$\mathcal{A}$	$\mathcal{B}$
2 Objekte	$\mathcal{C}$	$\mathcal{D}$
3+ Objekte	$\mathcal{E}$	$\mathcal{F}$

Tabelle 3.1: Kategorisierung von Anfragetypen.

Man sieht bereits, daß diese zwei Typen von Fragestellungen zueinander invers sind. Weiterhin sollte erwähnt werden, daß die Eigenschaften von Objekten in beiden Fällen bekannt sind (der Fragesteller weiß, wonach er fragt). Dies muß nicht immer so sein, wie z.B. das Problem des Data Mining zeigt. Diese Anwendung soll aber hier nicht betrachtet werden.

Weiterhin kann man nach der Anzahl der an einer Anfrage beteiligten Objekte unterscheiden. Diese Klassifikation ist insofern wichtig, da nur hierdurch Kongruenzprüfungen realisiert werden können. Zum Beispiel ist das Ermitteln eines Abstandes nicht möglich, wenn an der Anfrage nur ein Objekt beteiligt ist.

Tabelle 3.1 zeigt die Kategorisierung der möglichen auftretenden Anfragetypen. Im natürlichsprachlichen Gebrauch treten keine atomaren Relationen zwischen mehr als zwei Objekten auf. Etwaige Fragestellungen mit drei oder mehr beteiligten Objekten lassen sich durch Komposition erreichen (Klassen $\mathcal{E}$ und $\mathcal{F}$).

3.2.1 Eine alternative Kategorisierung

De Floriani et al. [FMP93] kategorisieren Anfragen über räumliche Sachverhalte in topologische, mengentheoretische und metrische Typen. Hierbei ist eine Anfrage Q wie folgt definiert: $Q = (q, S, \mathcal{R})$. q sei das zu erfragende Objekt, S sei eine Menge von Referenzobjekten, also ein Anfrageraum, und $\mathcal{R}$ sei eine Relation. Q soll nun eine Menge von Objekten liefern, für die zutrifft:

$$Q \subseteq S \quad \wedge \quad \forall x, x \in Q, q\mathcal{R}x \tag{3.1}$$

Die typischen Fragestellungen der kategorisierten Anfragen werden wie folgt angegeben:

topologisch orientierte Anfrage

- Nachbarschaft
- Rand (boundary)
- Inhalt bzw. äußeres Gebiet (co-boundary)

mengentheoretisch orientierte Anfrage

- Schnittmenge
- Enthaltensein
- Gleichheit

metrisch orientierte Anfrage

- Entfernung

3.2.2 Vergleich der Kategorisierungsschemata

Das Kategorisierungsschema von De Floriani et al. stellt nach eigenen Angaben einen Versuch dar, mögliche Anfragen über räumliche Sachverhalte an ein GIS systematisch zu kategorisieren. Die Autoren stellen fest, daß heutige GI-Systeme in der Lage sind, „einige“ räumliche Anfragen zu beantworten, daß ein vollständiger und vereinheitlichter Formalismus sowie dessen Implementierung jedoch immer noch ein ungelöstes Problem darstellen.

Dem vorgeschlagenen Kategorisierungsschema wird die effiziente und exakte Lösung geometrischer Probleme zugrundegelegt. Somit ist dieser Ansatz quantitativ orientiert. Fragestellungen der Form „ist x ähnlich y“ fehlen.

Das Ergebnis einer Anfrage ist gemäß der Definition stets eine Menge von Objekten, welche Gleichung 3.1 genügen. Wahrheitswerte gibt es zwar nicht als Ergebnis, jedoch können diese nachgebildet werden:

„Ist x enthalten in y?“, muß also angegeben werden als $Q = (x, \Omega, \subset)$ („Nenne alle Objekte, in denen x enthalten ist.“). Diese Anfrage liefert mit Q eine Menge von Objekten, die x enthalten. Nun muß noch festgestellt werden, ob y Element dieser Menge ist. Dies bedeutet, daß sämtliche Fragetypen in diesem Schema den unter Abschnitt 3.2 definierten iterativen Fragestellungen entsprechen. Der Vergleich mit Tabelle 3.1 zeigt, daß es sich hierbei ausschließlich um Anfragen aus Kategorie $\mathcal{B}$ handelt.

Es sei an dieser Stelle betont, daß die von De Floriani et al. vorgestellte Einteilung hinsichtlich der Anfragekategorien vollständig ist und auch auf qualitative Systeme angewandt werden könnte. Die Restriktion auf (generische) iterative Fragekonstrukte erfordert überdies bei den hier definierten deiktischen Fragetypen ein Anpassen des Anfrageraumes, um ebenfalls $\mathcal{O}(1)$ Laufzeitverhalten zu ermöglichen[1].

3.3 Klassen von Inferenzen

Bei Anfragen an die vorgestellte Komponente sollen die Fälle betrachtet werden, bei denen die Anzahl der beteiligten Objekte unterschiedlich ist. Folgende Varianten sollen uns interessieren:

1 Objekt Inferenzen dieses Typs können – durch Kombination primitiver Operationen der Komponente – Auskunft über bestimmte Eigenschaften des erfragten Objektes geben.

2 Objekte Hierbei gilt es die möglichen topologischen Relationen zwischen zwei Objekten zu berechnen. Ein Problem ist die nicht eindeutige Zuordnung von Präpositionen natürlicher Sprache zu diesen Relationen. Da die hier vorgestellten Relationen jedoch als objektiv anzusehen sind, kann das Problem als extern (d.h. durch eine andere[2] Komponente zu lösen) betrachtet werden.

3 bis n Objekte Treten drei Objekte in Relation miteinander, so kann durch Komposition der vorgenannten Inferenzen eine Lösung für diese Relation gefunden werden.

[1] Anstelle von Ω wird eine entsprechend restringierte Menge verwendet

[2] Gemeint ist eine übergeordnete Komponente zur Auflösung von Ambiguitäten

Der Operator *zwischen* wurde zuerst als ternärer Operator konzipiert. Er kann (aufwendig) durch Komposition dualer Operatoren gewonnen werden (vgl. hierzu Abb. 3.8 und Abb. 3.9).

Ein Redesign des hier verwendeten Formalismus erlaubt es jedoch auch diesen Operator als zweiwertig zu betrachten und seine Sonderstellung aufzugeben. Der hierfür erforderliche Algorithmus identifiziert nun nicht mehr ein Objekt als zwischen zwei gegebenen Objekten liegend, sondern liefert eine Liste von zwischenliegenden Objekten zu zwei gegebenen Referenzobjekten.

Damit ist auch möglich auf dieser Operation einen hochwertigen Operator für den Test auf Nachbarschaft aufzusetzen.

3.4 Datenrepräsentation und primitive Operationen

3.4.1 Datenrepräsentation

Die für die Inferenzkomponente relevanten Geometriedaten der Instanzen sind quantitativ als offener oder geschlossener Streckenzug gespeichert. Offene Streckenzüge werden intern in geschlossene umgerechnet (siehe Abb. 3.1 sowie Abschnitt 3.4.2) und dienen der Repräsentation länglicher und schmaler Objekte wie etwa Flüsse.

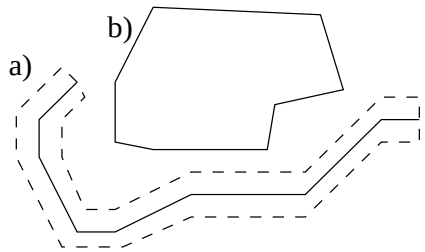

Abbildung 3.1: Beispiel für Umrechnung offener Linienzüge: offener (a) und geschlossener (b) Streckenzug

Intern erfolgt die Repräsentation durch eine Liste von Koordinatenpaaren, wobei das Kriterium durch die Übereinstimmung der ersten und letzten Koordinate der Liste gegeben ist.

Geschlossene Streckenzüge werden im folgenden als Polygone bezeichnet. Um von der Inferenzkomponente korrekt verarbeitet werden zu können unterliegen die Polygone folgenden Einschränkungen:

- einfach zusammenhängend
- nicht selbstüberschneidend
- keine kolinearen (also redundanten) Punkte

3.4.2 Primitive Operationen

Auf diesen quantitativen kartographischen Daten können nun folgende Operationen definiert werden:

Linienzug schließen

Wie bereits erwähnt, werden grundsätzlich zwei verschiedene Datenrepräsentationen verwendet: offene und geschlossene Streckenzüge. Diese Unterscheidung und die Forderung nach einer einheitlichen Repräsentation macht es erforderlich offene Streckenzüge in geschlossene umzurechnen (siehe Abb. 3.1). Die Unterscheidung geschlossen/offen wurde aus folgenden Gründen getroffen:

- **Erleichterung und Genauigkeit der Eingabe**
  Die für die Inferenzkomponente benötigten Daten werden mittels einer Hilfsapplikation eingegeben. Es ist wesentlich schneller einen (langen) Flußlauf mit einem einfachen Linienzug, als durch seine Umrisse nachzuzeichnen. Bei vielen dieser Objekte besteht wegen Ihrer geringen Breite überdies die Gefahr von Ungenauigkeiten bei der Eingabe, so daß aus Versehen selbstüberschneidende Polygone entstehen könnten.
- **Effizienz der Algorithmen**
  Durch die maschinelle Umrechnung der einfachen Linienzüge in Polygone, wird stets ein Minimum an Eckpunkten generiert, was sich natürlich positiv auf das Laufzeitverhalten der verwendeten Algorithmen auswirkt. Die hierdurch erzeilbare Beschleunigung liegt im Mittel bei Faktor 2, da nur die Hälfte der Randpunkte betrachtet werden muß.

Konvexe Hülle

Bedingt durch die Anforderungen an Polygone (siehe 3.4.1) gilt es lediglich zwischen konvexen und konkaven Polygonen zu unterscheiden. Desweiteren ist es nötig von konkaven Polygonen die konvexe Hülle zu bilden.

Schwerpunkt/Mittelpunkt

Da keine Gewichtung der einzelnen Randpunkte des Polygons vorliegt und keine kolinearen Punkte auftreten dürfen, wird der Schwerpunkt als gewichteter Mittelwert der X- und Y-Werte der einzelnen Punkte berechnet. Der Vorteil liegt in der Repräsentation der Daten als Liste von Punkten und nicht etwa Linienzügen: Egal ob es sich um ein offenes, geschlossenes, konkaves oder konvexes Polygon handelt - der Aufwand für das Ermitteln des Schwerpunktes ist stets linear.

Für den Schwerpunkt $X_s = (x_s, y_s)$eines Polygons Π der Länge N kann folgende Formel verwendet werden:

$$x_s = \frac{1}{N} \sum_{i=1}^{N} x_i \tag{3.2}$$

$$y_s = \frac{1}{N} \sum_{i=1}^{N} y_i \tag{3.3}$$

Entfernung zweier Punkte

Aufgrund des quantitativen Charakters der Datenbasis kann eine Entfernung zweier Punkte X, Y leicht ermittelt werden (siehe 3.6.1). Vorher müssen ggf. Besonderheiten des Koordinatensystems beachtet werden.

Richtung durch Start- und Endpunkt

In der Ebene können zwei als Start- und Endpunkt ausgezeichnete Punkte eine gerichtete Strecke beschreiben. Der Winkel, den diese Strecke mit der x-Achse eines zugrundeliegenden kartesischen Koordinatensystems einschließt, kann einen Wert im Intervall $[0, 2\pi)$ annehmen.

3.5 Operationen auf Objekten

In diesem Abschnitt soll eine Übersicht der bereitgestellten Operationen, welche auf den Objekten der Datenbasis ausgeführt werden können, gegeben werden. Eine detaillierte Beschreibung der diesen Operationen zugrunde liegenden Formalismen erfolgt in Abschnitt 3.8.

3.5.1 Inferenz über 1 Objekt

Bei Anfragen an ein System, welche sich lediglich auf ein konkretes Objekt $\boldsymbol{A}$ beziehen, können folgende Fragestellungen auftreten:

- Wie groß ist $\boldsymbol{A}$? ($|A|$)
- Welche Objekte enthält $\boldsymbol{A}$? ($\#A$)

Die Orientierung eines Objektes soll nicht näher betrachtet werden, da ihre Behandlung weitere Objektdaten erfordert, die in der gegenwärtigen Datenbasis nicht vorliegen.

3.5.2 Inferenz über 2 Objekte

Anfragen zu zwei Objekten $\boldsymbol{A}$ und $\boldsymbol{B}$ können folgende Typen sein:

- In welcher Relation steht $\boldsymbol{A}$ zu $\boldsymbol{B}$? ($A?B$ – nicht kommutativ, invers)
- Sind $\boldsymbol{A}$ und $\boldsymbol{B}$ gleich groß? ($A \approx B$ – kommutativ)
- Wie weit ist $\boldsymbol{A}$ von $\boldsymbol{B}$ entfernt? ($A \ominus B$ – kommutativ)
- In welcher Richtung von $\boldsymbol{A}$ liegt $\boldsymbol{B}$? ($A \oplus B$ – nicht kommutativ, invers)
- Ist $\boldsymbol{A}$ Nachbar von $\boldsymbol{B}$? ($A \asymp B$ – kommutativ)
- Welche Objekte liegen zwischen $\boldsymbol{A}$ und $\boldsymbol{B}$? ($A| \cdot |B$ – kommutativ)

3.5.3 Inferenz über 3 oder mehr Objekte

Komposition zweiwertiger Relationen

Diese Aufgabenstellung entspricht den bei Egenhofer [Ege91] und Grigni et al. [GPP95] beschriebenen Einschritt-Inferenzen. Hierbei wurden für sämtliche 64 Fälle die möglichen resultierenden Relationen aufgezeigt. Abbildung 3.2 zeigt die 16 Einschritt-Inferenzen bei den in dieser Arbeit verwendeten vier essentiellen topologischen Relationen.

Sieben der aufgezeigten Kombinationen lassen mehrere Schlußfolgerungen zu. Diese Zweideutigkeiten lassen sich nur auflösen indem für die Objekte A

A,B \ B,C	id	dj	sch	in
id	id	dj	sch	in
dj	dj	dj ∨ id ∨ sch ∨ in	dj ∨ sch	dj ∨ sch ∨ in
sch	sch	dj ∨ sch ∨ in	dj ∨ id ∨ sch ∨ in	sch ∨ in
in	in	dj	dj ∨ sch ∨ in	in

Abbildung 3.2: Die 16 Einschritt-Inferenzen der vier essentiellen topologischen Relationen.

und C direkt aus der quantitativen Datenbasis ihre Relation zueinander ermittelt wird. Ein Weiterverfolgen aller Alternativen ist nicht erfolgversprechend. Zum einen stellt dies ein exponentielles Verhalten dar, zum anderen sind die die Ambiguitäten verursachenden Relationen *dj* und *sch* wiederum in den Ergebnissen vorhanden, so daß keine Elimination erfolgt.

Anfragen, die eine Betrachtung von mehr als 3 Objekten erfordern, können durch Komposition der oben beschriebenen Operatoren, durch sukzessives Anwenden der Einschritt-Inferenztabelle sowie durch Verwendung der Quantoren $(\forall, \exists)$ abgearbeitet werden. Meist sind dies die iterativen Fragentypen, oder interne Berechnungen, die die Datenrepräsentation erfordert.

Für das folgende Beispiel soll Abbildung 3.7 als Visualisierungshilfe dienen. Ω ist hierbei unser Universum, also die Menge aller Objekte. Eine Frage der Form „Nenne Alle Objekte innerhalb von Objekt $\boldsymbol{A}$“ erfordert bei der gegenwärtigen Repräsentation folgende Vorgehensweise:

Algorithmus allin: Alle Innerhalb.

```
A := Ω                          ! 1 Liste aller Objekte
B := Ø                          ! 2 leer (Ergebnis)
foreach X in A:                 ! 3 Alle Elemente durchlaufen
    if X ? A = „innerhalb“      ! 4 Falls X innerhalb A
        add X to B              ! 5 Zu Ergebnis hinzu
return B                        ! 6 Ergebnis zurück
```

Dies würde im Fall von Abbildung 3.7 also als Ergebnis bedeuten: $\mathcal{B} = \{\boldsymbol{B},\boldsymbol{C},\boldsymbol{D},\boldsymbol{E},\boldsymbol{F},\boldsymbol{G}\}$. Wir sehen hier, daß auch Objekte welche sich innerhalb von Objekten welche sich innerhalb von $\boldsymbol{A}$ befinden aufgelistet werden. Mit anderen Worten: Es fehlt eine hierarchische Repräsentation. Man kann durch Verwendung des **allin**-Algorithmus, eine weitere Berechnungsvorschrift angeben, die

eine Hierarchie realisiert:

Algorithmus hierin: Innerhalb 1. Hierarchiestufe.

```
B := allin(A)                          ! 1 Alle Objekte in A (Ergebnis)
foreach X in B                         ! 2 Alle Elemente durchlaufen
   A := B                              ! 3 Arbeitsmenge von B
   del X from A                        ! 4 X ist Referenz
   foreach Y in A                      ! 5 Alle Elemente durchlaufen
      if Y ? X = „innerhalb“           ! 6 Falls Y innerhalb X
         del Y from B                  ! 7 innerhalb innerhalb entfernen
return B                               ! 8 Ergebnis zurück
```

Dies liefert als Ergebnis $\mathcal{B} = \{\boldsymbol{B},\boldsymbol{C},\boldsymbol{G}\}$.

Eine Anfrage der Form „Nenne alle Nachbarn von $\boldsymbol{A}$“ erfordert einen Algorithmus, der nahezu identisch mit **allin** ist. In einem Algorithmus **allnachbar** muß also lediglich die Abfrage in Zeile 4 lauten:

```
if X ≍ A = true                        ! 4 Falls X Nachbar von A
```

Aus diesem Grund empfiehlt es sich einen generischen Iterator für die **all...** Algorithmenklasse zu haben. Analog hierzu lassen sich Algorithmen definieren, welche Anfragen der Form „Alle Objekte nördlich, südlich, etc. von $\boldsymbol{A}$“ realisieren.

