UNSCHARFE SUCHE IN DATENBANKEN –

KONZEPTUELLE GRUNDLAGEN, SYSTEMENTWURF UND PROTOTYPISCHE IMPLEMENTIERUNG

Diplomarbeit
vorgelegt von

ERNST AUGUST WIEDEN

Technische Universität Hamburg-Harburg
Arbeitsbereich Softwaresysteme

Betreuerin: Ulrike Steffens
1. Korrektor: Prof. Florian Matthes
2. Korrektor: Prof. Joachim W. Schmidt

Meinen Eltern

VORWORT

Diese Arbeit verfolgt das Ziel, den Abfragemechanismus von Datenbanken intelligenter zu gestalten; dies insbesondere in Hinblick auf den Einsatz von Fuzzy-Logic-Methoden, die bisher vorwiegend Einsatz in der Regelungstechnik finden.

Die Arbeit entstand im Umfeld des Forschungsprojekts PIA an der TU-Harburg, welches im Anschluss an ein einleitendes Kapitel näher beschrieben wird.
Darauffolgend findet sich eine Einführung in die Fuzzy-Logic, soweit es zum Verständnis der Ansätze dieser Arbeit notwendig ist. Hierin werden auch die bisherigen Forschungsergebnisse auf dem Gebiet der Fuzzy-Datenbanken dargestellt. Das vierte Kapitel beschäftigt sich mit der Darstellung einer Applikation, die der prototypischen Realisierung der Ansätze dieser Arbeit zugrunde liegt. Diese konkrete Implementierung einer Fuzzy-Datenbank heißt FfA (Fuzzy for MS-Access).
Nachdem somit die Grundlagen der Problemlösung beschrieben sind, wird im fünften Kapitel ein Ideal-System entworfen. das Fuzzy-Zugriffe auf beliebige Datensätze zur Verfügung stellt. Es lernt über einen Feedback-Mechanismus individuelle Bewertungen der Ergebnismenge und erfüllt somit auch Personalisierungsanforderungen. Als theoretischer Systementwurf ist es jedoch von einer konkreten Implementierung weit entfernt.
Das sechste Kapitel befasst sich mit dem, im Rahmen dieser Arbeit realisierbaren Prototypen, der die FfA-Implementierung erweitert und korrigiert. Das Ergebnis ist eine Software, die von beliebigen Clients über das http-Protokoll angesprochen werden kann, überschaubar und damit wartbar ist, Implementierungsfehler des Originals korrigiert, notwendige Ergänzungen enthält und schließlich auch den Anforderungen des PIA-Objektmodells gerecht wird.
Abschließend findet sich eine Zusammenfassung mit Ausblicken auf mögliche weiterführende Untersuchungen zu dem Thema dieser Arbeit.

Mit den gesteckten Zielen rückt diese Arbeit ins Umfeld der viel gescholtenen Künstlichen Intelligenz, die in der Vergangenheit viele ihrer selbstbesteckten Ziele nicht erreichen konnte. Um solcher Art Enttäuschungen vorzubeugen, ist es klüger sich schon im Vorfeld von diesem Anspruch zu distanzieren und sich selbst einer nicht weniger ehrgeizigen Disziplin zuzuordnen:

> **Künstliche Dummheit** (KD) kann als der Versuch von Computerwissenschaftlern angesehen werden, Computerprogramme zu entwerfen, die in der Lage sind, die Art von Problemen zu erzeugen, die normalerweise in der menschlichen Denkweise begründet sind.
>
> Wallace Marshal, Journal of Irreproducible Results (1987) [in Ku93]

In diesem Sinne möge man sich geduldig mit der Technik zeigen. Ihre Intelligenz steht noch am Anfang der Evolution.

Ich danke meinen Betreuern Professor Florian Matthes und Ulrike Steffens für ihre Geduld und Unterstützung.

Hamburg, im August 2002

Inhaltsverzeichnis

Kapitel 1 - Einleitung und Motivation

Die vorliegende Arbeit beschäftigt sich mit der Problematik von Fuzzy-Zugriffen auf Datenbanken. Das englische Wort „Fuzzy“ lässt sich etwa mit „unscharf“ oder „verschwommen“ übersetzen und bezieht sich auf die den Abfragen zu Grunde liegenden Bedingungen.
Als Eindruck für die Problemstellung möge man sich den Bewertungsmechanismus vorstellen, der notwendig ist, um folgende Anfrage zu beantworten:
„Ich suche einen schnellen Computer, der nicht zu teuer ist“.
Dabei gehe man davon aus, dass in der zu durchsuchenden Datenbank nicht etwa Aussagen wie „Geschwindigkeit ist hoch“, sondern „CPU-Takt=266MHz“ getroffen werden.

Im Informationszeitalter ist die technische Verarbeitung von Wissen zur wesentlichen Aufgabe geworden. Wissen wird aus seiner ursprünglichen Umgebung, dem menschlichen Gehirn, heraus gelöst und immer weiter zur Maschine hin verlagert.
Gedächtnisforscher haben drei Phasen des Vergessens in der Menschheitsgeschichte ausgemacht. Die Erfindung der Schrift, des Buchdrucks und des Computers. Mit jedem Schritt verlagerte sich die Tätigkeit des Gehirns mehr in Richtung Informationsverknüpfung und weg von der Informationshaltung.
Mit fortschreitender Komplexität der Informationssysteme muss auch die Tätigkeit der Informationsaufbereitung immer mehr in die Maschine verlagert werden, da der menschliche Geist nicht nur mit dem Vorhalten der Informationen überfordert ist, nein, er kann sie auch nicht mehr vollständig überblicken und ihre Relevanz bewerten. Eigens hierzu hat sich die neue Disziplin des Data Minings entwickelt. Ein älterer Ansatz ist der des Decision Makings aus der Disziplin der Künstlichen Intelligenz (KI, engl. AI). Heute werden diese Ansätze in der Disziplin des Soft-Computings vereint und erweitert. Sie besteht aus Disziplinen wie Fuzzy-Logic, Künstlichen Neuronalen Netzen, Genetischen Algorithmen, verschiedenen Ausläufern der KI u.Ä.. Dies geschieht mit dem Ziel der Synergie. Der Ansatz steht damit im Kontrast zu der oft als dogmatisch empfundenen Position der KI.

Die Künstliche Intelligenz, wie sie sich in den Ingenieurswissenschaften etabliert hat, versteht Denken als eine effektive Suche in großen Lösungsräumen. Dies ist eine Spielart der KI, bei der versucht wurde, die verwertbaren Forschungsergebnisse praktisch umzusetzen. Man unterscheidet beste Lösung, erste gefundene Lösung und Lösung einer bestimmten Mindestqualität (Hybridform der vorgenannten). Der Lösungsraum muss hierfür in Form eines Netzes von möglichen Varianten vorliegen. Auf diesem Netz werden dann effektive Suchalgorithmen eingesetzt.
Neben der Effektivität der Suche, spielt die Sprache die zweite maßgebliche Rolle in der maschinellen Wissensverarbeitung. Wir kennen verschiedene Formen von synthetischen und natürlichen Sprachen.
Zu den synthetischen Sprachen gehören z.B. die Systemmodellierungssprachen der objektorientierter Programmierung, wie UML [BoRu99] (Unified Modeling Language), die in dieser Arbeit verwendet wird. Dabei versucht man zunächst Zusammenhänge und Abhängigkeiten im System so präzise wie möglich und nötig darzustellen, um dann Abläufe beschreiben zu können. Das System wird gedanklich durch Definition von Schnittstellen gegen seine Umwelt abgegrenzt und soweit vereinfacht, wie es für die gewünschte Funktionalität zulässig erscheint.
Mit diesen systemtheoretischen Einschränkungen lassen sich „scharfe“ Systeme modellieren, deren Objekte i.A. präzise bestimmten Mengen zugeordnet werden können. In den komplexen Systemen der realen Welt hingegen lässt sich so eine Zuordnung oft nicht treffen und wir

tragen dem mit unserer natürlichen Sprache Rechnung, indem wir von schnellen oder sehr schnellen Autos sprechen und den Übergang dazwischen als fließend bezeichnen.

Obwohl wir mit solchen Formulierungen den Informationsgehalt aus technischer Sicht scheinbar verringern, reduzieren wir doch in Wirklichkeit nur die Datenmenge und werten sie i.A. mit unserem Expertenwissen auf. So kann es als unwesentlich angesehen werden, ob ein Auto mit 180 oder 190 km/h verunglückt, nicht aber ob es schnell oder langsam war.
Maria Zermankova-Leech unterscheidet in ihrer oft zitierten Grundlagenarbeit „Fuzzy relational data bases – a key to expert systems“ [KaZe84] zwischen Daten und Information und spricht der Information die Eigenschaft zu, eine bedeutungstragende Interpretation von Daten zu sein, die beispielsweise durch Messung oder Statistik gewonnen wurden.
Eine scheinbar andere Sicht der Dinge liefert ein Zitat von George Klir (in [McFr94]):

> „Wenn die Komplexität zunimmt müssen wir eines aufgeben. Und das ist die Gewissheit.“

Dennoch zeigen beide nur die Dualität eines Weltbilds. Einerseits kann man versuchen die Welt in ihrer Komplexität präzise abzubilden. Unsere wachsenden technischen Möglichkeiten sind hierzu zunehmend in der Lage. Dies macht die Modelle zwar präziser, nicht aber unbedingt handhabbarer. Für eine sinnvolle Bearbeitung der Daten, etwa mit dem menschlichen Geist über die Schnittstelle des Perzeptionsvermögens, ist es aber andererseits notwendig, diese Information ggf. in iterativen Prozessen zu komprimieren, um das zuvor gewonnene Mehr an Präzision zu einem Mehr an Information zu machen. Diese Abläufe müssen zum Teil von Experten bewerkstelligt werden. Es gibt aber zunehmend Bestrebungen, das Wissen der Experten in Computerprogramme abzubilden, um ihr Wissen effektiver zu nutzen und die Verfügbarkeit zu erhöhen.

Am Arbeitsbereich Softwaresysteme der TU-Hamburg-Harburg wird ein Personal Information Assistent (PIA) entwickelt, der das Ziel verfolgt interaktive Kommunikation zwischen Anwendern und Produktkatalogen zu ermöglichen. PIA soll intuitiven Zugriff auf unterschiedlichste Informationsquellen bieten und im interaktiven Prozess vom Verhalten der Anwender lernen. Die Aufgabe des intuitiven Zugriffs erfordert neben einem angemessenen Frontend Datenbankabfragen, die natürlichsprachliche Queries beantworten. Der Ansatz dieser Arbeit ist es hierzu die Methoden der Fuzzy-Logic für Datenbanken und andere Informationssysteme zu adaptieren.. Mit dieser Technik lassen sich Zuordnungen finden für die Bedingung „schnelles Auto“, oder auch graduelle Zugehörigkeiten zu „rotes Bild“ oder phonetischen Ähnlichkeiten in Worten.
Um Fuzzy-Logic und Datenbanken in einem System zu vereinen, müssen die dazu nötigen Konzepte theoretisch untersucht werden. Außerdem soll ein „ideales“ System entworfen werden und zum Zwecke der Evaluation ausgewählter Methoden eine prototypische Implementierung vorgelegt werden.
Bei der Behandlung dieser Aufgabe ist dem vorliegenden Objektmodell und den übergreifenden Projektanforderungen Rechnung zu tragen, wie beispielsweise der Anforderung durch User-Feedback das System weiterentwickeln zu können.

Abschließend sei der renommierte Forscher Marvin Minsky zitiert (in [Ku93]), der auf einem Kongress zu seinen Kollegen aus einer fiktiven Zukunft sprach:

> „Können sie sich vorstellen, dass es damals Bibliotheken gab, in denen sich die Bücher nicht miteinander unterhielten?“

Diesem Ziel „autointelligenter Datenhaltung“ hat sich auch diese Arbeit verschrieben.

Kapitel 2 - PIA

Die Anwendung elektronischer Massenmedien, wie Internet und CD-ROMs, setzt sich zunehmend bei der Kommunikation zwischen Anbietern und Konsumenten von Dienstleistungen und Produkten durch. Im Übergang zwischen einem persönlichen Beratungsgespräch mit dem Kunden und der anonymen Informationsaufbereitung gegenüber einem Interessenten an einem Computer entsteht dabei eine Lücke. Diesem Mangel will der Personal Information Assistant, im folgenden auch als das PIA oder PIA-System bezeichnet, begegnen, indem er personalisierte Suchabfragen an den Computer erlaubt. Er wird am Arbeitsbereich Softwaresysteme (STS) der TU-Harburg entwickelt.
Dabei verfolgt man das Ziel, nicht nur auf der Clientseite mit dem Anwender in einen lang andauernden interaktiven Lernprozess zu treten, sondern auch auf der Anbieterseite von dem Verhalten des Anwenders zu lernen. Dieser Prozess wird als „gegenseitiges Verständnis" („Mutual Understanding") bezeichnet und wird von einem sogenannten „Relevance Feedback" gesteuert, einer Rückwirkung des vom Anwender bewerteten Angebots, welches ihm vom Anbieter unterbreitet wurde.
Neben weiteren Anforderungen des Systems, Benutzerfreundlichkeit, benutzerspezifische Schnittstellen, personalisierte Suchabfragen, Medienunabhängigkeit, u.ä. [MaSt99] ist für die vorliegende Arbeit besonders wichtig, quasi natürlichsprachliche, unscharfe Anfragen zuzulassen. Dabei wird zunächst mehr Augenmerk auf eine reiche Funktionalität der Sprache gelegt, als auf eine mathematisch exakte und wohl definierte Theorie.
Um dies zu erfüllen wurde zunächst ein Frontend zur Anwenderkommunikation entwickelt [Ca01]. Parallel dazu wurde ein offenes Objektmodell erstellt. Es hat u.a. die Aufgabe eine Brücke zwischen verschiedenen Arten von Informationssystemen schlagen. Insbesondere die Verschiedenartigkeit von strukturierter Information (Datenbanken) und unstrukturierter Information (Texten) soll berücksichtigt werden [St97]. PIA berücksichtigt dies in einem Domänen-Modell (siehe 2.3 Das PIA-Domänenmodell).
Das offene Modell sieht weiter vor, Beschreibungen beliebiger Produkte und Dienstleistungen als Daten verwalten zu können. Dieser Ansatz birgt einige Konflikte in semantischer Hinsicht. So hat das Wort „teuer" eine andere Bedeutung, wenn man es auf Butter bezieht, als wenn es den Preis eines Hauses bewertet. Das gegenwärtige Objektmodell ist im Abschnitt 2.2 Objektmodell beschrieben. Es ist Ergebnis einer Studienarbeit am Arbeitsbereich STS [Zh98].
Die unscharfen Suchanfragen werden dagegen in dieser Arbeit theoretisch untersucht und prototypisch implementiert. Sie umfassen Queries in der ersten Normalform auf numerischen, hierarchischen und Volltext-Domänen, wie im weiteren Verlauf der Arbeit noch erläutert werden wird.
Gerade wegen der Maßgabe einer prototypischen Umsetzung der Aufgabe wurde in dieser Arbeit ein besonderes Augenmerk auf eine solide theoretische Grundierung des Themas gelegt.
Die Realisierung der Personalisierung des Systems ist ein noch offenes Problem, gleichwohl es mündliche Anforderungsspezifikationen gibt, die in den einzelnen Arbeiten zu PIA berücksichtigt werden und auch im Lösungsentwurf dieser Arbeit ihren Niederschlag finden.
Da diese Arbeit Teil eines fortlaufenden Entwicklungsprozesses ist, sind die folgenden Beschreibungen als gegenwärtiger Stand des Systems zu betrachten. Einleitend wird das verwendete Vokabular beschrieben.

2.1. Das PIA-Vokabular

Um das Datenmodell von PIA zu erläutern seien zunächst einige Begriffe grundsätzlich beschrieben.

Produkt (Item)

Ein Produkt ist die Beschreibung eines verkaufbaren Artikels und wird einer bestimmten Produktkategorie zugeordnet. Es hat weitere beschreibende Attribute wie einen Namen oder auch eine Preisangabe sowie produktspezifische Attribute. Ein Beispiel für ein Produkt ist:
Kategorie Auto, Typ Mercedes E230, Baujahr 1998, Leistung 95 KW, Preis 12 000 €.

Attribut (Attribute, Merkmal, Eigenschaft)

Eine mit einem Namen (Bezeichner) versehene Eigenschaft eines Produkts. Häufig kommt ihre Bedeutung in ihrem Namen zum Ausdruck.
Beispielsweise informieren die Attribute „Produktname“, „Produktkategorie“, „Preis“, „Hersteller“ und „Anbieter“ über den Zustand eines Produktes in der realen Welt.

Ausprägung (Value, Attributwert)

Der Wert der einem Attribut für ein bestimmtes Produkt zugeordnet ist.
Ein konkretes Vorkommen (tatsächliches Auftreten) eines Attributes. In einer tabellarischen Darstellung werden Attributausprägungen als Tabelleneinträge dargestellt, während ein Attribut durch die Spaltenüberschrift einer Tabelle repräsentiert wird.
Beispiel: Im Produktkatalog wird die Produktkategorie „Personal Computer“ genannt, Attribut ist „Produktkategorie“, Attributausprägung ist „Personal Computer“.

Domäne (Domain)

In PIA ist jedem Attribut eine Domäne zugeordnet. Die Domäne enthält die Information über die gesamte Menge der Attributausprägungen, die bei diesem Attribut vorkommen könnten. Domänen lassen sich bestimmten Domänenkategorien zuordnen, wie numerisch, hierarchisch oder Volltext.
Ein Beispiel dafür ist die Domäne „Preis“, die positive reelle Zahlen repräsentiert und weitere Informationen zu Währungseinheiten, z.B. Dollar, € oder Yen usw. enthält.

Objekt (Entität)

Eine Repräsentation eines konkreten oder abstrakten Gegenstandes, die für ein gegebenes Anwendungssystem von Bedeutung ist. Ein Produkt ist z.B. eine Entität.

Prädikat (Predicate)

Um die Eigenschaften von Objekten gezielt abfragen zu können, werden mit Prädikaten Bedingungsoperatoren eingeführt.
Sie können etwa die Existenz einer Eigenschaft fordern oder die Relation zu einem Vergleichswert (z.B.: =, <, *ungefähr gleich, hoch*).

Attributanfrage (Criterion, Kriterium)

Eine Attributanfrage gibt ein Kriterium eines Attributes wieder, nach dem die Produkte gewählt werden sollen. Sie ist die konkrete Ausprägung eines Prädikats mit den dazugehörigen Parametern.
Wir bezeichnen z.B. „Kategorie ist Auto“ als eine Attributanfrage.

Suchauftrag (Alternative, Query, Interest, Interesse)

Ein Suchauftrag besteht aus einer oder mehreren Attributanfragen, die sich auf unterschiedliche Attribute beziehen. Ein Beispiel dafür wäre:
Kategorie ist Auto, Preis zwischen 2000 € und 10.000 €, Leistung mindestens 100 PS.

Auf diesen Begriffen aufsetzend wird nun das Objektmodell mit Hilfe eines UML-Diagramms beschrieben (Abbildung 2-1: Das PIA-Objektmodell). Die Sprachwahl ist Englisch und wird hier in Dualität mit den entsprechenden deutschen Begriffen verwendet.

