Studienarbeit

Konzeption und Implementation von Fuzzy-Prädikaten in digitalen Bibliotheken

von
Thomas Büchner

Betreuer:
Prof. Dr. Florian Matthes
Dipl.-Inform. Ulrike Steffens

Arbeitsbereich Softwaresysteme
Technische Universität Hamburg-Harburg

30. November 1999

Inhaltsverzeichnis

1 Einführung **1**

2 Das Katalogsystem PIA **3**

2.1 Darstellung des Kataloginhaltes 4

2.2 Domänen und Werte 6

2.2.1 Numerische Domänen 6

2.2.2 Diskrete Domänen 6

2.2.3 Volltextdomäne 7

2.3 Formulierung eines Kundeninteresses 7

2.4 Prädikate 8

3 Fuzzy Logic **11**

3.1 Zugehörigkeitsfunktionen 11

3.2 Aggregatoren 13

3.2.1 Konjunktion und Disjunktion 13

3.2.2 t-Norm 14

3.3 Relationen 15

3.4 Implementierte t-Normen 16

4 Diskrete Prädikate **17**

4.1 Betrachtungen über *is similar to* 17

4.2 Art der Beziehung zwischen den Konzepten einer diskreten Domänen 18

4.3 Relationswerte 20

4.4 Auswertung des Prädikates 21

4.5 Bestimmung der Kantenwerte 22

4.5.1 Minimum-Norm (nicht-interaktiv) 22

4.5.2 Verwendung einer interaktiven Norm 23

4.5.3 Algorithmus zur Unterstütznng des Katalogbetreibers bei der Bestimmung der Kantenrelationen 23

4.6 Implementierungsdetails 27

4.7 Mögliche Verbesserungen 28

5 Numerische Prädikate **29**
5.1 Anforderungen . . . 29
5.1.1 Grundbereich . . . 30
5.1.2 Zugehörigkeitsfunktion . . . 31
5.2 Mathematische Grundlagen . . . 32
5.3 Definition des *Cluster*-Begriffes . . . 34
5.4 Repräsentation der Cluster . . . 35
5.5 Anlegen der Clusterinformationen . . . 37
5.6 Auslesen der Clusterinformationen . . . 38
5.7 Arbeitsweise der numerischen Prädikate . . . 40
5.7.1 Klassifikation nach Gewinnung der Stützstellen . . . 41
5.7.2 Klassifikation nach Zugehörigkeitsfunktion . . . 43
5.7.3 Arbeitsweise . . . 43
5.8 Mögliche Verbesserungen . . . 43

6 Zusammenfassung **45**

Abbildungsverzeichnis

2.1 Ausschnitt des Klassendiagramms von PIA 5

3.1 Graph der L-Funktion 12
3.2 Graph der Γ-Funktion 12
3.3 Graph der Trapez-Funktion 13

4.1 Baumstruktur 18
4.2 Diskrete Domäne: Produktkategorie 19
4.3 Kante 20
4.4 Diskrete Domäne mit Kantenwerten 21
4.5 Auswertung von *is similar to* 22
4.6 Algorithmus: automatisch korrigierte Grundrelation 26
4.7 Algorithmus: Veränderungsqualifizierer 26
4.8 Algorithmus: Endresultat 26
4.9 Algorithmus: Beispielauswertung 27

5.1 Cluster 35
5.2 Klassendiagramm Cluster 36
5.3 *EntryCluster.setUp()* 38
5.4 Ablaufplan für *QueryCluster.setNumericAttributes()* 40
5.5 Graph der Trapez-Funktion 41
5.6 *FuzzyGraph* Objekte 43

Kapitel 1

Einführung

Die folgende Arbeit beschäftigt sich mit Teilen des Katalogsystems PIA (Personal Information Assistant), welches zur Einordnung von und zum Zugriff auf Produkt- und Dokumentbeschreibungen verwandt wird. Da Kataloge zumeist ein integraler Bestandteil digitaler Bibliotheken sind, findet PIA insbesondere auch in diesem Gebiet Anwendung. Gegenüber anderen Systemen mit ähnlichen Leistungsmerkmalen zeichnet sich PIA durch ein sehr hohes Maß an Flexibilität aus. Diese Flexibilität äussert sich durch das Fehlen eines starren Schemas zur Produktrepräsentation. Es bleibt dem Katalogbetreiber überlassen, durch welche Kombination von Eigenschaften er ein Produkt repräsentiert. Jede Eigenschaft ist einer Domäne zugeordnet, wodurch eine Typsicherheit gewährleistet ist.

Ein zentrales Element eines jeden Katalogsystems ist die Auswertungseinheit, welche repräsentierte Produkte bezüglich einer vom Kunden gestellten Anfrage bewertet und somit eine Suchfunktion realisiert. Diese Auswertung wird bei PIA mit Hilfe von Prädikaten gewährleistet und steht im Mittelpunkt dieser Arbeit. Die Arbeitsweise dieser Prädikate beruht auf der Theorie unscharfer Mengen, der *Fuzzy Logic*, die in verschiedensten Foschungsgebieten Einzug hält und in PIA die Vagheit der Entsprechung zwischen angebotener und gesuchter Information zum Ausdruck bringt.

Entsprechend den verschiedenen in PIA verwendeten Domänen existieren verschiedene Arten von Prädikaten. Im Rahmen dieser Arbeit werden zwei Typen von Prädikaten behandelt: diskrete Prädikate, die eine Aussage über eine Ähnlichkeit zwischen verschiedenen hierarchisch geordneten diskreten Konzepten treffen, und numerische Prädikate, die Aussagen über einen Grad der Zugehörigkeit eines numerischen Wertes zu einer unscharfen Menge treffen.

Die unscharfen diskreten Prädikate lassen sich im Sinne von Fuzzy Logic formal als unscharfe Relationen auffassen. Da die in Relation zu setzenden Konzepte innerhalb einer diskreten Domäne stets in einer Baumstruktur angeordnet sind, stellt die Auswertung eines diskreten Prädikates ansich keine große Herausforderung dar. Es stellt sich jedoch die Frage nach einer sinnvollen und einfachen Initialisierung dieser Baumstruktur mit geeigneten Gewichtungen. Zu diesem Zweck wird ein Algorithmus vorgestellt, der den Katalogbetreiber

bei dieser Aufgabe unterstützt.

Ein numerisches Prädikat repräsentiert formal eine unscharfe Menge. Aus dieser Aussage ergeben sich die zu berücksichtigenden Anforderungen: die Festlegung eines Grundbereiches und das anschließende Aufstellen einer Zugehörigkeitsfunktion. Durch die dem Katalogbetreiber willkommene Flexibilität des Systems stellt das Auffinden eines der Anfrage entsprechenden Grundbereiches eine anspruchsvolle Aufgabe dar, da kein starres Schema zur Produktrepräsentation vorhanden ist. Ziel dieser Arbeit ist es an dieser Stelle, eine Methode zu entwickeln, die sowohl exakte Ergebnisse im Sinne der höchsten erzielbaren Genauigkeit als auch eine effiziente Auswertung erreicht. Dies wird durch Anlegen von Metainformationen beim Initialisieren des Systems ermöglicht.

Die vorliegende Arbeit beschreibt nicht nur die Konzepte, die den diskreten und numerischen Fuzzy-Prädikaten prinzipiell zugrundeliegen, sondern erläutert ebenfalls ihre objektorientierte Implementation in Java sowie ihre Integration in das prototypische PIA-System.

Kapitel 2

Das Katalogsystem PIA

Eine ständig wachsende Anzahl von Unternehmen und Organisationen benutzt digitale Medien (z.B. Online Web-Server oder Offline CD-ROMs), um mit ihren Kunden in Kontakt zu treten. Es existieren zwei extreme Formen der Kommunikation. Ein Extrem ist die direkte Kommunikation zwischen einem bestimmten Kunden und einem gut geschulten Verkäufer; das andere Extrem ist die anonyme Informationsverbreitung über Massenmedien wie Fernsehen, Zeitschriften, usw. Digitale Kommunikation hat das Potential inne, die Lücke zwischen diesen Extremen zu schließen. Im folgenden wird ein System PIA (Personal Information Assistant) vorgetellt, das in der Rolle eines *aktiven Assistenten* einen Kunden bei der Suche in einem beliebigen Produktkatalog unterstützen kann (siehe [12]).

Es existieren drei kooperierende Gruppen von Akteuren, die mit PIA in Kontakt treten. *Kunden* interagieren mit PIA, um bei ihrer Suche nach einem/ mehreren Produkten unterstützt zu werden. *Katalogbetreiber* benutzen das System, um ihre Produkte bestmöglich zu präsentieren. Weiterhin existiert eine Gruppe von Mittlern zwischen Kunden und Katalogbetreibern, die für die Pflege und Wartung des Systems zuständig sind und somit als *Redakteure* des Sytems zu betrachten sind.

Im folgenden wird allgemein von einem *Katalog* von *Produkten* mit *Eigenschaften* gesprochen. Bei den Produkten kann es sich um unterschiedlichste Arten von Produkten handeln (z.B. Autos, Reisen, Serviceleistungen, Online-Dokumente, ...). In PIA ist es möglich, sowohl die homogene Produktpalette eines einzigen Herstellers (z.B. alle Autos von BMW), als auch eine heterogene Produktpalette ohne gemeinsame Struktur (z.B. alle Produkte eines Kaufhauses) zu repäsentieren.

Das PIA-Modell benutzt zwei sich ergänzende Formalismen. Zunächst werden die zentralen semantischen Konzepte, wie z.B. *Kunde, Katalog, Eigenschaft* usw. als Objekte in einem objektorientierten Datenmodell repräsentiert. Zur Beschreibung wird hier UML (Unified Modeling Language) verwendet. Der für diese Arbeit interessante Ausschnitt aus dem Datenmodell ist in Abbildung 2.1 abgebildet und wird im folgenden weiter erläutert.

Als ein zweiter Formalismus werden Erkenntnisse aus der Theorie unscharfer Mengen zur Repräsentation der dynamischen Beziehungen zwischen den in PIA existierenden Ob-

jekten verwendet.

Im weiteren Verlauf der Arbeit wird ein kleiner Beispielkatalog mit neun Produkten eingeführt. Der Aufbau und die Arbeitsweise des Kataloges werden vor dem Hintergrund des Datenmodells näher erläutert.

2.1 Darstellung des Kataloginhaltes

Ein PIA Katalog besteht aus einer Menge von Produkten. Jedes Produkt wird durch eine Menge von Eigenschaften beschrieben. Eine Eigenschaft ist ein Paar bestehend aus einem Attribut und einem Wert (siehe Abbildung 2.1), zum Beispiel wird eine 4-Sterne Reise in die Türkei zu einem Preis von 2300 DM in PIA wie in Tabelle 2.1 repräsentiert. Jedes Attribut ist zu einer Domäne zugehörig, welche die Menge der für dieses Attribut

Attribut	Wert
Produktkategorie	Reisen
Reiseziel	Türkei
Preis	2300 DM
Anzahl Sterne	vier

Tabelle 2.1: Reise in die Türkei

zulässigen Werte sowie die zulässigen Prädikate definiert. Mehrere Attribute können der selben Domäne angehören.

Dabei existieren keine Regeln, welche Eigenschaften ein Produkt aufweisen muss. Der Katalogbetreiber hat in diesem Punkt alle Freiheiten. Die Eigenschaft *Produktkategorie* ist in PIA eine Eigenschaft wie jede andere auch.

Im Falle des Beispielkataloges existieren fünf verschiedene Domänen, auf denen wiederum sechs Attribute definiert sind (siehe Tabelle 2.2).

Domain	Attribute
Preis	Preis, Stornokosten
Geschwindigkeit	Höchstgeschwindigkeit
formatierter Text	Produktname
Reisekategorie	Anzahl Sterne
Produktkategorie	Produktkategorie
Land	Reiseziel

Tabelle 2.2: Domänen und Attribute des Beispielkatalogs

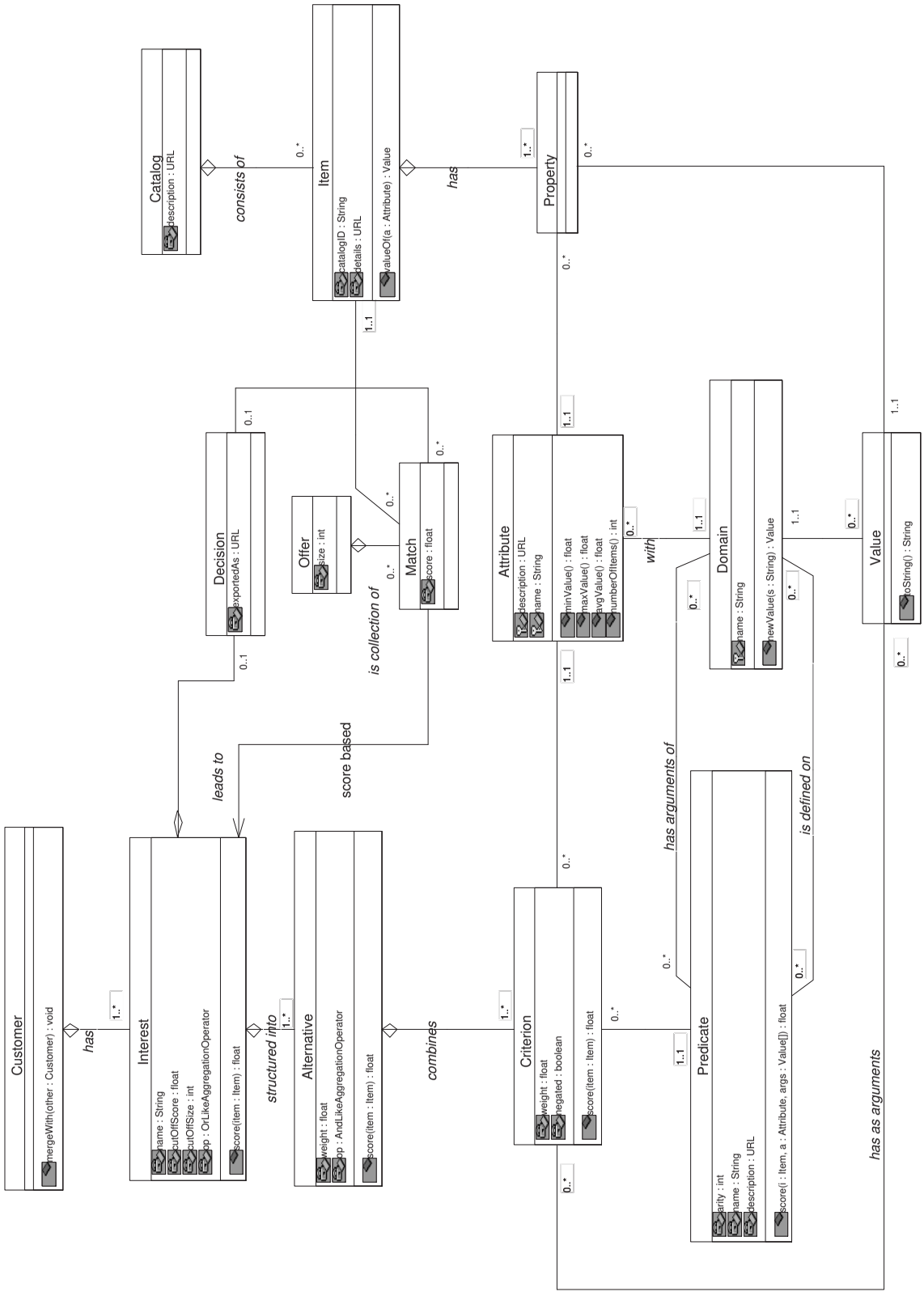

Abbildung 2.1: Ausschnitt des Klassendiagramms von PIA

2.2 Domänen und Werte

Da in PIA keine Anforderungen an die Zuordnung von Attributen zu Produkten gestellt werden, existieren eine Reihe von Anforderungen an die Domänen und die dazugehörigen Werte eines Katalogs. Diese sind erforderlich, um die Konsistenz und Anwendbarkeit des Katalogs zu gewährleisten. Domänen und Werte existieren unabhängig von konkreten Produkten, sie definieren potentiell mögliche Eigenschaften. PIA unterscheidet drei voreingebaute Klassen von Domänen mit unterschiedlichen Wertepräsentationen.