Kombinationen dieser Anfragen lassen sich direkt in Mengenoperationen umwandeln. Auch hier tritt das Problem natürlichsprachlicher Logik auf.

- „Zeige alle Nachbarn, die sich nördlich von X befinden.“
- „Zeige alle Nachbarn und alle nördlichen Objekte.“
- „Zeige alle Nachbarn oder alle nördlichen Objekte.“
- „Zeige alle Nachbarn, die sich nicht nördlich von X befinden.“
- „Zeige alle Objekte, die Nachbarn sind und sich nicht nördlich von X befinden, sowie alle Objekte, die sich nördlich befinden und nicht Nachbarn sind.

Der einfachste (da eindeutige) Fall dürfte die versteckte/implizite `UND` Verknüpfung sein. Punkt 1 läuft also auf eine Bildung der Schnittmenge der Ergebnisse von **allnachbar** und **allnord** hinaus.

Punkt 2 ist trotz der Formulierung ein inklusiv-ODER und läuft somit auf die Bildung der Vereinigungsmenge hinaus.

Punkt 3 zeigt das bekannte Problem der sprachlichen Verwendung von „oder“[3]. Es läßt sich lösen, indem man ein `XOR` annimt und nur eine der beteiligten Mengen als Ergebnis zurückliefert. Eventuell erfolgt die Auswahl zufallsgesteuert, um die Langeweile beim Benutzer in Grenzen zu halten.

Punkt 4 führt Differenzen in die Mengenoperationen ein (A ohne B bzw. B ohne A).

Punkt 5 ist ein Beispiel für symmetrische Differenz. Dieses Konzept scheint in natürlicher Sprache nicht oft Verwendung zu finden. Die Realisierung als

[3] meist meint der Sprecher ein exklusiv-ODER.

Mengenoperation gestaltet sich jedoch einfach. Die sprachliche Formulierung zeigt bereits, daß symmetrische Differenz durch Bildung der Vereinigungsmenge zweier Differenzen erreicht werden kann. Abbildung 3.3 soll das Beispiel veranschaulichen.

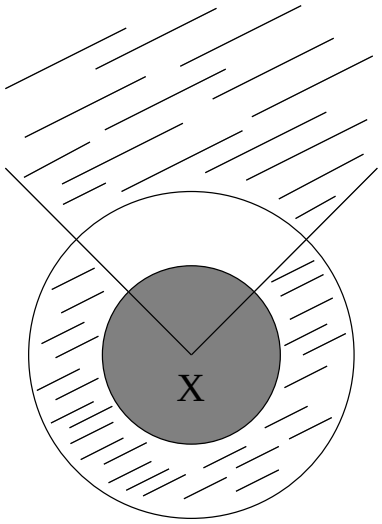

Abbildung 3.3: Symmetrische Differenz zweier räumlicher Eigenschaften: Das schraffierte Gebiet trifft auf die im Text erwähnte Anfrage zu.

3.6 Geometrische Grundlagen der Komponente

In diesem Abschnitt werden die Konzepte Abstand und Richtung genauer betrachtet. Dies geschieht für die Anwendungsfälle

- ebene Fläche
- Zylinderoberfläche

Die Anwendungsmöglichkeit einer Kugeloberfläche wird in Abschnitt 6.3 erläutert. Es sollte noch angemerkt werden, daß sowohl Zylinderoberfläche als auch Kugeloberfläche - trotz des dreidimensionalen Charakters der geometrischen Figur - ebenfalls zweidimensionale Probleme darstellen. Desweiteren sind Zylinderoberfläche und Kugeloberfläche nur einige Spezialfälle von 2D-Flächen. Andere Objekte wie etwa Torus oder Möbiusband sind denkbar, sollen aber hier nicht betrachtet werden.[4]

[4]Ein Möbiusband ist von der geometrischen Betrachtungsweise der 2D-Eigenschaften - Richtung und Entfernung - einem Zylinder sehr ähnlich. Ebenfalls zyklisch, jedoch mit doppeltem maximalen Abstand zwischen zwei Punkten.

3.6.1 Abstand

Ebene Fläche

Der Abstand zweier Punkte ist eine der wichtigsten Grundlagen für alle weiteren geometrischen Operationen. Der Abstand zweier Punkte $X(x_1, y_1)$ und $Y(x_2, y_2)$ in der Ebene wird quantitativ durch die einfache Formel in 3.4 beschrieben.

$$\overline{XY} = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2} \tag{3.4}$$

Man muß jedoch unterscheiden zwischen einer Ebene mit endlichen Ausmaßen und einer prinzipiell unbeschränkten Ebene. In anwendungsorientierten Applikationen kommt zumeist nur der erste Fall vor. Hierbei ist der maximale Abstand gleich der Diagonalen der Fläche.

Zylinderoberfläche

Das Ermitteln eines Abstands zweier Punkte auf einer Ebene, welche zu einer Zylinderoberfläche gekrümmt wurde, wird durch die Tatsache erschwert, daß mit dieser Oberflächenform zyklische Koordinaten beachtet werden müssen. Dies entspricht dem Sachverhalt bei den Segmentbildern des Behaim-Globus.

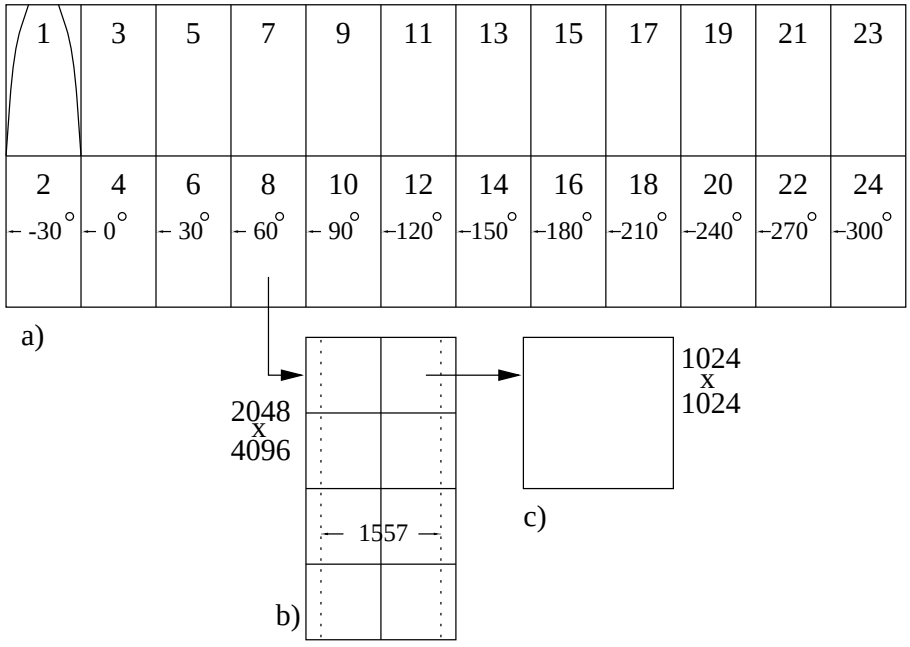

Abbildung 3.4: Organisation der Segment-Daten beim Behaim-Globus:
a) 12x2 Segmente mit Numerierung und Längengrad-Offset
b) Ein Segment (2048x4096 Pixel) mit angedeuteten realen Grenzen, die Ränder überlappen
c) 1024x1024 Pixel große Rasterbilder

Abbildung 3.4 zeigt die Aufteilung der Globusoberfläche in Segmente. Die Daten der Polkappen werden von diesem Schema getrennt behandelt. Jedes Seg-

ment erschließt einen Winkel von 30° auf der Zylinderoberfläche. Zwar ist die Rohbreite jedes Abschnittes 2048 Pixel, bedingt durch die Überlappungen ist die effektive Breite (und somit Bogenlänge) eines Segmentes jedoch im Schnitt lediglich 1557 Pixel. Die eigentlichen Daten beginnen bei einem x-Offset von 280 Pixeln und gehen bis x = 1837. Die Breite eines Segmentes ist jedoch als Sekante anzusehen, deren Endpunkte mit denen eines Kreisbogens für 30° zusammenfallen. Für den Radius des Zylinders ergibt sich theoretisch somit in Pixeln:

$$(1557 Pixel)/(2\sin(30/2)) = 3007 Pixel \tag{3.5}$$

Wegen diverser Anpassungen beträgt der reale Radius 2984 Pixel. Durch die 24 Segmente wird eine Fläche gebildet, die einen maximalen Abstand $\boldsymbol{l}$ (Länge der Diagonale) festlegt. Durch die Organisation als Zylinderoberfläche und die damit verbundene Einführung von zyklischen Koordinaten schrumpft die maximale Enfternung auf $\frac{\boldsymbol{l}}{2}$.

3.6.2 Richtung

Ebene Fläche

Bei Richtungsangaben in einer ebenen Fläche kommt es noch mehr als bei Entfernungsangaben darauf an, ob die Fläche begrenzt oder unendlich ist. Probleme treten bei den - für unsere Anwendungen - wichtigen begrenzten Flächen auf. Bei dieser Art müssen nämlich für die an den Rändern liegenden Punkte Sonderbedingungen beachtet werden. Je nach Lage eines Randpunktes gibt es für ihn einige nicht existente Nachbarn.

Abschnitt A.3 enthält die Implementierung für die Richtung einer Strecke im Intervall $[0, 360)$. Diese Basisdaten erlauben verschiedene qualitative Modelle für Richtungen. So sind die bei Frank [Fra92] erwähnten „cone-shaped directions“, bei denen Kegel die gewünschte Richtung angeben, ebenso wie die Interpretation der Richtung durch sog. „half-planes“ bei denen die Unterteilung der Ebene in verschiedene Mengen von Halbebenen erfolgt. Tabelle 3.2 zeigt die für die verschiedenen Modelle gültigen Intervalle. Zu beachten ist hierbei, daß es wegen des zyklischen Charakters durchaus Punkte gibt, welche z.B. der Eigenschaft $x : 315 \rightarrow 45 (x > 315 \wedge x < 45)$ (Ost-Kegel) genügen.

Im Gegensatz zu den bei Frank erwähnten Modellen für Richtungen in Tabelle 2.2, ist eine Identität bei Richtungen nicht notwendig, da dies einem „Nullabstand“ zweier Punkte entspricht und bereits bei der Betrachtung der Distanz zweier Punkte behandelt wird. Werden ganze Objekte betrachtet, entspricht eine „Identität“ der topologischen Relation *innerhalb*. Desweiteren muß eine Menge qualitativer Richtungsangaben keinesfalls eine äquidistante Unterteilung bedeuten. Eine Anwendung bei der lediglich von Interesse ist, welche Objekte sich in der nördlichen Halbebene relativ zu einem gegebenen Objekt befinden, in der es aber erforderlich ist, zwischen n südlichen Richtungen zu unterscheiden, ist hiermit ebenso realisierbar. Abbildung 3.5 c) soll den etwas exotischen Sachverhalt veranschaulchen.

Modelltyp	Wertebereich
4 Richtungen	`N` $45 \rightarrow 135$, `W` $135 \rightarrow 225$, `S` $225 \rightarrow 315$, `O` $315 \rightarrow 45$
8 Richtungen	`O` $337 \rightarrow 22$, `NO` $22 \rightarrow 67$, `N` $67 \rightarrow 112$, `NW` $112 \rightarrow 157$, `W` $157 \rightarrow 202$, `SW` $202 \rightarrow 247$, `S` $247 \rightarrow 292$, `SO` $292 \rightarrow 337$
N/S Halbebene	`N` $0 \rightarrow 180$, `S` $180 \rightarrow 0$
W/O Halbebene	`W` $90 \rightarrow 270$, `O` $270 \rightarrow 90$
Kombination Halbebenen	`NO` $0 \rightarrow 90$, `NW` $90 \rightarrow 180$, `SW` $180 \rightarrow 270$, `SO` $270 \rightarrow 0$

Tabelle 3.2: Beispiele von Intervallen für qualitative Modelle von Richtungen

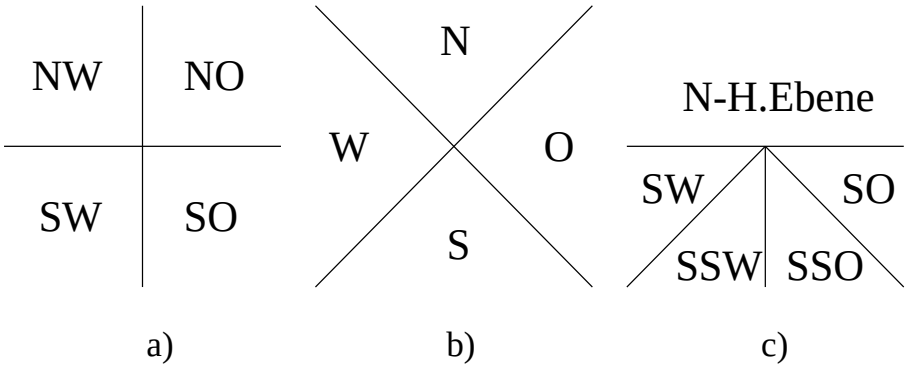

Abbildung 3.5: Verschiedene Modelle von Richtungen. a) Komposition einer Nord-Süd und West-Ost Halbebene; b) Nord-, West-, Süd- und Ost-Richtungskegel; c) Modell mit nicht-äquidistanter Unterteilung (siehe Text)

Zylinderoberfläche

Eine Zylinderoberfläche stellt prinzipiell eine endliche ebene Fläche dar, bei der durch Zusammenfügen zweier Enden zyklische Koordinaten entstehen. Somit werden Schlußfolgerungen für Punkte auf dieser Oberfläche nicht mehr eindeutig. Diese Ambiguität wird im Sprachgebrauch durch den jeweiligen Kontext aufgelöst.

Um also eine Richtung auf einer Zylinderoberfläche zu ermitteln[5], sind, wie bei der Richtungsangabe auf der Kugel (vgl. 6.3.1), drei Komponenten notwendig.

Da diese Daten bei Angabe von zwei Punkten nicht gegeben sind, nimmt die Komponente als Richtung diejenige, die eine kürzere Entfernung zwischen den beiden Punkten impliziert. Paris ist somit auch für die Komponente westlich von Nürnberg und nicht östlich.

[5]falls die Richtung definiert ist - speziell bei Randpunkten

3.7 Konversion: Von der Quantität zur Qualität

Bei einer Komponente, der quantitative Daten zur Verfügung stehen, die jedoch qualitativ orientiert arbeiten soll, muß früher oder später eine Konversion dieser Daten erfolgen. Im vorigen Abschnitt wurde ja bereits die in dieser Arbeit verwendete Konversion für verschiedene qualitative Modelle von Richtungen besprochen.

Wie aus diesen Konversionen ersichtlich ist, sind qualitative Aussagen nicht lediglich eine vergröberte Darstellung quantitativer Sachverhalte. Sie können zwar im Prinzip auf diese Weise gewonnen werden, es bleiben aber noch einige zusätzliche Aspekte die es zu beachten gilt:

äquidistante Unterteilung
: Die verschiedenen Modelle für qualitative Richtungsangaben in Tabelle 3.2 gehen von einer äquidistanten Unterteilung des quantitativen Datensatzes aus. Figur c) in Abbildung 3.5 zeigt ein Modell, das nicht diese Unterteilung aufweist. Äquidistanz ist für qualitative Modelle nicht zwingend.

schnittfreie Unterteilung
: Alle Modelle in Tabelle 3.2 gehen weiterhin von einer schnittfreien Unterteilung des quantitativen Datensatzes aus. Auch diese Annahme scheint nicht zwingend. Schließlich wäre es beim N,W,S,O-Modell ebenso möglich, daß sich die Richtungskegel an ihren Grenzen überschneiden und dort eben zwei mögliche Antworten generieren.

  Freilich läßt sich ein solcher Sachverhalt auch mit einer schnittfreien Unterteilung simulieren.[6]

Die gleichen Ausführungen gelten auch für die Konversion von quantitativen Entfernungsangaben in qualitative Aussagen über Distanz. So wird zur Modellierung von Entfernungen oft eine logarithmische Unterteilung vorgenommen. Kognitive Modelle beinhalten zwischen zwei qualitativen Größen häufig Sprünge in Größenordnungen, was die entsprechende quantitative Darstellung betrifft. Das gilt z.B. für eine subjektive Entfernungseinschätzung genauso wie für eine empfundene Lautstärkesteigerung.

Zweifellos gehen bei der Konversion in eine qualitative Datenbasis Informationen verloren. Eine Rückkonversion würde sich demnach erheblich schwieriger gestalten und zusätzliche Informationen benötigen, um gute Ergebnisse zu liefern.

3.8 Logische Grundlagen der Komponente

In diesem Abschnitt sollen die verwendeten Operatoren spezifiziert werden. Die verwendeten Datentypen und ihre Nomenklatur entnehme man Tabelle 3.3.

Der Datentyp bool kann die üblichen Wahrheitswerte annehmen. Wir werden das Paar `ja, nein` verwenden sowie von einer positiven Logik ausgehen. Ein Punkt $\boldsymbol{X}$ ist ein Paar Skalare, ein Polygonzug Π eine geordnete Liste von Punkten.

[6] Man führt einfach anstelle der entstandenen Schnitträume neue Richtungskegel ein.

Datentyp	Variable
bool	Kleinbuchstaben (Typewriter) `a,b,c,...`
skalar	Kleinbuchstaben $a, b, c, ...$
punkt	Großbuchstaben (Bold) $\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C}, ...$
polygon(zug)	griechische Großbuchstaben $\Pi, \Phi, \Psi, ...$

Tabelle 3.3: Datentypen und ihre Nomenklatur.

3.8.1 Übersicht der Operatoren

Operator: Ausmaß (| |)

$$|\ | : polygon \longrightarrow groesse$$

Die typischen Anfragen bezüglich der Größe eines Objektes lauten in der Regel:

- „Wie groß ist Objekt x?“
- „Welches der Objekte x und y ist größer(kleiner)?“

Wir wollen im folgenden zwischen *Größe* und *Ausmaß* unterscheiden. Unter Größe verstehen wir hierbei die tatsächliche Fläche eines Objektes. Das Außmaß eines Objektes ist dann von Bedeutung, wenn es um einen groben Test geht, ob ein Objekt in ein anderes eingepasst werden kann. Abbildung 3.6 zeigt zwei Objekte mit identischer Fläche (3cm^2), wobei Objekt b die größeren Ausmaße hat.