2.2. Das PIA-Objektmodell

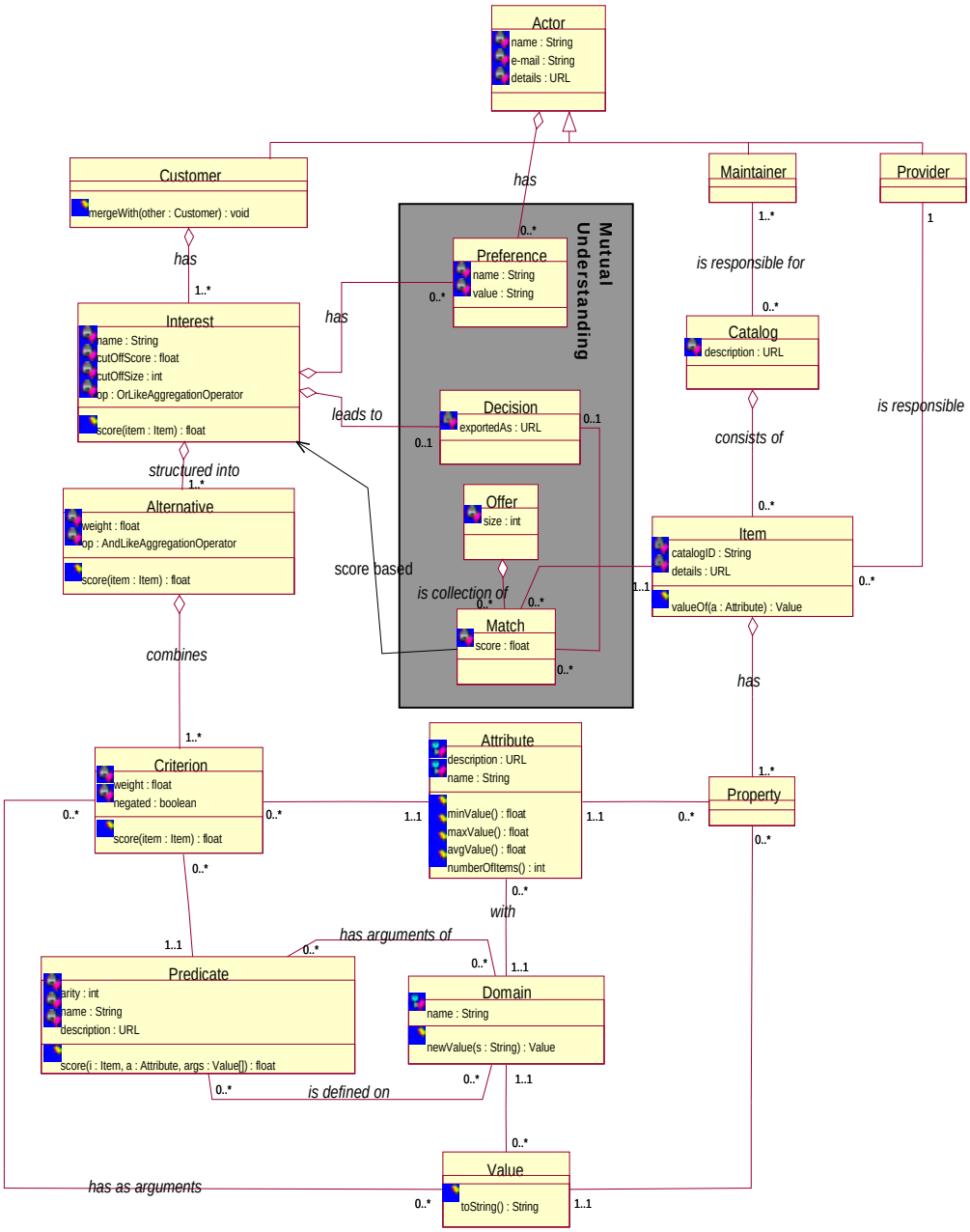

Abbildung 2-1: Das PIA-Objektmodell

An der Spitze des Diagramms findet sich ein allgemeiner Actor, der ein Anwender (Customer), d.h. jemand der Anfragen an das System stellen möchte, sein kann oder aber jemand, der Daten

ablegen oder verwalten möchte, Maintainer oder Provider (Überwacher oder Datenanbieter). Der Maintainer (Redakteur) ist jemand, der das System im Bertrieb kontrolliert, also Attribute und Domänen pflegt. Der Provider dagegen stellt Beschreibungen ein. Den Verwaltern des PIA lässt sich etwa die rechte, den Anwendern die linke Teil des Diagramms zuordnen.
In der Mitte findet sich ein Rahmen, der als Mutual Understanding (gegenseitiges Verstehen) bezeichnet wird. Er beschreibt die abstrakte Ebene eines Entscheidungsprozesses und interagiert mit dem Rest des Modells über die Klassen Interest (Interesse) und Item (Produkt).
Abgelegte Datensätze werden in diesem System als Items bezeichnet. Jedes Produkt weist bestimmte Einzelheiten auf, die durch Eigenschaften beschrieben werden, die Werte (values) haben können. Diese Attribute entstammen bestimmten Domänen.
Neben den Einträgen der Datenbanken, die im rechten unteren Teil des Diagramms beschrieben werden, ist auf der anderen Seite die Beschreibung der Queries wichtig. Sie werden übergeordnet als Interesse (Attributanfragen) beschrieben, welches sich aus verschiedenen, gewichteten Alternativen zusammensetzen kann (Oder-Verknüpfung). Alternativen können ihrerseits aus verschiedenen gewichteten Kriterien bestehen (Und-Verknüpfung). Kriterien sind Bedingungen an bestimmte Eigenschaften der Produkte, hier als Predicate (Prädikat) bezeichnet. Diesen Bedingungen ist eine Arity (Wertigkeit) zugeordnet, im Sinne einer z.B. ternären Funktion „a liegt zwischen b und c“(arity=3).

2.3. Das PIA-Domänenmodell

Das Domänenmodell von PIA stellt seit Anbeginn ein besonderes Problem innerhalb des Projekts dar. Es sei durch einen Auszug aus dem PIA-Objektmodell beschrieben (Abbildung 2-2). Als Domäne wird, wie zuvor beschrieben, die gesamte Menge möglicher Attributausprägungen bezeichnet, wie etwa Preis oder Geschwindigkeit oder Prozessortyp.

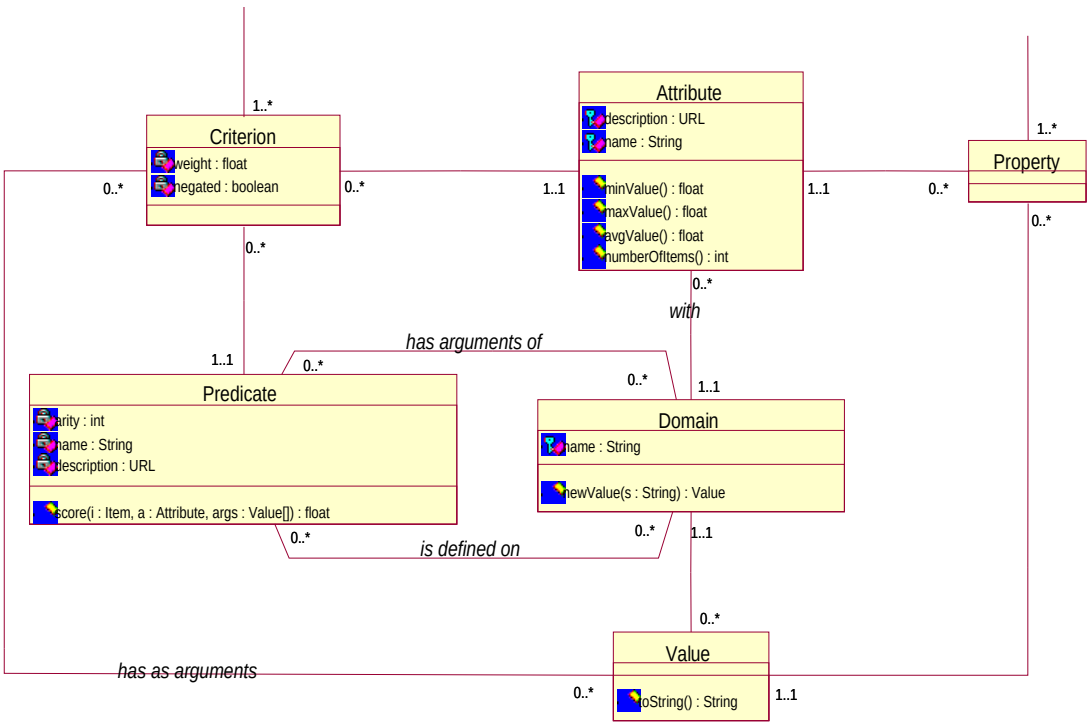

Abbildung 2-2: Das PIA-Domänenmodell

Umrechnungsfunktionen fangen dabei mögliche unterschiedliche Skalierungen ab (m/s versus km/h). Typische sprachliche Ungenauigkeiten, wie *schneller Prozessor* lassen sich durch Präzisierungen wie *Prozessor-Taktgeschwindigkeit ist hoch* abfangen.
Ein ungelöstes Problem hingegen stellt die Gleichbehandlung aller Items mit beispielsweise dem Attribut der numerischen Domäne Preis dar. So muss sich derzeit ein Gebrauchtwagen für 1000 € gefallen lassen als teuer bezeichnet zu werden, da sein Preis der gleichen Domäne entstammt, wie der eines Bleistifts. Diese Problematik wird in (5.3 Erweiterung des PIA-Objektmodells) noch näher erläutert. Das Problem rührt daher, dass Prädikate in diesem Modell auf Domänen (skalar / hierarchisch) definiert sind, und nicht auf den Attributen einzelner Domänen.
Um diesem Modell zu genügen, müssen Prädikate höheren Grades definiert werden:

Linguistic Value teuer **à**

Unäre Funktion teuer = hoch(Preis) **à**

binäre Funktion teuer = hoch(Preis, Auto)

Dieses Prinzip wird in *5.3 Erweiterung des PIA-Objektmodells* noch in einer formal strengeren Form beschrieben.
Wesentlich am Domänenmodell ist weiterhin die Einteilung der Domänen in Unterdomänen. Diese Aufteilung bewirkt die Integration der unterschiedlichen Informationssysteme IR und RDBMS's zu einem Einzigen. Die Domänen werden hierfür unterschieden nach skalarem, hierarchischem oder textuellem Charakter. Diese drei Gruppen ließen sich etwa als Oberdomänen bezeichnen. Jede Domäne gehört einer dieser Oberdomänen an. Hierbei enthält die hierarchische Oberdomäne mit einer Tiefe von Eins, den Spezialfall Aufzählungsdomäne (Set, Menge). In dieser Arbeit wird dem Problem numerischer Domänen das Hauptaugenmerk gewidmet.

2.4. Zusammenfassung (PIA)

Diese Arbeit beschäftigt sich mit der Evaluierung eines Interesses auf einem Katalog von Items, so dass dem Customer ein entsprechendes Angebot unterbreitet werden kann. Dabei wird auch die Pflegbarkeit des Angebots durch Maintainer und Provider mit in Betracht gezogen, sowie die Rückwirkung des Relevance Feedbacks auf das Mutual Understanding. Die Arbeit liefert somit den Unterbau für das Frontend. Vorrangig wird dabei das theoretische Fundament entwickelt und ein Prototyp bereitgestellt, mit dem sich die Relevanz der Konzepte und die Geschlossenheit des Modells in praktischen Versuchen überprüfen lassen.

Kapitel 3 - Stand der Technik (Fuzzy-Datenbanken)

In diesem Abschnitt soll der gegenwärtige Stand der Forschung und Entwicklung auf dem Gebiet der Fuzzy-Datenbanken beschrieben werden. Wegen der allgemeinen Verbreitung der Datenbanktechnologie wird auf eine nähere Beschreibung dieser Technologie verzichtet. Dagegen sei die Fuzzy-Logic in einer einführenden Form beschrieben (Konzepte der Fuzzy-Logic) und an den relevanten Stellen vertieft (Mathematik der Fuzzy-Logic). Dabei werden die allgemeinen Fuzzy-Theorien meist schon in ein Verhältnis zu ihrer Anwendung in der Datenbanktechnologie gesetzt.
Abschließend werden einige Literaturquellen herangezogen, um spezielle Problematiken der Fuzzy-Datenbanken näher zu betrachten.

3.1. Konzepte der Fuzzy-Logic

3.1.1. Begriff und Historie

Fuzzy-Logic wird häufig mit dem paradoxen Ausdruck „unscharfe Logik“ übersetzt. Inhaltlich präziser ist jedoch der Ausdruck „Die Logik der Unschärfe“.
Die Fuzzy-Logic stellt eine Erweiterung des binärlogischen Kalküls dar. Den mathematisch historischen Hintergrund bilden Arbeiten über mehrwertige Logiken, die insbesondere mit der Ereignisunbestimmtheit der Quantentheorie entstanden. Darin wurden die in der klassischen binären Logik möglichen Wahrheitswerte wahr und falsch (bzw. 0 und 1) einer Aussage, um weitere Zwischenzustände (z.B. unbestimmt bzw. ½) ergänzt.
Lotfi A. Zadeh [Za65] erweiterte diese Theorie zur mathematisch exakten Beschreibung von Variablen mit linguistisch und damit unscharf vorgegebenen Werten. Die Regeln für die Verknüpfung dieser Variablen sind das axiomatische Grundgerüst für die Fuzzy-Logic. In den folgenden Jahren entstand eine große Anzahl sowohl von theoretischen, als auch von anwendungsbezogenen Arbeiten, insbesondere in den Bereichen

- Steuerung und Regelung,
- Prozessüberwachung und –diagnose
- Mustererkennung
- Medizin und Psychologie,
- Wirtschaftswissenschaften,
- Mathematik.

Motor für die rasche industrielle Entwicklung war weniger die Notwendigkeit, für bis dahin unlösbare Aufgaben eine Lösungsmethode zu finden, sondern vielmehr schnellere Lösungsverfahren zu entwickeln, als sie von den klassischen Methoden zur Verfügung gestellt werden.
Im Falle unscharfer Datenbankabfragen hingegen müssen wir die Fuzzy-Logic als Methodik für ein sonst nicht beherrschbares Problem einführen. Die Abbildung scheinbar genau quantifizierbarer Werte auf sprachliche „Ungenauigkeiten“.

3.1.2. Linguistische Variable

Das menschliche Denken ist eng mit der menschlichen Sprache verbunden. Anfang des Jahrhunderts vertrat der Philosoph Arthur Wittgenstein die noch heute bei KI-Forschern beliebte These: „Worüber man nicht sprechen kann, darüber muss man schweigen.“ [Wi60] – und stellt in den weiteren Überlegungen zu diesem Satz heraus, wie eng der Zusammenhang ist zwischen der menschlichen Ausdrucksfähigkeit und der Möglichkeit, Sachverhalte gedanklich

zu erfassen und zu verarbeiten. Ähnliche Ansätze wurden später in der Kybernetik weiterentwickelt und sind heute in den systemtheoretischen Grundlagen der objektorientierten Programmierung zu finden, wenn es um die Vollständigkeit und Modellierbarkeit von Systemen geht.
Es steht jedenfalls fest, dass die Funktionalisierung von Sprache einer der entscheidenden Schritte hin zu intelligenten Informationssystemen ist. Die Fuzzy-Logic liefert mit ihrem Konzept der linguistischen Variablen einen wichtigen Baustein dafür.
Die Werte einer linguistischen Variablen sind Worte bzw. Terme einer natürlichen (oder synthetischen, d.h. standardisierten) Sprache. Sie werden durch unscharfe Mengen $\mathbf{A}_i$ bzw. deren Zugehörigkeitsfunktionen[1] $\mu_i(x)$ in Form von Verteilungsfunktionen über einer Basisvariablen x eines (physikalischen) Grundbereichs **X** repräsentiert. Diese Zugehörigkeitsfunktionen bilden eine linguistische auf eine numerische Werteskala ab. Die Dimensionierung dieser Abbildung ist neben der Regelerstellung (s.u.) i.A. Aufgabe des Entwicklers.

Beispiel 3-1: Ein möglicher syntaktischer Aufbau der linguistischen Variable Geschwindigkeit

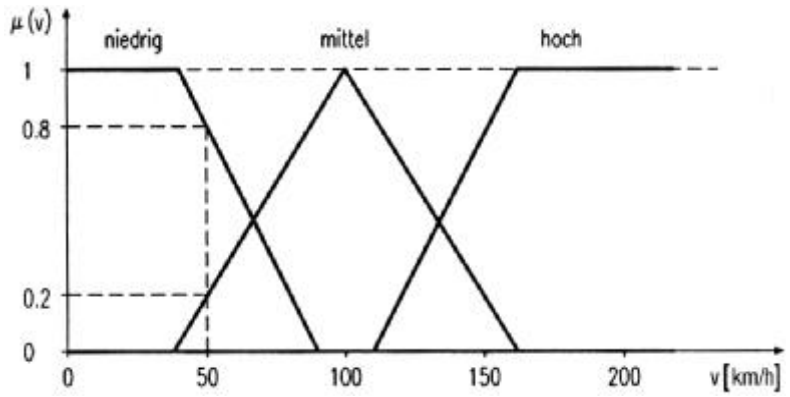

Abbildung 3-1: möglicher syntaktischer Aufbau der linguistischen Variable Geschwindigkeit

3.1.3. Unscharfe Mengen, unscharfe Relationen und Zugehörigkeitsfunktion

Wesentliches Konzept der Fuzzy-Logic sind unscharfe Bewertungen von Größen. Hier sind in erster Linie die unscharfe Menge und die unscharfe Relation zu nennen.
Die Definition einer *unscharfen Menge* **M** erfolgt mit Hilfe der charakteristischen Funktion μ_M.
Die Werte von μ_M liegen im normalisierten Fall, der hier immer angenommen werden soll, zwischen 0 (keine Zugehörigkeit) und 1 (volle Zugehörigkeit). Werte außerhalb dieses Bereichs können bei der Regelerstellung entstehen, wenn statistische Daten herangezogen werden. Diese lassen sich aber i.A. wiederum normieren.
Mit diesen Funktionen können im Gegensatz zur klassischen Mengenlehre auch „fließende“ Übergänge der Elementzugehörigkeit zu einer Menge (bzw. des Zutreffens einer Aussage) beschrieben werden[2]. Es bietet sich die Möglichkeit, im Rahmen der Idealisierungen bei der

[1] engl. Membership function
[2] Eine mathematische Definition der Zugehörigkeitsfunktionen und unscharfen Mengen findet sich im Abschnitt 3.2.1Diskrete vs. kontinuierliche Mengen

Systembeschreibung „elastisch“ zu modellieren. Der Fall scharf begrenzter Mengen ist im Fall unscharf begrenzter Mengen insofern enthalten, als dass dann die Zugehörigkeitsfunktionen nur die Werte Null und Eins annehmen können.

Beispiel 3-2:

Die in Abbildung 3-2 gestrichelt dargestellte Zugehörigkeitsfunktion $\mu_M(x)$ beschreibt die Menge aller reellen Zahlen ≥ 18. Sie kann beispielsweise die Menge aller „volljährigen“ Bundesbürger darstellen, da der Begriff „Volljährigkeit“ eindeutig (scharf) durch Gesetzesentschluss festgelegt ist; zur Beschreibung der Menge aller erwachsenen Bundesbürger eignet sie sich dagegen nur als grobe, vereinfachende Darstellung, da für das Merkmal Erwachsensein keine scharfe Altersgrenze angebbar ist. Ebenfalls unbefriedigend wäre eine Festlegung aller nicht unangenehm kühlen Raumtemperaturen mit Hilfe einer einfachen Sprungfunktion. Ein gleitender Übergang (durchgezogen gezeichnet) eignet sich hier erheblich besser; er erzeugt den Freiheitsgrad, unter Umständen auch ein auf 16°C aufgeheiztes Zimmer noch als nicht unangenehm kühl zu bezeichnen.

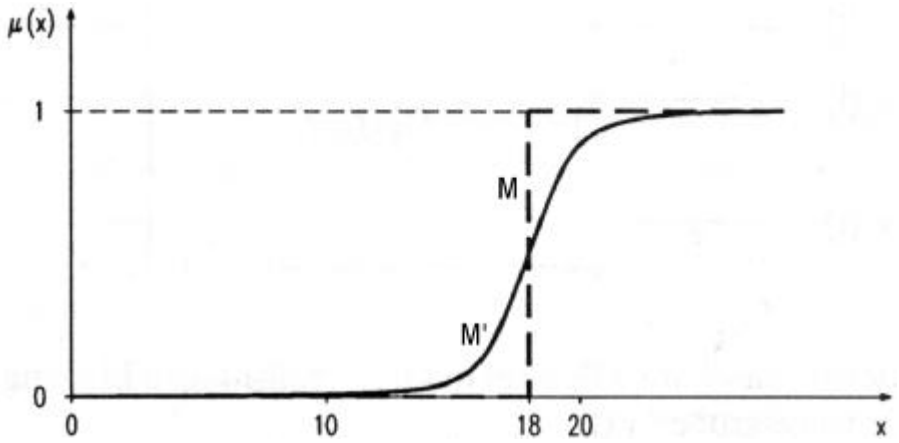

Abbildung 3-2: Zugehörigkeitsfunktion $\mu_M(x)$ für eine scharf begrenzte Menge **M** und $\mu'_{M'}(x)$ für eine unscharf begrenzte Menge **M‘**.

Das Modell der Unscharfen Menge beschreibt mathematisch betrachtet eine unäre Funktion, d.h. eine Funktion mit nur einem Parameter:

Warm: T -> $\mu_{warm}(T)$

Neben dieser Definition auf der Grundgesamtheit der Definitionsmenge, sind aber auch relative Funktionen wünschenswert (unscharfe Relation), die von einem oder mehreren Parametern abhängen. Beispiel:

sehr viel größer als: T **à** $\mu_{\text{sehr viel größer als}}(T, T_{Ref} = 10°C)$

Dabei handelt es sich um eine binäre Funktion, mit der Maßgabe, dass beide Parameter der gleichen physikalischen Domäne entstammen. So ist es nicht sinnvoll zu fragen, ob die Geschwindigkeit eines Autos sehr viel größer sei als die Grenztemperatur eines Transistors.

Schließlich müssen für den Ansatz von PIA, Abfragen der Form „T liegt ungefähr zwischen T_B und T_C“ zuzulassen, die vorgenannten Konzepte noch auf ternäre Funktionen erweitert werden:

Liegt ungefähr zwischen: T -> μ Liegt ungefähr zwischen (T, T_B, T_C)

Zur Lösung dieses Problems lassen sich zwei Ansätze finden. Zum einen könnte ein eigenes mathematisches Konzept entwickelt werden, wie im Kapitel Lösungsentwurf angeführt, zum anderen ließe sich die ternäre Funktion mit Verknüpfungsoperatoren in zwei binäre Funktionen zerlegen (siehe hierzu auch Particularization and Conditional Possibility Distributions in [KaZe84] S.91):

(T ist größer als T_B) und (T ist kleiner als T_C)

bzw.

(T ist ungefähr größer als T_B) und (T ist ungefähr kleiner als T_C).