2.2.1 Numerische Domänen

Numerische Domänen enthalten numerische Werte *(NumericValue)*, welche intern als Fließkommazahlen repräsentiert werden. Alle numerischen Werte in PIA werden als exakte Werte angesehen. Die Domänen *Preis* und *Geschwindigkeit* des Beispielkataloges sind numerische Domänen.

2.2.2 Diskrete Domänen

Eine diskrete Domäne besteht aus einer endlichen Menge diskreter Werte, welche durch Zeichenketten repräsentiert sind, z.B.:

```
Reisekategorie = {drei Sterne, vier Sterne, fünf Sterne}
Land = {USA, Kanada, Argentinien, Deutschland,...}
```

Ein diskreter Wert einer diskreten Domäne, ein *DiscreteValue,* wird also durch eine Zeichenkette repräsentiert. In vielen Fällen ist es jedoch wünschenswert, eine Beziehung zwischen den Werten einer diskreten Domäne widerzuspiegeln. Zu diesem Zweck eignet sich eine Baumstruktur, in die die diskreten Werte eingeordnet werden und in der die Kindsknoten z.B. jeweils eine Spezialisierung des Konzeptes des Elternknotens darstellen. Alle diskreten Domänen besitzen eine solche Baumstruktur, was sich am Beispiel der diskreten Domäne Land darstellen lässt:

```
Land =
  (Welt {
    (Nordamerika {Kanada, USA,...}),
    (Südamerika {Brasilien, Argentinien}),
    (Europa {
      (Nordeuropa {Norwegen, Schweden}),
      (Mitteleuropa {
        (Deutschland), Benelux {Niederlande}})})})
```

In PIA ist es erlaubt, dass sich ein diskreter Wert an mehreren Stellen einer Baumstruktur befindet. Diese mehrfachen Vorkommen stellen Kopien ein und desselben Konzeptes dar.

Demzufolge ist es ebenfalls möglich, dass ganze Subbäume als Kopien mehrfach in einer Baumstruktur vorkommen. Jedoch werden die Kindsknoten eines mehrfach vorkommenden Konzeptes nicht automatich kopiert. Zusammenfassend beschreibt ein *DiscreteValue* eine nichtleere Menge von Knoten eines Baumes.

Die Domänen *Land*, *Reisekategorie* und *Produktkategorie* des Beispielkataloges sind diskrete Domänen (siehe Abbildung 4.2).

2.2.3 Volltextdomäne

Die Werte einer Volltextdomäne werden durch beliebige Zeichenketten repräsentiert. Im Rahmen dieser Arbeit spielt die Volltextdomäne keine große Rolle. Im Beispielkatalog ist die Domäne *formatierter Text* eine Volltextdomäne.

2.3 Formulierung eines Kundeninteresses

Ein Kunde kann mehrere, zu einem Katalog gehörende, voneinander unabhängige Interessen haben (z.B.: ein Buch kaufen, ein Geschenk finden). Diese Interessen zerfallen ihrerseits in eine Menge von Alternativen. Demzufolge verwaltet PIA pro Kunde eine Menge von Interessen und pro Interesse eine Menge von Alternativen. Jede Alternative wiederum ist eine Menge von Kriterien. Wenn ein Kunde beispielsweise nach einer billigen 4-Sterne Reise nach Italien sucht, formuliert er eine Alternative entsprechend Tabelle 2.3. Ein

Attribute	*Predicate*	*Value*	Gewichtung
Produktkategorie	ist	Reisen	sehr wichtig
Preis	ist klein		normal
Anzahl Sterne	ist	vier	normal
Reiseland	ist ungefähr	Italien	normal

Tabelle 2.3: Beispiel-Alternative

Kriterium ist also die “kleinste Einheit” eines Kundeninteresses und entspricht einer Zeile dieser Tabelle. Wenn ein Kunde ein Kriterium für ein Attribut a und einen Value v definieren möchte, müssen a und v derselben Domäne d zugehören. Wie anhand des Beispiels ersichtlich wird in unserem Modell jedes Kriterium als die Anwendung eines Prädikates auf ein bestimmtes Attribut bezüglich eines oder meherer Vergleichswerte definiert. Jedes Kriterium ist weiterhin mit einer Gewichtung versehen. Diese Gewichtung spiegelt die Bedeutsamkeit des Kriteriums relativ zu den anderen Kriterien in bezug auf die gesamte Alternative wider. Die Gewichtung eines Kriteriums ist ein numerischer Wert im Intervall $[0, 1]$.

```
class Criterion {
  Predicate predicate; Attribute attribute;
```

```
  Value[] arguments; float weight; bool negated;
  float score (Item i) {
    if (negated) {
      return 1.0 - predicate.score(i, attribute, arguments);}
    else { return predicate.score(i, attribute, arguments);}}
}
```

Um die Benutzerfreundlichkeit zu erhöhen, wurde hier eine kleine Menge von Gewichten vordefiniert und entsprechend durch Zeichenketten repräsentiert.

Die Aufgabe des Systems besteht nun darin, zu einem formulierten Kundeninteresse Produkte zu finden, die diesem Interesse möglichst gut entsprechen, und diese dem Grad der Entsprechung nach sortiert aufzulisten. Zur Berechnung des Grades der Entsprechung wird ein Produkt relativ zu allen Kriterien einer Alternative bewertet. Zu diesem Zweck werden zu jedem Kriterium dem Attribut entsprechende Eigenschaften eines Produktes gesucht und bewertet. Die erzielten Ergebnisse aller Kriterien für ein Produkt werden UND-verknüpft.

```
class Alternative {
  Criterion[ ] criteria;
  AndLikeAggregationOperator op; float weight;
  float score (Item i) {
    float[ ] cs = new float[criteria.size()] ;
    float[ ] cw = new float[criteria.size()] ;
    for ( int j= 0, j < criteria.size(), j++) {
      c=criteria[j]; cs[j]=c.score(i); cw[j]=c.weight; }
    return op(i, cs, cw);}
}
```

Als Ergebnis wird somit der Grad des Entsprechens eines Produktes relativ zu einer Alternative ermittelt. Man erkennt hier die Bewertung eines Kriteriums als "kleinste Bewertungseinheit". Diese Bewertung ist die Aufgabe des jeweiligen Prädikates eines Kriteriums.

Wenn als Beispiel die Reise nach Tabelle 2.1 entsprechend der Alternative aus Tabelle 2.3 bewertet werden soll, muss das Prädikat *ist klein* entscheiden, inwiefern 2300 DM billig sind. Das Prädikat *ist ungefähr* muss entscheiden, zu welchem Grad die Türkei ähnlich zu Italien ist.

2.4 Prädikate

Im folgenden wird die Semantik der von Kunden verwendbaren Prädikate, welche Teil eines Kriteriums sind, erläutert. Ebenfalls sind nur für d definierte Prädikate zugelassen. Da drei voreingebaute Klassen von Domänen existieren, unterscheidet man drei Arten von

Prädikaten: numerische, diskrete und Volltext-Prädikate (*numeric*, *discrete*, *fulltext*). Die Funktionsweise dieser Prädikattypen unterscheidet sich erheblich, weil ihnen unterschiedliche Wertebereiche zugrunde liegen.

Alle diese Prädikate erben von der Superklasse *predicate.* Als durchgehende Beispiele seien das numerische Prädikat *is between* und das diskrete Prädikat *is subsumed by one of* angeführt. Das Prädikat *is between* bewertet inwiefern ein numerischer Wert eines Produktes zwischen zwei gegebenen Vergleichswerten liegt, und das diskrete Prädikat *is subsumed by one of* entscheidet ob ein diskreter Wert eines Produktes von gegebenenen Vergleichswertem "eingeschlossen" ist. Darauf wird in Kapitel 4 näher eingegangen.

Allen Prädikaten gemein sind folgende Variablen/ Methoden:

***name* : String** welche das Prädikat eindeutig charakterisiert (im Beispiel *"is between"* und *"is subsumed by one of"*).

isFinite : boolean, welche kennzeichnet, ob die Anzahl der zu übergebenden Argumente in *args[]* eindeutig begrenzt ist. Im Beispiel *is between* werden genau zwei Werte übergeben, *isFinite* ist demzufolge gleich *true.* Bei manchen Prädikaten (zum Beispiel beim Prädikat *is subsumed by one of*) ist die Zahl der übergebenen Argumente nicht explizit festgelegt, sondern hängt von der jeweiligen Anfrage ab. Zur Kennzeichnung ist der Wert der Variable *isFinite* bei solchen Prädikaten *false.* Zur Vereinfachung wird festgelegt, dass nur die Anzahl von Argumenten derselben Domäne unbestimmt sein kann. Argumente, deren Anzahl vorher festgelegt ist, könne hingegen aus beliebigen Domänen stammen. Argumente deren Anzahl unbestimmt ist werden am Ende des Arrays *args[]* übergeben.

***argsSignature :* String[]** welche die Domänen der zur Prädikatauswertung zu übergebenden Argumente festschreibt . Diese Variable ist ein Array von Strings, welche die jeweiligen Klassennamen der Domänen repräsentieren. Im Beispiel *is between* ist in dieser Variable festgelegt, dass zwei Variablen aus *NumericDomain* übergeben werden, *argsSignature* ist demzufolge gleich *["NumericDomain!", "NumericDomain"].* Falls *isFinite* gleich *false* ist, also beliebig viele Argumente übergeben werden können, geben die ersten $n-1$ Werte von *argsSignature* (bei einer Länge von *argsSignature* von n) die Domänen der ersten $n-1$ Argumente an, und der letzte Wert von *argsSignature* bezeichnet die Domäne der folgenden Argumente. Im Fall *is subsumed by one of* werden beliebig viele Werte aus *DiscreteDomain* übergeben, *argsSignature* ist gleich *[DiscreteDomain].*

boolean testIntegrity(AbstractValue[] args) welche überprüft, ob die übergebenen Parameter in *args* wohlgeformt sind. Es wird also ermittelt, ob Anzahl und Art der übergebenen Parameter mit der Beschreibung durch *isFinite* und *argsSignature* übereinstimmt. *TestIntegrity* wird aufgerufen, sobald ein neues Kriterium erzeugt wird.

float evaluate(Attribute att, AbstractValue val, AbstractValue[] args) welche das Resultat einer Auswertung eines Prädikates zurückliefert. In *val* wird der Wert innerhalb des jeweiligen Produktes übergeben, welches bezüglich einer *Alternative* ausgewertet werden soll. In *args* werden eventuelle Vergleichswerte des *Criterions* übergeben. Als Ergebnis liefert *evaluate()* eine Zahl aus $[0, 1]$, die den Grad des Zutreffens des Prädikates auf *val* unter Berücksichtigung von *args* ausdrückt. Die Methode *evaluate()* wird bei Aufruf der *score()*-Methode eines *Criterions* aufgerufen.

Kapitel 3

Fuzzy Logic

Fuzzy Logic ist eine Erweiterung der gewöhnlichen Mengenlehre durch Einführung *unscharfer Mengen*. Sie wurde 1965 von Zadeh begründet [7], und es folgte eine große Anzahl von Arbeiten zu diesem Thema. In dieser Arbeit soll nur eine kurze Einführung der im folgenden benötigten Grundlagen erfolgen. Zum weiteren Studium der Theorie der Fuzzy Logic sei auf die entsprechende Literatur (siehe [4], [3], [7], [8]) verwiesen.

3.1 Zugehörigkeitsfunktionen

Eine unscharfe Menge A wird charakterisiert durch eine verallgemeinerte charakteristische Funktion $m_A : \chi \to [0, 1]$, die *Zugehörigkeitsfunktion* von A genannt wird und über einem Grundbereich χ definiert ist. Für jede gewöhnliche Menge M ist deren charakteristische Funktion m_M eine derartige Zugehörigkeitsfunktion, deren Funktionswert nur 0 oder 1 werden kann. Gewöhnliche Mengen können demzufolge als Spezialfall der unscharfen Mengen aufgefasst werden. Die Zugehörigkeitsfunktion einer unscharfen Menge kann man für einen endlichen Grundbereich χ zum Beispiel durch eine Wertetabelle angeben. Für den Fall eines unendlichen Grundbereiches χ eignet sich ein Funktionsausdruck. Dabei ist zu beachten, dass jeder Graph für eine subjektive Modellierung von linguistischem Wissen steht. Demzufolge muss von Fall zu Fall neu entschieden werden, welche Zugehörigkeitsfunktion für welches Problem optimal ist.

Die in der Praxis verwendeten Funktionen besitzen oft einen einfachen mathematischen Zugang (siehe [3], [9]). Eine große Rolle spielt die Klasse der linearen konvexen Funktionen. Im Rahmen dieser Arbeit wurden drei verschiedene lineare konvexe Funktionen verwendet. Es handelt sich dabei um die L-Funktion (siehe Tabelle 3.1 und Abbildung 3.1), die Γ-Funktion (siehe Tabelle 3.2 und Abbildung 3.2) sowie die Trapez-Funktion (siehe Tabelle 3.3 und Abbildung 3.3). Diese Funktionen besitzen den Vorteil, dass sie durch wenige Parameter (L-Funktion und Γ-Funktion besitzen jeweils zwei Parameter, die Trapez-Funktion besitzt vier Parameter) beschrieben sind und sich sehr einfach auswerten lassen.

Bereich	
$x < a_1$	$m_A(x) = 0$
$a_1 \leq x < a_2$	$m_A(x) = \frac{(x-a_1)}{(a_2-a_1)}$
$x \geq a_2$	$m_A(x) = 1$

Tabelle 3.1: L-Funktion

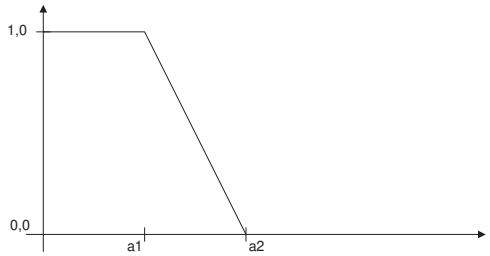

Abbildung 3.1: Graph der L-Funktion

Bereich	
$x < a_1$	$m_A(x) = 1$
$a_1 \leq x \leq a_2$	$m_A(x) = \frac{a_2-x}{a_2-a_1}$
$x > a_2$	$m_A(x) = 0$

Tabelle 3.2: Γ-Funktion

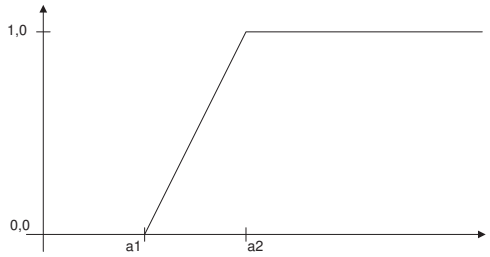

Abbildung 3.2: Graph der Γ-Funktion

Bereich	
$x < a_1$	$m_A(x) = 0$
$a_1 \leq x \leq a_2$	$m_A(x) = \frac{x-a_1}{a_2-a_1}$
$a_2 \leq x \leq a_3$	$m_A(x) = 1$
$a_3 \leq x \leq a_4$	$m_A(x) = \frac{a_4-x}{a_4-a_3}$
$x > a_4$	$m_A(x) = 0$

Tabelle 3.3: Trapez-Funktion

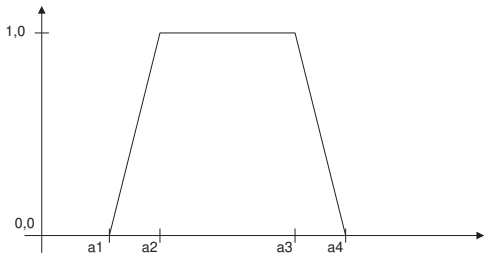

Abbildung 3.3: Graph der Trapez-Funktion

3.2 Aggregatoren

Im Falle einer Datenbank mit Fuzzy-Logic Funktionalität ist der Wert einer Zugehörigkeitsfunktion $m_A(x)$ als der Grad des Zutreffens eines Elementes $x \in \chi$ auf eine Anfrage A aufzufassen. Je größer dieser Wert ist, desto mehr trifft x auf A zu, desto eher ist x das gewünschte Ergebnis. Im Fall von PIA existiert zu jedem Kriterium eine Methode *score(item Item) float,* welche den Grad des Zutreffens des Produkts *item* auf das jeweilige Kriterium liefert. Diese Methode liefert also den Wert der Zugehörigkeitsfunktion $m_A(x)$ wobei x dem untersuchten Produkt *item* entspricht. Jede Alternative wiederum besteht aus mehreren Kriterien. Es wird also eine Methode benötigt, die Werte mehrerer Zugehörigkeitsfunktionen $m_A(x)$, $m_B(x)$, $m_C(x)$,..,$m_N(x)$ sinnvoll zu einem Gesamtergebnis zu kombinieren. Dieses Gesamtergebnis repräsentiert den Grad des Zutreffens eines Produkts auf eine Alternative. Eine ähnliche Aufgabe stellt sich, wenn die Zugehörigkeitsgrade mehrerer Alternativen, die durch ein Interesse aggregiert sind zu einem Gesamtergebnis zusammengefasst werden müssen.