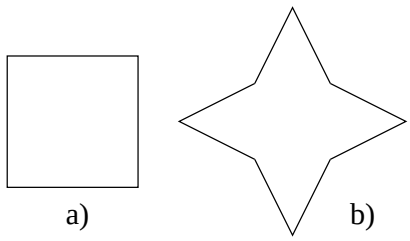

Abbildung 3.6: Beispiel für die Ausmaße zweier Objekte mit gleicher Fläche.

Im natürlichsprachlichen Gebrauch wird oft „groß“ für beide Begriffe verwendet, so daß es für die Komponente notwendig ist aus dem Kontext heraus die eigentliche Bedeutung zu erfassen.

Operator: Liste innenliegender Elemente (#)

$$\# : polygon \longrightarrow liste(polygon)$$

Dieser Operator liefert eine Liste der Polygone, die sich vollständig in dem erfragten Polygon befinden. Dabei wird keine implizite Hierarchie zwischen den Objekten vorausgesetzt. Der #-Operator entspricht somit dem in Abschnitt 3.5.3 vorgestellten *allin*-Algorithmus und basiert somit auf dem Relationsklassifikator ?.

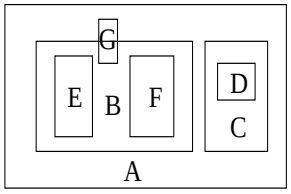

Abbildung 3.7: Liste der Objekte innerhalb eines Polygons.

Operator: Relationklassifikator (?)

$$? : polygon \quad \text{x} \quad polygon \longrightarrow relation$$

Identifiziert die Relation in welcher zwei Polygone zueinander stehen. Diese Relationen können sein: *disjunkt*, *identisch*, *innerhalb* und *schneidet*. Die Klassifikation erfolgt mit Hilfe des Schnittes zweier Polygone Π und Φ:

$$\Pi = \Phi \quad \Leftrightarrow \quad identisch \tag{3.6}$$

$$\Pi \cap \Phi = \emptyset \quad \Leftrightarrow \quad disjunkt \tag{3.7}$$

$$\Pi \cap \Phi = \Psi \neq \emptyset \quad \begin{cases} innerhalb, & \Psi = \Pi \vee \Psi = \Phi \\ schneidet, & \text{sonst} \end{cases} \tag{3.8}$$

Eine genaue Übersicht der Relationen findet man in Abschnitt 3.8.2.

Vergleichsoperatoren für Größe von Objekten ($<, =, >$)

$$\approx : polygon \quad \text{x} \quad polygon \longrightarrow bool$$

Die Operatoren sollen die Größe von Objekten vergleichen. Hierzu werden die Ausmaße der zu vergleichenden Objekte ermittelt und miteinander verglichen.

Intern erfolgt die Realisierung dieser Operation mit Hilfe des $|\ |$-Operators. Somit stellen diese Operatoren Vergleiche der Größenordnungen an. Diese Operation ist somit unbhängig davon, welches Modell bzw. welche Granularität für die möglichen Größen von Objekten festgelegt wurde. Desweiteren fällt eine Toleranzbetrachtung als Nebenprodukt des Vergleichs zweier qualitativer Größen ab.

Operator: Entfernung ($\ominus$)

$$\ominus : polygon \quad \text{x} \quad polygon \longrightarrow entfernung$$

Zusätzlich zu den Ausführungen unter 3.6.1 muß bei dieser Operation geklärt werden, wie verschiedene Entfernungsmodelle betrachtet werden. Da gegenwärtig nur nach Objekten „in der Nähe von“ Referenzobjekten gefragt wird, kann man das zugrundeliegende qualitative Entfernungsmodell auf „nah“ und „fern“ beschränken

Operator: Richtung ($\oplus$)

$$\oplus : punkt \quad \text{x} \quad punkt \longrightarrow richtung$$

Die verwendeten qualitativen Modelle von Richtungen sind derzeit die 4 Himmelsrichtungen (nördlich,westlich,südlich,östlich) sowie die Richtungsangaben „oberhalb“ und „unterhalb“.

Operator: Zwischen ($|\cdot|$)

$$|\cdot| : polygon \quad \text{x} \quad polygon \longrightarrow liste(polygon)$$

Die erste Herangehensweise zur Realisierung eines Zwischen-Operators, der ermitteln kann, welche Objekte sich zwischen zwei Referenzobjekten befinden, war durch Komposition der Nachbar-Relation angedacht. Abbildungen 3.8 und 3.9 zeigen jedoch, daß Nachbarbeziehungen diesen Sachverhalt nicht definieren können. Es stellte sich vielmehr heraus, daß die Nachbar-Eigenschaft die Verwendung des Zwischen-Operators bedingt.

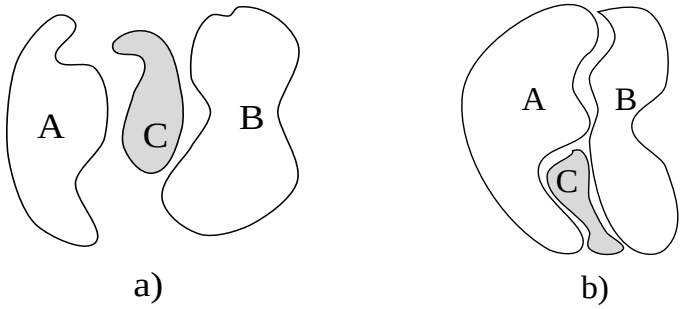

Abbildung 3.8: Zwei Fälle intermediärer Objekte: a) A und B sind nicht Nachbar, b) A und B sind Nachbarn.

Diese Fälle decken auch nicht die Möglichkeit ab, daß sich zwischen zwei beliebigen Objekten mehrere andere Objekte befinden können und daß hierbei die Entfernung der beiden Referenzobjekte beliebig groß sein kann.

Der Zwischen-Operator wird daher durch andere primitive Operationen approximiert, um beiden obigen Anforderungen gerecht zu werden. Abbildung 3.10 b) soll dieses Verfahren veranschaulichen. Man sieht, daß durch bestimmte Eckpunkte der umschließenden Rechtecke und ihre Verbindungslinien (die gleichzeitig Richtungsinformationen liefern), elf Zonen gebildet werden. Alle Objekte, die in den Zonen 4 bis 8 liegen, werden als *zwischenliegend* zwischen $\boldsymbol{X}$ und $\boldsymbol{Y}$ betrachtet. Der Vorteil dieser Approximation ist ihre effiziente Lösung.

Durch die zwei Paare von Verbindungslinien (ausgehend in den Punkten **e1** und **e2**) werden zwei Richtungskegel festgelegt. Man kann also nun den bereits vorhandenen Test - befindet sich Objekt X innerhalb eines gegebenen Richtungskegels - nutzen, um zwei Listen zu bekommen. Die Schnittmenge dieser zwei Listen entspricht Objekten, welche sich in den Zonen 4 bis 8 sowie 2 und 10 befinden.

Die jeweiligen Diagonalen in den umschließenden Rechtecken eliminieren die in den Zonen 2 und 10 liegenden Lösungen.

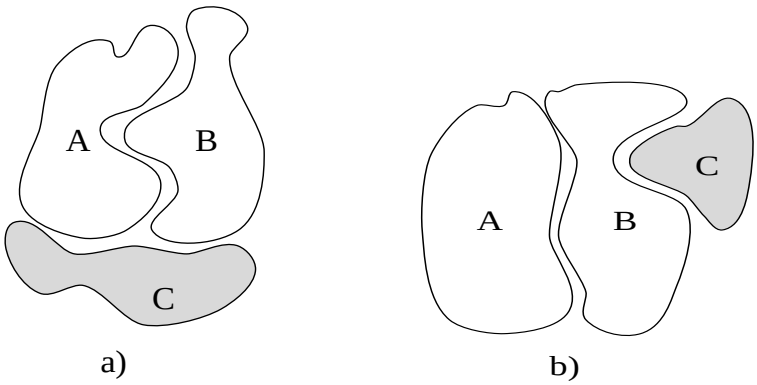

Abbildung 3.9: Zwei Fälle nicht-intermediärer Objekte: In beiden Fällen sind A und B Nachbarn.

Operator: Nachbar ($\asymp$)

$$\asymp: polygon \quad \text{x} \quad polygon \longrightarrow bool$$

Will man eine allgemeine Nachbar-Eigenschaft, so darf die Entfernung zwischen zwei Objekten, die dieser Eigenschaft genügen sollen, keine Rolle spielen. Der Raum zwischen zwei Objekten - seine Ausmaße kennzeichnen ja die Entfernung - ist nicht Bestandteil des Pools an Objekten (Universum), die für diese Eigenschaft betrachtet werden. Ein Beispiel der Handhabe am heutigen Globus: Madagaskar und Afrika sind Nachbarn (ca. 410km), ebenso wie Mallorca und Menorca (ca. 38km). Menorca und Ibiza (ca. 220km) wegen Mallorca jedoch nicht.

Die Entfernung spielt überhaupt keine Rolle, vielmehr die Frage, ob sich ein weiteres Objekt zwischen den Referenzobjekten befindet. Nun befinden sich zwischen Madagaskar und Afrika noch die Komoren und eine Unzahl kleiner (Küsten-)Inseln. Sicher wird man auch zwischen Mallorca und Menorca den einen oder anderen Felsen finden.

Das zwischenliegende Objekt muß also hinreichend klein sein, um ignoriert werden zu können. Es ist klar, daß keine absolute Schwelle für solch eine Größe angegeben werden kann, sondern daß eine Abschätzung relativ zur Größe der Referenzobjekte geschehen muß.[7]

Folgender Algorithmus kann für einen Nachbar-Operator verwendet werden:

Algorithmus nachbar: Test auf Nachbarschaft.

$\mathcal{A} := \boldsymbol{A} \mid \cdot \mid \boldsymbol{B}$	! 1 Objekte zwischen A und B
foreach $\boldsymbol{X}$ *in* $\mathcal{A}$	! 2 Alle Elemente durchlaufen
if $\lvert\boldsymbol{X}\rvert >= \lvert\boldsymbol{A}\rvert$ *or*	! 3 Falls zwischenliegendes El.
$\lvert\boldsymbol{X}\rvert >= \lvert\boldsymbol{B}\rvert$	! mind. gleiche Größenordnung
return false	! 4 $\Rightarrow$ keine Nachbarn
return true	! 5 Alle kleiner: Nachbarn

[7] Immerhin ist eine Insel der Komoren hinsichtlich ihrer Größe mit Mallorca vergleichbar.

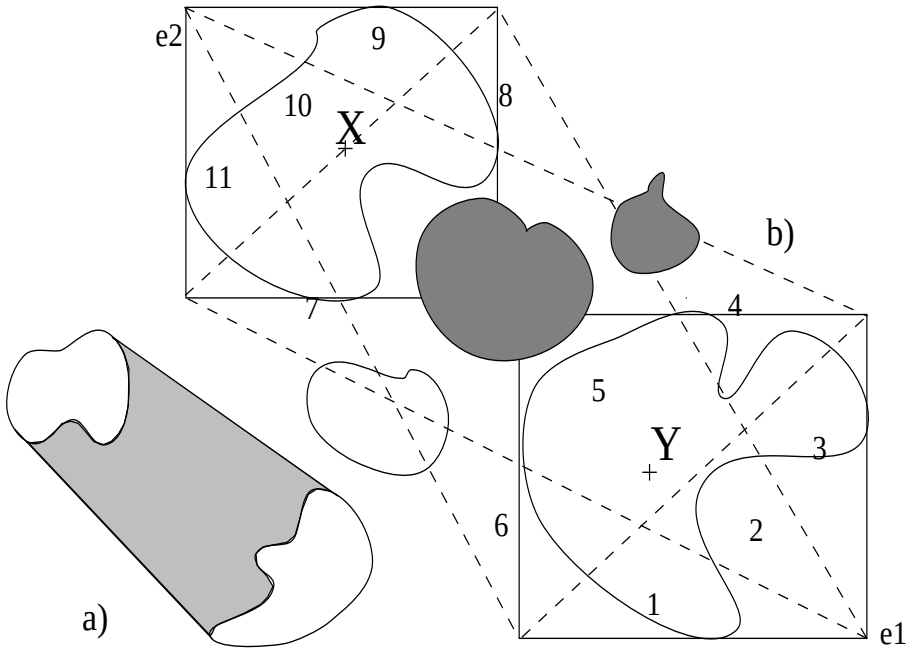

Abbildung 3.10: a) Idealer Test auf Kriterium „zwischen“ b) Approximation durch umschließendes Rechteck und Kombination komplementärer Richtungen.

3.8.2 Die vier essentiellen topologischen Relationen

Wie eingangs unter 2.1.1 erwähnt, lassen sich bei Vorhandensein der Konzepte „Abstand“ und „Richtung“ die in der Literatur erwähnten acht topologischen Relationen auf vier essentielle Relationen reduzieren **und** die Aussagekraft der Formalismen steigern. Der unter 3.8.1 erwähnte Relationsklassifikator soll aufgrund von zwei gegebenen Objekten die Relation liefern, in der diese beiden Objekte zueinander stehen. Im folgenden sollen die möglichen Relationen und ihre Eigenschaften besprochen werden.

Identisch

Zwei Objekte werden als identisch erkannt, wenn ihre Polygonzüge und ihre Position identisch sind. Hierbei kann es sich um zwei Instanzen eines Objektes handeln oder um zwei Referenzen auf ein Objekt. (Gleichung 3.6).

Die Identität zweier Polygonzüge festzustellen ist nicht so trivial, wie es auf den ersten Blick erscheinen mag. Zum einen ist bei einem solchen Vergleich die Orientierung beider Polygone zu berücksichtigen, zum anderen kann nicht davon ausgegangen werden, daß beide Polygone den gleichen Offset besitzen. Mit anderen Worten: Das erste Element in der geordneten Liste der Eckpunkte des Polygonzuges kann ein beliebiger Eckpunkt des Polygons sein.

Disjunkt

Zwei disjunkte Objekte haben keinen Punkt gemeinsam. Als schneller Test auf Disjunktheit lassen sich Umrandungen der erfragten Objekte miteinander ver-

gleichen. (Gleichung 3.7 und Abbildung 3.14)

Einen exakten Test erhält man nur beim Test eines Schnittes zweier beliebiger Polygone. Die resultierende Menge muß leer sein. Ein solcher Test ist jedoch quantitativ und mit einem quadratischen Aufwand an Rechenzeit verbunden, weswegen er hier nicht betrachtet werden soll. Als Approximation dient der Test mit einer einfachen geometrischen Figur.

Schneidet

Zwei Objekte schneiden sich genau dann, wenn sie mindestens einen Punkt gemeinsam haben. Somit fällt hier auch der in anderen Modellen differenzierte Sachverhalt „berührt“ in diese Kategorie. Der Schnitt zweier Objekte ist Vorbedingung für die Relation „Innerhalb“ (Gleichung 3.8).

In der von Grigni et al. liegt dieses Modell zwischen der dort postulierten *medium* und *low resolution*.

Hierbei werden die Relationen „schneidet“ und „berührt“ zu „schneidet“ zusammengefaßt. Die Relation „enthält“ wird nicht durch den Relationsklassifikator, sondern durch den allgemeineren #-Operator abgedeckt.

Innerhalb

Der vollständige Einschluß eines Objektes in einem anderen ist ein Spezialfall des Schnittes zweier Objekte. Daher gilt für diesen Fall das gleiche Kriterium wie für Relation „schneidet“, mit der Zusatzeinschränkung, daß die Schnittmenge mit dem innenliegenden Objekt identisch sein muß.

Randell et al. [RCC92] unterscheiden zwischen „topologisch innerhalb“ und „geometrisch innerhalb“ (vgl. Abb. 3.11). Als geometrisch innerhalb wird ein Objekt angesehen, welches sich vollständig innerhalb der konvexen Hülle eines anderen Objektes befindet. Für eine natürlichsprachliche Verwendung des Wortes „innerhalb“ ist jedoch die Eigenschaft *topologisch* zu scharf, *geometrisch* hingegen notwendig aber nicht hinreichend (vgl. Abb. 3.12).

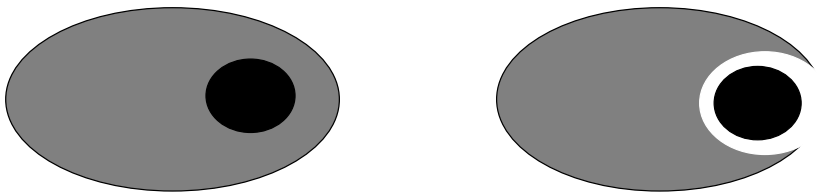

Abbildung 3.11: In der linken Abbildung befindet sich die schwarze Ellipse topologisch innerhalb der grauen Figur, in der rechten Abbildung geometrisch innerhalb.

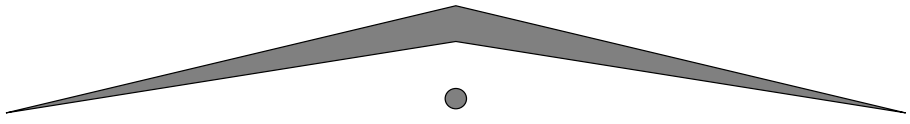

Abbildung 3.12: Obwohl sich der kleine Kreis „geometrisch innerhalb“ der großen Figur befindet, wäre die Formulierung eher „unterhalb“.

3.8.3 Sprachlich innerhalb - möglicher Lösungsansatz

Eine mögliche Lösung des Problems „sprachlich innerhalb“ könnte der in Abbildung 3.13 gezeigte Ansatz bringen. Objekte, die sich topologisch innerhalb eines gegebenen Referenzobjektes befinden, stellen keine Zweifelsfälle dar. Problematisch sind nur einige Objekte innerhalb der konvexen Hülle dieses Referenzobjektes.