Auch dieser Ansatz erfordert im Sinne eine theoretischen Fundierung eine weiterführende Diskussion, nämlich über den Operator „Und“. (Siehe Lösungsentwurf).
In der Literatur konnten zum Thema „unscharf vergleichende Zwischenwerte“ leider keine Vorschläge gefunden werden.

3.1.4. Definition von Unschärfe mittels Normalisierung

Ein besonders eleganter Weg, den Aufwand bei der Definition von Unschärfe zu reduzieren, besteht darin, allgemeine unscharfe Terme zu verwenden und sie als Attribute für i.A. alle Domänen einer Oberdomäne zu verwenden. Ein solcher Satz unscharfer Terme für numerische Domänen könnte sein:

```
{niedrig, mittel, hoch}.
```

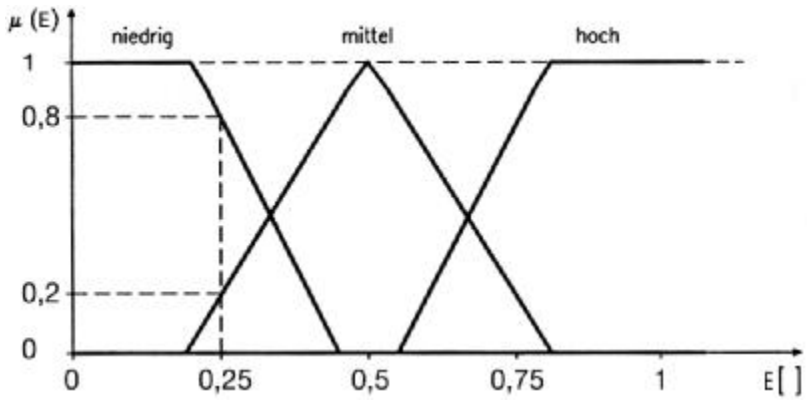

Abbildung 3-3: normierte Linguistische Werte

Mit einer solchen Menge lässt sich gleichzeitig die Unschärfe verschiedener Domänen beschreiben. So wird aus „schnell“ beispielsweise durch „Geschwindigkeit ist hoch“ beschrieben. Da die Unschärfe auf einem Einheitsintervall definiert ist, muss aber eine Transformation der Wertemenge der Domäne auf das Einheitsintervall definiert werden. Dies wird üblicherweise durch lineare Abbildung geleistet.
Die dafür nötigen Abbildungsparameter müssen für jede Domäne definiert werden. In erster Näherung bieten sich dafür Minimal- und Maximalwert der Domänen an. Wegen der Möglichkeit außergewöhnlich extremer Werte sollte man aber eher von üblichen Minimal- und Maximalwerten sprechen und die Ausreißer der Wertemenge auf diese abbilden.
Sei

$\mathbf{X}=\{x_i \mid x_i \in \mathbf{X}\}$.

$$T: x_i \rightarrow \begin{cases} x_i' = x_{Max} & \text{für alle } x_i > x_{Max} \\ x_i' = x_{Min} & \text{für alle } x_i < x_{Min} \\ x_i' = (x_i - x_{Min})/(x_{Max} - x_{Min}) & \text{für alle } x_{Min} \geq x_i \geq x_{Max,} \end{cases}$$

mit $x_{Max}, x_{Min} \in \mathbf{X}$ und $x_i' \in [0,1]$.

Auf diese Weise lässt sich durch geeignete Wahl der Stützwerte Min und Max auch eine Individualisierung der Unschärfe gewährleisten. So kann der Preis eines Hauses generell als teuer bewertet werden, wenn man als Minimalwert beispielsweise 0,- € annimmt. Nimmt man dagegen einen üblichen Mindestpreis von etwa 50.000,- € an, so lassen sich für Personen, die sich mit den üblichen Hauspreisen arrangieren können, auch günstige Häuser identifizieren. Andere Interessenten mögen einen Mindestpreis von 100.000,- € annehmen.

Die Bindung der Stützwerte an die Domänen gemäß dem Objektmodell von PIA stellt ein besonderes Problem dar, weil in diesem Objektmodell keine Unterscheidung zwischen den Preisdomänen von verschiedenen Kategorien vorgesehen ist. So ließen sich zunächst keine unterschiedlichen Minimalwerte für Autopreise und Streichholzpreise definieren. Diese Problematik wird in 5.3 Erweiterung des PIA-Objektmodells weiter diskutiert.

3.1.5. Modifikator

Eine weitere Möglichkeit, die Unschärfe technisch zu vereinheitlichen, besteht in der Verwendung linguistischer Modifikatoren. Mit der funktionalen Beschreibung von Begriffen wie „sehr“, „nicht so sehr“ oder ähnlichen lassen sich bestehende Terme einheitlich erweitern. Hierzu zählt auch die Komplementbildung „nicht“.

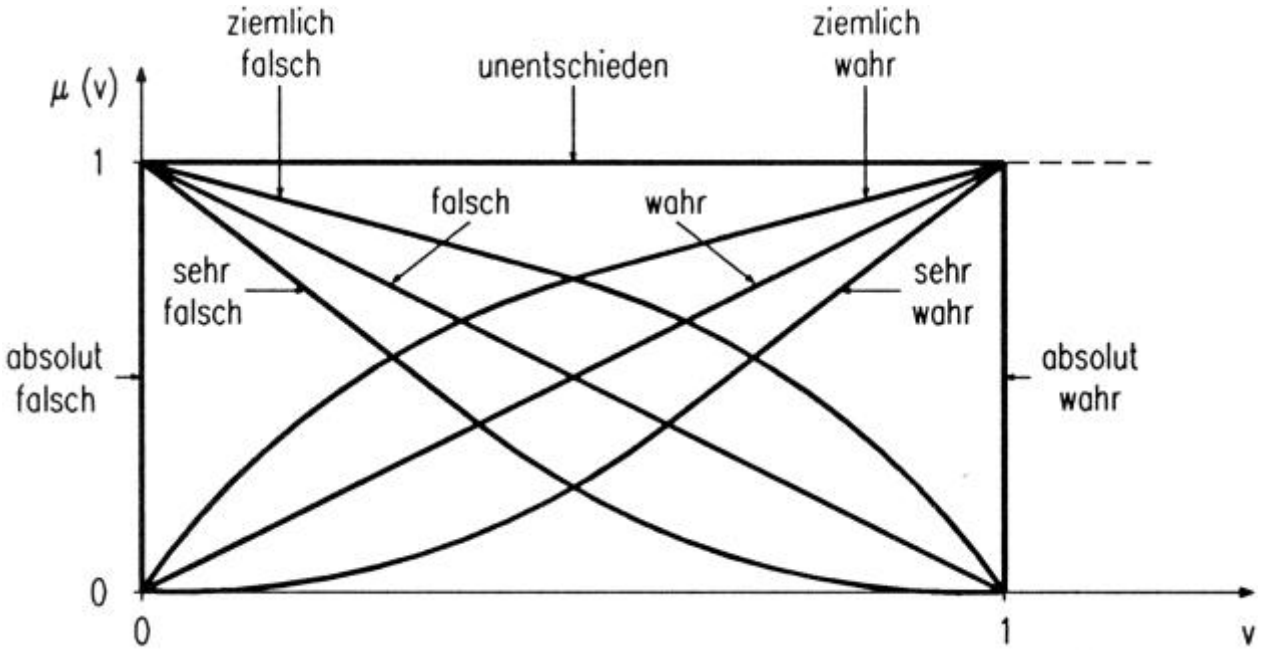

Abbildung 3-4: Auswirkung Linguistischer Modifikatoren auf die Linguistische Variable *wahr*

Beispiel 3-3: Ein möglicher syntaktischer Aufbau der linguistischen Variable Geschwindigkeit mit dem Primärterm schnell, dem Antonym langsam, sowie einer Auswahl möglicher Modifikatoren

Linguistische Variable			"Geschwindigkeit"
Linguistische Werte (Terme)	Primärterm	:	schnell
	Antonym	:	langsam
Modifikator		:	sehr, mehr, kaum, nicht, ...

Abbildung 3-5: Syntaktischer Aufbau der Linguistischen Variablen „Geschwindigkeit“

3.1.6. Weitere Schlüsselkonzepte am Beispiel eines Reglers

Will man die im Grunde geschichtlich scheinbar artfremden Disziplinen objektorientierter Programmierung und der Fuzzy-Logic einander näher bringen, so ließe sich an dieser Stelle resümieren, dass mit den oben genannten Begriffen die Klassen der Fuzzy-Logic beschrieben sind, die Einführung ihrer Methoden aber noch aussteht.
Dies soll nun an einem regelungstechnischen Beispiel geschehen. Nicht, dass die hier zu lösende Aufgabe einen direkt regelungstechnischen Hintergrund hätte, die Motivation hierfür rührt eher daher, dass die Regelungstechnik die Haupttriebfeder der Fuzzy-Logic war und daher für die Beschreibung der Fuzzy-Logic-Methoden besonders geeignet ist und im übrigen die ganze Fuzzy-Logic an sich auf der Anwendung von Fuzzy-Reglern beruht. Also wird auch hier versucht, Fuzzy-Logic-Abfragen auf Datenbanken als regelungstechnische Aufgabe zu interpretieren:
„Nenne alle Datenbank-Einträge die schnelle Autos betreffen.“, erzeugt einen Vorgang in dem alle Datenbank-Einträge mit dem Attribut Objektart=Auto ermittelt werden. Deren Wert der Domäne Höchstgeschwindigkeit wird als Eingangssignal an einen Regler gelegt, der ausgangsseitig den Zugehörigkeitswert zur Linguistische Variablen „schnell“ als Stellwert liefert. Man sagt in dem Regler sei die Regel „Wenn maximale Geschwindigkeit ist hoch dann Interesse ist hoch.“ implementiert.
Wir erhalten damit drei mögliche Darstellungen für die Beispielbedingung[3] :

Natürlichsprachlich	Nenne alle Datenbank-Einträge die schnelle Autos betreffen
Synthetische Sprache/ SQL-Syntax	SELECT *.* FROM PIA.db WHERE Kategorie=Auto AND Geschwindigkeit=schnell
Fuzzy-Regel	IF Kategorie=Auto AND max. Geschwindigkeit ist hoch THEN Interesse ist hoch

[3] Später wird eine Mischform der zweiten und dritten Darstellung zur Realisation vorgeschlagen werden.

Wenn das Ausgangssignal einen Schwellwert (α-cut, s.u.) überschreitet, wird die Regel als erfüllt angesehen und der Datensatz der Ergebnismenge zugeordnet.
Einen besonderen Reiz erhält diese Sicht auf unscharfe Abfragen, wenn man damit verschachtelte Definitionen realisiert. So ließe sich durch eine Regel "Wenn Gebrauchtwagenwert nach Durchschnittstabelle ist größer als Kaufpreis, dann Angebotsqualität ist hoch" ein vordefinierter, unscharfer, zusammengesetzter Wert *Angebotsqualität* festlegen, der seinerseits als Attribut einer Bedingung verwendet werden kann:
„IF Angebotsqualität ist hoch und Preis ist niedrig THEN Interesse ist hoch".
Mit dieser Umsetzung einer natürlichsprachlichen Abfrage in den Bedingungsteil eines Reglers, wird dieser zur Schnittstelle zwischen der Datenbank-Query und der Fuzzy-Logic.

Die Vorgänge in dem Regler werden durch drei Begriffe beschrieben, die im folgenden näher erläutert werden:
- Fuzzifikation,
- approximatives Schließen
- Defuzzifikation.

3.1.7. Fuzzifikation

Die Eingangswerte x_i des Reglers sind in der Regel, insbesondere und ausdrücklich nach dem Ansatz von PIA, scharfe Werte. Sie müssen zunächst auf die linguistische Werteskala abgebildet werden, auf der das Reglerverhalten in Form von IF...THEN...-Regeln vorgegeben ist, da die Ausgangswerte $y_i(x)$ auf der Basis dieser Regeln hergeleitet werden. Die x_i werden also zunächst in den Zugehörigkeitsraum der beteiligten linguistischen Terme transformiert. Wenn eine linguistische Variable durch n Terme beschrieben wird, dann entsteht als fuzzifiziertes Signal ein n-dimensionaler Vektor $\mathbf{s}(x)$ mit den Elementen $\mu_i(x) \in [0,1]$, i=1,...,n (siehe auch Beispiel 3-4). Man spricht auch vom Sympathievektor des Eingangswertes. So bedeutet s(22°C)= kalt/0,2 +mittel/0,7 +warm/0,7 in Zadeh-Mengennotation (s.u.), dass 22°C zu 0,2 kalt ist und zu jeweils zu 0,7 den mittleren und den warmen Temperaturbereichen zuzuordnen ist. Dieser Vektor wird beim Herleiten der Folgerungen weiterverarbeitet. Die Transformation heißt Fuzzifikation der Eingangssignale. Dieser Begriff ist insofern unglücklich gewählt, als dass es sich weniger um eine Verunschärfung handelt, als vielmehr um die Abbildung, der im allgemeinen scharfen Eingangswerte, auf die linguistische Werteskala des Reglers.

Beispiel 3-4: (Fuzzifikation)

In einigen Fällen kann es sinnvoll sein, die scharfen Eingangsmesswerte zunächst in unscharfe Werte umzuwandeln, bevor sie in den unscharfen Regler gelangen; die Motivation dafür liegt in der grundsätzlichen Toleranzbehaftung von Messungen, wie sie in der Fehlerabschätzung der Messtechnik theoretisch behandelt ist und berücksichtigt eine subjektive, bzw. objektive Vorabbewertung der Toleranzintervalle. Dieses Vorgehen ist dann sinnvoll, wenn beispielsweise aus Kostengründen relativ ungenaue Messwertaufnehmer oder A-D-Umsetzer gewählt werden. Im Falle eines Datensatzes einer Datenbank als Eingangswert entspricht dieses Vorgehen einem unscharf abgespeicherten Datensatz. Dies kann sowohl explizit („VB 1000 €" oder „Fahrzeug ist schnell") geschehen, als auch implizit („TÜV bis 2003" oder „Marke BMW", ohne Angabe des Typs).

3.1.8. Approximatives Schließen

Mit Hilfe von Linguistischen Variablen werden zur Beschreibung des Systemverhalten eine Reihe von Aussagen getroffen, die für bestimmte Eingangskombinationen die dazugehörigen Ausgangswertkombinationen in Form von IF...THEN-Regeln festlegen. Es seien u und v zwei linguistische Variablen (z.B. „Geschwindigkeit“, „Ankerstrom“ eines Elektrofahrzeugs), a und b deren Terme (z.B. groß und recht klein). Dann können die linguistisch formulierten Regeln zur Systembeschreibung auf folgende Art auch durch Implikationen (u=a)=> (v=b) beschrieben werden

p: u ist a
q: v ist b
IF p THEN q : (u=a) => (v=b) (modus ponens).

Einen anderen Ansatz mit gleichem Ergebnis verfolgen die modifizierten Horn-Klauseln. Diese können zu Implementierungszwecken angemessener sein. Der Ansatz findet sich in [DeDo90].
Bei Vorgabe konkreter (scharfer oder unscharfer) Eingangswerte für den Regler müssen zur Berechnung der Ausgangswerte zunächst die Einzelaussagen überprüft werden. Da die Eingangswerte i.a. nicht exakt die in den Implikationen festgelegten Bedingungen befriedigen werden, können nur unscharfe Schlüsse bzw. Folgerungen (engl. Inferences) gezogen werden. Der dazu erforderliche Vorgang wird als ***Approximatives Schließen*** bezeichnet und leitet sich aus den Methoden des Schließens in der Binärlogik ab. Eine einfache und beliebte Methode zur Umsetzung eines approximativen Schlusses ist in Beispiel 3-5 dargestellt. Sie wird als Max-Min-Inferenz-Methode bezeichnet und setzt Regeln in der ersten Normalform voraus. Das Modul, das diesen Schluss leistet, heißt auch Inference-Engine

Beispiel 3-5

Es soll der scharfe Stellwert φ_{aus} der Öffnung eines Kühlventils aufgrund gemessener scharfer Werte T_{EIN} der Eingangsgröße „Temperatur“ verstellt werden. Es liegen zwei Regeln in linguistischer Form vor. Die Terme werden der Abbildung 3-6 entsprechend durch unscharfe Mengen repräsentiert. Die Regeln zur Beschreibung des Reglerverhaltens lauten:

IF Temperatur = niedrig THEN Kühlventil = halb offen,
IF Temperatur = mittel THEN Kühlventil = fast offen.

Es wird die Temperatur T_{EIN}=18°C gemessen. Zur Berechnung des dazugehörigen Stellwerts φ_{AUS}(18°C) werden zunächst die beiden Zugehörigkeitswerte $\mu_{niedrig}$(18°C)=0,2 und μ_{mittel}(18°C)=0,5 zu den Termen *niedrig* und *mittel* der Temperatur ermittelt (Fuzzifikation). Diese werden als resultierende Aktivierungsgrade der Regeln angesehen[4] und zum Ausfüllen der beiden Zugehörigkeitsfunktionen $\mu_{halb\ offen}(\varphi)$ und $\mu_{fast\ offen}(\varphi)$ der dazugehörenden unscharfen Terme der Kühlventilöffnung übernommen (Maximumbildung). Der endgültige unscharfe Ausgangswert entsteht durch Überlagerung der beiden Einzelergebnisse (Minimumbildung). Er wird durch eine unscharfe Ausgangsmenge repräsentiert, aus der man den benötigten Einstellwert gewinnt (siehe 3.1.9 Defuzzifikation).

[4] Falls ein Aktivierungsgrad gleich Null ist, sagt man auch, dass die dazugehörende Regel nicht feuert.

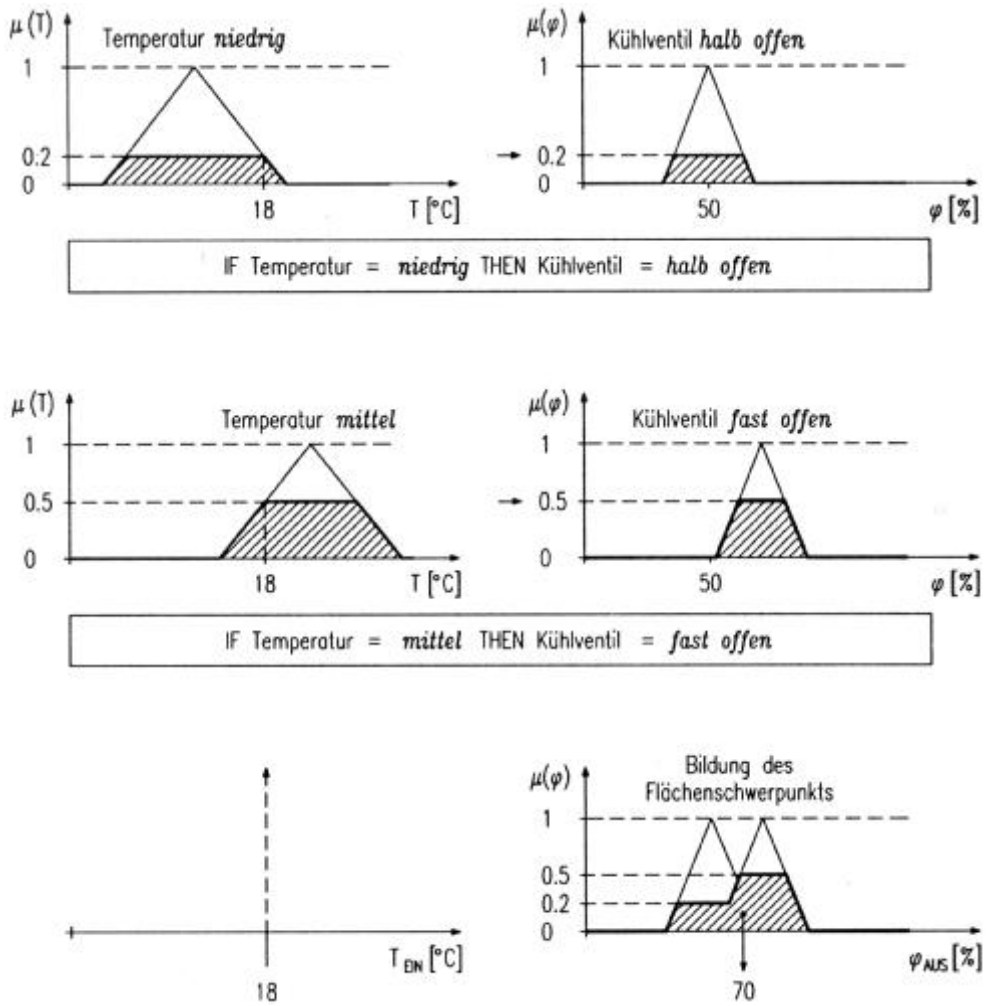

Abbildung 3-6: Max-Min-Inferenz-Methode
Das Verhalten eines unscharfen Reglers zur Einstellung der Kühlventilöffnung P_{AUS} in Abhängigkeit von einer gemessenen Temperatur T_{EIN} wird durch zwei linguistisch formulierte Regeln festgelegt. Bei einem Eingabewert von 18°C wird als Ausgabewert eine Öffnung φ=70% eingestellt.