Im Falle der klassischen Mengenlehre sind die Vereinigungs- und Durchschnittsbildung entsprechende Methoden. Demzufolge sucht man nach Verallgemeinerungen dieser Operationen für unscharfe Mengen. Die unscharfe Durchschnittsbildung löst das Problem der kombinierten Auswertung von Kriterien innerhalb einer Alternative, und die Vereinigungsbildung die kombinierten Auswertung mehrerer Alternativen innerhalb eines Interesses.

3.2.1 Konjunktion und Disjunktion

Bereits 1965 gab Zadeh [7] entsprechende Verallgemeinerungen an. Er wählte für den Durchschnitt zweier unscharfer Mengen A und B:

$$C := A \cup B \ : \ m_C(x) =_{def} \min\{m_A(x),\, m_B(x)\} \qquad \textit{für alle } x \in \chi, \tag{3.1}$$

sowie für die Vereinigung:

$$D := A \cup B \ : \ m_D(x) =_{def} \max\{m_A(x),\, m_B(x)\} \qquad \textit{für alle } x \in \chi. \tag{3.2}$$

Beide Operationen sind assoziativ und kommutativ. Im folgenden soll exemplarisch nur noch die Durchschnittsbildung betrachtet werden, da die dabei gewonnenen Erkenntnisse im weiteren benötigt werden. Obgleich die Wahl der Definition 3.1 intuitiv sinnvoll erscheint, muss nicht zwingend diese Definition gewählt werden. Darüber hinaus existieren eine Reihe von weiteren Durchschnittsbildungen, die einem übergreifendem Konzept entsprechen. Als unterscheidendes Merkmal dieser Operationen werden anstelle der Minimumsbildung in 3.1 sogenannte t-Normen benötigt, die im folgenden vorgestellt werden.

3.2.2 t-Norm

Unter einer t-Norm [8] versteht man eine zweistellige Funktion t von [0, 1] in [0, 1], die kommutativ, assoziativ und monoton wachsend ist sowie 1 als neutrales und 0 als Nullelement hat. Eine t-Norm erfüllt also für beliebige x, y, z, u, $v \in [0, 1]$ die folgenden Bedingungen:

$$y\,t\,x = x\,t\,y \tag{3.3}$$

$$x\,t\,(y\,t\,z) = (x\,t\,y)\,t\,z \tag{3.4}$$

$$wenn\ x \leq u\ und\ y \leq v,\ so\ x\,t\,y \leq u\,t\,v \tag{3.5}$$

$$x\,t\,1 = x\ und\ x\,t\,0 = 0. \tag{3.6}$$

Zu jeder t-Norm erhält man eine Durchschnittsbildung $\cap_t$ für unscharfe Mengen, indem man den (auf die t-Norm bezogenen) Durchschnitt $A \cap_t B$ definiert gemäß

$$C := A \cap_t B\ :\ m_C(x) =_{def} m_A(x)\,t\,m_B(x) \qquad für\ alle\ x \in \chi. \tag{3.7}$$

Wegen der Kommutativität und der Assoziativität der t-Norm sind die Durchschnittsbildungen $\cap_t$ gemäß 3.7 ebenfalls kommutativ und assoziativ. Aus den Bedingungen 3.5 (Monotonie) und 3.6 (neutrales und Nullelement) folgt, dass stets

$$x\,t\,y \leq x\,t\,1 = x \tag{3.8}$$

und ebenso

$$x\,t\,y \leq y \tag{3.9}$$

gelten.

Für $A \cap B$ gemäß 3.1 ist die zugehörige t-Norm die Operation t_{min}:

$$x\, t_{min}\, y = \min\{x,\, y\} \tag{3.10}$$

Fordert man von einer t-Norm $\widehat{t}$ zusätzlich noch $x\,\widehat{t}\,x = x$, so kann man zeigen, dass nur t_{min} diese Bedingung erfüllt. Ebenfalls ist die von t_{min} abgeleitete Durchschnittsbildung $\cap_{min}$ die einzige *nicht-interaktive* Durchschnittsbildung, d.h. es gilt stets:

$$x\, t\, y \in \{x, y\}\,. \tag{3.11}$$

Alle anderen möglichen t-Normen sind *interaktiv*, das heisst es gilt:

$$x\, t\, y < \min\{x,\, y\} \tag{3.12}$$

Wie man nun leicht sehen kann, erhält man eine Kombination der Ergebnisse vieler Zugehörigkeitsfunktionen m_A, m_B, m_C,... durch fortgesetztes Anwenden einer Durchschnittsbildung unter Benutzung einer beliebigen t-Norm als Parameter. Dieser Vorgang wird als *Aggregation* bezeichnet. Natürlich hängt das Ergebnis der Aggregation von der verwendeten t-Norm ab.

3.3 Relationen

Ein weiteres wichtiges Problem ist die Spezifizierung *unscharfer Relationen*: z.B. *größer als*, *ungefähr gleich*, u. ä. Es werden also Aussagen über das Zutreffen einer bestimmten Relation R auf ein Wertepaar $(x,\, y)$ gemacht. Diese Arbeit wird sich auf binäre, d.h. zweistellige Relationen beschränken. Solche Relationen werden formal als unscharfe Teilmenge über einem Grundbereich

$$Y = \chi_1 \times \chi_2 = \{(x,\, y) \mid x \in \chi_1 \quad \textit{und} \quad y \in \chi_2\}$$

aufgefasst, wobei der Zugehörigkeitswert $m_R(a,\, b)$ der Grad ist, mit dem die unscharfe Relation R auf ein Paar $(a,\, b)$ zutrifft. Wie üblich bedeutet $m_R(a,\, b) = 1$, dass R auf $(a,\, b)$ voll und ganz zutrifft, während $m_R(a,\, b) = 0$ bedeutet, dass R auf $(a,\, b)$ überhaupt nicht zutrifft.

Die im folgenden wichtigste Operation auf unscharfen Relationen ist die Bildung des *Relationenproduktes* (*Verkettung*) $R \circ S$. Die Definition des Relationenproduktes $R \circ S$ kann durch Rückgriff auf auf eine beliebige t-Norm gegeben werden:

$$T = R \circ S \;:\; m_T(x,\, y) =_{def} \sup_{z \in \chi_2} m_R(x,\, z)\, t\, m_S(z,\, y) \tag{3.13}$$

$$\textit{für alle } (x,\, y) \in \chi_1 \times \chi_3.$$

Wie man sieht, hängt das Ergebnis des Relationenproduktes von der verwendeten t-Norm ab. Ebenfalls lässt sich erkennen, dass man mehrere Relationsprodukte ähnlich wie in 3.2.2 hintereinander ausführen kann.

3.4 Implementierte t-Normen

- Minimum-Norm

$$u\, t_{min}\, v = \min\{u,\, v\} \tag{3.14}$$

- Einstein-Norm

$$u\, t_e\, v = \frac{uv}{1 + (1-u)(1-v)} \tag{3.15}$$

- Yager-Norm

$$u\, t_{Y,p}\, v =_{def} 1 - \min\left\{1,\, ((1-u)^p + (1-v)^p)^{\frac{1}{p}}\right\} \tag{3.16}$$

- Hamacher-Norm

$$u\, t_{h,\gamma}\, v = \frac{uv}{\gamma + (1-\gamma)(u + v - uv)} \tag{3.17}$$

- probabilistische Norm

$$u\, t_p\, v = uv \tag{3.18}$$

Kapitel 4

Diskrete Prädikate

Im folgenden wird die Arbeitsweise der Prädikate auf den diskreten Domänen des PIA-Systems erläutert. Die wichtigste Methode eines Prädikates ist dabei die Methode *evaluate()*. Die übergebenen Parameter *args[]* und *val* (siehe 2.4) stammen für die momentan existierenden Prädikate alle aus derselben diskreten Domäne, und somit aus derselben Baumstruktur. Allgemein soll die *evaluate()*-Methode der diskreten Prädikate nun Aussagen über Verwandtschaftsbeziehungen zwischen den übergebenen Knotenobjekten einer Baumstruktur treffen. Formal repräsentiert ein diskretes Prädikat eine Relation zwischen den übergebenen Werten. Mittels der *evaluate()* Methode wird also eine Aussage über das Zutreffen einer Relation auf bestimmte Wertepaare getroffen.

Von den implementierten Prädikaten (siehe Tabelle 4.1) repräsentieren *is exactly, is similar to* und *is subsumed by* binäre Relationen. *Is subsumed by one of* ist eine disjunktive Verknüpfung einer Menge von binären Relationen. Da lediglich das Prädikat *is similar to*

Name des Prädikates	*argsSignature*
is exactly	(DiscreteDomain)
is similar to	(DiscreteDomain)
is subsumed by	(DiscreteDomain)
is subsumed by one of	(DiscreteDomain, DiscreteDomain[])

Tabelle 4.1: Implementierte diskrete Prädikate

eine gewisse Unschärfe beinhaltet, ist es Gegenstand der weiteren Betrachtungen.

4.1 Betrachtungen über *is similar to*

Im folgenden soll dargelegt werden, was ein Kunde von der Funktionsweise des Prädikates *is similar to* erwartet. Der Kunde formuliert eine Alternative (siehe 2.3) mit einem diskreten Kriterium, das auf dem Prädikat *is similar to* aufbaut. Als Vergleichswert des Kriteriums übergibt er einen Wert einer diskreten Domäne und ist an Produkten "nahe"

diesem formulierten Wert interessiert. Dieser Wert repräsentiert ein bestimmtes Konzept der entprechenden diskreten Domäne.

Da die Baumstruktur der diskreten Domänen eine Spezialisierungs-/ Generalisierungsbeziehung widerspiegelt, was im folgenden noch motiviert wird, ist der Kunde an allen spezialisierteren Konzepten sowie an "eng benachbarten" Konzepten interessiert. Konzepte, für die im Baum zunächst wenige oder gar keine Stufen hinaufgegangen werden muss, sind also interessanter als Konzepte, für die erst viele Stufen aufgestiegen werden muss.

Am Beispiel der in Abbildung 4.1 dargestellten Baumstruktur übergibt der Kunde das Konzept des mit 1 beschrifteten Knotens. Um zu Knoten zwei zu gelangen muss nur ein Level hinaufgestiegen werden, auf dem Weg zu Knoten drei hingegen müssen drei Level überwunden werden. Das Konzept des Knotens zwei besitzt also eine höhere Relevanz bezüglich Knoten eins, als das Konzept von Knoten drei. Das Prädiket *is subsumed by*

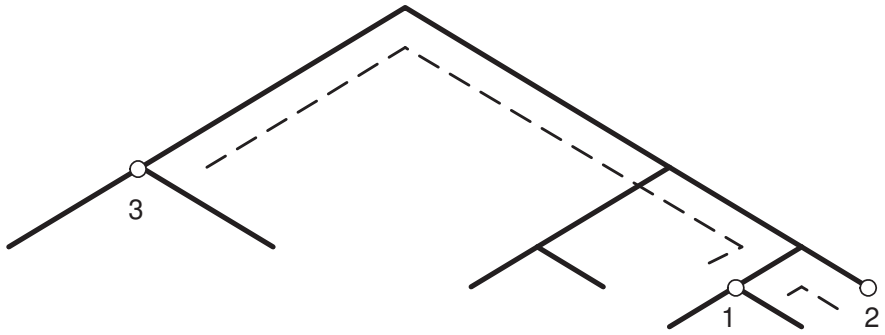

Abbildung 4.1: Baumstruktur

liefert im Unterschied zu *is similar to* als Resultat nur alle spezialisierteren Knoten. Daraus folgt, dass das Prädikat *is similar to* eine Fuzzy-Erweiterung des Prädikates *is subsumed by* darstellt.

Im folgenden wird eine formalere Diskussion der Arbeitsweise des Prädikates durchgeführt, welche zum oben formulierten Resultat führt.

Die hier zunächst umgangssprachliche Beschreibung der Arbeitsweise des Prädikates *is similar to* wird dabei durch eine formalere Diskussion des Aufbaus diskreter Bäume ergänzt, wodurch die in diesem Abschnitt nur oberflächlich verwendeten Begriffe mehr Klarheit erlangen.

4.2 Art der Beziehung zwischen den Konzepten einer diskreten Domänen

Die Baumstruktur der diskreten Domäne stellt eine Beziehung zwischen den Knoten des Baumes (DiscreteValues) her. Es gibt zwei verschiedene Arten von Beziehungen, die mittels einer Baumstruktur repräsentiert werden können. Häufig bestehen z.B. Teil-Ganzes-

Beziehungen zwischen Eltern- und Kindsknoten. Dadurch wird ausgedrückt, dass die Kindsknoten Teile des Elternknotenkonzeptes sind. Eine andere Art der Beziehung ist eine Spezialisierungs-/ Generalisierungsbeziehung zwischen verwandten Knoten. Das bedeutet, dass die Kindsknoten das Konzept des Elternknotens spezialisieren.

Im folgenden wird davon ausgegangen, dass die Baumstruktur der diskreten Domäne eine Spezialisierungs-/ Generalisierungsbeziehung widerspiegelt. Diese Entscheidung wurde getroffen, da Spezialisierungen für einen Katalog von entscheidender Bedeutung sind. Die wohl wichtigste diskrete Domäne eines jeden Produktkataloges ist die Domäne *Produktkategorie*. Es ist zwar in PIA nicht zwingend erforderlich, eine solche Domäne zu verwalten, doch empfiehlt es sich in jedem Fall. Mittels dieser Domäne wird eine Spezialisierungs-/ Generalisierungsbeziehung zwischen den Konzepten des Kataloges aufgebaut, wie zum Beispiel in Abbildung 4.2 dargestellt. Die meisten Produkte eines Kataloges werden jeweils in den Blattknoten der Domänen eingehängt sein. Es kann jedoch verschiedene Gründe geben, Produkte auch inneren Knoten des Baumes zuzuordnen. So ist es beispielsweise möglich, dass die entsprechende Spezialisierung in den Blättern des Baumes nicht vorhanden ist, oder der Katalogbetreiber sich nicht für eines der speziellen Konzepte entscheiden kann.

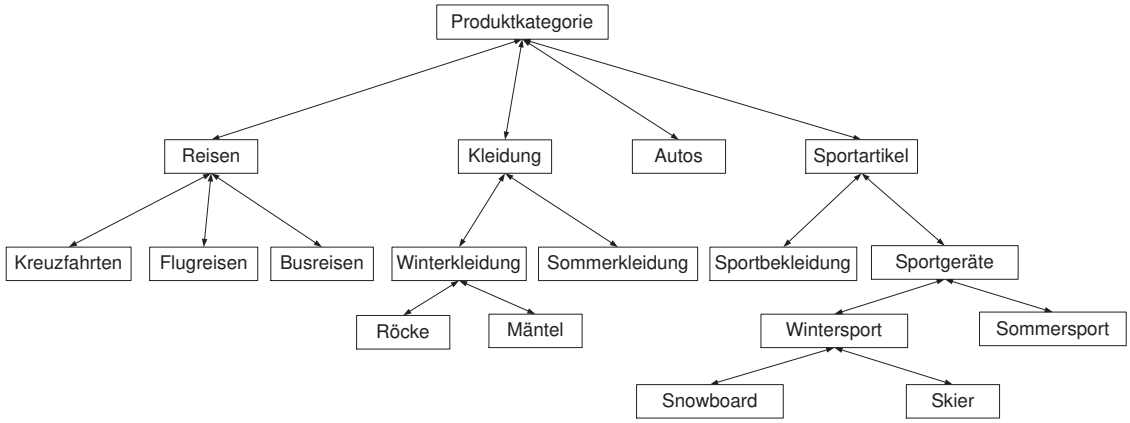

Abbildung 4.2: Diskrete Domäne: Produktkategorie

Die Domäne Produktkategorie spielt sicherlich in jedem Kundeninteresse eine zentrale Rolle. Je konkreter die Vorstellung eines Kunden ist, desto genauer weiss er, wo sein Wunsch in dieser Domäne einzuordnen ist, desto genauer spezialisiert ist seine Vorstellung bereits.