Man kann diese Problemobjekte in zwei (sich nicht ausschließende) Klassen einteilen.

Grenzgänger

Damit sind Objekte gemeint, welche sich - zwar noch innerhalb - sehr nahe der konvexen Hülle befinden und durch ihre Entfernung zu dem eigentlichen Körper des Referenzobjektes als nicht innerhalb angesehen werden. In der Abbildung entspricht dies den Objekten mit der Nummer 2a und 2b.

Quasi-Nebenliegende

Die Bezeichnung dieser Objekte ist etwas problematisch, da im Grunde nicht ihre Eigenschaft, sondern die des Referenzobjektes von Interesse ist. Gemeint sind hier Objekte, welche sich in einer sehr seichten konkaven Krümmung des Referenzobjektes befinden und dadurch als nebenliegend erscheinen. Objekt 3 und seine Lage sollen diesen Sachverhalt skizzieren. Oft - aber nicht immer - sind Quasi-Nebenliegende auch Grenzgänger.

Die Idee ist also, mit Hilfe von *Sichtbarkeitskegeln* einen Indikator für „sprachlich innerhalb“/nicht „sprachlich innerhalb“ zu erhalten. Ein solcher Sichtbarkeitskegel kennzeichnet einen Winkel, unter dem das erfragte Objekt von außerhalb der konvexen Hülle des Referenzobjektes sichtbar ist, bzw. welcher Außenbereich von ihm aus sichtbar ist - je nach Betrachtung.

Die Vermutung ist nun, daß ein spitzer Kegel (geringer Betrachtungswinkel vgl. Objekte 1 und 4) ein innenliegendes Objekt identifiziert, während ein sehr stumpfer Kegel ein Objekt anzeigt, welches in eine (oder beide) der oben angegebenen Kategorien fällt.

Es gilt in erster Linie herauszufinden, welche Winkelgrößen zum Grenzbereich bei der Unterscheidung gehören. Die Formulierung „spitz“ und „stumpf“ ist daher nicht exakt übereinstimmend mit dem in der Geometrie üblichen Sprachgebrauch. Allerdings kann ein überstumpfer Kegel als eindeutiges Indiz für „außerhalb“ gelten.

Gegeben seien die Objekte $\boldsymbol{A}$ (Referenzobjekt) und $\boldsymbol{B}$ (innenliegendes Objekt?). Folgende Schritte sind zum Ermitteln eines Sichtbarkeitskegels notwendig:

1. Befindet sich $\boldsymbol{B}$ innerhalb der konvexen Hülle von $\boldsymbol{A}$ ($\boldsymbol{A_h}$)? Falls ja: $\Rightarrow$ 2, falls nein: $\Rightarrow$ 4

2. Schnitt von $\boldsymbol{A}$ mit $\boldsymbol{A_h}$. Befindet sich $\boldsymbol{B}$ innerhalb eines der resultierenden Polygone ($\boldsymbol{C_i}$)? Falls ja: $\Rightarrow$ 3, falls nein: $\Rightarrow$ 5

3. Ermittle die beiden gemeinsamen Punkte von $\boldsymbol{C_i}$ mit $\boldsymbol{A_h}$. Verbinde diese mit geeigneten Randpunkten, oder dem Mittelpunkt von $\boldsymbol{B}$. Die Differenz der Steigungen der resultierenden Geraden ergibt den Winkel des

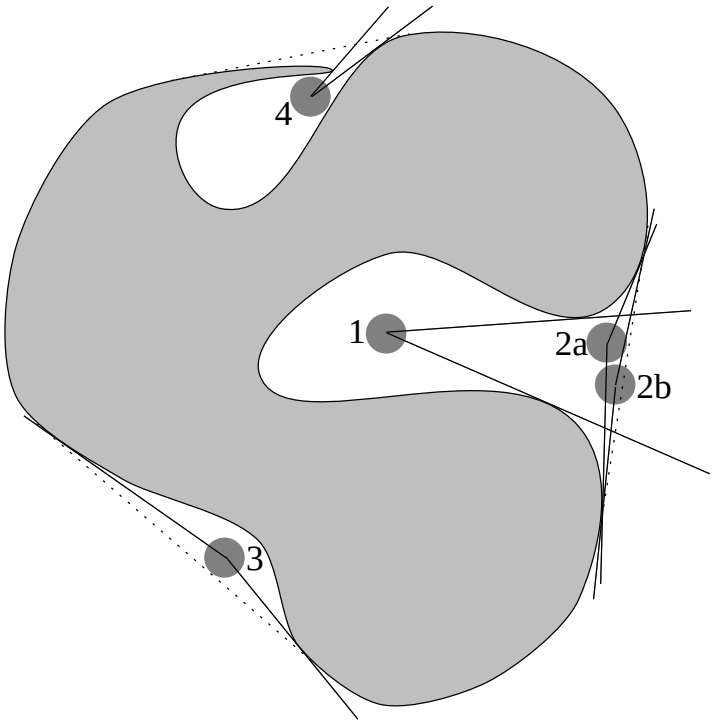

Abbildung 3.13: Mögliche Lösung für das Problem "sprachlich innerhalb": Sichtbarkeitskegel. Die gestrichelten Linien sollen die konvexe Hülle des Objektes kennzeichnen.

Sichtbarkeitskegels. Ist der Winkel oberhalb einer bestimmten Schranke (stumpf)? Falls ja: $\Rightarrow$ 4, ansonsten $\Rightarrow$ 5

4. Ende, $\boldsymbol{B}$ nicht innerhalb von $\boldsymbol{A}$
5. Ende, $\boldsymbol{B}$ ist innerhalb von $\boldsymbol{A}$

3.8.4 Methoden der Klassifikation topologischer Relationen

Die Genauigkeit mit der die bestehende topologische Relation zwischen zwei Objekten bestimmt werden kann, hängt in großem Maße von der verwendeten Funktion zur Schnittmengenbestimmung ab. Hierbei gibt es drei hinsichtlich Genauigkeit und Effizienz unterschiedliche Verfahren:

Approximation durch einfache geometrische Figur

Hierbei wird das zu schneidende Objekt durch eine einfache geometrische Figur wie z.B. Ellipse oder umschließendes Rechteck approximiert. Obwohl dies eine recht grobe Annäherung bedeutet, ist das damit erzielbare Ergebnis für qualitative Aussagen meist ausreichend und sehr effizient erzielbar.

Durch einen nachgeschalteten Flächenvergleich läßt sich die Qualität dieser Methode gegen einen kleinen Laufzeit-Aufpreis noch deutlich verbes-

sern. Der Schnitt zweier Objekte hat bei dieser Methode $\mathcal{O}(1)$ Komplexität, die Vorverarbeitung $\mathcal{O}(k)$, wobei k die Anzahl der Punkte des Polygons ist.

Approximation durch konvexe Hülle
Die konvexe Hülle eines Objektes stellt eine bessere Approximation dar und eignet sich gut zur Schnittmengenbestimmung. Der Schnitt zweier konvexer Objekte hat die Komplexität $\mathcal{O}(n+k)$, wobei k die Anzahl der Schnittpunkte zwischen diesen Objekten ist. Die Vorverarbeitung beträgt $\mathcal{O}(n)$ und $\mathcal{O}((n+k)\log n)$. Dies ist der Aufwand für das Erstellen der konvexen Hülle und des Verbindungsgraphen.

Exakter Schnitt
Beim exakten Schnitt müssen auch konkave Objekte mit konkaven Objekten geschnitten werden. Prinzipiell verhält sich hier die Laufzeitkomplexität ebenfalls $\mathcal{O}(n+k)$ mit gleicher Vorverarbeitung, jedoch kann vorkommen $k = \mathcal{O}(n^2)$, da zwei Polygone mit n Kanten $\frac{n^2}{2}$ Schnittpunkte haben können (worst case).

Es sollte angemerkt werden, daß die Vorverarbeitungszeiten (bis auf das Erstellen des Verbindungsgraphen) bei nichtdynamischen Anwendungen und entsprechender Organisation des Datensatzes nur einmal, etwa bei der Initialisierung des Systems, anfallen. Siehe hierzu auch Abschnitt 6.1.

3.8.5 Ergebnisse der einzelnen Methoden

In den folgenden Ausführungen bezeichnet $?_{\square}$ den Relationsklassifikator basierend auf dem Schnitt einfacher konvexer geometrischen Figuren (in unserem Fall Rechtecke), $?_{\diamond}$ basiert auf dem Schnitt konvexer Hüllen und $?_{\bowtie}$ auf einem exakten Schnitt (auch) konkaver Polygone. Abbildung 3.14 zeigt Ergebnisse und mögliche Interpretationen von $?_{\square}$.

$?_{\square}$ liefert	real	$?_{\square}$ liefert	real
identisch	identisch	innerhalb	disjunkt
	innerhalb		innerhalb
	schneidet		schneidet
schneidet	schneidet	disjunkt	disjunkt
	disjunkt		

Abbildung 3.14: Interpretationstabelle zum $?_{\square}$ Operator. Das Ergebnis „disjunkt“ ist korrekt.

Die Interpretation der Ergebnisse von $?_{\diamond}$ zeigt, daß diese Klassifikation hinsichtlich „identisch“, „schneidet“ und „disjunkt“ zu $?_{\square}$ äquivalent ist. Das Ergebnis „innerhalb“ zeigt wieder einmal deutlich die problematische Zuordnung natürlichsprachlicher Präpositionen. Die möglichen Interpretationen sind:

topologisch innerhalb
: Die konvexe Hülle einer innenliegenden Figur befindet sich vollständig inerhalb der konvexen Hülle einer größeren Figur. Siehe auch linke Abbildung unter 3.11.

geometrisch innerhalb
: Die innenliegende Figur hat keine gemeinsamen Punkte mit der umgebenden Figur, befindet sich jedoch innerhalb ihrer konvexen Hülle. Siehe auch rechte Abbildung unter 3.11.

schneidet
: Beide Operanden können sich zwar nur schneiden, der Kleinere befindet sich jedoch innerhalb der konvexen Hülle. Logisch kann man dies als Zwischenfall der beiden oben angegebenen Variationen von „innerhalb“ ansehen.

Randell et al. [RCC92] bezeichnen der Einfachheit halber diese Fälle von „innerhalb“ als „geometrisch innerhalb“. Man kann also sagen, daß $?_{\diamond}$ bei diesem Ergebnis hinreichend korrekt ist.

Der durch $?_{\bowtie}$ realisierte exakte Schnitt zweier (auch konkaver) Polygone liefert stets korrekte Ergebnisse bei „identisch“, „schneidet“ und „innerhalb“. Disjunktheit kann hier also auch „geometrisch innerhalb“' bedeuten.

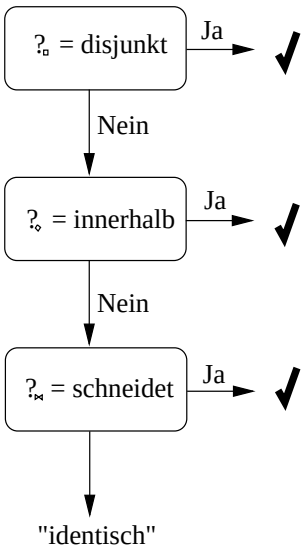

Abbildung 3.15: Komposition des ?-Operators. .

Kapitel 4

Anmerkungen zur Implementierung

4.1 Die Datenstruktur eines Objektes

Die Basisklasse der für die Inferenzkomponente relevanten Instanzen ist das `Globus-Objekt`. Die hier definierten Eigenschaften werden an die abgeleiteten Klassen vererbt. Der folgende Quellcodeausschnitt zeigt die entsprechende Definition:

```
(define-class (globus-objekt (:documentation
                              "Klasse aller Globus-Objekte"))
  (globus-root)
  (kommentar :type hytext)          ; inhaltlicher Kommentar
  (farbe :type farbe)
  (zustand :type zustand)
  (erhaltung :type erhaltungszustand)
  (quelle :type zitat)
  (metakommentar :type string)      ; Kommentar zu restl. Eigenschaften
  (globus-koord :type koordinate)
  (globus-polygon :type koordinate) ; Polygon um Obj. in (Breite Länge)
  (pixel-polygon :type pixelkoord)  ; Polygon um Obj. in x-Pixel y-Pixel
  (rasterfeld :type rasterfeld)
  (lfdnr :type posinteger)       ; laufende Nr. innerhalb des Rasterfelds
                                 ; 0 <= lfdnr <= 9999
                                 ; = 0, wenn Objekt größer als Rasterfeld
  (medienobj :type medienobjekt)
  (urheber :type string)            ; username+datum
)
```

Für die Inferenzkomponente sind die Felder `globus-koord`, `globus-polygon` und `pixel-polygon` von Interesse.

globus-koord
: Enthält das „Zentrum“ eines Objektes. Es handelt sich dabei lediglich um ein Koordinatenpaar im Format (Breite Länge), also einen Punkt. Diese Daten wurden geschätzt.

globus-polygon
: Ein Polygonzug um ein Objekt, repräsentiert als geordnete Liste von Punkten. Jeder Punkt ist in Kugelkoordinaten angegeben und hat das Format (Breite Länge).

pixel-polygon
: Der gleiche Polygonzug wie `globus-polygon`, jedoch angegeben in Pixelkoordinaten. Diese sind aus den Kugelkoordinaten mittels einer Mercator-Projektion auf eine Zylinderoberfläche errechnet worden. Objekte, welche sich auf den Polkappen befinden, werden durch diese Koordinaten nicht erfaßt.

```
(define-class (g-region (:documentation "Geographische Region"))
  (globus-region)
  (moderner-name :type string)
  (original-name :type name)
  (moderne-koord :type koordinate)
  (g-teil-von :type g-region)  ; eine Ebene uebergeordnete Region(en)
                               ; Hauptregion MUSS an 1.Stelle stehen
  (g-nachbar :type g-region)   ; gleichgeordnet
  (g-teile :type g-region)     ; eine Ebene untergeordnet
  (ng-teile :type ng-region)   ; eine Ebene untergeordnet
     ; Relation? disjoint tangency overlap contains included = / @border
)
```

4.2 Design der Inferenzkomponente

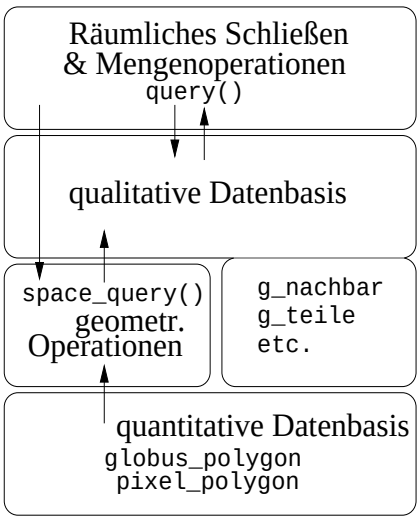

Abbildung 4.1: Design der Inferenzkomponente und die zugrundeliegenden Datenbasen.

Bei der Konzeption der Inferenzkomponente war zu beachten, daß sowohl quantitative als auch qualitative Daten vorliegen. Weiterhin sollten Anfragen zu den räumlichen Eigenschaften von Objekten mit Anfragen nach sonstigen

Attributen, wie *Farbe* oder *Erhaltungszustand*, kombiniert werden können. Bild 4.1 zeigt die Verarbeitungshierarchie der Komponente.

Ein Modul für geometrische Operationen, basierend auf quantitativen Daten, erstellt[1] Listen von Objekten, die qualitativen Eigenschaften wie *nördlich, südlich* etc. genügen. Diese Ergebnisse stehen, zusammen mit den bereits vorhandenen - qualitativen - Daten über Nachbarschaft, Teilgebiet etc., einem übergeordneten Modul zur Verfügung. Dieses kann nun logisch verknüpfte Anfragen zu diversen Attributen von Objekten in einfache Mengenoperationen auf den Ergebnislisten umwandeln (siehe hierzu auch Abschnitt 3.5.3).

Das Erstellen der Ergebnislisten bedeutet im Vergleich zu den qualitativ orientierten Operationen einen verhältnismäßig hohen Rechenaufwand. In Abschnitt 6.1 wird erläutert, wie man den Aufwand durch die Hinzunahme neuer Datenslots für jeweils eine Liste reduzieren kann. Eine weitere Möglichkeit für eine Beschleunigung dieser Konversion ist Parallelisierung: Alle Ergebnislisten sind bis zur Modifikation durch die Mengenoperationen voneinander unabhängig. Eine kombinierte Anfrage aus n, durch logische Verknüpfungen zusammengesetzten Teilanfragen, kann von n unabhängigen Maschinen bearbeitet werden. Es ist sogar denkbar, jede dieser Teilanfragen wiederum auf m Maschinen aufzuteilen, welche nur für bestimmte Teilbereiche der Datenbasis verantwortlich sind.

4.3 Schema einer Anfrage

Abbildung 4.2 zeigt den zentralen Dialog für Anfragen an die Datenbasis des Behaim-Projektes. Hiermit können Kombinationen verschiedener Attribute zu einer gewünschten Objektklasse angefragt werden. Ebenso ist es möglich Objekte zu erfragen, welche in einer bestimmten Relation zu einem beliebigen Referenzobjekt stehen. Räumliche Eigenschaften werden ebenfalls als Attribute bezeichnet. Die in der Anfrage gegenwärtig implementierten Typen sind `noerdlich_von, westlich_von, suedlich_von, oestlich_von, in_der_naehe_von, oberhalb_von` und `unterhalb_von`.

Die Richtungsangaben lassen sich direkt auf ein Modell mit vier Richtungskegeln abbilden (vgl. Tabelle 3.2). Die Attribute `oberhalb` und `unterhalb` können als ein Modell von Nord-Süd Halbebenen verstanden werden.