3.1.9. Defuzzifikation

Um aus den unscharfen Ausgangsmengen für die einzelnen Stellgrößen (als Repräsentanten der unscharfen Schlussfolgerungen) scharfe Einstellwerte zu erzeugen, wird oft die Fläche unter den Kurvenverläufen der Zugehörigkeitsfunktionen herangezogen. Der Abszissenwert des Flächenschwerpunkts wird als resultierender scharfer Stellwert herangezogen. In Beispiel 3-5 ergibt sich damit der scharfe Stellwert φ_{aus}=70% der Kühlventilöffnung. Dieser Vorgang wird als Defuzzifikation der Ausgangsgröße bezeichnet.
Andere Anwendungen verwenden hier den kleinsten Wert, bei dem sich das Maximum der Kurve findet. Für unseren Anwendungsfall entspricht der so gewonnene Wert dem Zugehörigkeitswert (MD) der Bedingung.

3.1.10. Linguistic Quantifier

Im Bereich der Konzepte der Fuzzy-Logic sei abschließend eine Funktion beschrieben, die man als Bewertung von Fuzzy-Aussagen beschreiben könnte. Als Beispiele mögen dienen:

1. „Die meisten Cabrios sind teuer“
2. „Die meisten der folgenden Bedingungen müssen erfüllt sein“

Trotz der gleichen Wortwahl sehen wir in den beiden Beispielen zwei grundlegend unterschiedliche semantische Konzepte. Das erste Beispiel stellt den Vertrauenswert einer Aussage dar. Er bewertet die Aussage „Cabrios sind teuer.“ mit „stimmt meistens“.
Dagegen finden wir im zweiten Beispiel einen Mechanismus zur Gewichtung von Verknüpfungsoperatoren wieder. Die Zugehörigkeitswerte eines Datensatzes zu einzelnen Bedingungen, die gewöhnlich Und- bzw. Oder-verknüpft sind[5], werden als Gruppe bewertet.
Es wird sich aber zeigen, dass sich die beiden unterschiedlichen Bedeutungen mit dem selben mathematischen Ansatz behandeln lassen. Die zwei grundlegenden mathematischen Konzepte zur Behandlung von Fuzzy Quantifiern, die Deutung nach Zadeh bzw. gemäß OWA, werden im Laufe des nachfolgenden Abschnitts über die Mathematik der Fuzzy-Logic behandelt. Das Umdeuten des Vertrauenswertes als Gewichtung von Verknüpfungsoperatoren findet sich im Kapitel Lösungsansatz.

3.2. Mathematik der Fuzzy-Logic

Auf dem Gebiet der unscharfen Mengen existieren mehrere unterschiedliche axiomatische Systeme. Regeln und Gesetze gestalten sich je nach aktueller Auswahl der verwendeten Verknüpfungen unterschiedlich, wobei die Widerspruchsfreiheit oft unerfüllt bleibt. Die Gesamtheit der bekannten Axiomatiken kann als „Theorie der unscharfen Mengen“ bezeichnet werden.
Die Definition unscharfer Mengen geht auf Lotfi A. Zadeh [Za65] zurück und gestaltet sich bis auf die Schreibweise einheitlich. Im folgenden wird ausschließlich die übliche Mengendarstellung gewählt, wobei die Elemente als Wertepaare (Linguistische Variable/ Zugehörigkeitswert) in runden Klammern geschrieben und mit einem Semikolon getrennt werden.

3.2.1. Diskrete vs. kontinuierliche Mengen

Definition 3-1: Unscharfe Menge

Sei X eine Menge von Elementen bzw. Objekten x, die hinsichtlich einer unscharfen Aussage mit einem Wahrheitswert $\mu_A(x)$ auf Zugehörigkeit zu bewerten sind. Dann ist die Menge **A** der Wertepaare $(x; \mu_A(x))$ mit

$$\mathbf{A} = \{(x; \mu_A(x)) \mid x \in X, \mu_A(x) \in R\}$$

eine unscharfe Menge auf **X** mit der Zugehörigkeitsfunktion $\mu_A(x)$. Die Menge **X** heißt „Grundbereich“ oder „Grundmenge“ von **A.**

Beispiel 3-6

Die Menge aller reellen Zahlen ungefähr gleich 8 lässt sich darstellen als

$$\mathbf{A}=\{(x; \mu_A(x)) \mid \mu_A(x) = [1+ (x - 8)^4]^{-1}\}.$$

Der Zugehörigkeitsverlauf $\mu_A(x)$ stellt sich dann nach Abbildung 3-7 dar:

[5] D.h. gemäß klassischem Ansatz wird nur der kleinste, bzw. größte Wert für das Ergebnis in Betracht gezogen.

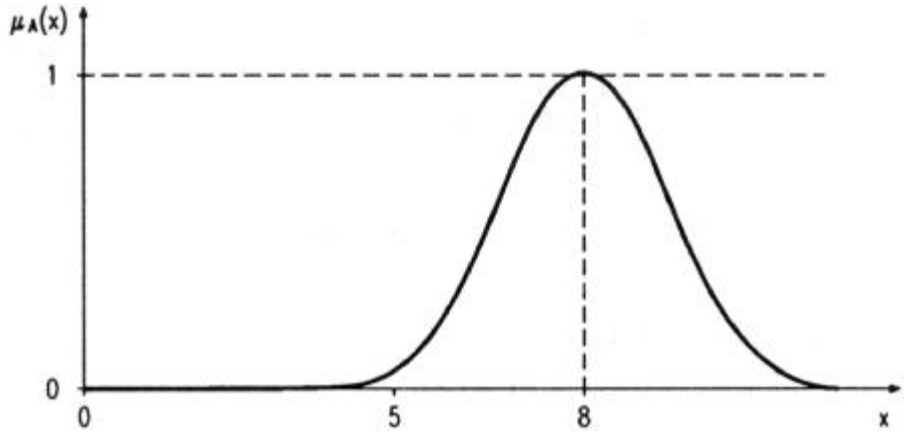

Abbildung 3-7: Zugehörigkeitsfunktion μ(x)= Funktion $[1+ (x - 8)^4]^{-1}$

Die hier verwendete Funktion $[1+ (x - a)^4]^{-1}$ ist wegen verschiedener mathematischer Eigenschaften für theoretische Betrachtungen in der Fuzzy-Logic sehr beliebt zur Darstellung von Zugehörigkeitsfunktionen. In der Praxis haben sich jedoch allgemein die Trapez- und die Dreiecksfunktion durchgesetzt. Obwohl diese Funktionen augenscheinlich nicht so präzise die weichen Übergänge der Bewertungen beschreiben können, genügen sie in einer Vielzahl der Fälle zum Modellieren der Unschärfe. Sie zeichnen sich dadurch aus, dass die vier Grundoperationen +, -, * und / sich ebenso leicht berechnen lassen, wie der Vergleich zweier Fuzzy-Sets. Die Bedeutung der Trapezfunktionen in der Fuzzy-Logic wird in der Literatur ausgiebig behandelt (siehe [KaZa97b] S. 42).

3.2.2. Konstruktion von Zugehörigkeitsfunktionen

Die Vorgehensweise zur Festlegung der Zugehörigkeitsfunktion einer unscharfen Menge A bei einer speziellen Problemstellung kann grundsätzlich auf zwei unterschiedliche Arten erfolgen [Bo87]. In beiden Fällen werden die Elemente $x \in \mathbf{X}$ bezüglich eines definierten unscharfen Kriteriums von einem Experten nach Zugehörigkeit bewertet:

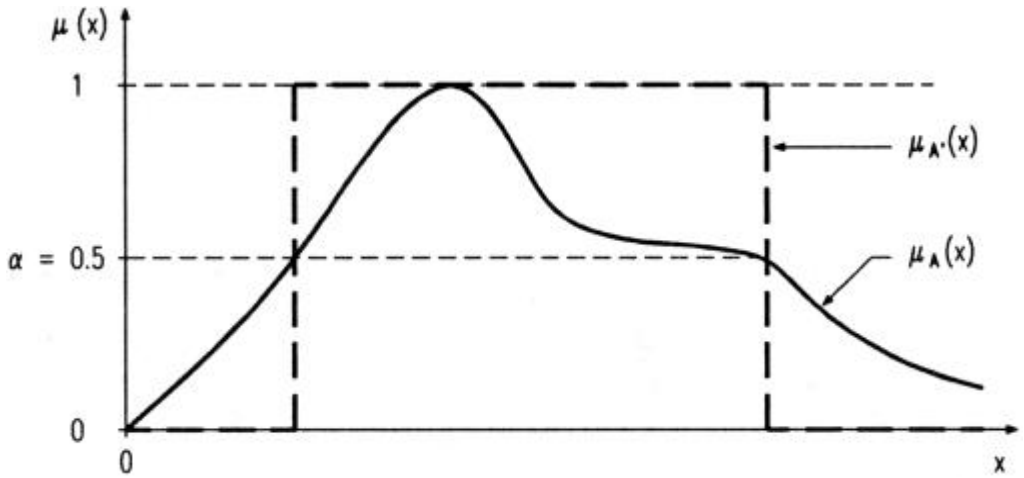

Abbildung 3-8: Konstruktion einer Zugehörigkeitsfunktion aus kontinuierlichen Teilmengen

1. Ausgehend von einer scharfen Teilmenge **A‘** $\subseteq$ **X** wird angenommen, dass die Zugehörigkeit der Randelemente zur Menge A geringer ist, als die der Zentralelemente, und die Randelemente eine Ausstrahlung über die (scharfen) Grenzen hinaus auf außerhalb liegende Nachbarelemente besitzen. Diese erhalten damit ebenfalls eine gewisse Zugehörigkeit. Zum Modellieren von $\mu_A(x)$ kann zunächst ein Randniveau a festgelegt und der Verlauf von $\mu_a(x)$ mit Hilfe von bekannten und geschätzten Systemeigenschaften modelliert werden.
   Ausgehend von einer geschätzten scharfen Menge A‘ kann eine realistische Zugehörigkeitsfunktion $\mu_A(x)$ dadurch konstruiert werden, dass Nachbarschaftsbeziehungen zwischen den einzelnen Elementen x den scharfen Kurvenverlauf deformieren. Die Nachbarschaftsbeziehungen können sich beispielsweise aus einem stochastischen Systemwissen als Wahrscheinlichkeitsverteilungen oder aus einer subjektiven Einschätzung ergeben.
2. Alle Elemente bzw. Objekte $x \in X$ sind a priori bekannt. Beispielsweise kennt man alle Einstellungen eines Systems, um sie bewerten zu können. Nachbarschaftsbeziehungen in Form von Ausstrahlungsmodellen sind nicht gefordert. Die Elemente liegen diskret vor.

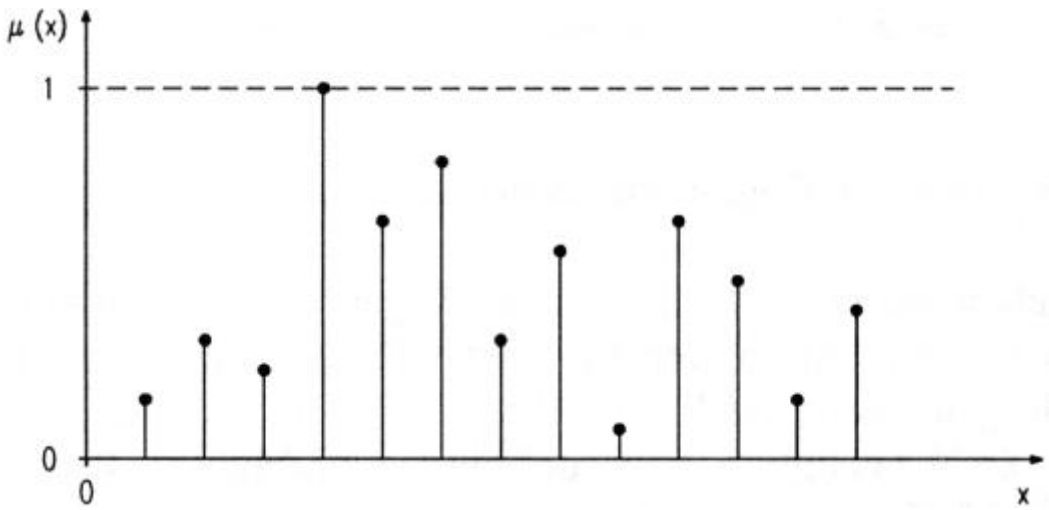

Abbildung 3-9: Konstruktion einer Zugehörigkeitsfunktion aus diskreten Teilmengen

Eine große, beziehungsweise repräsentative Auswahl diskreter Elemente ist bekannt und kann direkt bewertet werden. Falls die Bewertung von Zwischenwerten notwendig wird, kann sie durch Interpolation erfolgen.

Die Erzeugung von Zugehörigkeitsfunktionen hat sich als grundlegendes Problem der Fuzzy-Logic herausgestellt und war deshalb Thema intensiver Forschungen. Neben der Definition von Unschärfe mittels Normalisierung wurde der Einsatz Neuronaler Netze als möglicher Ansatz zur Behebung dieses Problems gefunden. Typische Ansätze benutzen RBNF’s (Radial Basis Network Functions), deren Ausgangssignale durch Gauß-Glocken-förmige Funktionen erzeugt werden und sich leicht als Fuzzy-Zugehörigkeitsfunktionen deuten lassen. Mittels Erzeugung eines Fehlersignals am Ausgang, d.h. eines Soll-Ist-Wert Vergleichs, kann das Neuronale Netz seine ggf. zufällig initialisierten Parameter korrigieren, ohne die Interpretierbarkeit zu verlieren.
Für den Fall der Definition von Unschärfe mittels Normalisierung, der im vorliegenden Fall zunächst einmal näher liegt, ließen sich verschiedene Möglichkeiten der Autoaggregation der Normierungsparameter vorstellen, die sich vermutlich auf statistische Werte stützen würden. So könnten etwa zur Ermittlung eines üblichen Höchstwerts, wie er in 3.1.4 Definition von

Unschärfe mittels Normalisierung beschrieben ist, Mittelwert und Varianz Verwendung finden und mit dem tatsächlichen Höchstwert abgeglichen werden.

3.2.3. Zwischenbilanz Fuzzy-Logic

Mit der Definition von Fuzzy-Mengen (Geschwindigkeit ist hoch) und Fuzzy-Relationen (Geschwindigkeit ist ungefähr gleich 130 km/h“ oder „Breite ist ungefähr zwischen 10cm und 20cm) sind die wesentlichen Konzepte zur Behandlung von atomaren Queries gegeben. Anders formuliert könnte man sagen, es sei eine Abbildungsfunktion definiert, um jedem Datensatz bei gegebener atomarer, unscharfer Bedingung einen Matching Degree MD=f(Datensatz) zuzuordnen.
Zur Auswertung dieser MDs wird zunächst noch die Funktion des α-cuts beschrieben und eine erste einstellige Mengenoperation, das Komplement, eingeführt.
Damit ist dann die Betrachtung atomarer MDs abgeschlossen.

Es folgen mehrstellige Mengenoperationen, um aus den atomaren MDs zunächst lokale und dann globale MDs zu errechnen, d.h. die MDs der Alternativen und Interessen zu ermitteln.
In diesem Zusammenhang sei die Anforderungsdefinition von PIA noch einmal angeführt[6] Queries in der ersten Normalform unter Verwendung von Gewichtung zu realisieren:

```
select <list of fields>
from <list of tables>
where
cond11 and cond12 and...and cond1k
or...or
condn1 and condn2 and condnm
```

Zur Umsetzung dieser Anforderung werden zunächst die Definitionen für Und und Oder benötigt. Sie werden mit Hilfe des Durchschnitts und der Vereinigung definiert und dann zu allgemeineren Formen hin überführt, bis zum frei skalierbaren γ-Operator.
Im Laufe dieser Überlegungen wird deutlich werden, dass ein Und oder ein Oder für unscharfe Größen eine deutlich komplexere Bedeutung hat als für scharfe Werte. Dies wird im anschließenden Abschnitt noch verschärft, wenn mit Einführung von Gewichtung (Importance) Mengenoperatoren wie „Und vielleicht“ eingeführt werden.
Diese Diskussion wird, von eigenen Überlegungen über den Vergleich zwischen „Und vielleicht“ und „Oder vielleicht“ über sogenannte Fuzzy Quantifier („Die meisten der wichtigen Bedingungen müssen erfüllt sein“) bis zu den neuesten Forschungsergebnissen von R. Fagin führen.
Die Verknüpfung von (Importance-) bewerteten Fuzzy-Mengen geschieht getrennt für Und und Oder. Für den schwierigeren Fall des Und’s wird hierfür die Betrachtung eines eigentlich nicht geforderten Fuzzy-Attributs angeführt, die des Fuzzy Quantifiers. Die Erklärung seiner ursprünglichen Bedeutung, in der Interpretation nach Zadeh, ist dabei eher als Einführung zu verstehen. Die neuere Deutung, gemäß Ordered Weighted Average (OWA) dagegen liefert gleichermaßen das Instrumentarium für einen allgemeinen Verknüpfungsoperator γ wie für Und-verknüpfte gewichtete Bedingungen. Außerdem trifft es sich, dass dieser Operator in der für die prototypische Realisation verwendeten Software FfA implementiert ist. Er bietet mit seiner Fähigkeit Verknüpfungsoperatoren frei zu definieren, die Möglichkeit, sich von der Beschränkung auf die 1. Normalform zu lösen.

6 (in SQL-Notation, siehe 4.4 Query-Design)

3.2.4. α-cut

In Fuzzy-Mengen müssen häufig diejenigen Elemente identifiziert werden, die gewissen Mindestanforderungen an eine Eigenschaft entsprechen. Dieser Auswahlvorgang wird mittels des sogenannten α-cut's gelöst.

Definition 3-2 α-cut

Sei **A** eine unscharfe Menge mit $\mathbf{A}=\{(x; \mu_A(x)), x \in \mathbf{X}\}$. Dann heißt

$\mathbf{A}_\alpha=\{x \in \mathbf{X} | \mu_A(x) \geq \alpha\}$

der α-Schnitt (α-cut) von **A**.

$\mathbf{A}_\alpha$ ist eine scharfe Menge mit den Elementen $x \in \mathbf{X}$, für deren Zugehörigkeitsfunktion $\mu_A(x) \geq \alpha$ ist mit α als positiver reeller Zahl. Im Fall einer Normalisierung entsprechend $\mu_A(x): x \rightarrow [0,1]$ muss $\alpha \in [0,1]$ gelten.
Im Fall $\forall x \in \mathbf{X}: \mu_A(x) > \alpha$ spricht man auch von einem strengen α-cut. Dieser findet in der prototypischen Realisierung Anwendung. Bildlich gesprochen werden alle Elemente des zu bewertenden Attributs, deren Zugehörigkeitswert einen bestimmten Schwellwert α überschreitet, einer scharfen Menge A_α zugeordnet.

3.2.5. Komplementbildung und weitere Modifikatoren

Die Komplementbildung ist eine einfache einstellige Operation. Ihre Bedeutung in der Fuzzy-Logic ist eine Transformationsvorschrift zu liefern, die beispielsweise die Aussage „Das Auto ist schnell." überführt in „Es ist falsch, dass das Auto schnell ist". Diese Aussage ist nicht notwendigerweise gleich mit „Das Auto ist nicht schnell".

Definition 3-3 Komplement:

Sei $\mathbf{A} \in \mathbf{P}(X)$. Wenn $\mu_A: \mathbf{X} \rightarrow [0,1]$ gilt, heißt $\mathbf{A}^C$ das Komplement von **A** auf **X,** mit

$\forall x \in \mathbf{X}: \mu_{A^c}(x)=1-\mu_A(x)$

Weitere einstellige Operatoren sind die oben eingeführten Modifikatoren. Im Rahmen der praktischen Umsetzung dieser Arbeit werden alle einstelligen Operatoren durch vordefinierte Zugehörigkeitsfunktionen modelliert. So wird etwa nicht $\mu_{schnell}$ durch den Modifikator ***sehr*** transformiert, sondern $\mu_{sehr\ schnell}$ direkt definiert.