Produkte werden in PIA als Ganzes abgelegt und durch eine Reihe von Eigenschaften charakterisiert. Zum Beispiel wird ein PC in PIA nicht als Summe aus Monitor, Gehäuse, Prozessor usw. repräsentiert, sondern als Ganzes und die Einzelteile werden durch bestimmte Eigenschaften spezifiziert: Bildschirmdiagonale des Monitors, Taktfrequenz des Prozessors usw.

Diese Eigenschaften ordnen das jeweilige Produkt in eine Reihe von diskreten Domänen

(jeweils Baumstrukturen) ein, und charakterisieren so die Art des Produktes. Produkte werden also nicht durch die Einzelteile aus denen sie bestehen gekennzeichnet. Dies äussert sich dadurch, dass die Knoten in den Baumstrukturen der Domänen keine Teilprodukte, sondern Konzepte der jeweiligen Domäne sind. Demzufolge wird durch die diskreten Domänen keine Teil-Ganzes Beziehung, sondern eine Spezialisierungs-/ Generalisierungsbeziehung repräsentiert.

4.3 Relationswerte

Es existiert also für jede Domäne eine Baumstruktur, welche eine Spezialisierungs-/ Generalisierungsbeziehung zwischen den Knotenkonzepten repräsentiert. Diese Beziehung lässt sich als Fuzzy-Relation $m_R(x, y)$ auffassen, wobei x und y verwandte Knoten (Konzepte) eines Baumes sind. Der Wert der Relation $m_R(x, y)$ spiegelt den Grad der Verwandtschaft des Konzeptes y zum Konzept x wider, wie in [1] dargestellt. Diese Relation ist im allgemeinen nicht symmetrisch, d.h. $m_R(x, y) \neq m_R(y, x)$.

Aus der Art der Beziehung folgt, dass der Wert der Relation zwischen Elter und Kind ($m_R(Elter, Kind)$) den Zahlenwert 1.0 erhält, da der Elternknoten eine Generalisierung des Kindknotenkonzeptes darstellt. Interessanter ist der Zahlenwert für die Relation zwischen Kind und Elter $m_R(Kind, Elter)$, der etwas über den Grad der Spezialisierung zwischen den Konzepten x und y aus. Ein Konzept, welches ein anderes Konzept spezialisiert, beschreibt nur eine Teilmenge des allgemeineren Konzeptes. Je kleiner diese Teilmenge ist, desto kleiner muss der Wert von $m_R(Kind, Elter)$ sein.

Je feiner eine Baumstruktur aufgeschlüsselt ist, desto kleiner ist diese Teilmenge durchschnittlich. Demzufolge ergeben sich durchschnittlich kleinere Werte für $m_R(Kind, Elter)$.

Baumstrukturen kann man graphisch anschaulich darstellen. Ebenso lassen sich die Kantenwerte in die graphische Darstellung einfügen. Die Darstellung unterliegt im folgenden stets dem in Abbildung 4.3 eingeführten Schema. Dabei bezeichnet b den Wert von $m_R(Elter, Kind)$ und a den Wert von $m_R(Kind, Elter)$.

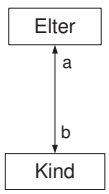

Abbildung 4.3: Kante

In Abbildung 4.4 ist ein Ausschnitt aus der diskreten Domäne *Produktkategorie* des Beipielkataloges, versehen mit Kantenwerten, dargestellt. Diese Kantenwerte wurden "von Hand" gewählt. Als Kriterium für die Wahl wurden die Verwandtschaftsgrade berücksichtigt.

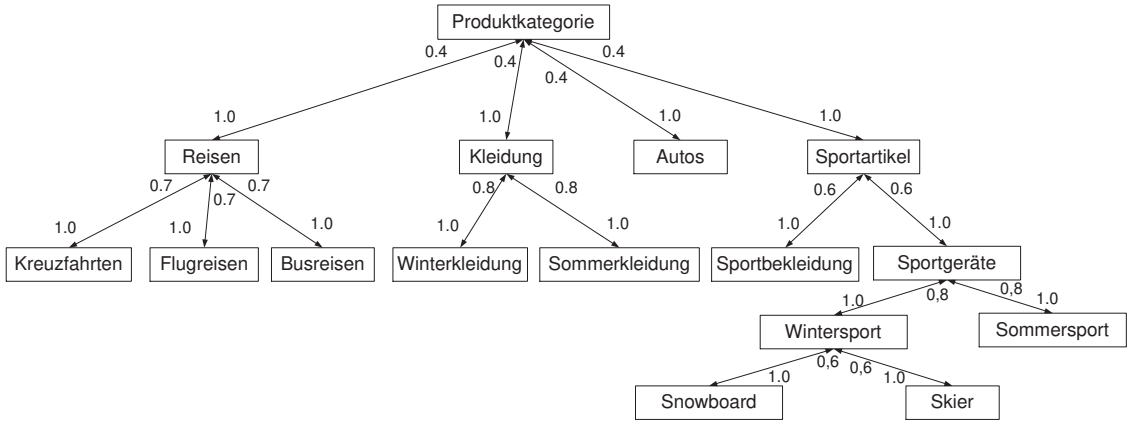

Abbildung 4.4: Diskrete Domäne mit Kantenwerten

4.4 Auswertung des Prädikates

Gegeben sind eine mit Kantenwerten versehene Baumstruktur der entsprechenden diskreten Domäne, die Wahl einer zu verwendenden t-Norm, ein Start- und ein Endknoten. Start- und Endknoten werden durch *args[0]* und *val* der *evaluate()* Methode übergeben. Dabei ist *args[0]* ein Vergleichswert und entstammt dem jeweiligen Kriterium (siehe 2.3), während *val* eine Eigenschaft des Produkts beschreibt (siehe 2.1), welches entsprechend dem Kriterium bewertet werden soll. Diese beiden Knoten stellen die Konzepte dar, zwischen denen ein Grad der Ähnlichkeit vom Prädikat *is similar to* ermittelt werden soll. In Fuzzy-Notation entspricht das einer Auswertung der unscharfen Relation $m_R(args[0], val)$.

Dies geschieht wie in [1] geschildert, indem der (eindeutige) Pfad vom Start- zum Endknoten gefunden wird und fortgesetzt das Relationenprodukt der abgelaufenen Kanten (vgl. Kapitel 3.13) ausgewertet wird. Da der Pfad als eindeutig angenommen wird, entfällt hier die Bildung des Supremums für das Relationenprodukt.

Da die gegebenen Start- und Endknoten jedoch Werte einer diskreten Domäne sind und wie in 2.2.2 beschrieben sich an mehreren Stellen der Baumstruktur befinden können müssen alle möglichen Kombinationen von Pfäden durchgespielt werden, wobei der Pfad mit dem höchsten Relationenprodukt das Ergebnis der Prädikatauswertung darstellt.

In dem in Abbildung 4.5 dargestellten Beispiel ist der Startknoten durch *Sportgeräte* und der Endknoten durch *Kleidung* gegeben. Auf dem Pfad werden drei Kanten abgegangen mit den Werten 0.6, 0.4 und 1.0. Das Ergebnis ist also durch $(0.6\,t\,0.4)\,t\,1.0$ gegeben.

Der Einfluss der Wahl einer t-Norm auf das jeweils erzielte Ergebnis wurde in [2] untersucht, wobei die Ergebnisse nicht vollständig auf den vorliegenden Sachverhalt übertragbar sind. Demzufolge wird die Wahl einer t-Norm im folgenden noch Gegenstand der Betrachtung sein.

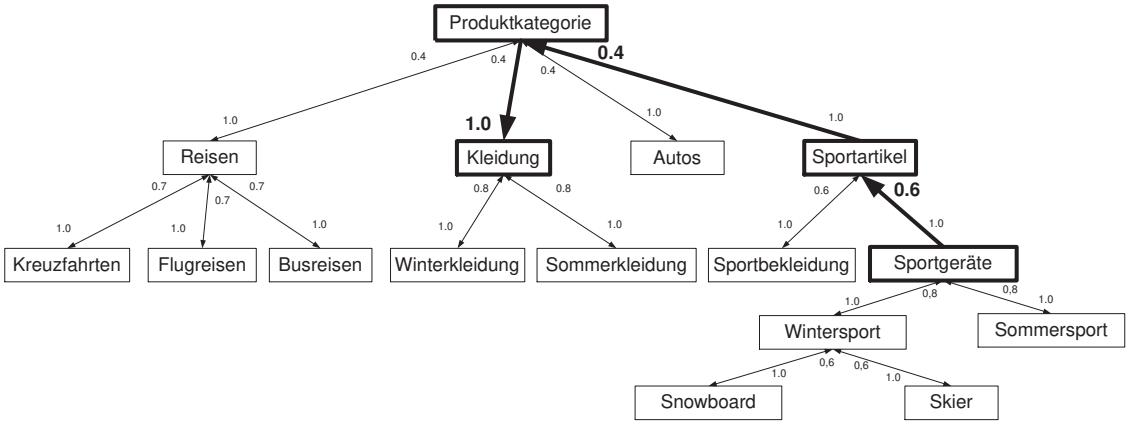

Abbildung 4.5: Auswertung von *is similar to*

4.5 Bestimmung der Kantenwerte

Als nächstes stellt sich die Frage nach einer sinnvollen Vorgabe der Kantenwerte sowie nach der Auswahl der t-Norm.[1] Prinzipiell gibt es zwei extreme Möglichkeiten, die Kantenwerte zu erlangen: manuell oder automatisch. Die erste Möglichkeit besitzt den Vorteil größter Variabilität und Freiheit für den Katalogbetreiber, erfordert aber ein großes Maß an Kenntnissen und Aufwand, um alle diskreten Domänen mit Relationen zu versehen. Aus diesen Gründen ist das PIA-System in der Lage, eine automatische Belegung der Kanten mit Relationswerten vorzunehmen . Ein gemischter Betrieb ist ebenfalls möglich. Auch hier sind mehrere Szenarien denkbar: der Katalogetreiber trägt die gewünschten Kantenwerte absolut ein, oder er gibt dem System einen Hinweis, den es im Algorithmus zur automatischen Generierung berücksichtigt. Dieser Hinweis kann z.B. über die Entscheidung zwischen zwei Menüpunkten *Beziehung verstärken*/ *Beziehung lockern* erfolgen oder über Schieberegler visualisiert werden. In Abhängigkeit dieser Angaben verändert das System nun die automatisch vorgeschlagene Relation.

Der Algorithmus zur automatischen Zuordnung von Relationswerten hängt grundlegend von der bei der Berechnung des Relationenproduktes verwendeten t-Norm ab (vgl. Kapitel 3.3). Im folgenden werden zwei verschiedene Arten von t-Normen bzgl. ihrer Eignung im diskreten Prädikat *is similar to* vorgestellt:

4.5.1 Minimum-Norm (nicht-interaktiv)

Die Anwendung der Minimum-Norm hat den Vorteil der Einfachheit und leichten Nachvollziehbarkeit ihrer Arbeitsweise. Ein Nachteil ist, dass das Verwenden eines einzigen Relationswertes a für alle Kanten nicht zu in 4.1 geforderten Ergebnissen führt. Wie man leicht einsehen kann, führt eine Auswertung des Prädikates "is similar to" entweder zu dem

[1] Der Begriff *Kantenwert* bezieht sich im folgenden auf den Wert von $m_R(Kind, Elter)$. Für $m_R(Elter, Kind)$ wird stets der Wert 1.0 angenommen.

Ergebnis 1.0 oder a. 1.0 erhält man für alle spezialisierten Konzepte (also alle Kinderknoten), während a das Resultat für alle anderen Knoten ist. Dies entspricht bei weitem nicht dem gewünschten Ergebnis einer Prädikatauswertung.

Ändern lässt sich das nur durch eine höhenabhängige Gewichtsverteilung. Je höher eine Kante sich in der Baumstruktur befindet, desto kleiner müsste ihr Wert sein. So kann man zwar das gewünschte Verhalten erzwingen, aber auf Kosten einer Verkomplizierung der Arbeitsweise des Algorithmus zur Kantenbewertung. Als Schlussfolgerung daraus ist von einer Verwendung der abzusehen. Die wurde zwar im Laufe der Arbeiten auf Seiten des Prädikates implementiert, aber von einer Unterstützung durch einen Algorithmus zur automatischen Bestimmung höhenabhängiger Kantengewichte wurde abgesehen.

4.5.2 Verwendung einer interaktiven Norm

Im Gegensatz zur Minimum-Norm führt aufgrund der in 3.2.2 vorgestellten mathematischen Zusammenhänge, insbesondere aufgrund von Formel 3.12, bei Verwendung einer interaktiven Norm ein beliebiger Wert < 1 für alle Kanten schon zu gewünschten Ergebnissen.

4.5.3 Algorithmus zur Unterstütznng des Katalogbetreibers bei der Bestimmung der Kantenrelationen

Dem im folgenden vorgestellten Algorithmus liegt die Idee zugrunde, einen solchen für alle Kanten gültigen Wert, im folgenden *Grundrelation* a genannt, zu finden. Dieser Wert wird im weiteren zunächst vom Algorithmus einmal korrigiert. Im Abschluss hat auch der Katalogbetreiber die Möglichkeit, manuell korrigierend einzugreifen.

Zunächst kommt dem Algorithmus also die Aufgabe zu, einen geeigneten Wert für die Grundrelation a zu finden. Wenn man von gleichgewichteten Kanten ausgeht, und alle abwärtsgerichteten Kanten mit 1.0 belegt sind, hat nur noch die Anzahl der Kanten, die im Baum hinaufführen, Einfluss auf das Ergebnis einer Auswertung des Prädikates "is similar to". Je mehr Aufwärtskanten in der Baumstruktur gefolgt werden muss, um zum Ziel zu gelangen, desto kleiner wird der Ergebniswert der Prädikatauswertung.

Dies entspricht auch der in Kapitel 4.1 formulierten intuitiven Vorstellung, die man von der Arbeitsweise des Prädikates hat: Je weiter man von einem bestimmten Startknoten erst den Baum hinauflaufen muss, desto geringer ist die Ähnlichkeit zwischen Start- und Zielknoten. Einfluss auf die Wahl der Grundrelation a hat, wie in Kapitel 4.2 angedeutet, die *Differenziertheit* der jeweiligen Baumstruktur. Unter der Differenziertheit einer Baumstruktur ist dabei der durchschnittliche Grad der Spezialisiertheit zwischen Elter- und Kindsknoten zu verstehen. Je verfeinerter eine Struktur ist, und je enger demzufolge die Beziehung zwischen Elter- und Kindsknoten ist, desto größer ist die Differenziertheit eines Baumes. Als Maß für die Differenziertheit wird ein Paar zweier positiver Zahlen (a, b) gewählt, wobei a eine ganze Zahl, und b eine reelle Zahl zwischen 0 und 1 ist. a heisst

Differenziertheitsniveau, und spiegelt die Anzahl der Stufen wider, die im Baum überwunden werden müssen, um einen gewissen Wert b (Differenziertheitsgrad) zu unterschreiten (z.B. (4, 0.1). Wählt man also zum Beispiel (4, 0.1) als Differenziertheit, dann führen vier hinaufgegangene Level zu einem Absinken des Ähnlichkeitsmaßes (fortgesetztes Relationenprodukt unter Verwendung einer t-Norm) auf unter 0.1. Im allgemeinen ist natürlich eine stark verfeinerte Struktur wünschenswert, da erst dann das Prädikat *is similar to* in der Lage ist, aussagekräftige Ergebnisse zu erzielen. Der Algorithmus ist auf eine Eingabe des Katalogbetreibers bezüglich der Differenziertheit des Baumes angewiesen, da es unmöglich ist diese nur aus der Struktur des Baumes zu schließen.

Aufgrund der für den Katalogbetreiber wenig einsichtigen Semantik von Paaren positiver Zahlen wurden drei Differenziertheitsmaße vordefiniert und dem Katalogbetreiber zur Verfügung gestellt. Diese drei Möglichkeiten mit den zugeordneten Differenziertheiten sind in Tabelle 4.2 dargestellt, und sind natürlich anpassbar und erweiterbar.