Die Kernsyntax der derzeit möglichen Anfragen in BNF:

```
<nenne/zeige> ::= nenne | zeige

<bool_op> ::= und | oder | und_nicht

<eigenschaft> ::= <attribut_1> <belegung_1>
                | ...
                | <attribut_n> <belegung_n>

<teilanfrage> ::= <nenne/zeige> <objklasse> [mit] <eigenschaft>

<anfrage> ::= <teilanfrage> | <anfrage> <bool_op> <teilanfrage>
```

[1] derzeit noch „on the fly“ (siehe Abschnitt 6.1)

Abbildung 4.2: Das JAVA-basierte Anfrageformular. Es können Kombinationen verschiedener Attribute zu einer gewünschten Objektklasse angefragt werden. Das Referenzobjekt ist hierbei beliebig.

Die Nichtterminalsymbole `objklasse, attribut_X` und `belegung_X` werden hier wegen ihres Umfangs nicht aufgeführt. Zu beachten ist auch die oben angedeutete Abhängigkeit der Belegungen von den Attributen.

Die oben erwähnten Richtungsangaben sind somit normale Attribute. Die gültigen Belegungen hierfür sind alle Instanzen der Objektklasse `globus_region` oder hiervon abgeleiteter Klassen. Eine Übersicht dieser Hierarchie gibt Abbildung 4.3.

4.4 Einbettung in den Anfragevorgang

Nachdem Anfragen über räumliche Eigenschaften von Objektklassen in der gleichen Form wie wie die sonstige Attribute erfolgen, muß auch ihre Behandlung einheitlich erfolgen. Insbesondere werden Anfragen durch einfache logische Operationen wie `UND` und `ODER` verknüpft. Diese wiederum lassen sich in die entsprechenden Mengenoperationen umformen (vgl auch Abschnitt 3.5.3 auf Seite 16.

Die Schritte einer Anfrage und die Einbettung der Komponente zum Räumlichen Schließen sind in Abbildung 4.4 abgebildet. Die vollständige An-

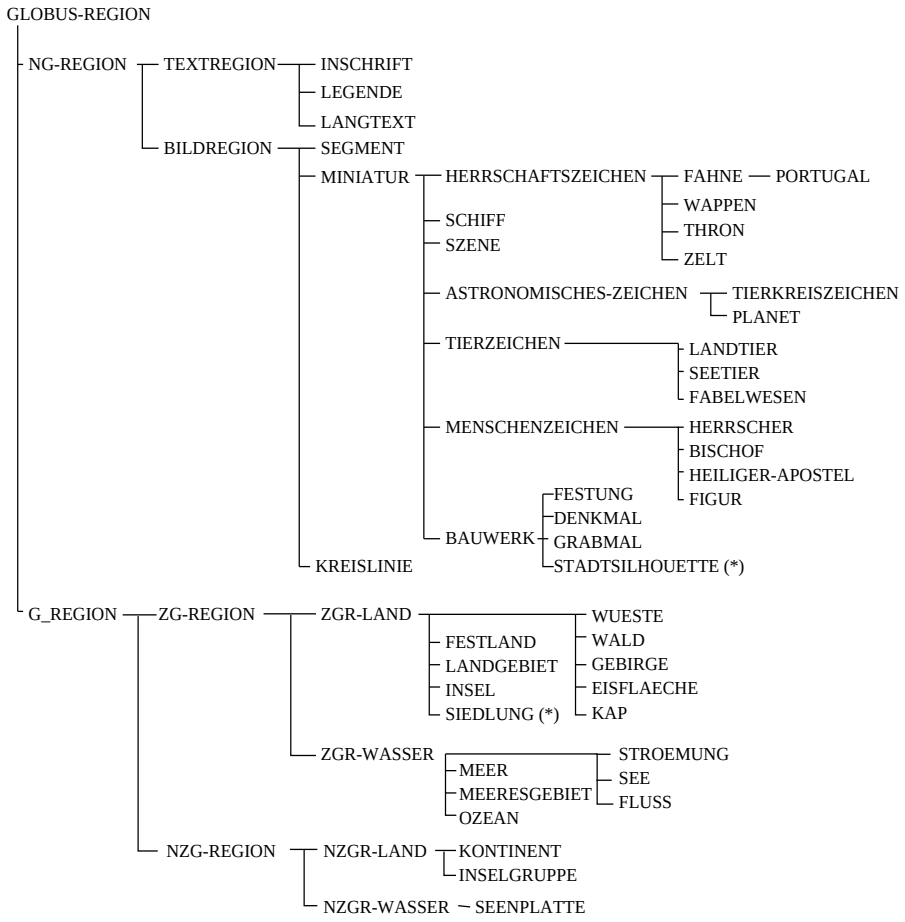

Abbildung 4.3: Die Hierarchie der für das räumliche Schließen interessanten Objektklassen.

frage (erzeugt mit dem in Bild 4.2 dargestellten Anfrageformular) wird an die zentrale Komponente - Query - geleitet. Diese zerlegt die Anfrage in einzelne Teilanfragen. Enthalten diese als Attribut räumliche Sachverhalte, wird die Anfrage mit den Instanzen des Suchraumes an die Space-Query Komponente weitergeleitet.

Diese fungiert nun wie ein Filter und gibt nur diejenigen Instanzen zurück, welche eine dem erfragten Attribut entsprechende Eigenschaft besitzen. Diese Ergebnisse werden dann - wie bereits vorhin erwähnt - von der zentralen Komponente mittels Mengenoperationen zu einer Liste von Instanzen zusammengesetzt, die der Kombination aller erfragten Eigenschaften gerecht wird. Diese Liste wird an die Ergebnisanzeige weitergegeben.

Derzeit geschieht die Bearbeitung der einzelnen Anfragen sequentiell. Da für den Test auf einzelne Eigenschaften der vollständige Datensatz[2] iteriert werden muß, steigt der Aufwand mit Zunahme der Objekte in der Datenbank.

Im Vergleich zu der direkten Datenbankanfrage der Space-Query Kompo-

[2]Natürlich nur die Instanzen der jeweils relevanten Objektklasse

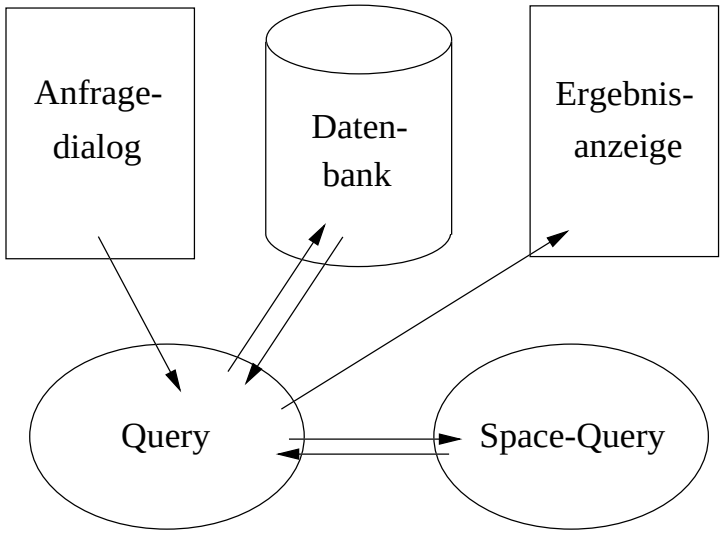

Abbildung 4.4: Einbettung der Inferenzkomponente `Space-Query` in den Anfragevorgang.

nente erlaubt das jetzige Vorgehen eine einfache Auslagerung einer (rechenintensiven) Inferenzkomponente. Sämtliche Teilanfragen lassen sich ja unabhängig voneinander bearbeiten. Bei einer vorhandenen Datenbank empfiehlt es sich, die zentrale Query-Komponente physikalisch auf der Maschine mit der Datenbank laufen zu lassen, da sie zwar datenbankintensive, jedoch nicht rechenintensive Operationen vornimmt.

Eine direkte Kommunikation von Space-Query mit der Datenbank erscheint erst sinnvoll, wenn diese auf mehrere Maschinen verteilt wäre. Hierbei würden auch restringierte Kopien der Datenbank genügen, da Space-Query ohnehin nicht alle Objektklassen betrachten muß.

Kapitel 5

Ergebnisse

Im Folgenden sollen einige Anfragen an die Inferenzkomponente sowie die daraus resultierenden Ergebnisse vorgestellt werden. Laufzeitmessungen der `space_query`-Komponente sollen Aufschluß über etwaige Performanzprobleme liefern. Zur Laufzeitmessung wird das in [Ste90] beschriebene time-Makro verwendet. Die hier gezeigten Ergebnisse sind der Log-Datei des bhmclos-Servers entnommen und beschreiben die Laufzeiteigenschaften von interpretiertem LISP-Code. Der verwendete Interpreter ist Allegro Common Lisp Version 4.2 für die SUN Plattform.

5.1 Richtungen

Eine Anfrage wird mittels get-query(`<query>`) protokolliert. Hierbei besteht `query` aus dem Schlüsselwort „Query“, gefolgt von der Identifikation des anfragenden Benutzers. Der Rest ist die eigentliche Anfrage entsprechend der in Abschnitt 4.3 beschriebenen Anfragesyntax.

Anfrage nach westlich liegenden Objekten

```
qet-query: ("QUERY" "USER-29441" "Nenne" "SEETIER"
                                 "westlich_von SCHIFF_12_3")
```

Der bei den folgenden Anfragen zugrundeliegende Suchraum (alle Instanzen der Klasse Seetier), ist stets gleich und soll daher hier nur einmal angegeben werden. Gegenwärtig handelt es sich um elf konkrete Instanzen. Die hier besprochenen Objekte können im Anhang B nachgesehen werden.

```
my-retrieve:erg: (#<SEETIER_SEESCHLANGE_18_5 (#1=a SEETIER)>
                  #<SEETIER_SEESCHLANGE_16_2 (#1# SEETIER)>
                  #<SEETIER_SEESCHLANGE_16_5 (#1# SEETIER)>
                  #<SEETIER_FISCH_14_4 (#1# SEETIER)>
                  #<SEETIER_SEESCHLANGE_14_2 (#1# SEETIER)>
                  #<SEETIER_FISCH_12_7 (#1# SEETIER)>
                  #<SEETIER_FISCH_2_12_5 (#1# SEETIER)>
                  #<SEETIER_FISCH_1_12_5 (#1# SEETIER)>
                  #<SEETIER_FISCH_12_3 (#1# SEETIER)>
                  #<SEETIER_SEESCHLANGE_10_2 (#1# SEETIER)>
                  #<SEETIER_FISCH_10_1 (#1# SEETIER)>)
```

Das Profil für Laufzeit und Speicherbedarf wird von `time` wie folgt angegeben:

```
"space-query" Referenzobjekt(Richtung):  (488905/44 246647/44)
; cpu time (non-gc) 400 msec user, 20 msec system
; cpu time (gc)     0 msec user, 0 msec system
; cpu time (total)  400 msec user, 20 msec system
; real time  420 msec
; space allocation:
 17,070 cons cells,;  3 symbols, 95,408 other bytes
```

Das korrekte Ergebnis sind folgende im Segment 10 (vgl. Abbildung B.2) befindlichen zwei Objekte.

```
SEETIER: FISCH_10_1
SEETIER: SEESCHLANGE_10_2
```

Anfrage nach östlich liegenden Objekten

```
qet-query: ("QUERY" "USER-29441" "Nenne" "SEETIER"
                                         "oestlich_von SCHIFF_12_3")


; cpu time (non-gc) 560 msec user, 50 msec system
; cpu time (gc)     40 msec user, 0 msec system
; cpu time (total)  600 msec user, 50 msec system
; real time  641 msec
; space allocation:
 34,131 cons cells,;  6 symbols, 190,736 other bytes
```

Die fünf resultierenden Ergebnisse befinden sich in den Segmenten östlich zu Segment 12 (14, 16 und 18):

```
SEETIER: SEESCHLANGE_14_2
SEETIER: SEESCHLANGE_16_2
SEETIER: FISCH_14_4
SEETIER: SEESCHLANGE_16_5
SEETIER: SEESCHLANGE_18_5
```

Kombinierte Anfrage

```
qet-query: ("QUERY" "USER-29441" "Nenne" "SEETIER"
                                         "westlich_von SCHIFF_12_3"
                                         "OR"
                                         "oestlich_von SCHIFF_12_3")
```

Diese Anfrage beinhaltet zwei Aufrufe von Space-Query, die hinsichtlich Ihrer Laufzeit und dem Speicherbedarf jeweils mit den beiden Einzelanfragen übereinstimmen.

```
"space-query" Referenzobjekt(Richtung):  (488905/44 246647/44)
; cpu time (non-gc) 330 msec user, 80 msec system
; cpu time (gc)     0 msec user, 0 msec system
; cpu time (total)  330 msec user, 80 msec system
```

```
; real time  417 msec
; space allocation:
 17,070 cons cells,;  3 symbols, 95,408 other bytes

"space-query" Referenzobjekt(Richtung):  (488905/44 246647/44)
; cpu time (non-gc) 350 msec user, 10 msec system
; cpu time (gc)     50 msec user, 0 msec system
; cpu time (total)  400 msec user, 10 msec system
; real time  410 msec
; space allocation:
 34,131 cons cells,;  6 symbols, 190,736 other bytes
```

Das Ergebnis entspricht der Vereinigungsmenge der beiden oben genannten Anfragen.

```
SEETIER: SEESCHLANGE_10_2
SEETIER: FISCH_10_1
SEETIER: SEESCHLANGE_14_2
SEETIER: SEESCHLANGE_16_2
SEETIER: FISCH_14_4
SEETIER: SEESCHLANGE_16_5
SEETIER: SEESCHLANGE_18_5
```

Jeder der beiden Anfragen ist hinsichtlich Speicherbedarf mit der zugehörigen oben erwähnten Einzelanfrage identisch. Die Rechenzeit für die zweite Anfrage ist signifikant kürzer, als das bei der Einzelanfrage der Fall ist. Da sich diese Abweichung nicht mit Meßungenauigkeiten erklären läßt, ist vermutlich internes Caching des Interpreters dafür verantwortlich.

Eine weitere Beschleunigung bei kombinierten Anfragen an die Space-Query Komponente könnte man (bei nicht verteiltem System) erzielen, wenn Space-Query selbst die logische Operation der Komposition durchführen würde. Das würde den mehrfachen Aufbau der Suchraumlisten einsparen. Der Code sollte dann überdies auch compiliert werden.

Anfrage nach nördlich und südlich liegenden Objekten

Eine den oben beschriebenen Anfragen äquivalente Version - jedoch mit nördlichen und südlichen Richtungen - liefert (Referenzobjekt ist wiederum `schiff_12_3`) bei der Frage nach nördlich gelegenen Seetieren als Ergebnis `fisch_12_3`. Die Anfrage nach südlich gelegenen Seetieren liefert die korrekten Instanzen `fisch_1_12_5` und `fisch_2_12_5`. Instanz `fisch_12_7` wird nicht angezeigt, da sie derzeit noch nicht erfaßt ist. Die Laufzeiten und Speicheranforderungen für diese Anfragen liegen in der gleichen Größenordnung wie ihre Pendants für „östlich“ und „westlich“.

5.2 Entfernung

Es gibt verschiedene Möglichkeiten für qualitative Modelle von Entfernungen. Der bereits bestehende Dialog (Abb. 4.2) ermöglicht Anfragen nach Objekten, welche sich `in_der_Naehe_von` einem angegebenen Referenzobjekt befinden. Das daraus resultierende Entfernungsmodell unterteilt Objekte somit in die Kategorien „nah“ und „fern“.

Nun gilt es noch die quantitativen Daten entsprechend auf diese Prädikate abzubilden. Hierbei erweist sich - wieder einmal - die Interpretation natürlichsprachlicher Präpositionen als das Hauptproblem.

Eine Anfrage an die Komponente kann also lauten: „Nenne alle zusammenhängenden geographischen Regionen in der Näche der Instanz schiff_12_3."

```
qet-query: ("QUERY" "USER-29441" "Nenne" "ZG-REGION"
                                          "in_der_Naehe_von SCHIFF_12_3")
```

Der Suchraum umfaßt bei dieser Anfrage in der gegenwärtigen Datenbasis 223 Objekte und wird daher hier nicht aufgeführt.

Folgende Modelle sind denkbar:

Fixe Intervallangabe

Die einfachste Herangehensweise würde darin bestehen, die maximale Entfernung, die zwei Objekte in dem System zueinander haben können, zu ermitteln und Entfernungen bis zu einem bestimmten Wert als „nah" zu betrachten.

Diese Herangehensweise ignoriert jedoch die Größe des Referenzobjektes selbst. Wählt man den Mittelpunkt eines Objektes als Referenzpunkt, so liegen bei großen Objekten viele - evtl. sogar alle - „Nahen" innerhalb.

Wählt man die Randpunkte, welche die kürzeste Entfernung zwischen den beiden Objekten definieren, dann werden alle Objekte unabhängig Ihrer Größe mit dem gleichen „Hoheitsbereich"[1] ausgestattet. Alles was sich in diesem Bereich befindet wird als „nah" angesehen. Diese Vorgehensweise mag für einige Aufgabenstellungen in GIS sinnvoll sein, es bleibt aber fraglich, ob sie dem natürlichsprachlichen Begriff von Nähe gerecht wird.

Die mit dieser Methode benötigte Rechenzeit sowie Speicherbedarf werden von `time` wie folgt angegeben:

```
; cpu time (non-gc) 1,550 msec user, 30 msec system
; cpu time (gc)     120 msec user, 0 msec system
; cpu time (total)  1,670 msec user, 30 msec system
; real time  1,953 msec
; space allocation:
 91,221 cons cells,;  97 symbols, 618,168 other bytes
```

Das Intervall (0 500) liefert insgesamt 19 Ergebnisse:

```
FLUSS: RIO_DE_MADALENA_12_4
FLUSS: RIO_DE_PATRON_12_4
FLUSS: RIO_SODEKOSO_12_4
KAP: CAP_DELTA_12_4
KAP: MUO_RUODO_12_4
LANDGEBIET: ANGRA_E_RIO_DE_FERNANTE_12_4
LANDGEBIET: CASTEL_PODEROSO_DE_SAN_AUGUSTIO_12_4
LANDGEBIET: GOLFO_DAS_ALMADIAS_12_4
LANDGEBIET: GOLFO_DA_SAN_MARTIN_12_3
LANDGEBIET: GOLFO_DE_JUDEO_12_3
```

[1] Man denke hierbei an maritimen Hoheitsgewässer von Staaten.

```
LANDGEBIET: PANTA_FORMOSA_12_3
LANDGEBIET: PONTA_ALTA_12_4
LANDGEBIET: PONTA_BIANCO_12_3
LANDGEBIET: PONTA_DE_MICH_12_4
LANDGEBIET: PONTA_FORMOSA_12_3
LANDGEBIET: REALI_12_3
LANDGEBIET: S_PAUL_12_4
STROEMUNG: 1_12_3
WUESTE: DESERTA_NA_12_3
```

Von der Objektgröße abhängige Intervallangabe

Die Annahme, die dieser Herangehensweise zugrundeliegt, besteht darin, daß bei der Betrachtung größerer Objekte eine höhere Toleranzschwelle für das Kriterium „Nähe“ vorliegt.