3.2.6. Verknüpfungsoperatoren

Im Folgenden werden Verknüpfungsoperatoren behandelt, mit denen sich komplexe unscharfe Bedingungen auswerten lassen (Etwa „{Größe ist groß und Preis ist mittel} oder {Größe ist mittel und Preis ist niedrig}").
Es handelt sich hierbei um die Übertragungen boolescher Funktionen in die Methodik der Fuzzy-Logic. Bereits Lotfi A. Zadeh [Za65] gelang es ein allgemeines Prinzip zur Transformation beliebiger klassischer mathematischer Funktionen in die Algorithmik der Fuzzy-Logic zu definieren. Diese Methodik wird als Erweiterungsprinzip bezeichnet.
Auf diese Weise ergeben sich für das Und und das Oder durch Herleitung aus der Booleschen Algebra folgende Definitionen:

Definition 3-4 Und, Oder

Sei $\mathbf{A_i}=\{\mathbf{(x,}\ \mu_i(x)) \mid x \in X\}_{i=1,..,n}$. mit $\mathbf{A_i} \in \mathbf{P}(X)$. Dann bezeichnet man als

und-Verknüpfung der $\mathbf{A}_i$ die unscharfe Menge mit

$$\mu_{Und}(x) = \min [\mu_1(x), ..., \mu_n(x)]$$

oder-Verknüpfung der $\mathbf{A}_i$ die unscharfe Menge mit

$$\mu_{Oder}(x) = \max [\mu_1(x), ..., \mu_n(x)]$$

Tatsächlich haben sich diese Definitionen in der Fuzzy-Logic durchgesetzt, da sie in ihrer Einfachheit an Performanz nicht zu überbieten sind und sie sich für die meisten praktischen Anwendungen als qualitativ ausreichend herausgestellt haben.
Dennoch resultiert ihr Ergebnis aus der Erfülltheit einer einzigen Bedingung (Min oder Max). Anschaulich lassen sich daher aus dem Stand heraus eine Reihe sinnvollerer Algorithmen finden, wie beispielsweise der arithmetische und der geometrische Mittelwert:

$$\mu_{Und}(x) = [\mu_1(x) \cdot ... \cdot \mu_n(x)]^{1/n}$$
$$\mu_{Oder}(x) = 1/n\ [\mu_1(x) + ... + \mu_n(x)]$$

Für eine höchstmögliche Präzision der Ergebnisse wurde gar ein sogenannter Gamma-Operator entwickelt, der mittels einer numerischen Konstanten beliebige Einstellungen in einem kontinuierlichen Übergang zwischen Min und Max ermöglicht:

Definition 3-5 Gamma Operator

Seien $\mathbf{A_i} \in \mathbf{P}(X)$. Die γ-Verknüpfung der $\mathbf{A_i}=\{\mathbf{(x,}\ \mu_i(x)) \mid x \in X\}$ führt zu einer unscharfen Menge A_γ mit der Zugehörigkeitsfunktion

$$\mu_\gamma(x) = [\Pi_i\ \mu_i(x)]^{1-\gamma} \cdot [1 - \Pi_i\ (1 - \mu_i(x)]^{\gamma}$$

und $\gamma \in [0,1]$ als Kompensationsgrad.

Mit diesem, auch als „Kompensatorisches Und“ bezeichneten Operator lassen sich alle oben angeführten Verknüfungsoperatoren beschreiben.

3.2.7. Linguistic Quantifier und Importance

In diesem Abschnitt sollen verschiedene Möglichkeiten beleuchtet werden, wie sich Teile zusammengesetzter Bedingungen quantifizieren lassen. Hierzu zählen zum einen die Methoden zur Behandlung von Linguistischen Quantifizierern nach L. Zadeh und OWA (Ordered Weighted Average). Diese lassen sich zusätzlich mit Gewichtung (Importance) verknüpfen und treten in diesem Sinne in eine Gruppe mit zwei weiteren Ansätzen zur Behandlung von Gewichtungen: Einem neueren Ansatz von R. Fagin und einer selbst entwickelten Theorie. Diese Eigenentwicklung bietet einen besonders einfachen Ansatz den Projektanforderungen von PIA nach Gewichtung zu genügen, ohne den Ballast der Allgemeingültigkeit der anderen Theorien genügen zu müssen.
Um einen Begriff, wie Gewichtungs-Behandlungsroutinen o.ä. zu vermeiden, sei diese gesamte Gruppe vom Berechnungsvariationen für Gewichtung in Folge als Quantifier bezeichnet.

Reguläre Quantifier

Quantifizierer seien durch Beispiele wie „für *alle* oder *für die meisten* oder *für einige* Werte gilt“ eingeführt. Die konventionelle Logik kennt derer Art Beispiele nur mit scharfen Werten, wie „für alle“ oder „für einen“. Mit solchen Quantifizierern lassen sich unscharfe Regeln

definieren, wie „Nur wenige große Autos sind billig.“ oder, wie in unserem Anwendungsfall, „Die meisten der angegebenen Bedingungen müssen erfüllt sein.“.
Diese regulären Quantifier werden zwar vom PIA-Projekt nicht gefordert, sind aber in der für den Prototypen genutzten Software implementiert. Außerdem stellen sie die übliche Lösung dar, um Gewichtungen zu unterscheiden.

Allgemein lässt sich schreiben

Q y’s sind F

mit Q als Linguistic Quantifier (z.B. „die meisten“), Y={y} als Menge von Objekten (z.B. Experten) und F als Eigenschaft (z.B. „befragt worden“). Der gleiche Sachverhalt lässt sich noch mit einer Gewichtung (Importance) belegen:

Q Imp y’s sind F .

In diesem Fall lässt sich eine Regel/Bedingung wie „Die meisten der wichtigen Experten sind befragt worden.“ modellieren.

Quantifier nach Zadeh

Der erste praktikable Weg mit Quantifiern umzugehen wurde von Zadeh beschrieben [Za83].
Der Einfachheit halber, und weil sein Ansatz nur ein Schritt auf dem Weg zu einem anderen für diese Arbeit wichtigen Ziel ist, sei sein Ansatz nur mit Beispielen erläutert.
Er definierte

$$\mu_{most}(\mu)= \begin{cases} 1 & \text{für } \mu > 0.8 \\ 2\mu-0.6 & \text{für } 0.3 \le \mu \le 0.8 \\ 0 & \text{für } \mu < 0.3 \end{cases} \quad (1)$$

Man erkennt hierin die Abbildung **R à R**, eines Zugehörigkeitswertes auf einen anderen. Es wird gewissermaßen die Erfülltheit einer Erfülltheit bewertet. Dabei kann die zu bewertende Erfülltheit eine Komposition von Zugehörigkeitswerten sein, die besonders dann Sinn macht, wenn ein nicht hartes Und eingesetzt wird.

Für den Fall vorhandener Gewichtung ist die Berechnungsgrundlage deutlich komplizierter.
Hier definierte Zadeh zunächst eine Nicht-Fuzzy-Kardinalität, ein sogenanntes *sigma-count*, die als ΣCount(A) notiert wird. Hier die Definition für eine diskrete Fuzzy-Menge $\mathbf{A}= \mu_A(x_1)+...+ \mu_A(x_n)$

$$\Sigma\text{Count}(\mathbf{A})= \Sigma\mu_A(x_i) \ .$$

Mit

Q Imp y’s sind F .

Er schließt damit auf

$$\mu_{\text{Q Imp y's sind F}} .= \mu_Q(\Sigma\text{Count}(\mathbf{Imp} \text{ und } \mathbf{F}) / \Sigma\text{Count}(\mathbf{F}))$$

Im Falle des Oder-Operators, in unserer Notation also auf Ebene der Verknüpfung gewichteter Alternativen, führt diese Definition zur Bewertung eines arithmetischen Mittelwerts über den mit der Gewichtung (weight) multiplikativ gewichteten MDs der Alternativen. Der so gewonnene Wert wird wie im obigen Beispiel mit der Zugehörigkeitsfunktion des gewählten Quantifiers bewertet:

$$\mu_{most}(\text{Interest})= \mu_{most}(\Sigma\text{Imp}_i*\mu(\text{alternative}_i)/\Sigma\text{Imp}_i)$$

Eine detailliertere Diskussion findet sich in [Ya83] und Kacprzyk und Yager [KaYa84a] [KaYa84b].

Quantifier nach OWA

Zadeh's Ansatz wurde 1988 von Yager [Ya88] verallgemeinert zum OWA-Operator (Ordered weighted Average oder geordneter gewichteter Durchschnitt). Damit ist es möglich sowohl Zadeh's Definition nachzubilden als auch erweiterte Formen der Interpretation durchzuführen. Als Spezialfälle dieser Erweiterungen treten die Verknüpfungsoperatoren Und und Oder auf, sowie deren kontinuierlicher Übergang, wie wir ihn beim γ–Operator kennen gelernt haben.

Quantifier nach Fagin

Ronald Fagin und E. Wimmers stellten diesen Ansätzen zur Behandlung von Gewichtung jüngst einen neuen Algorithmus entgegen, der in [FaWi97] beschrieben wird. Die entwickelte Formel wird als „A Formula for Incorporating Weights into Rules" bezeichnet oder wegen ihrer Einfachheit „The Magic Formula".
Sei f (M) eine beliebige Verknüpfungsfunktion (z.B. Und oder Oder), für die gilt:

$f(M): [0,1]^n \rightarrow [0,1]$
mit $M=(\mu_i)$ ist ein Vektor von Zugehörigkeitswerten.

Sei weiter

$Q=(\theta_i)$ ein M zugeordneter, absteigend geordneter Gewichtsvektor
mit $i=1,...,n$,
$\theta_k > \theta_m \ , \forall\ k<m$
und $\Sigma\ \theta_i = 1$.

Dann gilt

$f(Q, M): [0,1]^{2 \times n} \rightarrow [0,1]$
mit $f(Q, M) = (\theta_1 - \theta_2) \cdot f(\mu_1)$
$+2 \cdot (\theta_2 - \theta_3) \cdot f(\mu_1,\mu_2)$
$+3 \cdot (\theta_3 - \theta_4) \cdot f(\mu_1,\mu_2,\mu_3)$
$+...+$
$+n \cdot \theta_m \cdot f(\mu_1,...,\mu_n)$

Bemerkung:
Der geforderte absteigend geordnete Gewichtsvektor erleichtert die Notierung und Evaluierung der Funktion. Um die Sortierung zu gewährleisten, ist gegebenenfalls M umzustellen.

Quantifier für PIA

Für die Queries die mit PIA realisiert werden sollen sind weder die beliebigen Verknüpfungsfunktionen aus Fagin's Ansatz f (Q, M) noch die Linguistic Quantifier Q aus Zadeh's Ansatz notwendig. Es sollen lediglich Und- bzw. Oder-verknüpfte Bedingungen gewichtet werden können.
Zwar fand sich in der Literatur kein geeignetes einfaches Verfahren, doch lässt sich durch Überlegung leicht ein einfacher Lösungsansatz finden, dessen Widerspruchsfreiheit sich im Rahmen dieser Arbeit allerdings nicht belegen ließ.
Hierfür unterscheide man die logische Bedeutung der Gewichtung einzelner Bedingungen im Rahmen einer Oder- und im Rahmen einer Und-Verknüpfung.

Im Falle einer Oder-Verknüpfung, die durch die übliche Maximumbildung realisiert sei, und einer angenommenen Standard-Gewichtung von 1 (keine Normierung $\Sigma\ \theta_i = 1$, wie bei Fagin)

bedeutet jede Gewichtung, die kleiner als dieser Wert ist, eine Abschwächung der Bedeutung dieser Bedingung (siehe Abbildung 3-10). So kann aus der einfachen Oder-Funktion:

Oder $(M)=\mu_{max}: [0,1]^n$ **à** $[0,1]$
mit $M=(\mu_i)$ ist ein Vektor von Zugehörigkeitswerten.

mittels der linearen Funktion $\mu'_i = \theta_i \cdot \mu_i$, die Oder-Funktion für gewichtete Bedingungen abgeleitet werden:

$$\text{Oder } (Q, M)=(\theta_i \cdot \mu_i)_{max}=(\mu'_i)_{max}: [0,1]^{2 \times n} \text{ à } [0,1] \ .$$

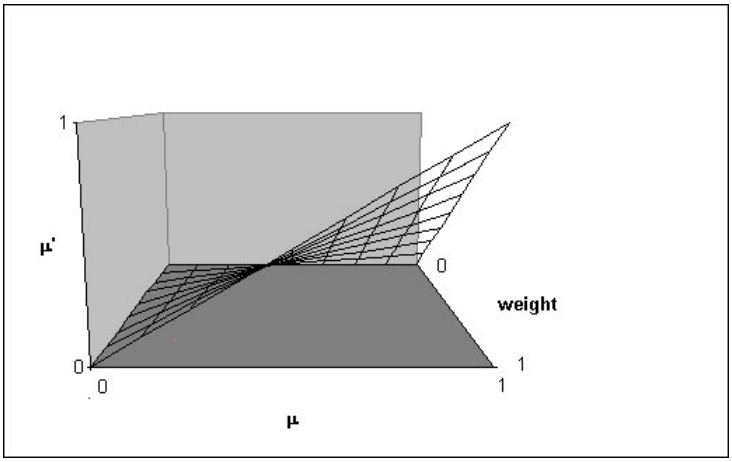

Abbildung 3-10: Gewichtung für Oder-Verknüpfungen

Im Falle einer Und-Verknüpfung dagegen, die durch die übliche Minimumbildung realisiert sei, soll aus der Nichterfülltheit unwichtiger Einzelbedingungen nicht auf die Nichterfülltheit der Komposition geschlossen werden (siehe Abbildung 3-11). Insbesondere soll ein Element der Gewichtung 0 immer die Zugehörigkeit 1 haben (um im Rahmen der Minimumbildung nicht zur Bestimmung der Lösungsmenge beizutragen), während ein Element der Gewichtung 1 immer auf seine reale Zugehörigkeit abgebildet werden soll. Für Zwischenwerte der Gewichtung soll die Erfülltheit der Bedingung zur 1 „hingezogen“ werden, um im Sinne des Und „unwichtiger“ zu werden. So wird aus der einfachen Minimum-Bildung ungewichteter Bedingungen

Und $(M)=\mu_{min}: [0,1]^n$ **à** $[0,1]$

mittels der linearen Funktion $\mu'_i = (\mu_i - 1) \cdot \theta_i + 1 = \theta_i \cdot \mu_i + (1 - \theta_i)$, die Und-Funktion für gewichtete Bedingungen abgeleitet werden:

$$\text{Und } (Q, M)=(\theta_i \cdot \mu_i + (1 - \theta_i))_{min} = (\mu'_i)_{min} : [0,1]^{2 \times n} \text{ à } [0,1] \ .$$

Die Graphen für die Und- und die Oder-Verknüpfung realisieren für $\theta_i=1$, also die Standardgewichtung, die Identität der Zugehörigkeitswerte $\mu'_i = \mu_i$.
Für $\theta_i=0$ dagegen, also der Auszeichnung einer Bedingung als „unwichtig“, realisieren die Graphen das Unbeachtetlassen dieser Zugehörigkeit unabhängig vom seinem Wert - also $\mu'_i = 0$ im Falle der Oder-Verknüpfung und $\mu'_i = 1$ im Falle der Und-Verknüpfung.

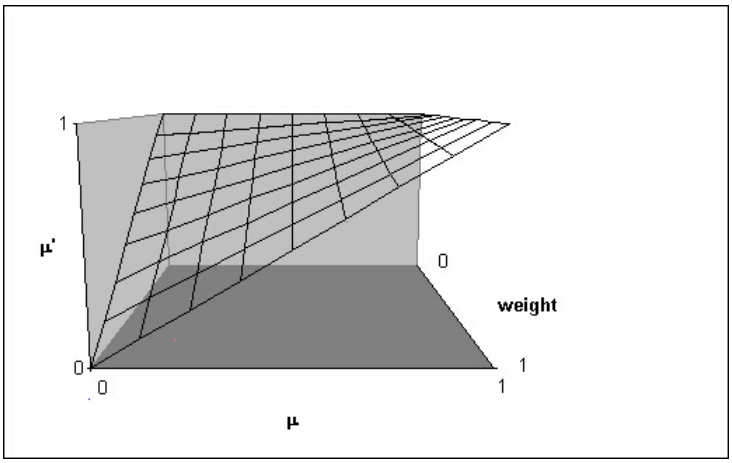

Abbildung 3-11: Gewichtung für Und-Verknüpfungen

Die Funktion des Graphen für die Oder-Verknüpfung $\mu'_i = \theta_i \cdot \mu_i$, ist offensichtlich die einfachste Funktion die den Flächen-begrenzenden Geraden

$\mu'_i (\mu_i = 0) = 0$,
$\mu'_i (\theta_i = 0) = 0$,
$\mu'_i (\mu_i = 1) = \theta_i$
$\mu'_i (\theta_i = 1) = \mu_i$

genügt.
Die entsprechende Gleichung für die Und-Funktion $\mu'_i = (\mu_i - 1) \cdot \theta_i + 1$ findet sich durch die einfache lineare Transformation

$\mu'^{T}_i = \mu'_i - 1$,
$\mu^{T}_i = \mu_i - 1$,
$\theta^{T}_i = \theta_i$,

so dass dieselbe Flächenform gegeben ist.

Dass es noch weitere Lösungen für die gegebenen Randbedingungen gibt, ist sicher. Diese werden aber im Allgemeinen komplizierter ausfallen als die hier vorgestellte. Die Gleichung ist eine Spezialfall der allgemeinen Quadrikengleichung, mit der Flächen zweiter Ordnung im Raum beschrieben werden.
Mit der gefundenen Lösung scheint jedenfalls eine deutlich einfachere Berechnung der Gewichtung, als die in der Literatur vorgeschlagenen Varianten (s.o.) möglich zu sein.

3.3. weitere Ansätze

Neben den vorgenannten Ansätzen finden sich in der Literatur weitere Quellen, die sich mit dem Thema Fuzzy-Zugriffe auf Datenbanken in der einen oder anderen Weise auseinandersetzen. Diese können aus quantitativen Gründen hier nur insofern angeführt werden, wie sie einen essentiellen Aspekt behandeln, der noch einer genaueren Untersuchung bedarf, oder als wichtige Literaturquelle dienen können, für ein vertiefendes Studium bestimmter Problematiken.

3.3.1. Erste Grundlagenarbeit: FRDB

Der vollständige Titel dieser Arbeit lautet Fuzzy relational data bases – a key to expert systems, [KaZe84] verfaßt von Maria Zermankova-Leech als Dissertation an der Florida State University

im Jahre 1984 mit Abraham Kandel als betreuendem Professor, der noch heute eine wichtige Rolle in der Erforschung von Informationstechnologien spielt. Die Arbeit, als Buch weltweit veröffentlicht, wird zu Recht in beinahe jeder Arbeit zitiert, die sich mit dem Themengebiet von Fuzzy-Datenbanken auseinandersetzt, da sie trotz ihres Alters von 16 Jahren ein solides theoretisches Fundament auf diesem Feld liefert.
Dem selbstgesteckten Ziel ein Schlüssel für Expertensysteme zu liefern indes kann sie heute nicht mehr gerecht werden. Heutzutage beruhen Expertensysteme nämlich nicht mehr primär auf den Techniken der Fuzzy-Logic, sondern auf ausgefeilten Methoden der Wissensrepräsentation, der Verkettung von Regeln und des approximativen Schließens, sowie effizienten Suchalgorithmen in großen Lösungsräumen. Die Fuzzy-Logic spielt in diesen Ansätzen nur noch eine untergeordnete Rolle im Sinne eines mathematischen Instruments.
Maria Zermankova-Leech hebt in ihrer Arbeit ausschließlich auf die Aspekte der Speicherung und Verarbeitung unpräziser Daten, sowie des unscharfen Schließens ab. Da in dieser Arbeit von präzisen Daten ausgegangen wird und der Schlußfolgerungsprozeß dem User übertragen wird, sei diese Arbeit lediglich wegen Ihrer grundsätzlichen Relevanz erwähnt.

3.3.2. Fuzzy Prolog databases

In dieser Quelle finden sich wichtige Hinweise zur Behandlung des in der Datenbankwelt immer wieder kritischen Problems von Nullwerten. Damit ist die Behandlung von Unbestimmtheit in einzelnen Datenfeldern gemeint, also etwa wenn das Baujahr eines Autos schlicht nicht angegeben wird. Neben der Möglichkeit andere Merkmale des Datensatzes in semantische Beziehung zu dem fehlenden Wert zu setzen, werden in dieser Arbeit Wege aufgezeigt, über Wahrscheinlichkeitsverteilungen sogenannte FEV's (Fuzzy Expected Values) zu definieren. (Vergleiche hierzu auch [Du91] „Approximate Reasoning by Analogy to Answer Null Queries.“)

3.3.3. Architekturmodelle

Bosc und Pivert beschreiben 1997 in ihrem Artikel „Extending SQL Retrieval Features for the Handling of Flexible Queries” Systemerweiterungen relationaler Datenbanksysteme um unpräzise Abfragen behandeln zu können. Sie behandeln Syntax und Semantik ihrer erweiterten Komponenten, sowie die dazu nötigen Spracherweiterungen von SQL. Sie nennen ihre Sprache SQLf. Außerdem beschäftigen sie sich mit der Realisierbarkeit und Implementierungsaspekten, welche uns in diesem Zusammenhang besonders interessieren.
Die Autoren schreiben, die Berechnung von Queries im Rahmen deklarativer Sprachen wie SQL sei ein offenes Problem in sich[7]. Im Allgemeinen könne bei einer gegebenen Query kein optimaler Algorithmus zu deren Berechnung gefunden werden. Wegen der höheren Komplexität, die bei der Berechnung unscharfer Queries notwendig ist, gelte diese Effektivitätsbeschränkung hier besonders[8]. Die Erhöhung der Komplexität beruhe vor allen Dingen auf der größeren Menge an Daten, die für die Berechnung der Zugehörigkeitswerte benötigt werde und der Durchführung von Berechnungen, im Gegensatz zu einfachen booleschen Vergleichen.
Dies in Bedacht könne man zur Realisierung, der von ihnen vorgestellten Spracherweiterungen für SQL, zwei Modelle unterschiedlicher Effektivität heranziehen, denen hier zwei weitere Architekturen gegenübergestellt werden sollen.
Die Autoren bezeichnen ihre Ansätze als a) „derivation“ Strategie und b) „direct evaluation“ Strategie.