Maß	Differenziertheit
wenig	(3, 0.2)
mittel	(3, 0.4)
stark	(4, 0.4)

Tabelle 4.2: eingebaute Differenziertheitsmaße

Falls der Katalogbetreiber nichts spezifiziert wird eine mittlere Differenziertheit angenommen.

Weiterhin muss dem bereits geäusserten Gedanken, dass der Zahlenwert von $m_R(Kind, Elter)$ etwas über den Grad der Spezialisierung zwischen den Konzepten x und y aussagt Rechnung getragen werden. Ein Konzept, welches ein anderes Konzept spezialisiert beschreibt nur eine Teilmenge des allgemeineren Konzeptes. Je kleiner diese Teilmenge ist, desto kleiner muss der Wert von $m_R(Kind, Elter)$ sein. Je mehr Kindsknoten ein Elternknoten hat, desto wahrscheinlicher ist es, dass ein Kindsknoten eine kleine Teilmenge des übergeordneten Konzeptes beschreibt. Demzufolge muss ein kleinerer Wert für die Relation gewählt werden. Dies wird erreicht, indem der Wert der Grundrelation gemäß Differenziertheit mit einem Wert laut Tabelle 4.3 multipliziert wird. Die Werte der Tabelle sind anpassbar und erweiterbar und haben sich in Zusammenhang mit dem von uns verwendeten Produktkatalog als sinnvoll erwiesen. Denkbar wäre auch eine Skalierung der Faktoren über die maximal vorkommende Anzahl von Kindsknoten.

Die dem letzten Schritt zugrundeliegende Annahme ist, dass sich die Kindsknoten das Konzept des Elternknoten untereinander zu gleichen Teilen "aufteilen". Dies ist sicherlich nicht immer der Fall, und es müssen daher Möglichkeiten existieren, dem gerecht zu werden. Zu diesem Zweck hat der Katalogbetreiber die Möglichkeit einzelne Kanten gezielt zu beeinflussen. Es existieren vier vorgefertigte *Veränderungsqualifizierer*, welche jeweils einen der in Tabelle 4.4 aufgeführten Faktoren repräsentieren. Falls also ein Kindskon-

Anzahl Kindsknoten	Faktor
1	1,5
2	1,2
3	1,0
4	0,8
> 4	0,6

Tabelle 4.3: Mulitplikatoren

enger	1.2
viel enger	1.5
lockerer	0.8
viel lockerer	0.6

Tabelle 4.4: Veränderungsqualifizierer

zept einen größeren oder kleineren Anteil des Elternkonzeptes als die übrigen Kindsknoten widerspiegelt, kann der Katalogbetreiber dem Rechnung tragen.

Zusammenfassend ergeben sich folgende Möglichkeiten der Einflussnahme des Katalogbetreibers:

- Entscheidung für eine bestimmte Norm: Einfluss auf Wahl des Algorithmus und Arbeit des Prädikates
- Eingabe eines Maßes für die Differenziertheit der Baumstruktur: Einfluss auf die vom Algorithmus berechnete Grundrelation a
- Eingabe eines Veränderungsqualifizierers: Einfluss auf die vom Algorithmus berechnete Kantenrelation $m_R(Kind, Elter)$

Alle diese Möglichkeiten können nebeneinander wahrgenommen werden.

Am Beispiel der Domäne Produktkategorie sollen nun diese Möglichkeiten konkret erläutert werden. Zur bildlichen Veranschaulichung wurden die Schritte im Abbildung **??** grafisch dargestellt. Als erstes entscheidet man sich für die Verwendung einer bestimmten t-Norm. Im Beispiel wird die probabilistische Norm angewandt. Als nächstes wird die Differenziertheit der Baumstruktur angegeben, im Beispiel wird eine mittlere Differenziertheit gewählt. Nun berechnet der Algorithmus die Grundrelation a, es ergibt sich ein Wert von 0.55. Anschließend werden die vorläufigen Kantenrelationen durch Multiplikation von a mit den entsprechenden Multiplikatoren gemäß Tabelle 4.3 berechnet. Es ergibt sich ein vorläufiges Resultat wie in Abbildung 4.6 dargestellt.

Als letztes können noch manuelle Korrekturen an den Kantenwerten mittels Veränderungsqualifizierern vorgenommen werden. Diese Veränderungen werden in Abbildung 4.7 grafisch dargestellt. Gestrichelte Linien verdeutlichen den Qualifizierer *lockerer*, während dicke Linien den Qualifizierer *enger* verdeutlichen. Daraus berechnet das System nun die endgültigen Kantenwerte. Im Beispiel ergibt sich daraus Abbildung 4.8 .

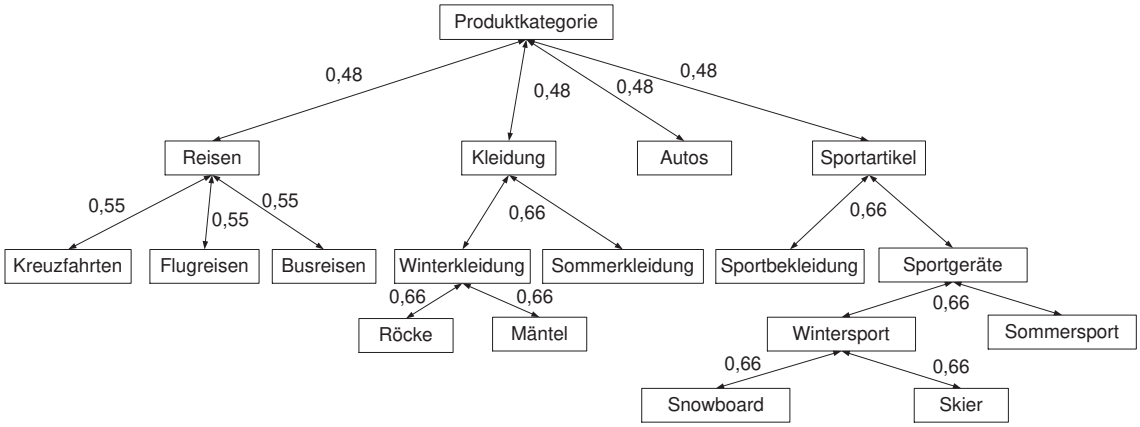

Abbildung 4.6: Algorithmus: automatisch korrigierte Grundrelation

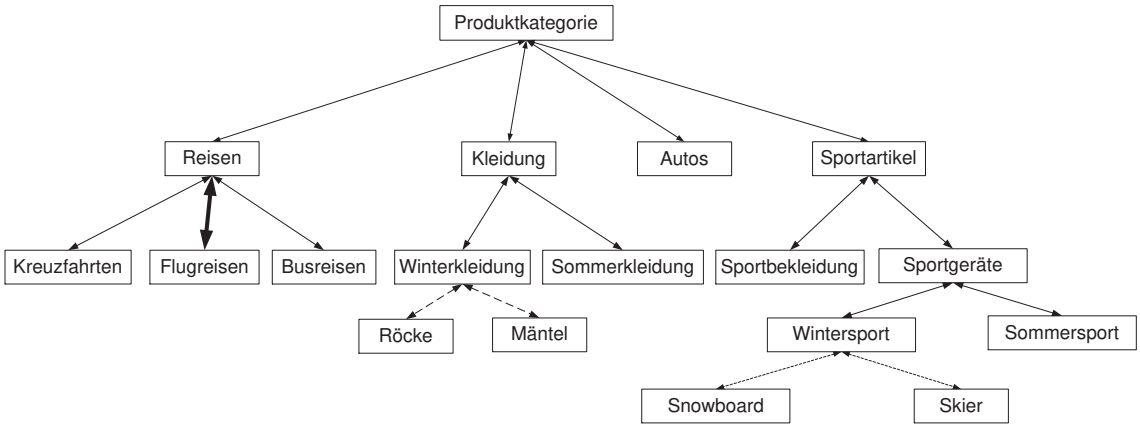

Abbildung 4.7: Algorithmus: Veränderungsqualifizierer

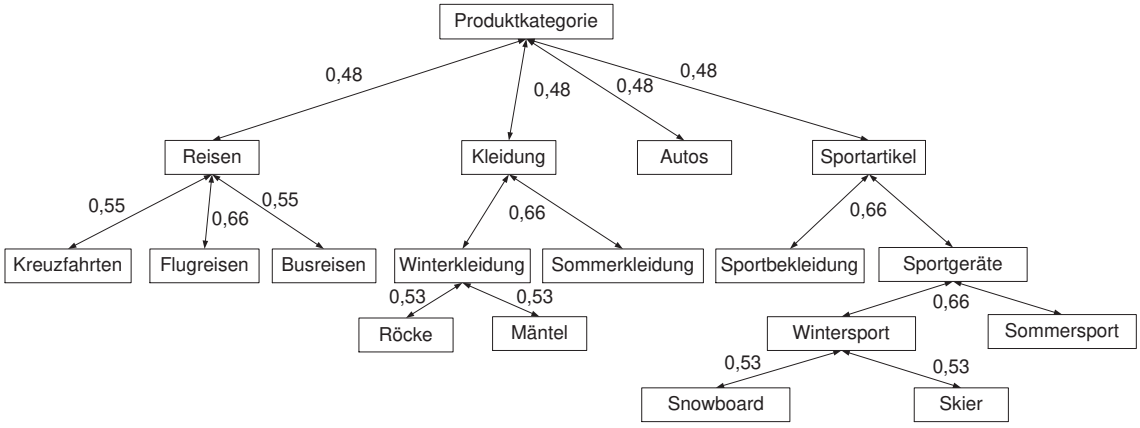

Abbildung 4.8: Algorithmus: Endresultat

Die gleiche Anfrage wie in 4.4 ($m_R(Sportgeraete, Kleidung)$) führt nun zum Ergebnis $0.66 * 0.48 * 1.0 = 0.32$, siehe Abbildung 4.9 . Die Auswertung des Prädikates *is similar to*

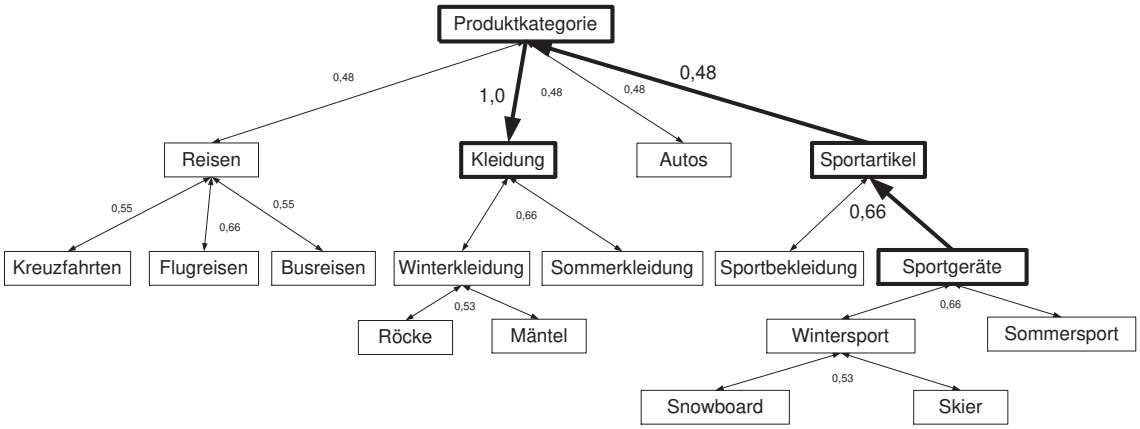

Abbildung 4.9: Algorithmus: Beispielauswertung

führt hier also zum Ergebnis 0.32.

4.6 Implementierungsdetails

Die diskreten Domänen werden in PIA mittels einer Baumstruktur repräsentiert (siehe Abschnitt 2.2.2). Die entsprechende Klasse heisst *DiscreteDomainTree* und beinhaltet Knotenobjekte der Klasse *DomainUserObject* (siehe Abbildung 5.2). Diese wiederum speichert die Kantenwerte in der Klasse *Relation*. Jeder Knoten speichert also die Relationen zum Elternknoten.

Zu Beginn werden die Domänen initialisiert und das eventuell spezifizierte *Differenziertheitsmaß* eingelesen. Daraus werden Differenziertheitsniveau und -grad entsprechend Tabelle 4.2 ermittelt und gespeichert. Zu jeder Domäne werden anschließend die *DiscreteDomainTree* Objekte eingelesen und die entsprechende Struktur erzeugt. Beim Einlesen der Knotenobjekte werden eventuell vorhandene *Veränderungsqualifizierer* eingetragen.

Anschließend wird die *initiateTree* Methode aufgerufen, welche die Grundrelation a berechnet. Diese kann anschließend wie in 4.5.2 beschrieben verändert werden, und der errechnete Relationswert wird in das *Relation* Objekt eines jeden Knoten eingetragen.

Als Resultat liegt eine mit den Vorstellungen des Betreibers entsprechenden Kantenwerten versehene diskrete Domäne vor.

Die *evaluate()* Methode des diskreten Prädikates *is similar to* braucht nun lediglich alle möglichen Pfade zwischen den übergebenen *val* und *args* abzulaufen. Dies geschieht mittels der Methode *getPathFromThisToB()* in *DiscreteDomainTree*. Für jeden Pfad werden jetzt mit der gewählten t-Norm die Relationen wie in 3.3 beschrieben ausgewertet und das jeweilige Relationenprodukt zwischen Anfang und Ende bestimmt. Das größte gefundene Relationenprodukt ist dann das Ergebnis der Prädikatauswertung und wird übergeben.

4.7 Mögliche Verbesserungen

Denkbar wäre eine Erweiterung der Baumstruktur dahingehend, dass man nicht nur direkte Relationen zwischen Elter- und Kinderknoten, sondern auch zwischen weiter entfernten Knoten innerhalb eines Baumes zulässt. Dies ist in der Domäne Produktkategorie in Abbildung 4.2 zum Beispiel wünschenswert, um eine engere Beziehung zwischen *Sportbekleidung* und *Kleidung* darzustellen. Diese Relationen müssten dann vom Katalogbetreiber von Hand eingegeben werden. Ausserdem ist das Finden eines minimalen Relationenproduktes zwischen zwei Knoten dann keine triviale Aufgabe mehr, da nun mehrere Pfade zwischen den Knoten möglich sind.

Ein ähnliches Resultat kann man auch mit den vorhandenen Mitteln erreichen, indem man ganze Teilbäume an mehrere Stellen im Baum einhängt. Im Beispiel der diskreten Domäne Produktkategorie könnte man den Knoten *Sportbekleidung* auch noch unterhalb von *Kleidung* einhängen, um die Verwandtschaft zur *Kleidung* auszudrücken. Auf diesem Wege bläht man zwar die Baumstruktur auf, aber sie bleibt erhalten und lässt sich in der beschriebenen Weise verarbeiten.

Kapitel 5

Numerische Prädikate

In diesem Kapitel werden die numerischen Prädikate untersucht. Ähnlich wie bei den diskreten Prädikaten (siehe Kapitel 4) stellt die Methode *evaluate()* die eigentliche Funktionalität der Prädikate zur Verfügung. Die übergebenen Parameter *args[]* und *val* (siehe Abschnitt 2.4) stammen alle aus derselben numerischen Domäne. Ganz allgemein soll die *evaluate()* Methode der numerischen Prädikate nun Aussagen über *val* allein (z.B. ist *val* besonders groß?) oder über *val* in Relation zu Werten aus *args[]* (z.B. ist *val* größer als *args[0]*?) treffen. Von den implementierten Prädikaten (siehe Tabelle 5.1) sind bis auf *is exactly* alle unscharf.

Name des Prädikates	*argsSignature*
is exactly	(NumericDomain)
is average	()
is between	(NumericDomain, NumericDomain)
is large	()
is small	()
is larger than	(NumericDomain)
is smaller than	(NumericDomain)
is approximately	(NumericDomain)
is as large as possible	()
is as small as possible	()

Tabelle 5.1: numerische Prädikate

5.1 Anforderungen

Zur Motivation soll die Problemstellung an einem kleinen Beispiel verdeutlicht werden. In unserem Beispielkatalog befinden sich neun Produkte (drei Autos und sechs Reisen). Ein Kunde sucht nach einem billigen Auto. Dies wird im PIA-Modell durch eine *Alternative* repräsentiert. Diese wiederum besteht aus mehreren Kriterien (siehe Abschnitt 2.3). Er

erzeugt sich zunächst ein diskretes Kriterium

```
(Produktkategorie, ist, Auto, normal)
```

Anschliessend formuliert er ein numerisches Kriterium

```
(Preis, ist klein, , normal)
```

Das diskrete Kriterium stellt kein Problem dar, da die Auswertung des Prädikates *is exactly* für alle drei Autos den Wert $1,0$ ergeben wird.