Bei dieser Vorgehensweise ist dann jedoch die Aussage, „$\boldsymbol{X}$ ist nahe bei $\boldsymbol{Y}$“ nicht mehr kommutativ. Wenn man dann davon spricht, daß zwei Objekte „nahe beieinander“ seien, so bedeutet dies immer, daß eines von Ihnen das Referenzobjekt ist.

Eine weitere Alternative wäre, die Aussage „alle in der Nähe“ mit „alle Nachbarn“ gleichzusetzen. Diese Vorgehensweise hat aber auch keine natürlichsprachliche Berechtigung.

Implementiert wurde eine von der Objektgröße abhängige Intervallangabe. Hierbei wird ein Intervall angegeben, welches einen Ring um das Zentrum des Referenzobjektes bildet. Die Parameter für diesen Ring wurden experimentell ermittelt. Sei W_{max} die maximale Ausdehnung des Referenzobjektes, so betragen die Werte für die Intervallgrenzen $I_{min} = 0.75 \cdot W_{max}$ $I_{max} = 1.25 \cdot W_{max}$. Abbildung 5.1 soll die einzelnen Größen veranschaulichen.

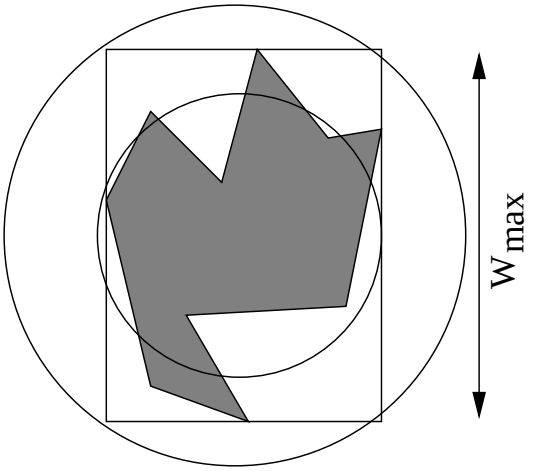

Abbildung 5.1: Ermitteln naher Objekte. Die maximale Kantenlänge des umschließenden Rechteckes gibt die maximale Ausdehnung W_{max}. Der innere Kreis entspricht I_{min}, der Äußere I_{max}.

Die objektabhängige Intervallberechnung ist hinsichtlich ihrers Speicherbedarfs und Laufzeit mit der festen Intervallangabe vergleichbar:

```
; cpu time (non-gc) 1,470 msec user, 50 msec system
; cpu time (gc)     120 msec user, 0 msec system
; cpu time (total)  1,590 msec user, 50 msec system
; real time  1,722 msec
; space allocation:
 93,788 cons cells,;  97 symbols, 626,632 other bytes
```

Hinsichtlich der Ergebnisse ist diese Vorgehensweise allerdings differenzierter. Anzahl der Ergebnisse: 12

```
FLUSS: RIO_SODEKOSO_12_4
KAP: CAP_DELTA_12_4
KAP: MUO_RUODO_12_4
LANDGEBIET: ANGRA_E_RIO_DE_FERNANTE_12_4
LANDGEBIET: GOLFO_DAS_ALMADIAS_12_4
LANDGEBIET: GOLFO_DA_SAN_MARTIN_12_3
LANDGEBIET: PANTA_FORMOSA_12_3
LANDGEBIET: PONTA_BIANCO_12_3
LANDGEBIET: PONTA_DE_MICH_12_4
LANDGEBIET: PONTA_FORMOSA_12_3
STROEMUNG: 1_12_3
WUESTE: DESERTA_NA_12_3
```

Kapitel 6

Zusammenfassung und Ausblick

In dieser Arbeit wurde ein System von Operatoren vorgestellt, mit dem es möglich ist Algorithmen bereitzustellen, die Anfragen qualitativer Natur bearbeiten und beantworten können. Diese Bearbeitung geschieht unter Zuhilfenahme von vorliegenden quantitativen und/oder qualitativen Daten.

Als größtes Hindernis konnte das Problem der Zuordnung natürlichsprachlicher Präpositionen zu qualitativen Sachverhalten in einem 2D-Szenario identifiziert werden. Diese Problematik bezieht sich auf topologische, mengentheoretische und metrische Anfragetypen.

Desweiteren wurde eine Komponente zum Räumlichen Schließen entwickelt, welche Anfragen nach Richtung und Entfernung von Objekten effizient beantworten kann.

6.1 Effizienzsteigernde Maßnahmen

Die vorliegende Inferenzkomponente wurde in Hinblick auf größtmögliche Flexibilität und geringen Speicherbedarf entworfen. Es gibt einige Möglichkeiten, um die Ausführungsgeschwindigkeit auf Kosten der beiden anderen Parameter beträchtlich zu erhöhen.

Sämtliche zum räumlichen Schließen notwendigen Schritte werden bei jedem Durchlauf neu berechnet. Dies ermöglicht zwar den Einsatz in einer nichtstatischen Datenbasis, wie etwa bei kontinuierlich aufgenommenen Satellitenbildern, geht jedoch zu Lasten der Performanz.

Folgende Maßnahmen können die Inferenzkomponente beschleunigen:

neue Datenfelder

Einige Zwischenergebnisse werden stets berechnet. Bei einer Datenbasis wie sie im Behaim-Projekt vorliegt, ist dies jedoch nicht notwendig, da kaum davon auszugehen ist, daß die vorhandenen Objekte ihre Position verändern werden. So könnte der bislang stets ermittelte Schwerpunkt fest abgespeichert werden.

Das umgebende Rechteck einer Figur sowie ihre konvexe Hülle sind ebenfalls Kandidaten für derartige Optimierungen. Beide Größen zu ermitteln

erfordert linearen Laufzeitbedarf hinsichtlich der Anzahl der Punkte pro Objekt. Geht man von k Punkten pro Figur und n Objekten aus, so ist der Gesamt-Laufzeitbedarf um z.B. alle Objekte zu ermitteln, deren Zentrum sich innerhalb eines umschließenden Rechtecks eines Referenzobjektes befindet, $\mathcal{O}(k \cdot n)$. Bei Vorhandensein der Schwerpunktdaten sowie der zu den Objekten zugehörigen Umrandungen reduziert sich dies auf $\mathcal{O}(n)$.

vereinheitlichte Daten

Die Komponente wurde so konzipiert, daß einige formale Abweichungen in der Repräsentation der Objekte toleriert werden. So ist es nicht wichtig, mit welcher Orientierung ein Polygonzug eingegeben wurde. Die Orientierung kann ermittelt und den von dieser Information abhängigen Algorithmen mitgeteilt werden.

Die Sonderstellung diverser länglicher Objekte mit geringer Breite könnte beibehalten werden. Während der Initialisierungsphase könnten diese Daten jedoch in das allgemeinere Polygonformat umgerechnet werden.

Restrukturierung der Datensätze

Eine erhebliche Steigerung der Geschwindigkeit der Inferenzkomponente würde die Strukturierung der Daten auf einem höheren Abstraktionsniveau bedeuten. In einer statischen Datenbasis sind z.B. die Eigenschaften „Nachbar von“ und „Enthält“ ebenfalls nur einmal zu berechnen. Die gegenwärtigen Slots einer `[n]g-region` enthalten manuell erfaßte und nicht vollständige Daten bezüglich dieser Eigenschaften (vgl. Definition von g-region in Abschnitt 4.1). Wünschenswert wäre es, wenn die von der Inferenzkomponente ermittelten Daten einmalig ausgewertet in diese Felder eingetragen würden. Gleiches ist für die möglichen topologischen Relationen denkbar. Hier würde - mit Ausnahme von „disjunkt“ - pro Relation eine Liste der Objekte stehen, welche mit der jeweiligen Instanz in eben jener Relation stehen.

6.2 Algorithmenbibliothek für 2D-Probleme

Die in dieser Arbeit beschriebenen Verfahren stützen sich auf einige wenige geometrische Hilfsmittel wie Schwerpunktbestimmung, umschließendes Rechteck, Steigung einer Geraden und Entfernung zweier Punkte. Obwohl man mit diesen Methoden bereits zahlreiche Problemstellungen im 2D-Raum lösen oder approximieren kann, wären weitere Verfahren wünschenswert, um genauere Ergebnisse zu erhalten.

Eine 2D-Geometrie Bibliothek sollte daher folgende Funktionalität bieten:

- Steigung einer Geraden
- Entfernung zweier Punkte
- Vergleich zweier Polygone
- Orientierung eines Polygons
- Mittelpunkt/Schwerpunkt eines Polygons
- Umschließendes Rechteck eines Polygons

- Konvexe Hülle eines Polygons
- Fläche eines Polygons
- Durchmesser eines Polygons in Abhängigkeit der Richtung von der aus auf das Polygon gesehen wird
- Schnitt zweier beliebiger (auch konkaver) Polygone

Die C++ Bibliothek Leda bietet effiziente Algorithmen zur Lösung des Schnittes zweier beliebiger Polygone. Es gibt darüberhinaus eine Scheme-Implemetierugn des Sutherland-Hodgeman Algorithmus zum Schnitt/Clipping eines beliebigen Polygons an einem konvexen Polygon.

Eine frei verfügbare 2D-Bibliothek, mit mindestens dem oben genannten Funktionsumfang, ist allerdings nicht bekannt. Eine solche Bibliothek ist jedoch bei quantitativ basierten Daten, unabdingbare Voraussetzung für eine vollständige Implementierung von Algorithmen zum Räumlichen Schließen.

6.3 Ein alternatives Kartenmodell

Die bisherigen Lösungsansätze im 2D-Raum behandeln Daten, welche in einer Ebene repräsentiert sind. Dabei geht man entweder von einer sich unendlich ausbreitenden Ebene aus oder von einer endlichen Ebene mit klar definierten Begrenzungen. Wird ersteres Modell vor allem in topologischen Problemen der Mathematik betrachtet, so finden endliche Ebenen zumeist in praktischen Aufgabenstellungen der Geographie Verwendung. Sie werden insbesondere dann eingesetzt, wenn sich ein zu inferierender Sachverhalt vollständig innerhalb des dargestellten Gebietes befindet (vgl. Abb. 6.1 rechts).

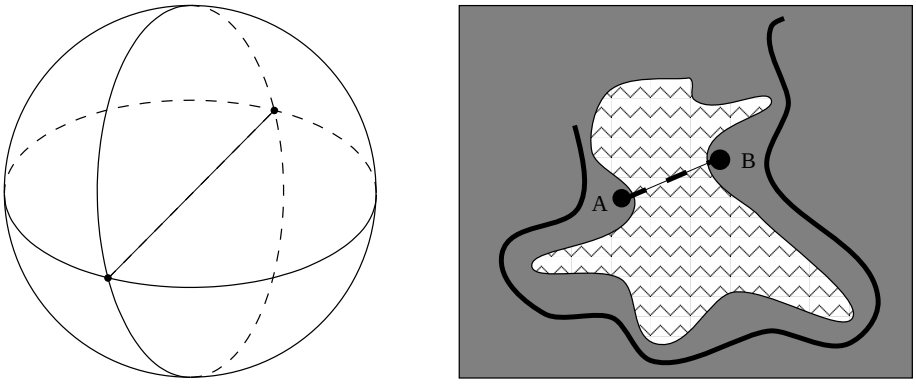

Abbildung 6.1: Zwei unterschiedliche Kartenmodelle: links das Kugelmodell, rechts eine endliche Fläche

Eine Alternative zu dem im Behaim-Projekt verwendeten 2D-Kartenmodell wäre ein Kugelmodell. Die Position eines Punktes auf dieser Kugel wird im folgenden stets durch seinen Längen- und Breitengrad angegeben. Der Hauptunterschied zu den konventienellen Kartenmodellen besteht in der Verwendung zyklischer Koordinaten und zweier singulärer Punkte/Pole. Hierdurch ergeben sich etwas andere Eigenschaften, welche von der Inferenzkomponente mitberücksichtigt werden müssen.

- Zwei Objekte auf der Kugeloberfläche, die sich auf unterschiedlichen Längengraden, jedoch auf dem selben Breitengrad befinden, sind hinsichtlich ihrer Ost/West-Beziehung dual.

- Transitivität ist nur noch eingeschränkt gültig: Aus C *westlich von* B und B *westlich von* A folgt nicht stets C *westlich von* A.

- Zwischen zwei Punkten auf der Kugeloberfläche kann es zwei gleichberechtigte Verbindungen geben.

- Aus Punkt 3 folgt insbesondere, daß ein Polygon nicht einfach durch Angabe seiner Randpunkte definiert werden kann, wie dies in der Ebene der Fall ist. Vielmehr muß zusätzlich die Richtung jeder Kante mit angegeben werden.

- In der Koordinatenrepräsentation auf einer Kugel stellen die Pole zwei singuläre Punkte dar. Diese sind hinsichtlich ihrer Koordinatendarstellung nicht eindeutig $((90\ 0) \equiv (90\ [-180, 180]))$.

6.3.1 Die Navigation auf einer Kugeloberfläche

In vielen Anwendungen der Astronomie und Navigation bedient man sich zur Standortbestimmung der Angabe von Längen- und Breitengrad eines Punktes auf der Kugeloberfläche. Als Basis für die folgenden Definitionen soll die Erdkugel dienen.

Position:

Der Breitengrad eines Punktes ist der Winkel zwischen diesem Punkt und dem Äquator. Der Wertebereich erstreckt sich von 0° bis 90° für jede Hemisphäre. Zur weiteren Bestimmung wird noch angegeben, ob es sich bei der Breitenangabe um die nördliche oder südliche Erdhalbkugel handelt: $35°N$ bedeutet also „35 Grad nördlicher Breite“.

Der Längengrad bezeichnet den Winkel zwischen zwei Großkreisen (siehe 3.6.1), die beide durch Nord- und Südpol gehen. Einer dieser beiden Längengrade (der nullte Längengrad) geht zusätzlich durch das Observatorium in Greenwich, England, und wird als Referenz bei der Längengradangabe verwendet. Der Wertebereich erstreckt sich von 0° bis 180° jeweils östlich und westlich des nullten Längengrades. Zur weiteren Bestimmung wird daher zwischen östlicher und westlicher Längenangabe unterschieden: $135°E$ bedeutet „135 Grad östlicher Länge“.

Abstand:

Um den Abstand zweier Punkte auf einer Kugeloberfläche definieren zu können, benötigt man zunächst die Definition eines Großkreises:

Ein *Großkreis* entsteht durch den Schnitt einer (Hohl-)Kugel mit einer Ebene, welche durch den Mittelpunkt der Kugel verläuft. Ein Großkreis hat somit stets maximalen Umfang. Die Längengrade und der Äquator sind Beispiele für Großkreise.

Gegeben seien also zwei Punkte A und B welche sich auf der Oberfläche einer Kugel befinden. Sei C der Mittelpunkt dieser Kugel. Die Entfernung

zwischen A und B ist die Länge des Bogens $\widehat{AB}$, welche bestimmt wird durch den Winkel $\angle ACB$ und den Radius der Kugel.

Das Ziel ist es nun, den Abstand zweier Punkte zu ermitteln ohne Transformationen in Raumkoordinaten vornehmen zu müssen, da die Koordinaten bereits in der Form <geogr. Breite, geogr. Länge> verfügbar sind. Siehe auch Abschnitt 4.1 (Datenstruktur) und A.1 (Implementierung). Jennings [Jen94] gibt in Kapitel 2 eine gute Zusammenfassung der Anwendungen sphärischer Geometrie. Die Implementierung basiert auf den dort vorgestellten Methoden.

Richtung:

Die Richtung auf einer Kugeloberfläche wird in der Form $Xyy°Z$ angegeben, wobei X die Referenzrichtung (N oder S) angibt, $yy°$ die Abweichung und Z die Orientierung (westlich oder östlich). $S35°W$ bedeutet „35 Grad westlich in südlicher Richtung".

6.4 Bestehende Probleme und mögliche Erweiterungen

6.4.1 Objekte über Segmentgrenzen

Einige der in den Segmentabbildungen in Anhang B befindlichen Objekte verlaufen über die Grenzen der jeweiligen Segmente hinweg. Das betreffende Objekt wird als „nicht geschlossen" aufgefaßt.

Da es sich häufig um große Objekte handelt, entstehen so Ringe anstelle von Flächen. Würde man die offenen Endpunkte entlang der Segmentgrenze einfach miteinander verbinden, hätte man die Flächen abgeschnittener Objekte. Angaben zum Mittelpunkt und zu den Ausmaßen eines Objektes würden verfälscht.

Das Problem stellt sich, da die Applikation *globus_koord*, die zum Erfassen der Objektkoordinaten dient, segmentorientiert arbeitet, d.h. nur genau ein Segment gleichzeitig darstellen kann. Auch in der zugrundeliegenden Datenbasis werden derart übergreifende Objekte dem Segment zugeordnet, in dem ihr Mittelpunkt liegt.

Es besteht nun entweder die Möglichkeit diese Objekte manuell zu korrigieren oder *globus_koord* derart zu modifizieren, daß anstelle einer Segmentnummer ein Zentrum sowie die Expansion des zu betrachtenden Ausschnittes angegeben wird. Letzteres ist zwar die aufwendigere, sicherlich aber auch die elegantere Lösung.