[7] Anmerkung: Wegen des Problems mittels deklarariver Konstrukte Prozeduren auszuführen.
[8] Die technische KI, wie sie sich in den Ingenieurswissenschaften findet, beschäftigt sich schwerpunktmäßig mit der Optimierung von Suchalgorithmen in komplexen Lösungsräumen.

a)
Der Name der „derivation“ (engl. Ursprung) Strategie rührt wohl daher, dass hierbei die Fuzzy-Query auf eine Standard Datenbank-Query zurückgeführt wird. Das genaue Vorgehen ist leider nicht angeführt. Statt dessen wird auf [BoPi93] verwiesen. Da sich aber die angeführten Beschränkungen nicht mit den für diese Arbeit gestellten Anforderungen decken, genügt hier die angegebene Zusammenfassung des Artikels.

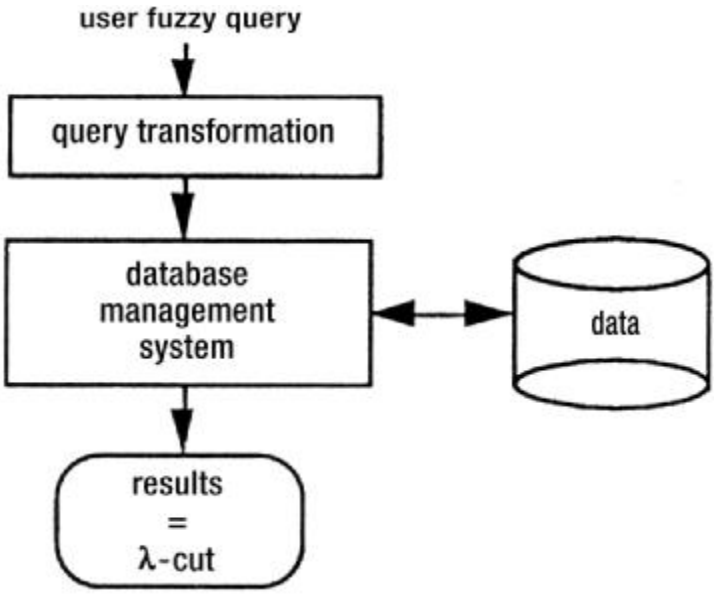

Abbildung 3-12: Architekturmodell „Derivation“

Die Architektur der Strategie wird in Abbildung 3-12 beschrieben. Die Fuzzy-Query wird mittels „query transformation“ auf eine gewöhnliche SQL-Query abgebildet. Dabei wird gefordert, dass die Query einen λ-cut[9] enthalte. Kern dieser Lösung scheint die Implementierung eines nicht näher spezifizierten, „komplexen Prädikats“ zu sein, dass die Berechnung des λ-cuts vornimmt. Es handelt sich dabei vermutlich um eine „stored procedure“ im DBMS. Als Einschränkung dieses Systems wird angegeben, es seien damit weder komplex verschachtelte Abfragen möglich, noch die Verwendung von Linguistic Quantifiers, die im PIA-System zum Anforderungsprofil gehören. Auf der anderen Seite sei das System in der Lage, die effektiven Abfragemechanismen der gängigen DBMS’s zu nutzen.

b)
Bei der zweiten Lösungsstrategie („direct evaluation“) beruht die Architektur auf einem speziellen Fuzzy-DBMS (Abbildung 3-13). Dieses spezielle Datenbanksystem trennt die Query in elementare Unterfunktionen, die mit Hilfe effizienter Algorithmen berechnet werden, die als direkter Zugriff auf die Datenbank ausführbar sind. Diese Funktionen können beispielsweise Ermittlung des Sympathie-Vektors, Verknüpfung von Fuzzy-Mengen, Komplementbildung und ähnliche sein. Erste beispielhafte Implementierungen seien in [Bo97] zu finden. Das Erscheinungsdatum der Referenz war 1997.

[9] entspricht dem in dieser Arbeit beschriebenen α-cut

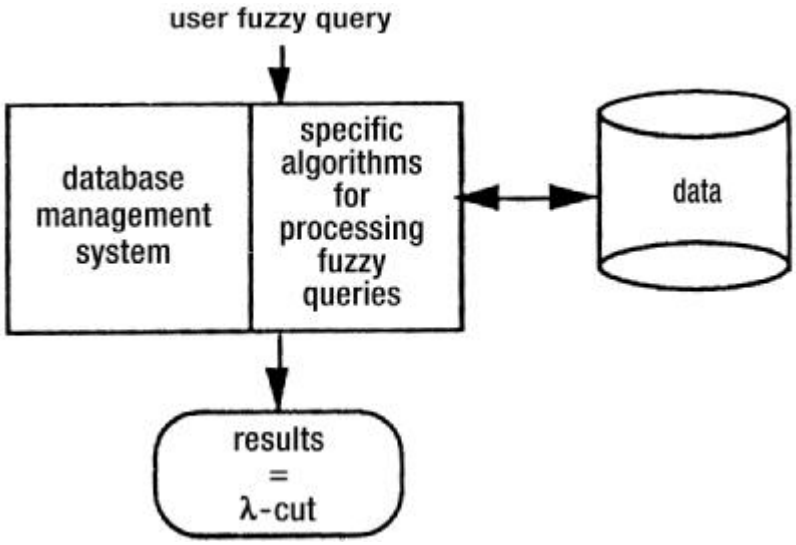

Abbildung 3-13: Architekturmodell „Direct Evaluation“

Die Effizienzsteigerung dieser Vorgehensweise lässt sich am Beispiel folgender Query nachvollziehen:

```
Select λ * from R where [fc1 and exists
                                (select * from S where fc2 and R.A θ S.B)].
```

Die Bedeutung der Terme im Einzelnen lässt sich in dem Artikel nachlesen, erläutert sei aber der Zusammenhang zwischen der ersten und der zweiten Zeile der Query. Die zweite Zeile der Query liefert nämlich eine eigene Tabelle und, ausgedrückt durch das „exists“ der ersten Zeile, die Bedingung der ersten Zeile wird nur auf diese Tabelle angewandt. So müssen die Zugehörigkeitswerte der ersten Bedingung nur auf den relevanten Datensätzen ausgeführt werden. Weitere verbesserte Algorithmen dieser Art finden sich auch in [Bo95].
Dieser zweite Ansatz umgeht die beschriebenen Restriktionen deklarativer Sprachen, da er in dem Fuzzy-Datenbanksystem eine prozedurale Sprache kapselt.

c)
Da weder der erste, noch der zweite dargestellte Ansatz im Rahmen dieser Arbeit zu einem befriedigenden Ergebnis führt, bzw. zu realisieren ist, wird diesen beiden im Abschnitt „Lösungsentwurf“ eine dritte Architektur gegenübergestellt, die ein **Pre-** und ein **Postprocessing** der Query vorsieht. Im Preprocessing wird die Anfrage in einen Fuzzy- und einen Datenbank-relevanten Teil zerlegt. Dann werden die relevanten Daten über ein SQL-Query aus der Datenbank abgefragt und sequentiell im Postprocessing genannten Teil von einem Fuzzy-Regler bewertet. Der Fuzzy-Regler wird mittels Importance-Koeffizienten von den Fuzzy-spezifischen Teilen der Query gesteuert (adaptiver Regler).
Dieser Ansatz vereint den Vorteil der ersten Strategie, mit Standard-Datenbank-Systemen arbeiten zu können, mit denen der zweiten, komplexe Anfragen zu ermöglichen. Sein Nachteil ist der Verlust von Effizienz. Zu deren Steigerung müsste die Datenbank-Query mit dem adaptiven Regler interagieren, was die Implementierung höherer Intelligenz erforderte, als sie hier vorgesehen ist.

d)
Als vierter Gesichtspunkt möglicher Systemarchitekturen sei noch die Software FfA erwähnt, der wegen ihrer grundsätzlichen Bedeutung für diese Arbeit, ein eigenes Kapitel zugeteilt wurde, welches an dieses anschließt.
Die mit dieser Software durchgeführte Lösung, lässt sich als eine Art Simulation der zweiten Strategie (direct evaluation) deuten. Die Implementierung setzt auf dem bekannten Datenbanksystem Access von Microsoft auf. Die vor dem User gekapselte prozedurale Sprache, die die Fuzzy-Funktionen evaluiert, ist Visual Basic for Applications (VBA). Von Simulation des zweiten Ansatzes muss man insofern sprechen, als dass die Prozeduren nicht direkt in der DBE realisiert werden, sondern die Datensätze, ähnlich der dritten Strategie, nach dem Auslesen nachbewertet werden. Damit gibt diese Lösung auch die Vorteile der zweiten Architektur Preis, eine hohe Effizienz zu gewährleisten.

3.4. Zusammenfassung (Stand der Technik)

In diesem Kapitel wurden eine Vielzahl von Möglichkeiten und Ansätzen aus der bisherigen Forschung dargestellt. Es hat sich einerseits gezeigt, dass die Fuzzy-Logic aus dem Stadium eines sich erst entwickelnden Forschungsgebiets in die alltägliche Anwendung übergegangen ist. Andererseits wurde sie offensichtlich vor gut zehn Jahren als Grundlage für intelligente Datenbankzugriffe entdeckt. Seitdem sind einige Forschungsaktivitäten zu verzeichnen gewesen. Trotzdem sich schon einige Anwendungen beispielhaft im World Wide Web befinden (siehe www.heise.de/suchen), hängt die Selbstverständlichkeit, mit der diese Anwendungen Verwendung finden, doch denen der klassischen Fuzzy-Logic hinterher. Da die Anwendung in Informationssystemen aber wenigstens genauso vielversprechend und dringend erforderlich scheint, wie in den klassischen Anwendungsgebieten, ist bei der erkennbar hohen Dynamik des Informationsmarktes mit einer raschen Entwicklung und Verbreitung dieser Technologie zu rechnen.
Im folgenden Kapitel wird ein über Jahre gepflegtes und fortentwickeltes Beispiel einer solchen Anwendung genauer dargestellt.

Kapitel 4 - FfA (Fuzzy for Access)

Zur Realisation eines Prototyps wurde die Software FuzzyForAccess (FfA) eingehend betrachtet. Sie wurde am Systems Research Institut der Polish Acadamy of Sciences entwickelt. Federführend war dabei der renommierte Forscher und Verfechter der Fuzzy-Logic Janusz Kacprzyk.
In einem Artikel zu diesem Thema [KaZa97] beschreibt er das Ziel dieser Anwendung so:
Die Funktionalität von Microsofts Access 2.0 wird exemplarisch für alle gängigen RDBMs um Fuzzy-Techniken erweitert. Ziel sei es, der Anfragen oft eigenen Ungenauigkeit Rechnung zu tragen. Die Anwendung werde in kommerziellen Projekten eingesetzt. Hierfür ließen sich leider keine Beispiele finden. Sicher jedoch ist, dass die Software bereits auf eine geraume Entwicklungsgeschichte seit 1986 zurückblicken kann. Die einzelnen Entwicklungsstufen werden unter dem Dach der FQUERY-Familie zusammengefasst. Zunächst entstand eine Folge von dBASE basierten DBMSs (FQUERY I bis Version III+). In den letzten Jahren dagegen konzentriert man sich am Institut ganz auf die Entwicklung in der Access-Umgebung (FQUERY for Access v.1, v.2, 97WWW). Dabei sprechen für die Verwendung dieser verbreiteten Software verschiedene Aspekte. In dem Artikel wird die offene Architektur und die einfache Implementierbarkeit des FfA-Tools als Add-In (4.2.3 Add-In) genannt. Sicherlich ist aber auch die Verbreitung von Access und der Programmiersprache Visual Basic for Applications (VBA) ein wichtiger Aspekt, der der Fortentwicklung der Software dienlich ist - wie diese Arbeit zeigt.
Außerdem zeichnet sich Access mit dem Quasi-Standard ODBC als offenes System aus. Über diese Schnittstelle kann Access auf eine Vielzahl gängiger Datenbanksysteme zugreifen. Die enge Bindung der systeminternen, prozeduralen Sprache VBA an die Datenbanksprache SQL tut ein übriges hinzu.

Um die von der FQUERY-Familie angestrebten Fuzzy-Elemente unterstützen zu können, müssen im wesentlichen drei Dinge vorgehalten werden:

1. Es muss eine Fuzzy-Syntax definiert werden, die in die bestehende SQL-Sprachstruktur eingebunden werden kann.
2. Die so entstandenen Fuzzy-Objekte müssen leicht editierbar sein, d.h. es muss eine entsprechende Schnittstelle implementiert werden.
3. Der Syntax muss eine entsprechende Semantik hinterlegt werden, die die erzeugten Abfragen beantworten kann.

Die Beschreibung der Fuzzy-Syntax erfolgt im anschließenden Abschnitt mit einer formalen Erweiterung der SQL-Syntax. Daran schließt der Abschnitt Implementierung mit der Beschreibung der semantischen Umsetzung an. Abschließend wird in FfA-Toolbar die Schnittstelle zur Verwaltung der Fuzzy-Objekte beschrieben.

4.1. Query-Design

Die Standard SQL-Syntax für eine Datenbank-Anfrage in der ersten Normalform lässt sich wie folgt beschreiben:

```
select <list of fields>
from <list of tables>
where
cond11 and cond12 and...and cond1k
or...or
condn1 and condn2 and condnm
```

mit $cond_{ij}$ als *atomaren Bedingungen*, im Sprachgebrauch der FfA-Entwickler. Im PIA-Objektmodell (Abbildung 2-1) entsprechen diese Bedingungen den *Kriterien.* Sie beschreiben die kleinste, d.h. nicht mehr zusammensetzte Abfrage-Bedingung, wie

```
Preis>100
```

oder

```
Lagerbestand<Zahl der Anforderungen.
```

Eine Und-Komposition atomarer Bedingungen wird als *Unterbedingung* bezeichnet (PIA: *Alternative*).
Eine Oder-Komposition der Unterbedingungen ergibt eine *Sequence of Subconditions* (PIA: *Interesse*).
Diese Form einer Bedingung wird in der Datenbank-Theorie auch als erste Normalform bezeichnet.

In FQUERY wird dieses Schema nun um Fuzzy-Terme erweitert. Die verallgemeinerte Form lässt sich wie folgt beschreiben:

```
<query> :=
select <list of fields>
from<list of tables>
where <fuzzy quantifier>
<sequence of subconditions> ;

<sequence of subconditions> :=
<subcondition> |
<subcondition> or <sequence of subconditions>

<subcondition> :=
<fuzzy quantifier> <importance coefficient>
<sequence of atomic conditions>

<sequence of atomic conditions>:=
<atomic condition> |
<atomic condition> and
<sequence of atomic conditions>

<atomic condition> |:=
<attribute> = <fuzzy value> |
<attribute> <fuzzy relation> <attribute> |
<attribute> <fuzzy relation> <number> |
<andere Formen in gültiger Microsoft Access Syntax>

<attribute> := <numeric field>

<fuzzy quantifier> := <OWA-tag> <quantifier name> |

<OWA - tag> := OWA |

mit | als Kennzeichen einer alternativen Erscheinungsform.
```

Wir sehen hierin im wesentlichen vier Fuzzy-Konzepte realisiert:
- fuzzy quantifier,
- fuzzy relation,
- fuzzy value,
- importance coefficient

Bemerkenswert dabei ist, dass die Gewichtung auf Ebene der Subconditions (Alternativen), die Quantifier auf Ebene der Subconditions und Sequence of Subconditions (Alternativen und Interessen) definiert sind. Dagegen fordert das Objektmodell von PIA speziell für die Gewichtung die Zulässigkeit auch auf der Ebene der atomaren Bedingungen (Kriterien).
Außerdem sei erwähnt, dass das Konzept des Fuzzy-Quantifiers sich in zwei Alternativen spaltet, je nachdem, ob man die Interpretation nach Zadeh oder OWA wählt.

Die verwendeten Begriffe lassen sich wie folgt weiter spezifizieren:

<attribute>

Dieser Begriff beschreibt ein Attribut auf einer numerischen Domäne.
Um als Fuzzy-Term angesprochen werden zu können, muss eine obere (UpperLimit) und untere Grenze (LowerLimit) des betrachteten Wertebereichs definiert werden. UL und LL werden bei der Berechnung des Matching Degrees als Transformationsparameter der Abbildung auf das Einheitsintervall benutzt, auf dem die Fuzzy-Funktionen definiert sind.
Beispielhaft sei ein Auszug aus der Tabelle FfA_Attributes für das Beispiel der mitgelieferten houses.mdb Datenbank gegeben:

Database	Table	Field	Field_range1	Field_range2
HOUSES.MDB	HOUSES	BATHROOMS	1	10
HOUSES.MDB	HOUSES	BEDROOMS	1	10
HOUSES.MDB	HOUSES	CAR SPACE	1	10
HOUSES.MDB	HOUSES	PRICE	50000	1000000

<fuzzy value>

Dies sind unäre Fuzzy-Funktionen, die auf einem Einheitsintervall definiert sind.
FfA verwendet als Einheitsintervall [-10, +10]. Die Form der Zugehörigkeitsfunktionen wird auf Trapezfunktionen beschränkt mit den oben (3.2.1 Diskrete vs. kontinuierliche Mengen) angeführten Vorteilen. Zur Definition einer jeden Funktion ist die Angabe von vier Parametern ausreichend. Ausgehend von einer Zugehörigkeit Null am Anfang und Ende des Intervalls reichen

- Beginn des Anstiegs auf Eins,
- Beginn des Plateau's,
- Ende des Plateau's und
- Ende des Abfalls auf Null

aus, um den Kurvenverlauf zu beschreiben.

Bildbeispiel „groß“:

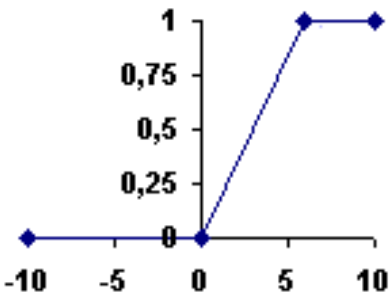

Abbildung 4-1: Linguistischer Wert „large“ = (0, 6, 10, 10)

Um einen Zugehörigkeitswert zu ermitteln wird ein gegebener Datensatz zunächst mittels seiner UL und LL auf das Normintervall [-10, +10] abgebildet und dann auf den Fuzzy Value projiziert.
Die vordefinierten Fuzzy-Values sind:

ID	Name	par1	par2	par3	par4
13	Any	-10	-5	8	8
5	Average	-2	-1	1	2
4	High	0	6	10	10
16	Large	0	6	10	10
9	Low	-10	-10	-5	0
10	Medium	-5	-2	2	5
15	Reasonable	-10	-10	2	8
14	Soon	-10	-10	-5	0
11	Very high	2	8	10	10
12	Very low	-10	-10	-8	-2

<fuzzy relation>

Eine unscharfe Relation wie „not much greater than“ wird als binäre Fuzzy-Funktion mit einer Trapezfunktion abgebildet. Die Operanden können Attributnamen oder Zahlen sein. Um zwei Werte auf die genannte Weise zu vergleichen, wird die Differenz der Werte D auf die Varianz der zugehörigen Bereiche abgebildet [LL1 - UL2, UL1 - LL2], wobei LL1, UL1, UL2, LL2 die Grenzen der beiden zu vergleichenden Werte bezeichnen. Wenn nur ein Operand ein mit UL und LL definiertes Fuzzy-Attribut ist und der zweite Operand eine Zahl, dann wird für den zweiten Operanden angenommen, er habe die gleichen Grenzen, wie der erste. Der Zugehörigkeitswert wird schließlich durch Projektion auf die Zugehörigkeitsfunktion der Fuzzy-Relation gefunden.
Die vordefinierten Fuzzy-Relationen sind:

ID	Name	par1	Par2	par3	par4
10	Around	-2	-1	1	2
15	Definitely greater than	0	2	10	10
16	Definitely less than	-10	-10	-2	0
5	Much greater than	0	5	10	10
3	Much less than	-10	-10	-5	0
17	Not much greater than	-10	-10	0	3
19	Not much less than	-2	0	10	10
13	Rather greater than	-3	-1	10	10
14	Rather less than	-10	-10	1	3
11	Slightly less than	-2	-2	0	0
12	Slightly more than	0	0	2	2

<Fuzzy quantifier>

Im Folgenden werden verschiedene Formen von Quantifiern eingeführt und ihr unterschiedliches Zusammenspiel mit Gewichtungen beleuchtet. Es wird unterschieden nach gesetzter und nicht gesetzter Gewichtung, nach lokaler und globaler Gewichtung und nach der Interpretation gemäß Zadeh oder OWA. Es ergeben sich folgende Fallunterscheidungen:

Keine importance / Gewichtung	OWA	Zadeh
Lokal	Ja a) / Nein	Ja a) / Nein
Global	Ja b) / Nein	Ja b) / Ja c)

Ja und Nein beschreiben, ob das jeweilige Konzept in FfA realisiert wurde. Die drei nicht realisierten Konzepte wurden im Rahmen der prototypischen Realisierung nachimplementiert. Die Buchstaben stehen für die Referenzen der Erläuterungen.

Zunächst die Beispiele ohne Gewichtung:
Mit den Fuzzy-Quantifiern wird ein besonderes Bewertungskriterium für zusammengesetzte Abfragen eingeführt. Dabei geht es um die Bewertung einer zusammengesetzten Bedingung, mit einem linguistischen Ausdruck der Form „Die meisten, der angegebenen Bedingungen müssen erfüllt sein“.
Es wird die defuzzifizierte Zugehörigkeit des Datensatzes zu der zusammengesetzten Bedingung auf ein Einheitsintervall [0,10] [10] projiziert und dort gemäß [Za83] mit einer stückweise linearen Zugehörigkeitsfunktion bewertet.