Die Aufgabe des numerischen Prädikates *ist klein* besteht nun darin, eine Aussage über den Grad der Zugehörigkeit eines Produktpreises zur unscharfen Menge der kleinen Preise, zu treffen. Die allem folgenden zugrundeliegende Prämisse lautet also, dass jeweils ein numerisches Prädikat eine unscharfe Menge repräsentiert. Daraus ergeben sich zwei Anforderungen, die im folgenden gelöst werden müssen.

Zunächst ist ein geeigneter Grundbereich zu finden, über dem die unscharfe Menge definiert wird. Der gefundene Grundbereich soll dabei eine möglichst hohe Differenziertheit aufweisen, worauf im folgenden noch detailiert eingegangen wird.

Als zweites Problem muss auf einem gefundenen Grundbereich und einer gegebenen linguistischen Formulierung der unscharfen Menge eine Zugehörigkeitsfunktion aufgestellt werden, welche die entsprechende Menge repräsentiert.

5.1.1 Grundbereich

Die erste zu lösende Anforderung bestand im Finden eines geeigneten Grundbereiches. Man könnte auch formulieren, dass man die Wertemenge sucht, relativ zu der ein bestimmter Wert eingeschätzt werden soll.

Zunächst soll ein erster intuitiver Ansatz und die sich daraus ergebenden Probleme aufgezeigt werden. Der Ansatz besteht darin, die Gesamtheit der Produkte des Kataloges als Grundbereich heranzuziehen. Man bestimmt also beim Laden des Kataloges Mittelwert und Standardabweichung aller numerischen Attribute, und stellt anhand dieser die Zugehörigkeitsfunktion auf. Die Verwendung von Mittelwert und Standardabweichung zum Aufstellen der Zugehörigkeitsfunktion wird im folgenden Abschnitt 5.1.2 motiviert. Im Beispiel der Suche nach einem billigen Auto werden also Mittelwert und Standardabeichung der Preise aller neun Produkte des Kataloges benutzt. Der Mittelwert der Preise beträgt 13300 DM, und die Standardabweichung 16300 DM. Mit der Zugehörigkeitsfunktion wie oben ergibt sich für alle Autos ein Grad der Zugehörigkeit von 0.0. Dies entspricht nicht dem erwarteten Ergebnis. Die Ursache dafür liegt im zu großen Grundbereich, da man auf diese Weise die unscharfe Menge der billigen Produkte und nicht der billigen Autos auswertet.

Ein verbesserter Ansatz muss also die formulierten diskreten Kriterien einer Alternative berücksichtigen. Man muss im Beispiel also zunächst alle Autos des Kataloges finden und

erst dann Mittelwert und Standardabweichung der Preise bestimmen. Im Beispielkatalog ergibt sich ein Mittelwert von 35700 DM, und eine Standardabweichung von 6800 DM. Wenn man nun das Prädikat auswertet, ergibt sich für das billigste Auto ein Grad der Zugehörigkeit von 1.0, das teuerste Auto wird zu 0.0 ermittelt und beim Auto mit dem mittleren Preis ergibt sich ein Wert von 0.4.

Zur Motivation des Ansatzes kann zusätzlich auch gesagt werden, dass der Kunde eine Alternative als Ganzes im Sinn hat. Für das System muss sie jedoch in einzelne Kriterien zerlegt werden. Bei der Auswertung wird die Alternative auf diesem Wege wieder zusammengesetzt. Die Prämisse lautet also, dass der Kunde durch die formulierten diskreten Kriterien den gewünschten Grundbereich formuliert.

Wie man jedoch erkennen kann bringt der zweite Ansatz noch einige Probleme mit sich. Der Katalog muss zweimal durchlaufen werden, im ersten Durchlauf werden Mittelwert und Standardabweichung zum Aufstellen der Zugehörigkeitsfunktion bestimmt und erst im zweiten Durchlauf findet die eigentliche Auswertung des Prädikates statt. Wenn diskrete Prädikate vorhanden sind und sich gegenseitig widersprechen (keine Schnittmengen ergeben), ergeben sich keine Werte für Mittelwert und Standardabweichung. Für diesen Fall müsste man - ohne einen plötzlichen Genauigkeitsverlust zu erleiden - einen zweiten Durchlauf mit leicht veränderten diskreten Kriterien durchführen, allerdings ebenfalls ohne Erfolgsgarantie. Abschliessend kann man sagen, dass eine solche Herangehensweise mit großen Problemen verbunden und somit nicht praktikabel ist.

Im Rahmen dieser Arbeit wurde eine alternative Methode entwickelt, bei der die Nachteile der o.a. Herangehensweise entfallen. Die Grundidee besteht darin, bereits beim initialen Einlesen des Kataloges die später beim Auswerten benötigten Informationen anzulegen. Allgemeiner gesprochen werden Metainformationen angelegt, welche die Suche erleichtern. Dies ist von der Herangehensweise z.B. mit dem Erzeugen eines invertierten Index bei einer relationalen Datenbank [10] oder in Information Retrieval Systemen [11] vergleichbar.

Es werden bereits beim Einlesen des Kataloges Mittelwerte und Standardabweichungen gesammelt, welche später nur noch kombiniert werden müssen. Auf diese Art und Weise wird die Effizienz des ersten Ansatzes mit der Genauigkeit des zweiten Ansatzes kombiniert.

Als Nachteil ergibt sich der leicht erhöhte Speicherbedarf des Systems.

5.1.2 Zugehörigkeitsfunktion

Im folgenden werden ein Grundbereich von Werten und eine linguistische Formulierung eines Prädikates aus der Tabelle 5.1 als gegeben vorausgesetzt. Die Aufgabe besteht nun darin, eine Zugehörigkeitsfunktion wie in Abschnitt 3.1 beschrieben aufzustellen. Wie dort dargestellt, wurden als Zugehörigkeitsfunktionen drei verschiedene lineare konvexe Funktionen (L-Funktion, Γ-Funktion und Trapez-Funktion) gewählt. Die verbleibende Fragestellung besteht im Finden der maximal vier Parameter, durch die die drei Funktionen

beschrieben werden.

Es stellt sich also allgemein das Problem, einer gegebenen Menge von numerischen Werten eine der Semantik des Prädikates angemessene Zugehörigkeitsfunktion zuzuordnen. Zu diesem Zwecke bietet es sich an, Mittelwert und Standardabweichung der gegebenen Werte heranzuziehen. Diese Parameter haben den Vorteil, dass sie sich wie in Abschnitt 5.2 gezeigt wird rekursiv ermittlen lassen. Der Mittelwert μ wird für die Prädikate *is average, is large, is small, is as large as possible* und *is as small as possible* benötigt. Die Standardabweichung δ wird als Maß für die Unschärfe herangezogen.

Die Zugehörigkeitsfunktion *is small* des Beispieles ist eine L-Funktion, und die Stützstellen berechnen sich wie folgt:

$$a_1 = \mu - \delta$$

$$a_2 = \mu.$$

Mittelwert und Standardabweichung lassen sich rekursiv ermitteln, was in Abschnitt 5.2 gezeigt wird.

5.2 Mathematische Grundlagen

In diesem Abschnitt werden die mathematischen Grundlagen, die zum Sammeln und Kombinieren von Mittelwerten und Standardabweichungen notwendig sind, vorgestellt.

Der Mittelwert über einer Menge von N Werten ist definiert als

$$\mu_N = \frac{1}{N} \sum_{i=1}^{N} v_i, \tag{5.1}$$

wobei v_i die Werte sind. Die Standardabweichung δ_N von N Werten ist definiert als

$$\delta_N = \sqrt{\frac{1}{N} \sum_{i=1}^{N} (v_i - \mu)^2} = \sqrt{\delta_N^2}, \tag{5.2}$$

wobei δ_N^2 die Varianz ist. Nun besteht das Problem, dass man unter Verwendung von 5.1 und 5.2 die Wertemenge zwei mal durchlaufen müsste, um μ und δ zu bestimmen. Dies ist jedoch in diesem Fall nicht wünschenswert. Als Resultat möchte man μ_N und δ_N als Funktionen von μ_{N-1} und δ_{N-1} und v_N berechnen können.

Der Mittelwert lässt sich sehr einfach rekursiv berechnen, es gilt nämlich

$$\mu_N = \frac{1}{N} \left(\sum_{i=1}^{N-1} v_i + v_N \right), \tag{5.3}$$

worin sich $\sum_{i=1}^{N-1} v_i$ durch $(N-1)\mu_{N-1}$ ersetzen lässt. Als Resultat erhält man:

$$\mu_N = \frac{1}{N}(v_N - \mu_{N-1}) + \mu_{N-1}. \tag{5.4}$$

Für die Varianz δ^2 findet man ebenfalls eine rekursive Formel, indem man das nichtzentrale Moment 2. Ordnung m_N mit

$$m_N = \frac{1}{N}\sum_{i=1}^{N} v_i^2 \tag{5.5}$$

einbezieht. Es gilt

$$\delta_N^2 = m_N - \mu_N^2. \tag{5.6}$$

Beweis:

$$\delta_N^2 = \frac{1}{N}\sum(v_i - \mu_N)^2 = \frac{1}{N}\sum(v_i^2 - 2v_i\mu_N + \mu_N^2)$$

$$= \frac{1}{N}(\sum v_i^2 - 2\mu_N \sum v_i + N\mu_N^2)$$

Hierin lässt sich $\sum v_i^2$ durch Nm_N und $\sum v_i$ durch $N\mu_N$ ersetzen:

$$\delta_N^2 = \frac{1}{N}(Nm_N - 2N\mu_N^2 + N\mu_N^2) = m_N - \mu_N^2.$$

Nun lässt sich zeigen, dass man m ähnlich wie μ rekursiv bestimmen kann, es gilt

$$m_N = \frac{1}{N}(v_N^2 - m_{N-1}) + m_{N-1}. \tag{5.7}$$

Demzufolge kann man in jedem Schritt μ_N und m_N rekursiv bestimmen und anschließend aus beiden δ_N^2 und anschliessend δ_N berechnen.

Ein weiteres Problem entsteht, wenn man zwei Mengen von Werten hat, getrennt μ_{N_1}, δ_{N_1} und μ_{N_2}, δ_{N_2} berechnet und an $\mu_{N_1+N_2}$ und $\delta_{N_1+N_2}$ der vereinigten Wertemenge interessiert ist. Für den Mittelwert gilt:

$$\mu_{N_1+N_2} = \frac{1}{N_1+N_2}(\mu_{N_1}N_1 + \mu_{N_2}N_2). \tag{5.8}$$

Für das nichtzentrale Moment 2. Ordnung $m_{N_1+N_2}$ gilt:

$$m_{N_1+N_2} = \frac{1}{N_1+N_2}(m_{N_1}N_1 + m_{N_2}N_2). \tag{5.9}$$

Nun lässt sich mittels 5.6 wiederum die Varianz $\delta_{N_1+N_2}^2$, und anschließend die Standardabweichung $\delta_{N_1+N_2}$ berechnen.

5.3 Definition des *Cluster*-Begriffes

Wie bereits weiter oben in diesem Kapitel festgestellt wird ein Produkt durch seine diskreten Eigenschaften in einen Katalog eingeordnet. Da es in PIA kein vorgeschriebenes Schema gibt, welches gewisse Eigenschaften eines Produktes erzwingt, existiert von vornherein keine äussere Ordnungsstruktur. So ist z.B. für ein Produkt keine Belegung des diskreten Attributes *Produktkategorie* o.ä. erforderlich, nicht einmal die Existenz eines solchen Attributes wird vorausgesetzt. Natürlich ist eine solche Eigenschaft für alle Produkte sinnvoll und wünschenswert, sie wird jedoch nicht erzwungen. Es ist jedoch einleuchtend, dass die diskreten Attribute die Aufgabe innehaben, eine Ordnungsstruktur zu schaffen. Ähnliche Produkte besitzen die gleiche bzw. ähnliche Kombinationen von diskreten Eigenschaften.

Eine solche Kombination von diskreten Eigenschaften beschreibt einen sogenannten *Cluster*. Alle Produkte mit der selben Kombination diskreter Eigenschaften bilden eine homogene Gruppe, so dass der Begriff Cluster hier als angemessen erscheint [5].

In Abbildung 5.1 sind die diskreten Eigenschaften einer Vier-Sterne Reise nach Italien gemäß Tabelle 5.2 grafisch veranschaulicht.

diskretes Attribut	Value
Produktkategorie	Reisen
Reisekategorie	vier
Reiseziel	Italien

Tabelle 5.2: Vier-Sterne Reise nach Italien

Diese Kombination beschreibt einen Cluster. Da ein Wert eines diskreten Attributes - ein *DiscreteValue* - durch eine nichtleere Menge von Knoten einer Baumstruktur bestimmt wird (siehe Abschnitt 2.2.2), wird ein Cluster durch eine Menge diskreter Attribute und pro diskretem Attribut durch eine Menge von Knotenobjekten repräsentiert.

Der Begriff des Clusters wird im folgenden eine zentrale Bedeutung erlangen, da mit seiner Hilfe die Kategorisierung der Produkte eines Kataloges möglich wird. Zur Verwaltung der Cluster wurden zwei Klassen erstellt: *EntryCluster* und *QueryCluster* (siehe Abbildung 5.2). Mittels EntryCluster werden die Cluster erstellt und Informationen eingetragen. Durch QueryCluster ist es möglich, auf einen Cluster zuzugreifen und die angelegten Informationen zu erlangen. Die Unterteilung in zwei verschiedene Klassen ist hier notwendig, da sich die Clusterrepräsentierung für beide Aufgaben intern geringfügig unterscheidet.

Die in einen Cluster eingetragene Clusterinformation besteht aus Mittelwert und Standardabweichung der im Cluster enthaltenen Produkte und wird in einem *ClusterCharacteristic*-Objekt gespeichert (siehe Abbildung 5.2).

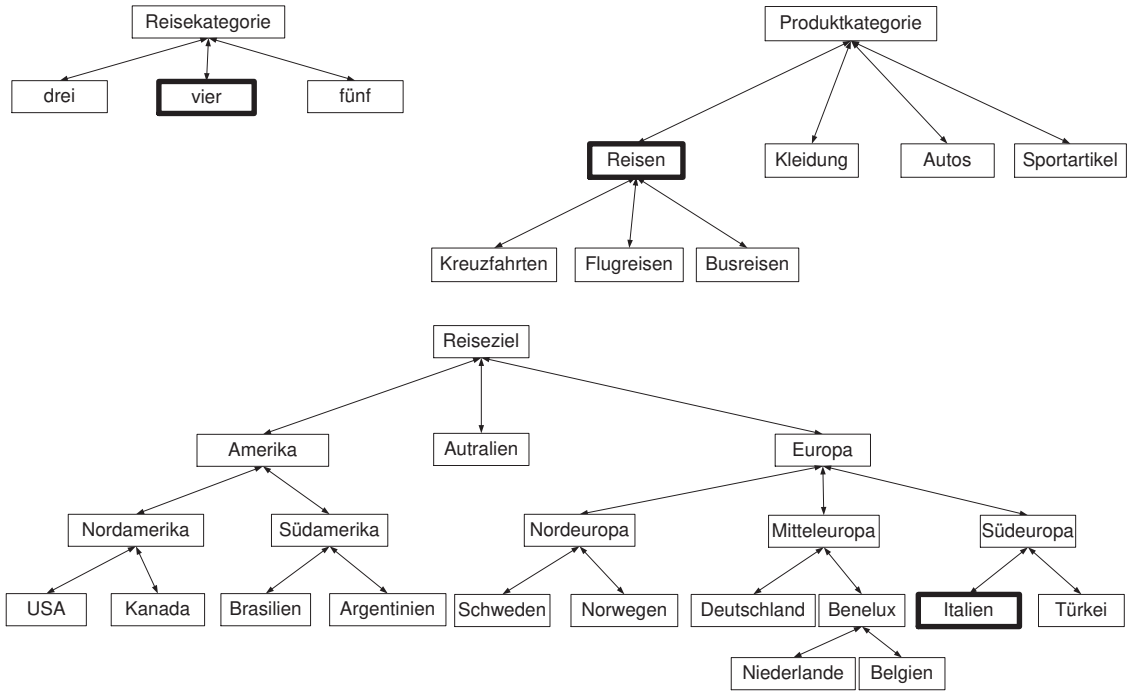

Abbildung 5.1: Cluster

5.4 Repräsentation der Cluster

Die Informationen über die Produke innerhalb eines Clusters werden durch ein *ClusterCharacteristic* Objekt repräsentiert. Ein *ClusterCharacteristic* Objekt wiederum besteht aus mehreren *AttributeDistribution* Objekten und einem Integerwert namens *arity*, welcher die Anzahl der verwendeten diskreten Eigenschaften speichert.