6.4.2 Planare Erfüllbarkeit

Wie in Abschnitt 2.1.4 erwähnt, haben Grigni et al. [GPP95] das Problem der Erfüllbarkeit einer, durch Angabe von topologischen Relationen definierten Szenerie in der Ebene herausgestellt.

Das Problem besteht im wesentlichen darin, daß bei einer gegebenen Menge von Objekten die zwischen diesen Objekten vorliegenden Relationen durchaus konsistent sein können, jedoch kein planares Modell besitzen. Angenommen man hat 14 Objekte wie in Bild 6.2 angeordnet. Die Objekte X_1 bis X_5 werden durch die Objekte Y_1 bis Y_9 verbunden (überlappen). Hat man die Forderung, daß die

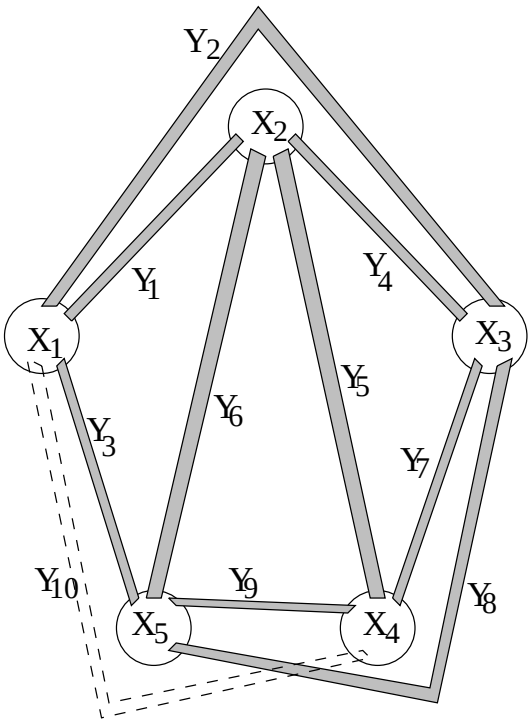

Abbildung 6.2: Beispiel eines Falles, in dem zwar relationale Konsistenz, jedoch nicht planare Erfüllbarkeit vorliegt.

verbindenen Objekte außer zu den durch sie Verbundenen jeweils zwei Regionen zu allen anderen Objekten disjunkt sind, so hat man sowohl relationale Konsistenz als auch ein planares Modell. Fügt man dieser Menge noch ein Objekt Y_{10} hinzu welches X_1 und X_4 verbinden soll und ebenfalls zu allen sonstigen Objekten diskunkt ist, so hat man zwar noch relationale Konsistenz aber kein planares Modell mehr. Y_{10} muß zwangsläufig mindestens eines der anderen Objekte schneiden.

Da dieses Subproblem als NP-vollständig erkannt worden ist, werden wohl nur Heuristiken mit einer approximativen Lösung eine akzeptable Laufzeitkomplexität liefern können.

6.4.3 Undefinierte Begrenzungen

Eine Besonderheit bei den im Behaim-Projekt vorhandenen Daten stellen Objekte mit nicht klar definierten Begrenzungen dar. Dabei handelt es sich meist um Gebiete, welche - zwar namentlich erwähnt - gekennzeichnete Begrenzungen vermissen lassen. Das gegenwärtige Objektkonzept geht mit dem Polygonzug von klaren Trennlinien zwischen den Objekten aus. Bei diffusen Grenzen stellt sich das Problem, daß bei zwei aneinandergrenzenden Objekten nicht einmal der Status „schneidet“ oder „berührt“ eindeutig bestimmt werden kann. Ebenso problematisch wird die Anwendung eines etwaigen | |-Operators zur Bestimmung der Ausmaße eines solchen Objektes.

Man könnte diesem Problem begegnen, indem für ein solches Objekt zwei

Polygone angegeben werden. Das innere Polygon kennzeichnet dann die minimale, das äußere die maximale Ausdehnung. Eine Gewichtungsfunktion zwischen zwei korrespondierenden Punkten der beiden Polygone bewirkt ein Gefälle das der Wahrscheinlichkeit entspricht, mit der an dem entsprechenden Punkt das Objekt noch anzutreffen ist. Bild 6.3 soll den Sachverhalt veranschaulichen.

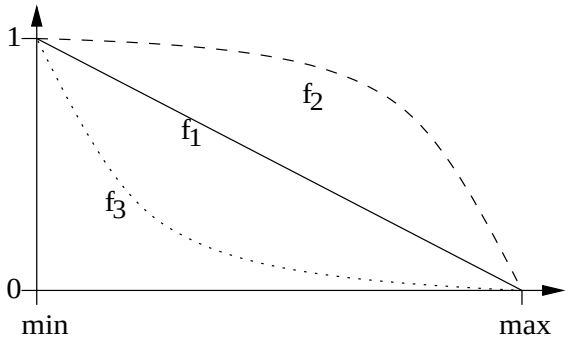

Abbildung 6.3: Beispiele verschiedener Gewichtungen für den Grenzbereich diffuser Objekte. f_1: linearer Abstieg - einfache Interpolation; f_2: progressiver Abstieg - Objekt mit „dichter Hülle“; f_3: regressiver Abstieg - Objekt mit „dünner Hülle“

Der mit dieser Betrachtungsweise verbundene erhöhte Rechenzeitaufwand muß allerdings noch ermittelt werden. Dieses Konzept der Darstellung einer Objektgrenze ist zwar allgemeiner jedoch nicht mehr eindeutig. Es gibt mindestens drei Möglichkeiten den speziellen Fall einer genauen Objektgrenze, wie er jetzt durch ein Polygon dargestellt wird, zu verwirklichen:

1. Inneres und äußeres Polygon fallen zusammen. Unabhängig von der verwendeten Gewichtungsfunktion tritt ein $1 \rightarrow 0$ Übergang auf. (Eigenschaft des äußeren Polygons). Mit anderen Worten: Die Gewichtungsfunktion wird ignoriert.

2. Inneres Polygon wird ignoriert, die Gewichtungsfunktion ist $f(x) = 1$.

3. Äußeres Polygon wird ignoriert, die Gewichtungsfunktion ist $f(x) = 0$.

Dies sollte bei einer evtl. Implementierung bedacht und konsistent behandelt werden.

Anhang A

Quellcodereferenzen

A.1 Abstand zweier Punkte auf der Kugeloberfläche

```
;; Globale Variable, die den Kugelradius bestimmt
(setq *kugelradius* 6367.445)


;; Umwandlung Grad -> Radian
(defun deg2rad (deg)
  (/ (* deg PI) 180)
  )

;; Umwandlung Radian -> Grad
(defun rad2deg (rad)
  (* (/ 180 PI) rad)
  )

;;
;; Ermittelt die Differenz zwischen den Längengraden der angegebenen
;; Punkte und liefert das Ergebnis in Radian zurück
;;
(defun rad_dl (p1 p2)
  (setq dl (abs (- (cadr p1) (cadr p2))))
  (if (> dl 180) (setq dl (- 360 dl)))
  (deg2rad dl)
  )

;;
;; Ermittelt die Differenz zwischen den Breitengraden der angegebenen
;; Punkte und liefert das Ergebnis in Radian zurück
;;
(defun rad_db (p1 p2)
  (deg2rad (abs (- (car p1) (car p2))))
  )

;;
;; Behaim to Norm: konvertiert eine Koordinatenangabe in den
```

```
;; Behaim-Daten in die geographische Normalform:
;; Behaim: (Breite Länge), wobei Breite [-90,90] und Länge [0 360]
;; Normal: (Breite Länge), wobei Breite [-90,90] und Länge [-180,180]
;;
;; negative Breitenangabe ist südliche Breite
;; negative Längenangabe ist westliche Länge
;;
(defun beh2nrm (p1)
  (if (> (cadr p1) 180) (list (car p1) (- (cadr p1) 360)) p1)
)

;;
;; Test, ob sich zwei Punkte auf der Kugel diametral gegenüberliegen
;;
(defun is_diametral (p1 p2)
  (cond ((= (+ (abs (car p1)) (abs (car p2))) 180) T)
        ((and (= (+ (car p1) (car p2)) 0)
              (= (+ (abs (cadr p1)) (abs (cadr p2))) 180)) T)
        (T NIL))
  )

;;
;; Test, ob es sich bei dem angegebenen Punkt um einen Pol handelt
;;
(defun is_pol (p1)
  (if (= (abs (car p1)) 90) T NIL)
  )

;;
;; Ermittelt den Abstand der angegebenen Punkte auf einer Kugel.
;; Das Ergebnis ist abhängig von der Variable *kugelradius*
;;
(defun kug_abstand (p1 p2)
  (cond ((is_diametral p1 p2) (setq abstand (* *kugelradius* PI)))
        ((is_pol p1) (setq abstand
                     (* *kugelradius*
                        (deg2rad (abs (- (car p1) (car p2)))))))
        ((is_pol p2) (setq abstand
                     (* *kugelradius*
                        (deg2rad (abs (- (car p2) (car p1)))))))
        (T (setq db1 (deg2rad (- 90 (car p1))))
           (setq db2 (deg2rad (- 90 (car p2))))
           (setq winkel (acos (+ (* (cos db1) (cos db2))
                                 (* (sin db1) (sin db2)
                                    (cos (rad_dl p1 p2))))))
           (setq abstand (* winkel *kugelradius*))))
  )
```

A.2 Abstand zweier Punkte

```
;;
;; ### Entfernung zweier Punkte
;;
```

```
(defun get-distance (two)
  (defun sqr (x) (* x x))
  (sqrt (+ (sqr (- (caar two) (caadr two)))
           (sqr (- (cadar two) (cadadr two)))
           ))
  )
```

A.3 Richtung einer Strecke

```
;;
;; ### Richtung einer Strecke bestimmen (gegeben Anfangs- & Endpunkt)
;;
(defun get-direction (two)
  (setq d (get-distance two))
  (defun deg (x) (round (/ (* x 180) PI)))
  (setq cost (deg (acos (/ (- (caadr two) (caar two))   d))))
  (setq sint (deg (asin (/ (- (cadadr two) (cadar two)) d))))


  (cond ((= (signum sint) -1) (- 360 cost))
        (T cost))
  )
```

A.4 Orientierung eines Polygons

Für viele Anwendungen, in denen Polygone zum Einsatz kommen, ist die Kenntnis über ihre Orientierung wichtig. Folgende Definition ist [BK95] entnommen:

> Ein Polygon Π heißt **positiv orientiert**, wenn beim Durchlaufen die Punkte von Π „links“ liegen (mathematisch positiver „Drehsinn“).

Hierzu ermittelt man einen geeigneten Extrempunkt. Im folgenden Algorithmus ist dies derjenige mit der kleinsten X-Koordinate. Sollte es davon mehrere geben, nimmt man aus dieser Menge den mit der größten Y-Koordinate. Anschließend vergleicht man die Steigung der Strecke zwischen diesem Punkt und seinem Vorgänger $\overline{S1}$, sowie die Steigung der Strecke zwischen ihm und seinem Nachfolger $\overline{S2}$.

Die Orientierung erfolgt im Uhrzeigersinn, wenn die Steigung von $\overline{S1}$ kleiner ist als die von $\overline{S2}$. Andernfalls ist das Polygon **negativ orientiert**. Abbildung A.1 zeigt den durch diese drei Punkte definierten relevanten Ausschnitt.

```
;;
;; ### Test, welche Orientierung Polygon hat
;;     Rückgabe: T   = Im Uhrzeigersin
;;               NIL = gegen Uhrzeigersinn
;;
(defun test-orientation (cls_list)
  (setf pm (get-xminymax cls_list))     ; Punkt in der Mitte
  (setf pv (get-pred cls_list pm))      ; Vorgänger
  (setf pn (get-succ cls_list pm))      ; Nachfolger

  (if (< (get-gradient (list pm pv)) (get-gradient (list pm pn)))
```

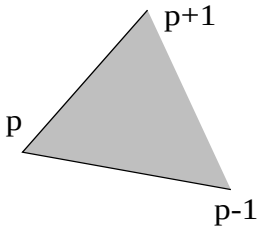

Abbildung A.1: Ermitteln der Orientierung eines Polygons. Das graue Gebiet kennzeichnet das Innere des Polygons.

```
      T NIL)
  )

;;
;; ### Minimalen X-Extrempunkt mit maximaler Y-Koordinate liefern
;;     (Hilfsfunktion zum Ermitteln der Orientierung eines Ploygons)
;;
(defun get-xminymax (any_list)
  (if (test-open any_list) (setf any_list (close-poly any_list)))

  (setf minx (caar any_list))
  (setf maxy 0)                          ; Never negative
  (dolist (var (cdr any_list) (list minx maxy))

          (cond ((> minx (car var)) (setq minx (car var))
                                    (setq maxy (cadr var)))
                ((= minx (car var)) (if (< maxy (cadr var))
                                        (setf maxy (cadr var))))
                )
          )
  )

;;
;; ### Vorgänger-Punkt innerhalb einer geschlossenen Liste zu einem
;;     gegebenen Punkt finden. Der angegebene Punkt MUSS sich in der
;;     Liste befinden.
;;
(defun get-pred (cls_list point)
                                  ; Falls erster Punkt der gegebene:
                                  ; Vorletzter Eintrag ist Vorgänger
  (cond ((equal point (first cls_list))
         (nth (- (length cls_list) 2) cls_list))
        ((equal point (second cls_list)) (first cls_list))
                                  ; Ansonsten: Liste durchlaufen und
                                  ; bei Übereinstimmung vorangehenden
                                  ; Punkt liefern (rekursiv)
        (T (get-pred (cdr cls_list) point))
        )
  )

;;
```

```
;; ### Nachfolger-Punkt innerhalb einer geschlossenen Liste zu einem
;;     gegebenen Punkt finden. Der angegebene Punkt MUSS sich in der
;;     Liste befinden.
;;
(defun get-succ (cls_list point)
  (cond ((equal point (first cls_list)) (cadr cls_list))
        (T (get-succ (cdr cls_list) point))
        )
)
```

A.5 Umschließendes Rechteck bestimmen

Wie in Abschnitt 3.8.4 gezeigt wurde, ist die Approximation eines Polygonzuges durch eine einfache geometrische Figur - zum Beispiel also ein Rechteck - hinsichtlich der Aussagekraft von Schnittbestimmungen ähnlich aussagekräftig wie der Schnitt zweier konvexer Polygone. Da die Laufzeitkompexität des Schnittes zweier Rechtecke jedoch konstant ist, bietet sich dieses Verfahren als primäre Methode der Klassifikation von Relationen an.

```
(setq min_xpixel 0) (setq max_xpixel 99999)
(setq min_ypixel 0) (setq max_ypixel 99999)

;;
;; ### umschließendes Rechteck liefern (ist eine cls_list)
;;
(defun bounding-rectangle (cls_list)
  (setq minx max_xpixel)
  (setq maxx min_xpixel)
  (setq miny max_ypixel)
  (setq maxy min_ypixel)
  (dolist (var (cdr cls_list) (list (list minx miny) (list maxx miny)
                                    (list maxx maxy) (list minx maxy)
                                    (list minx miny)))

    (cond ((> minx (car var)) (setq minx (car var))))
    (cond ((< maxx (car var)) (setq maxx (car var))))
    (cond ((> miny (cadr var)) (setq miny (cadr var))))
    (cond ((< maxy (cadr var)) (setq maxy (cadr var))))
    )
  )
```

A.6 Schwerpunkt eines Polygons

Durch die Unzulässigkeit kolinearer Punkte in einem Polygonzug kann der Schwerpunkt eines Polygons ermittelt werden, indem man den Schwerpunkt der Eckpunkte ermittelt.

```
;;
;; ### Mittelpunkt/Schwerpunkt liefern
;;
(defun get-midpoint (cls_list)
  (setq len (length (cdr cls_list)))
```

```
(setq x 0)
(setq y 0)
(dolist (var (cdr cls_list) (cons (/ x len) (/ y len)))
  (setq x (+ x (car var)))
  (setq y (+ y (cadr var)))
  )
)
```

Anhang B

Segmentdaten

Die folgenden Abbildungen zeigen eine Übersicht der bereits eingegebenen Datensätze für verschiedene Segmente des Behaim-Globus. Erfaßt wurden die Segmente 10, 12, 14, 16 und 18. Wie man sich anhand Abbildung 3.4 orientieren kann, umfaßt dies die südliche Halbkugel zwischen dem neunzigsten und 240ten Längengrad. Hauptaugenmerk liegt also auf Afrika und den umgebenden Objekten.

Die hier gezeigten Daten wurden direkt aus den in der Datenbasis vorhandenen Rohdaten gewonnen und in das Zeichenprogramm *xfig* eingesetzt. Die Originaldaten z.B. von der Instanz `schiff_12_3` im Slot PIXEL-POLYGON sehen wie folgt aus:

```
:PIXEL-POLYGON    (list '(11192 5453) '(11147 5434) '(11137 5467)
'(11173 5486) '(11169 5494) '(11156 5491) '(11156 5510) '(11159 5518)
'(11154 5533) '(11088 5485) '(11083 5493) '(11096 5505) '(11056 5547)
'(11046 5591) '(11033 5587) '(11025 5603) '(11029 5608) '(11018 5643)
'(10946 5623) '(10928 5643) '(10943 5671) '(10946 5696) '(10992 5713)
'(11009 5734) '(11024 5735) '(11035 5754) '(11048 5760) '(11124 5765)
'(11148 5761) '(11182 5777) '(11204 5757) '(11149 5722) '(11180 5700)
'(11190 5671) '(11176 5672) '(11174 5632) '(11213 5599) '(11234 5607)
'(11238 5596) '(11166 5540) '(11172 5526) '(11192 5535) '(11197 5512)
'(11178 5498) '(11192 5453) )
```

Hiervon werden einfach alle Klammern, Quotes und Schlüsselwörter entfernt. Das xfig-Format benötigt noch einige Daten über das Aussehen des so erzeugten Polyline (Linienzug). Diese sind das umschließende Rechteck, Informationen über Linientyp und die Anzahl der vorhandenen Linien - in diesem Fall 45.

```
6 450 675 2655 3155
2 1 0 1 -1 7 0 0 -1 0.000 0 0 -1 0 0 45
       2300 1000 2024 883 1963 1085 2183 1202 2159 1251 2079 1232
       2079 1349 2098 1398 2067 1489 1663 1195 1632 1244 1712 1318
       1467 1575 1406 1845 1326 1820 1277 1918 1301 1949 1234 2163
       793 2041 683 2163 775 2334 793 2488 1075 2592 1179 2720
       1271 2726 1338 2843 1418 2879 1883 2910 2030 2886 2239 2984
       2373 2861 2036 2647 2226 2512 2288 2334 2202 2341 2190 2096
       2428 1894 2557 1943 2582 1875 2141 1532 2177 1447 2300 1502
       2330 1361 2214 1275 2300 1000
-6
```

Die Figur muß noch skaliert und verschoben werden. Beides kann jedoch bereits in xfig selbst geschehen. Das Resultat ist in Abbildung B.1 zu sehen. Die vorliegenden Daten wurden auf diese halbautomatische Weise gewonnen. Ein einfaches Shellskript unter Verwendung von sed und awk[1] könnte diese Transformation automatisieren. Das Programm *transfig*[2] kann die so gewonnenen, im .fig-Format vorliegenden Daten z.B. in Postscript umwandeln.