Beispiel 4-1:

Die Aussage

„Das Auto ist rot und schnell und billig“

ist zu 0,7 erfüllt.
0,7 entspricht nach der Projektion dem Wert 7.
Die Aussage

>>Die meisten Bedingungen der Aussage „Das Auto ist rot und schnell und billig“ sind erfüllt<<

ist gemäß Abbildung 4-2 zu 0,67 erfüllt.

Um speziell die Realisierung dieser Methodik in FfA darzustellen muss zwischen der Anwendung auf Ebene der einzelnen Alternativen oder des Interesses unterschieden werden.

a) Quantifier für Alternativen
Auf der Ebene der Subqueries wird folgendermaßen verfahren:
Es gibt eine Reihe atomarer Bedingungen, die Und-verknüpft sind:

„Preis ist hoch und Geschwindigkeit ist hoch und ...“

Die MDs der einzelnen Bedingungen werden gemäß der Definition der Und-Bedingung zu einem Gesamt-MD verknüpft (Min). Der Gesamt-MD wird wie oben angegeben mit der Zugehörigkeitsfunktion des Fuzzy-Quantifiers bewertet.

[10] aus technischen Gründen abweichend vom sonstigen Einheitsintervall

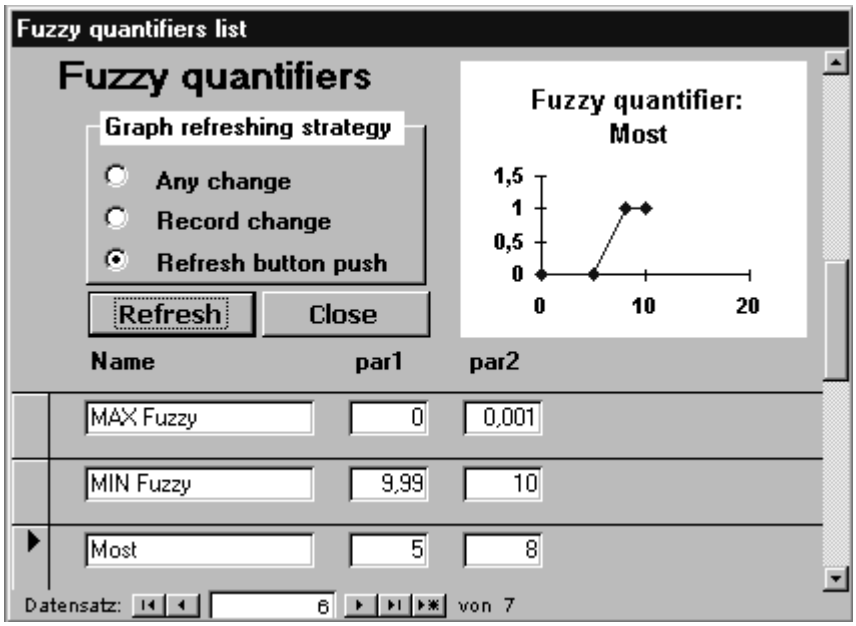

Abbildung 4-2: Fuzzy Quantifier „most“

b) Quantifier für Interessen
Auf der Ebene der (Gesamt-) Query findet sich dagegen folgendes:
Es gibt eine Reihe von Subqueries, die Oder-verknüpft sind:

„Subquery1 oder Subquery2 oder ...“

Die MDs der einzelnen Subqueries werden nun aber nicht, wie ohne Quantifier üblich, gemäß der Definition der Oder-Bedingung zu einem Gesamt-MD durch Maximumbildung verknüpft, sondern zu ihrem arithmetischen Mittel. Der Gesamt-MD wird wie oben angegeben daraus mit der Zugehörigkeitsfunktion des Fuzzy-Quantifiers bewertet.

Dieser Konsistenzbruch durch Umdefinieren der Fuzzy-Oder-Funktion wird im Abschnitt über die OWA-Operatoren noch näher erläutert. Es sei allerdings schon hier auf Abschnitt 3.2.6 Verknüpfungsoperatoren zurückverwiesen, in der verschiedene Formen von Fuzzy-Oder mit Hilfe des kompensatorischen Und‘s erläutert wurden. Gemäß den dort gemachten Erläuterungen sind beide Formen des Oder‘s zulässig.

<Fuzzy importance coefficient>

c) Wenn man verschiedene Subconditions Oder-verknüpft, in denen man beispielsweise zum Ausdruck bringt, dass man ein besonderes Auto für viel Geld kaufen möchte, oder ein gewöhnliches für wenig, so scheint es sinnfällig diesen unterschiedlichen Möglichkeiten unterschiedliche Bedeutungen beizumessen. So könnte man z.B. mehr Wert auf ein besonderes Auto legen.
Diesem Zweck dient der Fuzzy importance coefficient, der mit dem Wert „Eins“ initialisiert ist und dem Intervall [1,0] entstammen muss. Folglich lassen sich mit ihm einzelne subconditions abschwächen. Dies kann einerseits bedeuten, dass die Erfülltheit einer solchen abgeschwächten Subcondition in der ausgegebenen ranked-list zu einer niedrigeren Einstufung führt, oder aber, dass sie wegen des α-cuts dort gar nicht erscheint.
FfA ermöglicht die Gewichtung der Subconditions nur in der Interpretation nach Zadeh.

Für die Prototypische Implementierung gemäß dem PIA-Objektmodell ist außerdem die Gewichtung der Kriterien (Atoms) notwendig. Diese Funktionalität und die Gewichtung von Subconditions nach OWA wurden im Rahmen dieser Arbeit zusätzlich implementiert.

4.2. Implementierung

4.2.1. Versionen

FfA bildet, wie bereits erwähnt, die Grundlage für die prototypische Realisation der Fuzzy-Datenbank. Deshalb wird an dieser Stelle ein Abriss über die Implementierung der Software gegeben. Wegen der Länge des mehrere Tausend Zeilen umfassenden Quellcodes, kann eine vollständige Erläuterung aber nicht erfolgen. Der Code wird auch als zu lang erachtet, um als Anhang beigefügt zu werden. Statt dessen ist er in digitalisierter Form auf CD beigefügt (siehe Anhang A2. Inhalt der CD).

Die Software FfA fand sich auf dem FTP-Server des Systems Research Instituts der Polish Acadamy of Sciences. Mittlerweile existiert der Bereich auf dem Server nicht mehr.
Der WWW-Server des Instituts http://www.ibspan.waw.pl/ schwieg sich seit jeher zum Thema FfA aus. Veröffentlichungen finden sich hauptsächlich zur FfA-Sprache FQUERY auf verschiedenen internationalen Konferenzen von den Autoren Kacprzyk J. und Zadrozny S. .
In dem erwähnten Verzeichnis fanden sich verschiedene Dateien und Versionen der Software. Es sind dies die hier verwendete Version für Access 2.0 und eine Version für das neuere Access 6.0 (gleichbedeutend mit Access97). Die neuere Version wartet zwar mit einigen neuen, zum Teil wünschenswerten Features auf, die in Abschnitt FfA97 Erweiterungen näher erläutert werden, doch leider liegt sie nur in kompilierter Form vor, so dass eine Anbindung an eine selbstdefinierte Schnittstelle nicht möglich war. Die Vorgänger-Version hingegen liegt mit Quelltext vor und das Institut hat auf Anfrage keine Einwände gegen dessen Verwendung vorgebracht. Man zeigte sich im Rahmen seiner zeitlichen Möglichkeiten kooperativ.

Beide Versionen bestehen aus mehreren Dateien:
Eine `readme.txt` gibt Installationshinweise,
`houses.mdb` bietet eine Beispielsdatenbank in der Häuser angeboten werden,
`Ffa_help.hlp` ist eine Windows-Hilfedatei zu dem Programm und
`ffa97www.mde` bzw. `ffa20.mda` stellen die eigentlichen Anwendungen dar.

Die drei „w's" der `ffa97www.mde` stehen laut `readme.txt` für „Webedition", was die Vermutung nahe legt, dass die Software noch in weiteren Formen beim Institut vorliegt. Eine Internet-Funktionalität war in dieser Version jedenfalls nicht zu erkennen. Sollte sich die Performance und die Anpassungsfähigkeit in der praktischen Anwendung als ausreichend herausstellen, so könnte man sich hierum weiter bemühen.

4.2.2. Access rapid description

Die drei Dateiendungen *.mdb, *.mda und *.mde haben folgende Bedeutungen:
*.mdb bezeichnet allgemeine Microsoft Datenbanken,
*.mda bezeichnet spezielle Datenbanken, sogenannte Add-In's (s.u.) und
.mde bezeichnet Datenbanken (.mdb oder *.mda) mit kompiliertem Quellcode.

Der Inhalt von Access-Datenbanken ordnet sich grundsätzlich in sechs verschiedenen Funktionsgruppen ein:

- Tabellen,
- Abfragen,
- Formulare,
- Berichte,
- Makros und
- Module.

Hierin sind Berichte und Makros für unsere Zwecke unbedeutend, „Module“ bezeichnet Programmbibliotheken in VBA und „Tabellen“, „Abfragen“ und „Formulare“ sind bekannte Datenbank Begriffe.

4.2.3. Add-In

Unter einem Add-In versteht man in Access die funktionale Überlagerung einer vorgefertigten, zweiten Datenbank über eine geöffnete erste. Im vorliegenden Fall bedeutet dies, dass zunächst eine mit Fuzzy-Techniken zu behandelnde Datenbank von dem User geöffnet wird (Anwendungs-Datenbank). Dann wird über `Menu Extras, Add-Ins` das vorher dort installierte Ffa20-Add-In zusätzlich aufgerufen. Beide Datenbanken interagieren jetzt. In der internen Repräsentation werden sie vermutlich zu einer Datenbank vereint. Beide Datenbanken sind im übrigen auch manipulierbar. So lassen sich beispielsweise die Tabellen des Add-Ins editieren, wenn die in dem Add-In implementierten Funktionen dies zulassen. Ein direkter Zugriff auf die Tabellen, Formulare, Abfragen und Module des Add-Ins ist dagegen nur möglich, wenn man es direkt als Datenbank aufruft.
Auf diese Weise lassen sich die Funktionen, die in Ffa20.mda implementiert wurden, für beliebige Access-Datenbanken nutzen. Der Anwendungsbereich vergrößert sich auf viele weitere Datenbanksysteme, nämlich um die, die sich mittels ODBC einbinden lassen.

4.2.4. Funktionsprinzip

Die Funktionsweise der Software lässt sich am einfachsten untersuchen, indem man sich die Trennung von Anwendungs-Datenbank und Add-In zunächst wegdenkt. Dies ist leicht zu simulieren, indem man das Add-In direkt in Access öffnet und die Anwendungs-Datenbank als Tabelle importiert. Damit hat man eine Datenbank vorliegen, die sowohl die zu bearbeitenden Daten enthält, als auch die Fuzzy-Funktionen und das User-Interface für den Zugriff, die in dem Add-In implementiert sind.
Zur einfachen Generation von Queries greift FfA auf die Query by Example- (QBE-) Funktion von Access zurück. Dieses Tool erlaubt es dem User, sich Queries gewissermaßen „zusammenzuklicken“. Der Anwender gibt an, aus welcher Tabelle er Daten abfragen möchte (from-clause), welche Bedingungen die Abfrage erfüllen muss (where-clause) und welche Attribute angezeigt werden sollen (select-clause). Außerdem kann er noch Sortierkriterien angeben (order by-clause). Alle Felder, außer der where-clause, lassen sich mittels Pulldown-Menü in der original Access-Funktionalität editieren, so dass eine fehlerfreie Auswahl gewährleistet wird.
FfA stellt nun mittels einer Toolbar genannten Form (siehe FfA-Toolbar) eine Möglichkeit zur Verfügung, die where-clause nach dem gleichen Prinzip zu editieren, wie die select- und from-clause, und zwar mit den spezifischen Fuzzy-Funktionen. Ein spezieller GO-Button evaluiert dann die Anfrage.
Bei der Evaluierung wird zunächst ein „matching degree threshhold“ (FfA-Notation für α-cut, $\alpha_{default}$ = streng 0) abgefragt, dann wird die Query bearbeitet. Die Erzeugung der Antwort

geschieht in zwei Schritten. Zunächst wird der SQL-String der Query[11] nach FfA eigenen Bestandteilen untersucht. Dabei wird zum einen ein Syntaxcheck durchgeführt, zum anderen werden die Parameter der einzelnen Zugehörigkeitsfunktionen der Fuzzy-Operatoren eingelesen und in einer Matrix zur späteren Berechnung vorgehalten. So auf semantische und syntaktische Richtigkeit überprüft wird der SQL-String in eine Access-konforme Schreibweise (s.u.) überführt und von der Access-DBE evaluiert. Das Ergebnis wird wie üblich in einer temporären Tabelle dargestellt, die neben den in der select-clause spezifizierten Attributen zusätzlich ein Attribut MD (Matching Degree) enthält, in der das Bewertungsergebnis dargestellt ist. Vor der Präsentation der Ergebnistabelle wird der alte SQL-String noch wiederhergestellt.
Access-konforme Schreibweise bedeutet, dass VBA-Funtionsaufrufe, wie der der Funktion `FfA_MD (p As Integer) as double`, erlaubt sind, wohingegen sich die von FfA erweiterte SQL-Syntax mehr an dem deklarativen Charakter von SQL orientiert. So wird aus

> SELECT HOUSES.PRICE FROM HOUSES WHERE (((HOUSES.PRICE)=[FfA_FR
> Not much greater than|250000]));

nach der Transformation

> SELECT HOUSES.PRICE FfA_MD(PutValue(0,HOUSES.PRICE,False,0)) As MD
> FROM HOUSES
> WHERE FfA_MD(PutValue(0,HOUSES.PRICE,False,0)) > 0; .

„As MD“ bewirkt die Erzeugung einer neuen Spalte mit dem Attributnamen MD und ist Teil gültiger SQL-Syntax.
„PutValue“ und „FfA_MD“ sind dagegen Funktionsaufrufe von VBA-Funktionen. „FfA_MD“ berechnet die Matching-Degrees, nachdem die Werte des jeweils aktuellen Datensatzes mittels „PutValue“ in die Variablen der Funktion übergeben wurden.
Ein Abriss der implementierten Funktionen ist in Absatz 0 Der Name der Schnittstelle steht für „CGI to Access“. Grundidee dieser Delphi-Applikation ist es die FfA Software in ihrer MS-Access-Umgebung zugänglich zu machen.
Der Webserver Sambar bekommt die Query in Form von Post-Parametern. Diese werden gemäß der WIN-CGI Spezifikation in einer temporären Ini-Dateien abgelegt. Daraufhin wird c2a.exe mit drei Kommandozeilen-Parametern aufgerufen, die Verweise auf temporäre Dateien sind. Dies sind die erwähnte Ini-Datei, eine Datei mit den Parametern im Get-Format und eine Datei die zur Ergebnisausgabe vorgesehen ist. Diese wird vom Webserver beobachtet. Tritt ein Schließen-Ereignis auf ihr auf, so wird sie als Ergebnis an den fragenden Client gesandt.
Um mit den beschriebenen Konzepten auf eine Formularanfrage mit einer html-Seite zu reagieren müssen verschiedene Funktionen implementiert werden

- Übernahme der Kommandozeilen-Parameter
- Auslesen der Ini-Datei
- Übersetzung der Formularparameter in das FfA eigene SQL (FQUERY)
- Ergebnistabelle gemäß Query errechnen
- Ergebnistabelle als html-Datei formatieren
- Ergebnis unter dem spezifischen Dateinamen speichern

Das Access eigene Visual Basic erweist sich für diese Aufgabe als ungeeignet. Es kann beispielsweise keine Kommandozeilen-Parameter übernehmen und enthält auch keine Methoden, die der Syntax von Windows-Ini-Dateien Rechnung tragen. Zur Steuerung der Abfrage wurde daher Delphi gewählt. Das FfA-Access-Modul wird als OLE-Prozess angesprochen.

[11] Die QBE wird intern als SQL-String behandelt und läßt sich auch vom User als solcher betrachten und editieren.

Das Programm besteht aus Initialisierung (Query und Umgebungsvariablen in interne Variablen einlesen), erzeugen eines entsprechenden FQUERY-Strings und Steuerung der Access-Datenbank über OLE (Ergebnis berechnen und als Html-Datei abspeichern). Dabei werden typische Fehlerfälle abgefangen und gegebenenfalls in der Ergebnisdatei als Fehler benannt.
Im Falle einer Fagin-Interpretation von Gewichtung wird die notwendige Sortierung und Normierung vorgenommen. Die Anwendung wird lässt sich über eine Datei d2a.ini steuern, die im gleichen Verzeichnis wie die Applikation selbst liegt.
Die Datei enthält folgende Parameter:

```
[config]
database=\dbms\ffa_ole.mdb
default_in=\tmp\debug\default_ffa.ini
default_query=\tmp\debug\default_ffa.txt
default_out=\tmp\debug\default_ffa.out
Access_OLE =Access.Application.8
```

Die Pfadangaben sind relativ zur Wurzelverzeichnis des Webservers. „database“ ist die Datenbank, die durchsucht werden soll. Die anderen drei Dateien dienen Debugzwecken. Zu Entwicklungszwecken arbeitet das Programm auch ohne Kommandozeilenparameter. In diesem Fall werden die drei genannten Dateien verwendet.
Der Wert von „Access_OLE” findet sich in der Windows-Registry unter “HKEY_CLASSES_ROOT\Access.Application\CurVer”. Es handelt sich um den internen (OLE-)Namen der Access-Version. Er muss gegebenenfalls maschinenabhängig angepasst werden.
Ffa-Ole als Realisation der FDBMS beschrieben.

4.2.5. Installation und Portierung

Die beiden Versionen von FfA (FfA 2.0 und FfA97WWW) sind für unterschiedliche Systeme geschrieben, und, um es vorweg zu nehmen, auf einem heutigen Standard-PC funktioniert zunächst einmal keine der beiden.
Die im Prinzip interessantere Version FfA97 lässt sich zwar problemlos als Add-In in Access installieren und als Tool aus einer Datenbank wie `houses.mdb` aufrufen, doch scheint im Code das Window-Handling nicht sauber installiert zu sein. Die bei geöffnetem, korrektem SQL-String nicht zutreffende Fehlermeldung „A select clause must be opened in the current window“, deutet darauf hin, dass das aktuelle Fenster nicht erkannt wird. Dafür lassen sich vielerlei Gründe vorstellen. Beispielhaft sei erwähnt, das VBA landesspezifische Versionen herausgebracht hat, so dass möglicherweise vergeblich nach einem Objekt gesucht wird, das auf den polnischen Namen für „Window“ hört.
Jedenfalls konnte die Funktionalität dieser Version nicht getestet werden.

FfA20 hingegen wurde für ein heute veraltetes System geschrieben. In einer Testumgebung unter Windows 3.11 arbeitete es einwandfrei mit Access 2.0 zusammen. Allerdings funktionierte auch hier zunächst die Fensterreferenz nicht. Dies ließ sich allerdings, wegen des editierbaren Quellcodes, beheben.
Um es zu einer 32-bit Version von Windows zu portieren, mussten im wesentlichen nur bestehende Aliase auf die neuen Windows32-DLL’s (Dynamic Link Libraries) referenzieren.

Beispiel 4-2: Portierung von FfA20 in 32-bit Windows-Versionen:

```
Declare Function wu_GetWindowText Lib "User32" Alias
"GetWindowTextA" (ByVal h%, ByVal lpstr$, ByVal n%) As Integer
```

statt

```
Declare Function wu_GetWindowText Lib "User" Alias
"GetWindowTextA" (ByVal h%, ByVal lpstr$, ByVal n%) As Integer
```

Nach diesen Änderungen funktionierte FfA 2.0 weitestgehend auch unter Windows 95 bzw. Windows NT mit Access 97.
Da die Editierfunktionen von FfA97WWW funktionieren, nicht aber die von FfA 2.0, es sich mit den Abfragefunktionen aber genau umgekehrt verhält, wird in der prototypischen Realisierung dieser Arbeit FfA97WWW als Editier-Interface benutzt und FfA 2.0 als Abfrage-Schnittstelle.

4.3. FfA-Toolbar

Mit FfA-Toolbar stellt FfA ein User-Interface zur Verfügung. Es ist als Simulation eines Menüs mit Hilfe eines Formulars realisiert.

Abbildung 4-3 : FfA-Toolbar Version 2.0

Wenn das Add-In gestartet wird, öffnet sich das Formular und stellt verschiedene Editierhilfen zur Verfügung. Außerdem gibt es eine Hilfe Funktion, einen Button um das Add-In zu verlassen und einen um die Query auszulösen. Dieser Button „Go!“ ist dem Access-eigenen Button zur Auslösung einer Query nachempfunden (siehe Abbildung 4-4). Er ist notwendig, um die von FfA erweiterte SQL-Syntax in eine Access konforme Darstellung zu übersetzen. Dies geschieht mittels der in VBA implementierten Prozedur „process_query“, die von dem Button ausgelöst wird. Ihr vorgeschaltet ist noch eine Abfrage nach dem „matching degree threshhold“, der bestimmt, ab welchem MD ein Datensatz der Lösungsmenge zugeordnet wird. Wird statt des Go!-Buttons der Access eigene Evaluierungs-Button „!“ gedrückt reagiert Access mit einer Fehlermeldung, da die FfA-SQL-Syntax nicht SQL-konform ist. Sie wird in „process_query“ umgewandelt und dann von der Prozedur ausgelöst.