Für jedes numerische Attribut, das von mindestens einem im Cluster enthaltenen Produkt belegt ist, speichert ein *AttributeDistribution* Objekt Mittelwert, Standardabweichung und die Anzahl der repräsentierten Produkte. Mittels der Methoden *addDistributionValue()* und *removeDistributionValue()* können numerische Werte des entsprechenden Attributes jederzeit hinzugefügt und entfernt werden. Mittelwert, Standardabweichung und Anzahl werden gemäß den in Abschnitt 5.2 vorgestellten Berechnungsvorschriften aktualisiert.

Ein diskretes Attribut wird in PIA durch eine Baumstruktur (*DiscreteAttributeTree*), ähnlich der einer diskreten Domäne (vergleiche 4.6), repräsentiert. Der einzige Unterschied besteht in der Art des Knotenobjektes, welches im Falle einer diskreten Domäne ein *AttributeUserObject* ist und seinerseits mehrere *ClusterCharacteristic* Objekte speichert, die dort entsprechend abgelegt sind.

Zusammenfassend beinhaltet ein *ClusterCharacteristic* Objekt die Informationen über die im Cluster enthaltenen Produkte. Die *ClusterCharacteristic* Objekte wiederum sind in den dem Cluster entsprechenden Knotenobjekten der diskreten Attribute gespeichert. Ein

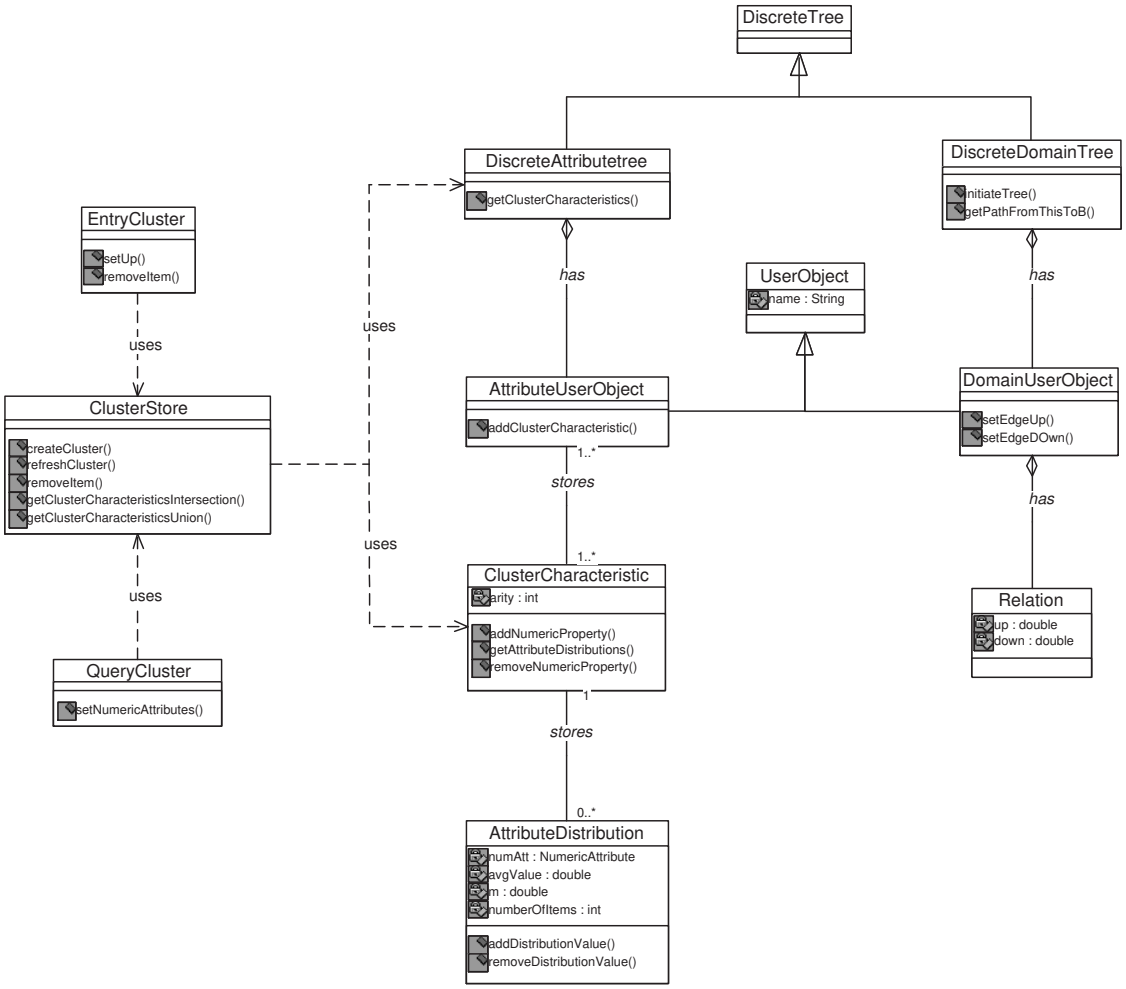

Abbildung 5.2: Klassendiagramm Cluster

Zugriff auf einen Cluster ist also nur über die diskreten Domänen möglich.

Um das Clustermanagement zu vereinfachen und übersichtlicher zu gestalten, wurde die Klasse *ClusterStore* angelegt, welche über statische Methoden den Klassen *QueryCluster* und *EntryCluster* die notwendige Funktionalität zur Verfügung stellt. Die Methoden sind im Detail:

- *createCluster(NumericProperties : Vector, discreteAttTrees : HashSet, arity : int)*; ein neues *ClusterCharacteristic* Objekt wird in die durch *discreteAttTrees* repräsentierten Knoten eingetragen

- *refreshCluster(NumericProperties : Vector, ClusterChar : ClusterCharacteristic)*; eine neue numerische Eigenschaft wird in *ClusterChar* eingetragen

- *getClusterCharactericticsIntersection(DiscreteAttributeTrees : HashSet) : HashSet*; liefert den Durchschnitt der in *DiscreteAttributeTrees* enthaltenen *ClusterCharacteristic* Objekten

- *getClusterCharacteristicsUnion(DiscreteAttributeTrees : HashSet) : HashSet*; liefert die Vereinigung der in *DiscreteAttributeTrees* enthaltenen *ClusterCharacteristic* Objekten

5.5 Anlegen der Clusterinformationen

Nachdem das PIA-System die Domänen und Attribute geladen hat, werden anschließend die Produkte initialisiert. Dabei werden pro Produkt getrennt die numerischen und diskreten Eigenschaften gesammelt und mit den *DiscreteProperties* ein *EntryCluster* Objekt erzeugt. Innerhalb von EntryCluster wird der Cluster durch ein *HashMap* mit dem diskreten Attribut als *Key* und einem *HashSet* von *DiscreteAttributeTree* Objekten als *value* repräsentiert.

Anschließend werden der *setUp()*-Methode von QueryCluster die gesammelten numerischen Eigenschaften übergeben. Innerhalb dieser Methode wird als erstes untersucht, ob in dem repräsentierten Cluster bereits Eintragungen existieren (siehe Abbildung 5.3). Zu diesem Zweck wird mittels *ClusterStore.getClusterCharacteristicsIntersection()* der Durchschnitt über alle bereits eingetragenen *ClusterCharacteristic* Objekte gebildet. Ist der Durchschnitt leer, wurde noch kein Produkt in diesen Cluster eingetragen und ein neues *ClusterCharacteristic* Objekt muss erzeugt werden. Besteht der Durchschnitt aus mehreren *ClusterCharacteristic* Objekten, muss jeweils überprüft werden, ob *arity* des neuen Clusters mit dem existierenden übereinstimmt. Ist das der Fall, wird das existierende *ClusterCharacteristic* Objekt aktualisiert und die numerischen Eigenschaften werden mittels der *addNumericProperty()* Methode eingetragen.

Im Beispielkatalog existieren, wie bereits erwähnt, neun Produkte, welche drei verschiedene Cluster belegen (siehe Tabelle 5.3). In den jeweiligen Clustern existieren *Attribute-Distributions* wie Tabelle 5.4 dargestellt.

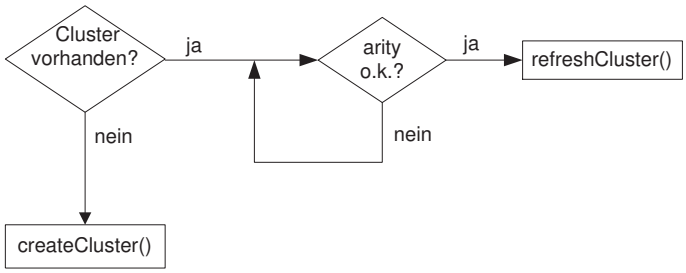

Abbildung 5.3: *EntryCluster.setUp()*

diskretes Attribut	#1	#2	#3
Produktkategorie	Autos	Reisen	Reisen
Reisekategorie		vier	
Reiseziel		Italien	Italien

Tabelle 5.3: alle Cluster

	#1	#2	#3
Preis	ja	ja	ja
Stornokosten	nein	ja	nein
Höchstgeschwindigkeit	ja	nein	nein

Tabelle 5.4: in den Clustern vorkommende Verteilungen

5.6 Auslesen der Clusterinformationen

Als nächstes wirft sich die Frage auf, wie auf die eingetragenen Clusterinformationen in Gestalt der *ClusterCharacteristic*-Objekte sinnvoll zugegriffen werden kann. Diese Aufgabe stellt sich, sobald ein Benutzer eine Alternative formuliert hat (vergleiche 2.3) und nun die *score()* Methode der Alternative ausgewertet werden soll. Beim Berechnen der *score()* Funktion einer Alternative werden alle *score()* Methoden der Kriterien der Alternative ausgewertet und anschließend wie in 3.2 beschrieben zusammengefasst. Im folgenden wird davon ausgegangen, dass die vom Kunden formulierte Alternative sowohl diskrete als auch numerische Kriterien enthält.

Dabei beschreiben die diskreten Kriterien der Alternative einen Cluster (im folgenden *Anfangscluster* genannt). An Produkten aus genau diesem oder einem nahe diesem gelegenen Cluster ist der Kunde interessiert. Demzufolge möchte er die numerischen Kriterien relativ zu Produkten aus diesem oder nahe gelegenen Clustern ausgewertet wissen. Um die

entsprechenden Zugehörigkeisfunktionen wie in Kapitel 5 besprochen erstellen zu können, werden also Clusterinformationen aus einem Cluster nahe dem Anfangscluster gesucht.

Zur Veranschaulichung der Aufgabenstellung dient im folgenden ein kleines Beispiel. Der einfachste Fall ist der, dass ein Kunde mit seinen diskreten Kriterien einen existierenden Cluster spezifiziert und numerische Kriterien für im Cluster vorkommende numerische Attribute aufgestellt hat. Als Beispiel würde eine Suche nach einer billigen 4-Sterne Reise nach Italien sofort zu denVerteilungen des Clusters #2 führen. Da im Cluster #2 Verteilungen für das numerische Attribut Preise enthalten sind, werden diese sofort genutzt. Sucht der Kunde z.B. allgemeiner nach einer billigen Reise nach Italien, werden die Preis-Verteilungen von Cluster #2 und #3 verschmolzen und für die Zugehörigkeitsfunktion verwandt. Sucht der Kunde aber nach einer billigen Reise in die Türkei, so existiert ein solcher Cluster nicht. Zur Lösung dieses Problems wird der vom Kunden spezifizierte Anfangscluster (Reise, Türkei) zu (Reise, Südeuropa) expandiert. Das führt zum Ergebnis der Preisverteilungen aus Cluster #2 und #3.

Die Umsetzung dieser im Beispiel verdeutlichten Mechanismen wird im folgenden allgemein dargestellt. Die gewünschte Funktionalität liefert die Klasse *QueryCluster*, beziehungsweise ihre Methode *setNumericAttributes()*.

Zunächst wird eine Instanz der Klasse *QueryCluster* mit den diskreten Eigenschaften erzeugt. Dieses Objekt beschreibt jetzt den Anfangscluster. Innerhalb von *QueryCluster* wird ein Cluster ähnlich wie in *EntryCluster* repräsentiert. Der Unterschied besteht darin, dass zu einem Elternknoten alle Kindsknoten mitgespeichert werden. Zu jedem diskreten Attribut existiert also eine weitere *HashMap* mit den Wurzeln der Unterbäume als *Key* und einem *HashSet* jeweils aller Kindsknoten als *Value*. Es werden also sozusagen alle *Untercluster* mitrepräsentiert.

Nach Erzeugung des Objektes wird die *setNumericAttribute()*-Methode aufgerufen. Übergeben werden die numerischen Attribute, für die eine Prädikatauswertung durchgeführt werden soll, für die also eine Verteilung (Mittelwert und Standardabweichung) gesucht wird. Der Ablauf ist in Abbildung 5.4 dargestellt. Als erstes wird im Anfangscluster plus mitrepräsentierten Unterclustern nach entsprechenden *AttributeDistributions* gesucht. Zu diesem Zweck wird pro diskreter Domäne die Vereinigung der eingetragenen *ClusterCharacteristic* Objekte gebildet. Anschließend wird der Durchschnitt über alle Domänen berechnet. Im Resultat sind alle Verteilungen aller in Unterclustern eingetragenen Produkte enthalten. Diese werden jetzt nach den gesuchten numerischen Attributen durchsucht und gegebenenfalls vereinigt. Wenn für jedes gesuchte Attribut Verteilungen vorliegen, werden diese mittels *setResult()* eingetragen. Wenn nur für einen Teil der gesuchten Attribute Verteilungen gefunden werden, werden die gefundenen übernommen, und der durch *QueryCluster* repräsentierte Cluster wird durch *expand()* erweitert. Dabei wird in der Repräsentation ein Wurzelknoten durch seinen Elternknoten ersetzt und die Kindsknoten entsprechend angepasst. Anschließend wird eine neue Suche durchgeführt und eventuell gefundene Verteilungen übernommen.