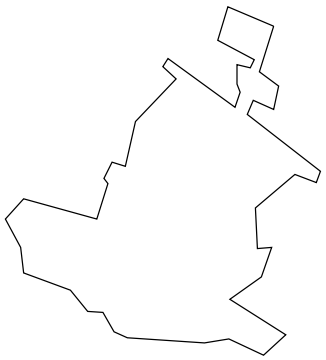

Abbildung B.1: Instanz schiff_12_3 aus Segment 12.

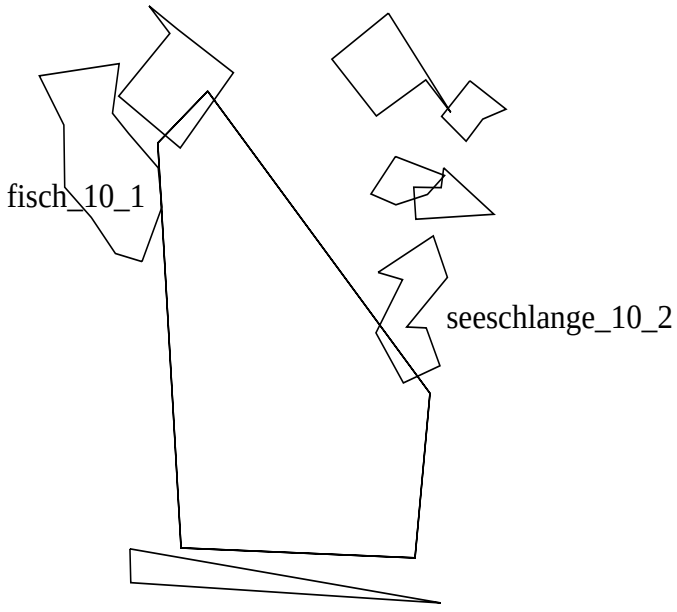

Abbildung B.2: Erfaßte Objekte in Segment 10.

[1] oder ein Perlskript
[2] gehört zu xfig.

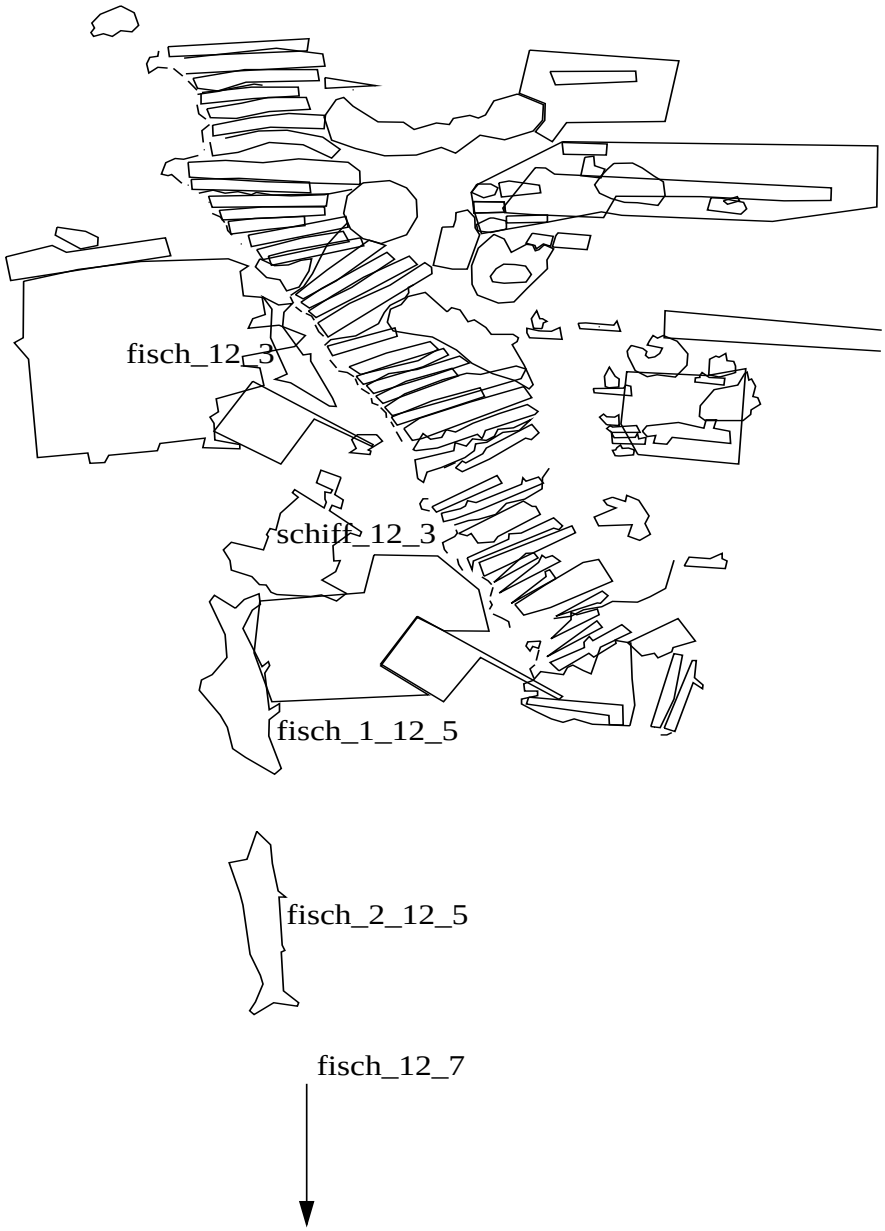

Abbildung B.3: Objekte entlang der Westküste Afrikas in Segment 12.

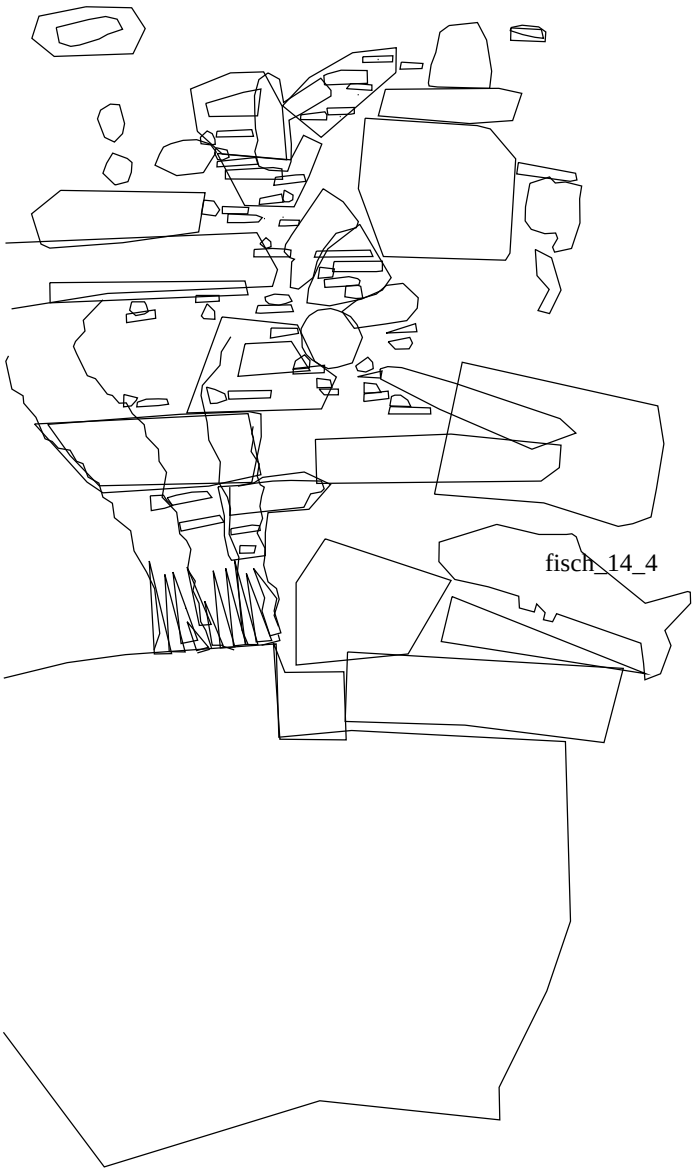

Abbildung B.4: Erfaßte Objekte in Segment 14.

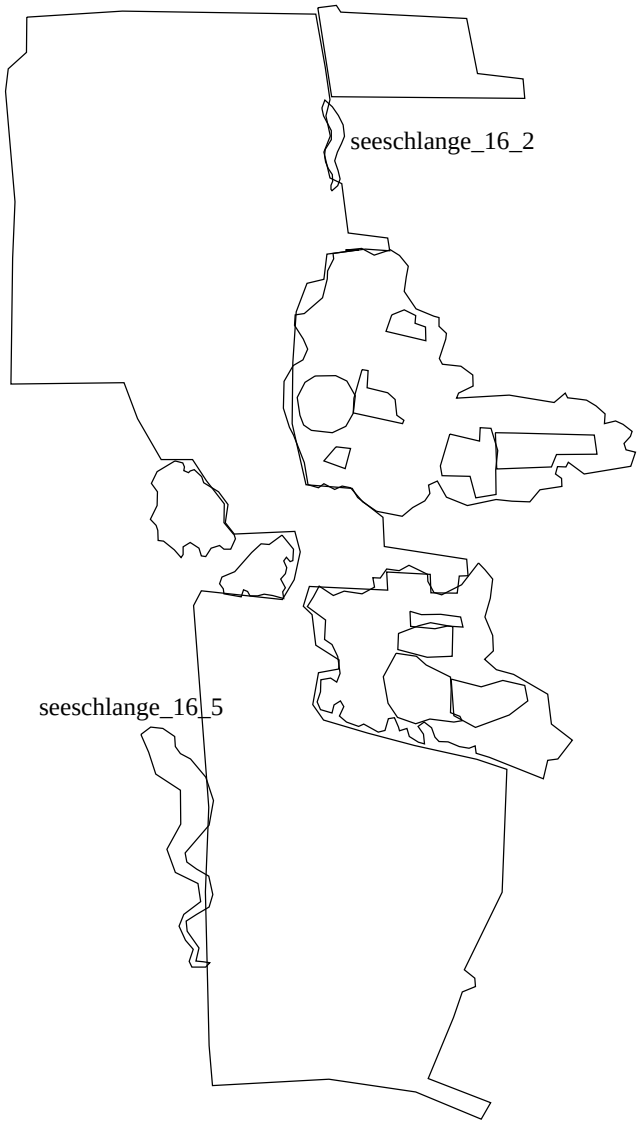

Abbildung B.5: Erfaßte Objekte in Segment 16.

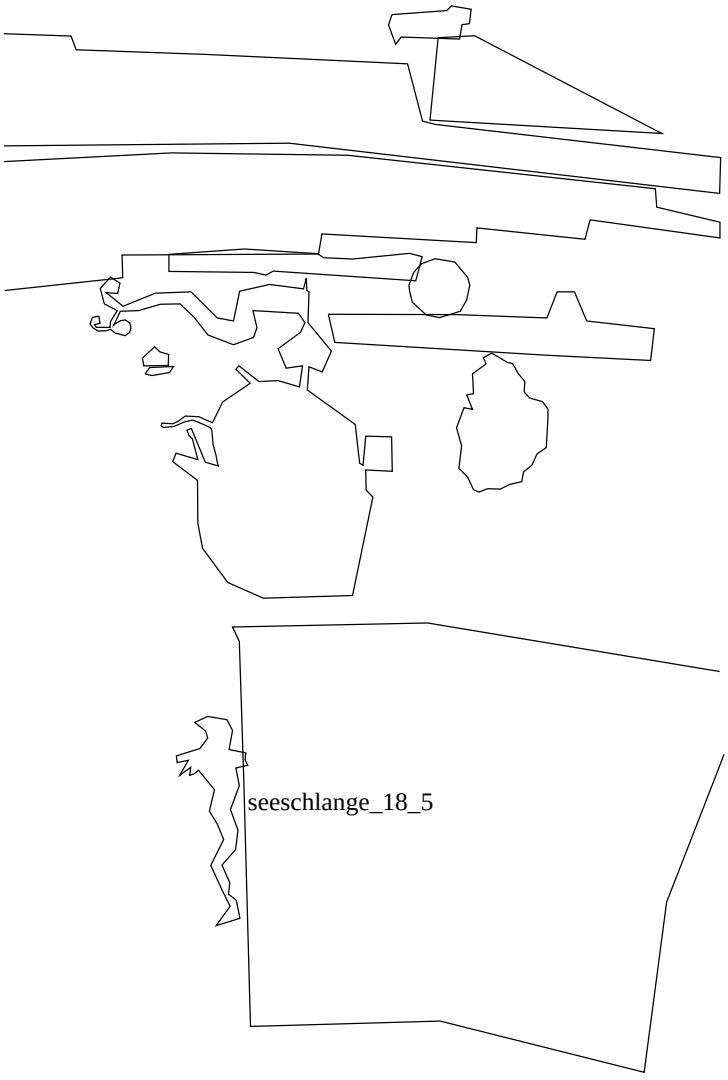

Abbildung B.6: Erfaßte Objekte in Segment 18.

Literaturverzeichnis

[AS93] Günter Aumann and Klaus Spitzmüller. *Computerorientierte Geometrie*. BI Wissenschaftsverlag, Mannheim, Leipzig, Wien, Zürich, 1993.

[BK95] Erhart Bohne and Wolf-Dieter Klix. *Geometrie - Grundlagen für Anwendungen*. Fachbuchverlag Leipzig, Leipzig, Köln, 1995.

[CJ96] Shi-Kuo Chang and Erland Jungert. *Symbolic Projection for Image Information Retrieval and Spatial Reasoning*. Academic Press, London, 1996.

[DS82] Roger M. Downs and David Stea. *Kognitive Karten*. Harper & Row, New York, 1982.

[EF91] Max J. Egenhofer and Robert D. Franzosa. Point-set topological spatial relations. In *International Journal of Geographical Information Systems*, volume 5(2), pages 161–174, 1991.

[Ege91] M. J. Egenhofer. Reasoning about binary topological relations. In Oliver Guenther and Hans-Joerg Schek, editors, *Proceedings of Advances in Spatial Databases (SSD '91)*, volume 525 of *LNCS*, pages 143–160, Berlin, Germany, August 1991. Springer.

[Euz95] Jérôme Euzenat. An algebraic approach to granularity in qualitative time and space representation. In Chris S. Mellish, editor, *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence*, pages 894–900, San Mateo, August 20– 25 1995. Morgan Kaufmann.

[FMP93] L. De Floriani, P. Marzano, and E. Puppo. Spatial queries and data models. In I. Campari A. U. Frank and U. Formentini, editors, *Information Theory: a Theoretical Basis for GIS*, pages 113–138. Springer-Verlag, Lecture Notes in Computer Science, N.716, 1993.

[Fra92] Andrew U. Frank. Qualitative spatial reasoning about distances and directions in geographic space. *Journal of Visual Languages and Computing*, 3:343–371, 1992.

[Fre92] C. Freksa. Temporal reasoning based on semi-intervals. *Artificial Intelligence*, 54(1-2):199–227, March 1992.

[FvDF+94] J. O. Foley, A. van Dam, S. K. Feiner, J. F. Hughes, and R. L. Phillips. *Grundlagen der Computergraphik*. Addison Wesley, Bonn, 1994.

[GPP95] Michelangelo Grigni, Dimitris Papadias, and Christos Papadimitriou. Topological inference. In *IJCAI*, pages 901–907, 1995.

[Gör95] Günther Görz, editor. *Einführung in die künstliche Intelligenz*. Addison-Wesley, Reading, Massachsetts, 1995.

[Her94] Daniel Hernandez. *Qualitative representation of spatial knowledge*, volume 804 of *Lecture Notes in Artificial Intelligence and Lecture Notes in Computer Science*. Springer-Verlag Inc., New York, NY, USA, 1994. Revision of thesis (Technische Universität München, 1992).

[Jen94] George A. Jennings. *Modern Geometry with Applications*. Springer-Verlag, Berlin Heidelberg, 1994.

[Kee89] Sonya E. Keene. *Object-Oriented Programming in Common Lisp*. Addison-Wesley, Reading, Massachsetts, 1989.

[MS95] Michael Montag and Peter Struß. *Qualitatives und modellbasiertes Schließen*, pages 87–128. In Görz [Gör95], 1995.

[RCC92] David A. Randell, Zhan Cui, and Anthony G. Cohn. A spatial logic based on regions and connection. In B. Nebel, W. Swartout, and C. Rich, editors, *Principles of Knowledge Representation and Reasoning: Proceedings of the 3rd International Conference*, pages 165–176, Cambridge, MA, Oct. 1992. Morgan Kaufmann.

[Ste90] Guy L. Steele, Jr. *Common Lisp: The Language (HTML-Version)*. Digital Press, Woburn, MA 01801, 2 edition, 1990.

Index