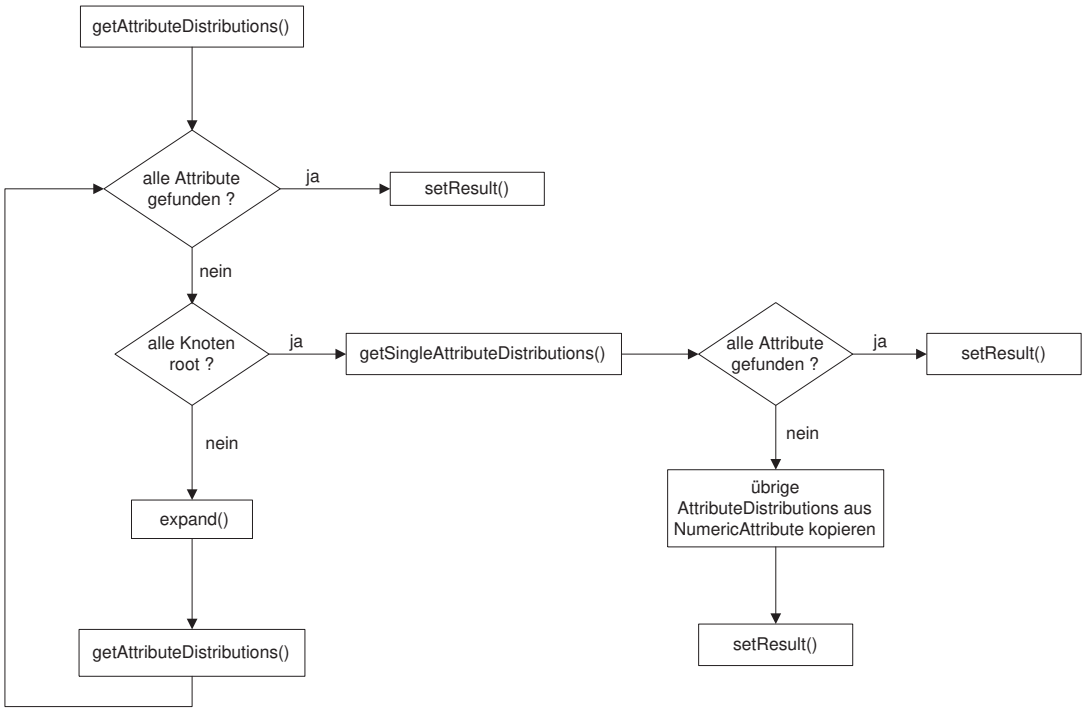

Abbildung 5.4: Ablaufplan für *QueryCluster.setNumericAttributes()*

Irgendwann ist eine Erweiterung von *QueryCluster* nicht mehr möglich, da alle Knoten bereits die Wurzeln der gesamten Baumstruktur sind. Jetzt wird in allen diskreten Attributen einzeln nach den gesuchten numerischen Attributen gesucht und die Resultate vereinigt. Dies kann insbesondere dann erforderlich werden, wenn zuviele diskrete Eigenschaften für zuviele diskrete Attribute aufgestellt worden sind. Eventuell gefundene Verteilungen werden wiederum übernommen. Falls auch jetzt noch nicht für alle numerischen Attribute Verteilungen gefunden wurden, werden die Verteilungen der gesamten Attribute übernommen, welche bereits beim Einlesen der Produkte ermittelt werden.

Auf diese Weise ist sichergestellt, dass auf jeden Fall ein Resultat ermittelt wird, welches die maximal mögliche Genauigkeit aufweist.

Die gefundenen Mittelwerte und Standardabweichungen werden mittels der Methode *setResult()* in die entsprechenden numerischen Attribute eingetragen, und können später von den numerischen Prädikaten zugegriffen werden.

5.7 Arbeitsweise der numerischen Prädikate

Zur Cluster-bezogenen Auswertung der numerischen Prädikate wird jetzt lediglich noch die Implementation ihrer Zugehörigkeitsfunktion benötigt. Dieser Abschnitt liefert zwei mögliche Klassifikationen der numerischen Prädikate.

5.7.1 Klassifikation nach Gewinnung der Stützstellen

Die numerischen Prädikate erben wie jedes Prädikat von *Predicate* die Methode *evaluate (Attribute att, AbstractValue val, AbstractValue[] args) : double*. Je nach Prädikat werden mehrere Argumente in *args* übergeben (siehe 2.4). Im folgenden soll eine Einteilung der Prädikate in vier verschiedene Gruppen vorgenommen werden.

Die Methode *evaluate* eines numerischen Fuzzy-Prädikates wertet eine Zugehörigkeitsfunktion einer unscharfen Menge an der Stelle *val* aus. Diese Zugehörigkeitsfunktion lässt sich auf verschiedene Arten und Weisen bilden. Als Beispiel sei die Zugehörigkeitsfunktion des Prädikates *is between* angeführt. Hierfür eignet sich die in Abschnitt 3.1 vorgestellte Trapezfunktion. Es gibt vier verschiedene Arten, die Stützstellen der Zugehörigkeitsfunktion zu erlangen. Diese Orientierungspunkte der Funktion (im Beispiel $a1$ bis $a4$, siehe Abbildung 5.5) können sowohl absolute Werte, gewonnen aus *args*, als auch relative Wer-

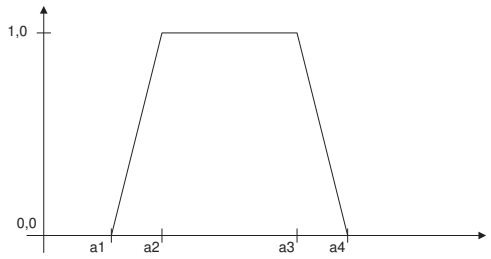

Abbildung 5.5: Graph der Trapez-Funktion

te, gewonnen aus der Verteilung des Clusters (meist Mittelwert) sein. Als zweites Unterscheidungskriterium können die Abweichungen vom Orientierungspunkt sowohl absolut angegeben werden, als auch relativ zur Standardabweichung. Daraus ergeben sich vier Gruppen von Prädikaten:

1. absolut angegebene Orientierungspunkte und absolut angegebene Toleranzen
2. absolut angegebene Orientierungspunkte und relativ zur Standardabweichung definierte Toleranzen
3. relativ zur Verteilung des Clusters definierte Orientierungspunkte und absolut angegebene Toleranzen
4. relative Orientierungspunkte und realtive Toleranzen

In die 1. Gruppe fällt nur das numerische Prädikat *is exactly*, welches allerdings kein Fuzzy-Prädikat ist.

In die 2. Gruppe fallen die Prädikate *is approximately*, *is between*, *is larger than* und *is smaller than*. Zur Beschreibung der Zugehörigkeitsfunktion sind zusätzlich zu den Verteilungsdaten und *args[]* Parameterwerte des Prädikates notwendig. Es ergeben sich die

Prädikat	*approximately*	*between*	*larger than*	*smaller than*
Funktionstyp	Trapez	Trapez	Γ	L
Parameter	D_{min}, D_{max}	D	D	D
$a1$	$args[0] - D_{max}\delta$	$args[0] - D\delta$	$args[0] - D\delta$	$args[0]$
$a2$	$args[0] - D_{min}\delta$	$args[0]$	$args[0]$	$args[0] + D\delta$
$a3$	$args[1] + D_{min}\delta$	$args[1]$		
$a4$	$args[1] + D_{max}\delta$	$args[1] + D\delta$		

Tabelle 5.5: Gruppe 2

in Tabelle 5.5 dargelegten Stützstellen. Die Parameter D_{min}, D_{max} bzw. D stellen Einflussmöglichkeiten bezüglich des Grades der Unschärfe der aufgestellten Zugehörigkeitsfunktionen dar.

Die 3. Gruppe ist leer, während die Prädikate *is average*, *is large*, *is small*, *is as large as possible* und *is as small as possible* der 4. Gruppe angehören. Ähnlich wie bei den Prädikaten der 2. Gruppe sind zwei zusätzliche Parameter notwendig. Die Zusammenhänge sind in Tabelle 5.6 dargestellt.

Prädikat	*average*	*large*	*small*	*large as possible*	*small as possible*
Funktionstyp	Trapez	Γ	L	Γ	L
Parameter	D_{min}, D_{max}	D_1, D_2	D_1, D_2	D_1, D_2	D_1, D_2
$a1$	$\mu - D_{max}\delta$	$\mu + D_1\delta$	$\mu - D_1\delta$	$\mu + D_1\delta$	$\mu - D_1\delta$
$a2$	$\mu - D_{min}\delta$	$\mu + D_2\delta$	$\mu - D_2\delta$	$\mu + D_2\delta$	$\mu - D_2\delta$
$a3$	$\mu + D_{min}\delta$				
$a4$	$\mu + D_{max}\delta$				

Tabelle 5.6: Gruppe 4

Parameter D_{min}, D_{max}, D_1, D_2 und D

Die Parameter dienen zur Feineinstellung der Zugehörigkeitsfunktionen der numerischen Prädikate. Sie bestimmen den Grad der Unschärfe des jeweiligen Prädikates. Alle Parameter sind positiv und werden wie in Tabelle 5.5 und 5.6 dargestellt mit der jeweiligen Standardabweichung mulitipliziert.

Die Parameter D_{min} und D_{max} werden bei Trapezfunktionen um einen Punkt herum angewendet (*approximately*, *average*). D_{min} beschreibt den Abstand zwischen dem Orientierungspunkt ($args[0]$ und $args[1]$ bzw μ) und $a2$ bzw. $a3$ (siehe Abbildung 3.3). D_{max} dementsprechend beschreibt den Abstand zwischen dem Orientierungspunkt und $a1$ bzw. $a4$.

Die Parameter D_1 und D_2 kommen bei L- und Γ-Funktionen, bei denen der Mittelwert μ als Orientierungspunkt verwendet wird zum Tragen (*large, small, large as possible, small as possible*). D_1 beschreibt den Abstand zwischen dem Orientierungspunkt ($args[1]$ bzw μ) und $a1$ (siehe Abbildung 3.1 und 3.2), und D_2 beschreibt den Abstand zwischen dem

Orientierungspunkt und $a2$.

Der Parameter D kommt bei L- und Γ-Funktionen, bei denen das übergebene Argument $args[0]$ als Orientierungspunkt verwendet wird zum Tragen (*between, larger than, smaller than*). D beschreibt den Abstand zwischen dem Orientierungspunkt $args[0]$ und $a2$ (siehe Abbildung 3.1 und 3.2).

5.7.2 Klassifikation nach Zugehörigkeitsfunktion

Eine weitere Unterteilung ist natürlich hinsichtlich der verwendeten Zugehörigkeitsfunktion möglich. Es wurden drei Typen verwendet (siehe 3.2, 3.1 und 3.3). Diese Zugehörigkeitsfunktionen wurden durch drei Klassen *MemberGamma*, *MemberL* und *MemberTrapezium* implementiert, welche ihrerseits von *FuzzyGraph* erben (siehe Abbildung 5.6).

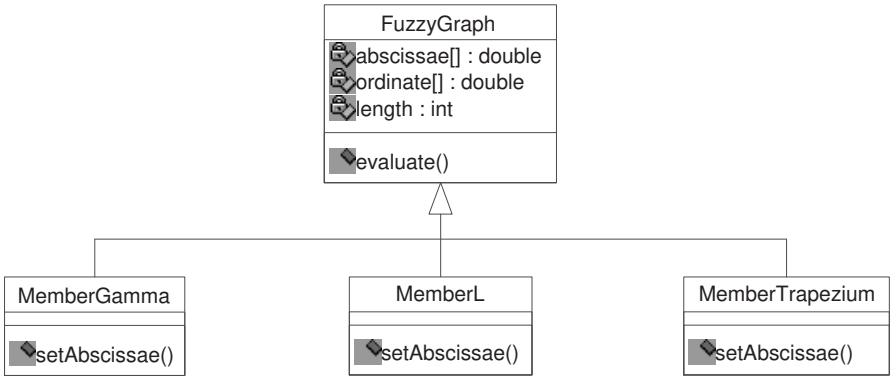

Abbildung 5.6: *FuzzyGraph* Objekte

5.7.3 Arbeitsweise

Ein numerisches Prädikat erzeugt beim Aufruf seines Konstruktors ein *FuzzyGraph* Objekt (siehe Abbildung 5.6), welches die jeweilige zu benutzende Zugehörigkeitsfunktion gemäß Tabelle 5.5 und 5.6 implementiert. Bei Prädikaten der 2. Gruppe muss die Zugehörigkeitsfunktion bei Aufruf der Methode *evaluate()* an die übergebenen Parameter aus *args[]* angepasst werden. Bei Auswertung der *evaluate()*-Methode des Prädikates wird *evaluate(val)* des *FuzzyGraph*-Objektes gerufen, welche das Ergebnis der Prädikatauswertung liefert.

5.8 Mögliche Verbesserungen

Als für die Funktionsweise der numerischen Prädikate wichtiger Freiheitsgrad bleibt die Wahl der Parameter D_{min}, D_{max}, D_1, D_2 und D (siehe Abschnitt 5.7.1). Sinnvolle Werte für diese Parameter müssen in Versuchen mit realen Daten ermittelt werden. Natürlich lassen sich diese Parameter vom Katalogbetreiber beeinflussen.

Weiterer Raum für Verbesserungen besteht in der Wahl einer Strategie für die in Abbildung 5.4 enthaltene *expand()* Funktion. Spielraum besteht in der Wahl des durch den Elternknoten zu ersetzenden Kindsknoten. Denkbar wäre eine Zuhilfenahme der bestehenden Kantenrelationen, so dass die loser gebundenen Knoten zuerst expandiert werden.

Kapitel 6

Zusammenfassung

Das Katalogsystem PIA zeichnet sich, aufgrund seiner Objektstruktur, durch ein hohes Maß an Flexibilität gegenüber anderen Systemen aus (vgl. Kapitel 2). Diese Objektstruktur liegt dieser Arbeit zugrunde und schafft besondere Anforderungen an das Konzept und die Implementation von Anfrageprädikaten.

Ein weiterer dieser Arbeit zugrundeliegender Formalismus ist die Theorie unscharfer Mengen, die in Grundzügen in Kapitel 3 vorgestellt wird. Benötigt wird insbesondere die Verkettung unscharfer Relationen unter Zuhilfenahme einer t-Norm zur Realisierung der Arbeitsweise des unscharfen diskreten Prädikates *is similar to*.

Da die Werte der diskreten Domänen, entsprechend der Objektstruktur von PIA, in einer Baumstruktur angeordnet sind, reduziert sich die Auswertung des diskreten Prädikates *is similar to* auf die wiederholte Anwendung einer bestimmten t-Norm, so dass der Spielraum für diese Arbeit lediglich bei der Initialisierung der Baumstruktur und der Wahl der t-Norm liegt. Beide Aspekte werden in Kapitel 4 erörtert. Die Idee zur Vereinfachung der Initialisierung der Baumstruktur besteht darin, dem Katalogbetreiber semantisch bedeutsame Auswahlmöglichkeiten zur Verfügung zu stellen.

Der Arbeitsweise der numerischen Prädikate, wie in Kapitel 5 vorgestellt, liegt die Prämisse zugrunde, dass ein numerisches Prädikat eine unscharfe Menge der entsprechenden Semantik repräsentiert. Als Zugehörigkeitsfunktionen werden drei verschiedene lineare Funktionen gewählt, und die Werte des Grundbereiches werden durch Mittelwert und Standardabweichung repräsentiert.

Das Auffinden des jeweiligen Grundbereiches einer Anfrage eines Kunden stellt in PIA eine Herausforderung dar. Zu diesem Zweck werden die diskreten Kriterien einer Anfrage herangezogen, da sie den vom Kunden gewünschten Grundbereich repräsentieren. Um dies effizient und gleichzeitig so genau wie möglich zu gestalten, werden beim Initialisieren des Systems geeignete Metainformationen angelegt und bei Auswertung einer Anfrage effizient zugegriffen.

Literaturverzeichnis

[1] Larsen and Yager, "The Use of Fuzzy Relational Thesauri for Classificatory Problem Solving in Information Retrieval and Expert Systems" *IEEE Transactions on Systems, MAN and Cybernetics,* Vol 23, no. 1 pp 31-41

[2] R. M. Tong, "The representation of uncertainty in an expert system for information retrieval," in *Fuzzy Logic in Knowledge Engineering.* H.Prade, and C. V. Negoita, Eds. Köln: Verlag TÜV Rheinland 1986 pp. 58-72

[3] A. Grauel, "Fuzzy Logik," BI-Wiss.-Verlag, 1995

[4] H. Bandemer, S. Gottwald, "Einführung in Fuzzy-Methoden," Akad. Verlag 1993

[5] J. Bacher, "Clusteranalyse," Oldenbourg 1996

[6] R. Fagin, "Fuzzy Queries in Multimedia Database Systems," Invited paper, Proc. Seventeenth ACM Symp. on principles of Database Systems, Seattle, 1998, ed. J. Paredaens, pp. 1-10

[7] L. A. Zadeh, "Fuzzy Sets," Information and Control 8 (1965) pp. 338-353

[8] D. Dubois and H. Prade "Fuzzy Sets and Systems: Theory and Applications", Academic Press, New York, 1980

[9] B.Biewer "Fuzzy-Methoden", Springer-Verlag, 1997

[10] P.C. Lockemann und J.W. Schmidt "Datenbankhandbuch", Springer-Verlag, 1987

[11] Gerard Salton "Introduction to Modern Information Retrieval" Michael J. McGill 1983

[12] F. Matthes and U. Steffens "PIA - A Generic Model and System. for Interactive Product and Service Catalogs", *Research and Advanced Technology for Digital Libraries*, volume 1696 of Lecture Notes in Computer Science, pages 403-422. Springer-Verlag, September 1999.