Abbildung 4-4 : Access GO-Button

Die verbleibenden fünf Schalter dienen dem Editieren der Fuzzy-Funktionen. Zum einen lassen sich damit die Begriffe der vier Fuzzy-Operatoren Fuzzy-Value, Fuzzy-Relation, Fuzzy-Quantifier und Fuzzy-Importance fehlerfrei in Queries setzen, zum anderen lassen sich deren Werte, sowie die der Attribute editieren, was im Folgenden näher erläutert wird.

Editieren von Queries

Im Regelfall dient die Toolbar dem Editieren von Queries. Deshalb verlangt das Betätigen der letztgenannten fünf Schalter zwingend, dass eine Query im Editier-Modus geöffnet ist. Für den Fall, das man nur eine Fuzzy-Funktion editieren möchte, könnte man dies als Bug bezeichnen.
Drückt man bei zum Editieren geöffneter Query auf einen der vier Schalter für die Operatoren, so erscheint zunächst ein Formular, wie in Abbildung 4-5 beispielhaft für die Fuzzy-Relation dargestellt. Hier kann man einen Fuzzy-Operator auswählen, z.B. *Not much greater than.* Er wird dem in der Query ausgewählten Attribut zugeordnet, indem man den symbolisierte Bleistift drückt. Da es sich bei der Fuzzy-Relation um eine binäre Funktion handelt, muss bevor der Operator geschrieben wird, noch der zweite Operand gewählt werden. Er kann entweder ein Attribut der Tabelle sein („Zahl der Badezimmer ist viel größer als Zahl der Garagen“) oder aber ein absoluter Wert („Zahl der Badezimmer ist viel größer als 3“). Die anderen Fuzzy-Operatoren funktionieren entsprechend.

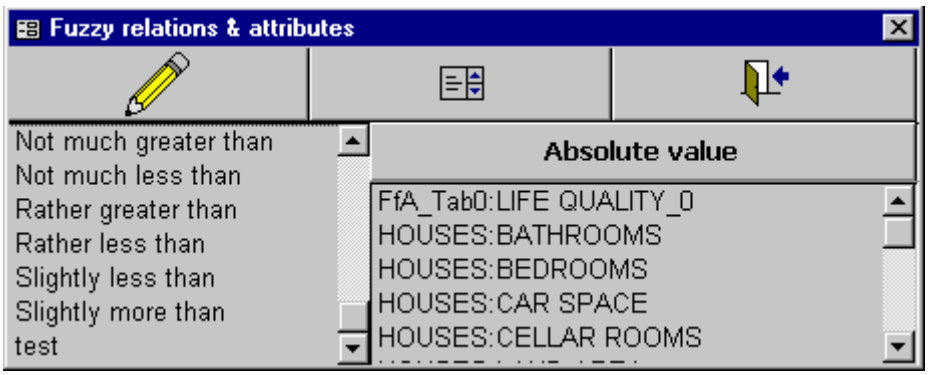

Abbildung 4-5 : Administrations-Fentster für Fuzzy-Relations

Fuzzy Operatoren editieren

Um die Operatoren zu editieren, z.B. hinzufügen einer Fuzzy-Values „Not High“ mit entsprechenden Parametern, betätigt man in dem entsprechenden Formular den Schalter, der ein Auswahlfeld symbolisiert. Er öffnet ein Editier-Fenster, wie in Abbildung 4-6 gezeigt. Hier lassen sich sowohl bestehende Werte veranschaulichen und verändern, als auch neue erzeugen. Diese Option gibt es aus ersichtlichen Gründen nicht für die Fuzzy-Importance, da dies lediglich eine Zahl aus dem Intervall [0,1] ist.

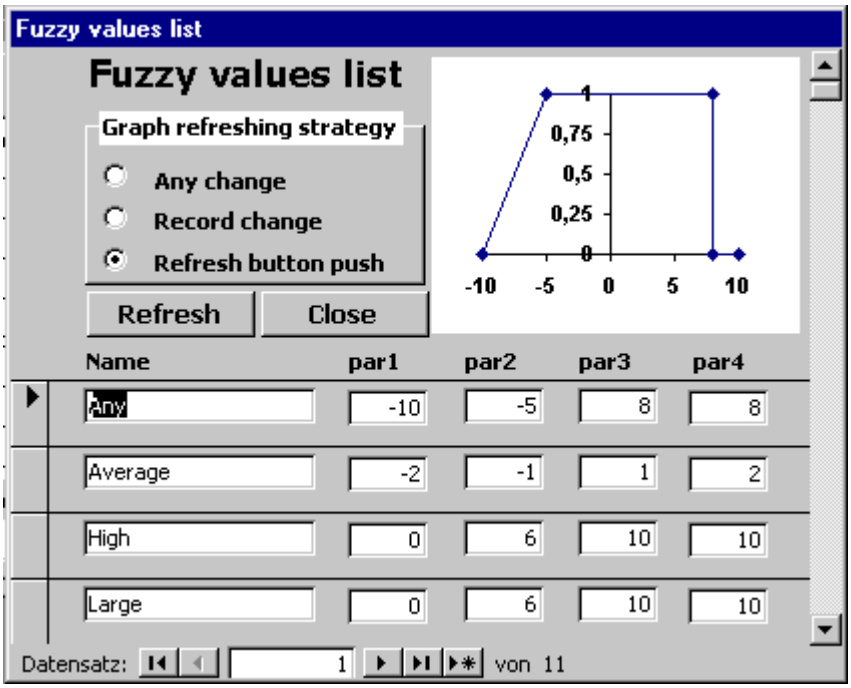

Abbildung 4-6 : Editier-Fenster Fuzzy-Values

Fuzzy Attribute editieren

Bei der Zuordnung eines Minimal- und Maximalwert zu Attributen gelangt man, nach Auslösen des FA-Buttons der Toolbar, direkt in den Editiermodus. Es wird ein Formular geöffnet, mit dessen Hilfe sich die UL- und LL-Werte in eine Tabelle eintragen lassen.
Die vier genannten Fuzzy-Funktionen lassen sich alternativ auch direkt in den Tabellen FfA_Attributes, FfA_Fuzzy Relations, FfA_Fuzzy Values und FfA_Linguistic Quantifiers editieren, wenn das Add-In direkt, d.h. als Datenbank geöffnet wurde.

4.4. FfA97 Erweiterungen

Die zweite unter den öffentlich zugänglichen Versionen der Software, FfA97, zeichnet sich durch einige funktionale Erweiterungen aus, die nicht untersucht werden konnten, da sich die Software nicht starten ließ. Ihre englischsprachige Erläuterung, die der Hilfefunktion des Systems entnommen wurde findet sich im Anhang. Es handelt sich um

- Fuzzy set constants
- Compatibility operator
- Single- and multi-valued attributes

Fuzzy set constants sind dabei nicht im Sinne eines kombinierten Fuzzy-Attributs zu verstehen, wie es im Lösungsentwurf beschrieben ist. Es scheint sich dabei vielmehr um eine Möglichkeit zu handeln Elemente einer hierarchischen Domäne explizit mit den zugehörigen Werten einer Linguistischen Variablen verknüpfen zu können.

- Diese Funktionalität spielt in nicht genau identifizierbarer Weise mit den anderen beiden Funktionen (Compatibility operator und Single- and multi-valued attributes) zusammen.

Kapitel 5 - Lösungsentwurf

Nachdem in den vorangegangenen Kapiteln Ansätze und Lösungen für Fuzzy-Zugriffe auf relationale Datenbanken beschrieben wurden, soll nunmehr als Ergebnis dieser Überlegungen ein „wünschenswerter“ Lösungsentwurf dargestellt werden. Im Gegensatz dazu wird im nächsten Kapitel die tatsächlich durchgeführte Lösung beschrieben, die den praktischen Möglichkeiten der Diplomarbeit Rechnung trägt.
Dieses Kapitel gliedert sich grob in drei Unterabschnitte:
Zunächst wird ein geschlossenes Modell eines Fuzzy-DBMS (FDBMS) beschrieben, dass den gegebenen Randbedingungen des Projekts Rechnung trägt.
Anschließend findet sich eine Diskussion über die Schnittstelle des FDBMS.
Schließlich wird das PIA-Objektmodell (Klassendiagramm) diesen Theorien angepasst.

5.1. Modellentwurf FDBMS

Ziel dieser Arbeit ist es neben der prototypischen Realisierung, einen Lösungsvorschlag für das Problem der Fuzzy-Queries vorzulegen. Das hierfür entwickelte Datenbank-System wird in der Annahme, dass dieser Begriff noch nicht für ein bestehendes System als Eigenname verwendet wird, als Fuzzy Database Management System (FDBMS) bezeichnet und will nicht mehr sein, als eine mögliche Ausprägung eines solchen Systems.
Einleitend lässt sich hierzu feststellen, dass sich im Rahmen des PIA-Projekts keine Eigenentwicklung einer leistungsfähigen FDBE wird realisieren lassen. Deshalb muss das anvisierte FDBMS auf einer existierenden DBE aufsetzen und die dort mit scharfen Bedingungen abgefragten Datensätze selbst mit einem MD bewerten. Diese Bewertung wird aus Gründen der Anschaulichkeit mit einem Fuzzy-Regler modelliert.

Das Konzept sieht also vor, ein Standard DBMS als Datenquelle zu verwenden und die Datensätze gemäß ihrer Zugehörigkeit zur Query zu bewerten. Diese Bewertung geschieht mit Hilfe eines adaptiven Fuzzy-Reglers (AFR, s.u.), der von der Query gesteuert wird.
Dieser Entwurf bietet als weiteren Vorteil die Möglichkeit, in einer beliebigen Sprache implementiert werden zu können, insbesondere der, in der PIA implementiert wird: JAVA. Er benötigt auch keine Spracherweiterungen, der SQL-Syntax, da die Fuzzy-Funktionen sämtlich außerhalb des DBMS’s realisiert werden. Die SQL-Query für die Datenbankabfrage kannn aus der frei definierbaren Erscheinungsform gemäß Objektmodell generiert werden. Außerdem weißt er eine hohe Modularität auf, so dass er in Teilschritten, und für den Fall wohldefinierter Schnittstellen, auch von verschiedenen Personen realisiert werden kann.

Die Beantwortung einer unscharfen Query in allgemeiner Form erweist sich dabei in einer solchen Weise als komplex, dass davon abgesehen wird, den Vorgang in einem einzigen durchgehenden Gedankengang umfassend zu beschreiben. Stattdessen wird die Lösung in zwei Schritten beschrieben.
Zunächst folgt eine nur anschauliche Beschreibung, die einen Überblick über das Gesamtsystem erlaubt und Raum für Überlegungen bzgl. möglicher Ausprägungen oder Erweiterungen des Systems gibt. Hierbei werden die einzelne Komponenten des Systems beschrieben.
Danach findet dann eine eingehende Diskussion der Phänomene statt, die auftreten, wenn man sich die Details der Implementierung vor Augen führt. Hierbei wird der sequentielle Ablauf eines Evaluierungsvorgangs (Berechnung einer Query) beschrieben.

5.1.1. Beschreibung der Komponenten des Systems

Das System sei zunächst in einem raschen Überblick in der vom Autor favorisierten Version beschrieben, um als Diskussionsgrundlage für nachfolgende Überlegungen zu möglichen Variationen und Detailbetrachtungen des Abschnitts 5.1.2 Sequentielle Beschreibung des Systems zu dienen.

Das System besteht aus sechs Teilen:
- Regler
- Regelaktivierungseinheit (ANN)
- Personalisierung
- Datenbank
- Ablaufsteuerung inkl. Interpreter
- Feedback (backprop)

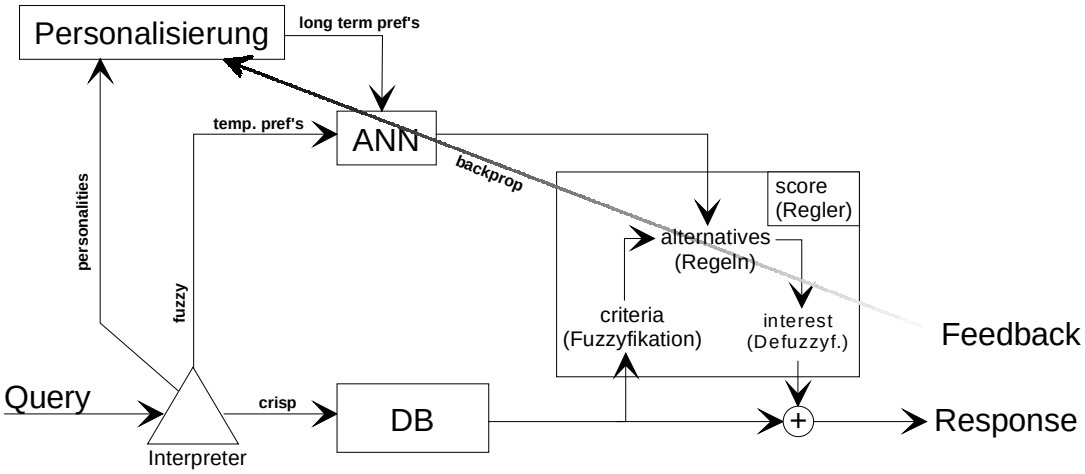

Abbildung 5-1: Systementwurf

5.1.1.1. Regler (AFR)

Kernstück der Anlage ist der Regler. An ihn werden Daten zur Bewertung angelegt (Eingangssignal). In dem Regler werden die Daten entsprechend dem Interesse bewertet und präsentieren dann ausgangsseitig die Zugehörigkeit zu diesem Interesse als MD. Die Abbildung des Interesses in den Regler geschieht mit der Regleraktivierungseinheit (s.u.) in Abhängigkeit von der Query und der individuellen Person respektive Personengruppe (Individualisierung). Der Regler wird aus diesem Grund adaptiv genannt (AFR). Zunächst sei jedoch angenommen die Regel, bzw. das Interesse sei fest implementiert.

Kriterien (Fuzzifizierung)
Der Regler besitzt Domänen-spezifische Eingänge (Geschwindigkeit, Farbwert, ...). Die Attributwerte der Datensätze werden fuzzifiziert ($\mu_{v=hoch}=0.7$, $\mu_{F=rot}=0.6$,...).

Alternativen
Die Zugehörigkeitswerte werden gewichtet und zu alterantive.score zusammengefasst.

Interesse
Die Zugehörigkeitswerte der Alternativen werden zu interest.score zusammengefasst und ausgegeben.

5.1.1.2. Regleraktivierungseinheit (RAE)

Damit der Regler für verschiedene Interessen Zugehörigkeitswerte berechnen kann, wird er für alle criterion.score initialisiert. D.h. die Zugehörigkeit jedes Attributwertes zu dem gegebenen

Satz an linguistischen Variablen (z.B. *hoch, mittel, niedrig*) wird berechnet. Gemäß dem spezifischen Interesse blendet er dann nicht relevante Zugehörigkeitswerte durch weight=0 aus. Hierbei muss beachtet werden, dass in der gewählten Interpretation lokaler Gewichtung weight=0 auch wirklich zu einem Nichtbeachten dieses criterion.score führt.

Die RAE lässt sich idealerweise von einem neuronalen Netz (ANN, Artificial Neural Network) realisieren, dessen natürliche Eigenschaft inhibitorische (hemmende) und exhibitorische (verstärkende) Reize sind. Es bietet zudem mit seiner Fähigkeit zu maschinellem Lernen, die Möglichkeit ein User-Feedback zur Verbesserung des Systems zu nutzen.

5.1.1.3. Personalisierungs-Einheit (PE)

Das Neuronale Netz der RAE kann individuell, d.h. personenbezogen initialisiert, bzw. parametrisiert werden. Wenn es somit z.B. Anjas Ansicht von einem Billigauto von der von Bernd unterscheiden kann, wären damit alle Anforderungen des PIA-Systems an das DBMS erfüllt. Diese PE wäre idealerweise wiederum ein Neuronales Netz um aus dem User-Verhalten Rückschlüsse ziehen zu können. Wenn die RAE als Neuronales Netz ausgelegt wird, wäre die PE lediglich eine weitere Schicht in diesem Netz.

5.1.1.4. Datenbank

Das Standard Datenbank Management System muss keine besonderen Vorraussetzungen erfüllen außer im Rahmen der Systemarchitektur erreichbar zu sein. So kann es auf dem gleichen Rechner oder Rechnernetz liegen und seine Daten etwa per ODBC, JDBC oder OLE zur Verfügung stellen. Die Entwicklungsumgebung könnte aber auch native Treiber für die Datenbank haben, was die schnellste Verbindung darstellen würde. Die Daten könnten aber auch auf einem entfernten Server im Internet liegen und über eine CGI-Schnittstelle abfragbar sein.

Die Datenbank funktioniert in klassischer Weise. Aus einer gestellten Anfrage

```
Ich möchte einen billigen, roten Opel Manta.
```

werden die SQL-spezfischen scharfen Teile herausgefiltert

```
Select *
from PIA1.dbf
where Kategorie=Auto, Fabrikat=Opel, Typ=Manta, Farbe=rot;
```

Hierin sind verschiedene Phänomene zu erkennen:
Die unscharfe Bedingung *billig* ist nicht in der Datenbank-Anfrage enthalten. Sie wird im AFR realisiert.
„*“ repräsentiert die Bereitstellung aller Attribute der Datensätze. Sie werden benötigt um den Datensatz im AFR bewerten zu können.
Das ungelöste Problem der unscharfen Bewertung in hierarchischen Domänen verlangt eine präzise (scharfe) Formulierung der Anfrage. Anstatt nach einem billigen, roten Manta zu fragen, muss die ganze Hierarchie angegeben werden (Auto, Opel, Manta).

5.1.1.5. Ablaufsteuerung

Es verbleibt die Erwähnung eines ausführenden, prozeduralen Moduls, dem die Interpretation und Zerteilung der Eingangsquery obliegt, die zeitliche Synchronisation der Datenbank-Query mit dem Regler, sowie der sequentiellen Zuführung der Daten an den Regler und der Aufbereitung des Ergebnisses.
Es handelt sich gewissermaßen um die steuernde Hauptprozedur des Systems respektive die Funktionalität eines Objekts der Klasse Fuzzy-Application, gemäß dem erweiterten

Objektmodell Abbildung 5-8. Sie ruft die verschiedenen Module auf, übergibt die Parameter und sorgt für die zeitliche Synchronisation.

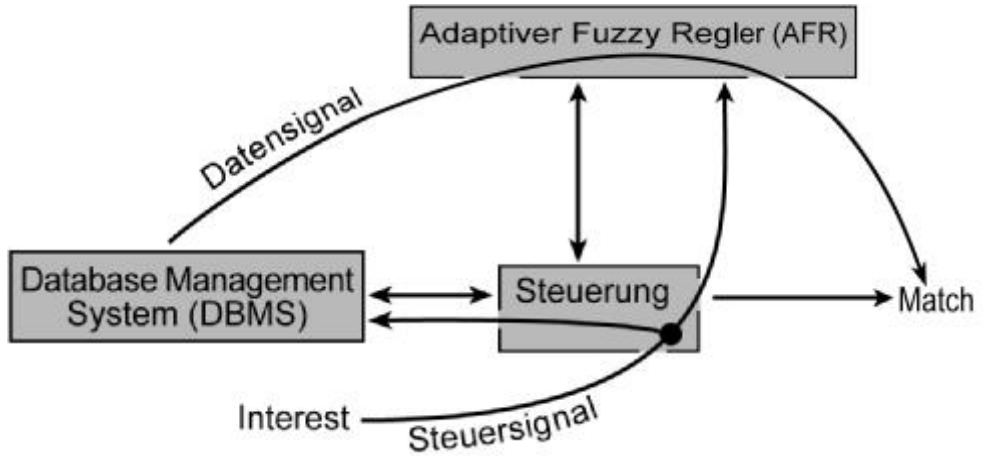

Abbildung 5-2: Daten und Steuerungssignale

5.1.1.6. Feedback

Durch Rückgabe eines Fehlersignals, d.h. durch Angabe einer Differenz zwischen Soll- und Ist-Wert am Ausgang, lassen sich Neuronale Netze, wie sie für die Regleraktivierungseinheit und das Personaliserungsmodul vorgeschlagen werden, leicht trainieren. Dieser Vorgang wird als Backpropagation bezeichnet, da es sich um eine Rückwärtsprojektion des Fehlersignals durch das Netz handelt.
Dieses Vorgehen wird im PIA-Objektmodell als „Veränderliches Verständnis“ („Mutual Understanding“) bezeichnet. Als Begriff für die Fehler-Rückwirkung wird dort „Relevance Feedback“ verwendet.

5.1.1.7. Abgleich des Entwurfs mit menschlichen Entscheidungs- und Bewertungsverfahren

Es hat sich als treffliche Testmethode für KI-Anwendungen erwiesen, zu überprüfen, inwiefern sie mit den Funktionsweisen des menschlichen Gehirns übereinstimmen. Dies kann natürlich nur ein heuristischer Wert und kein eigentliches Qualitätsmaß sein. Dennoch sei eine solche Überlegung an dieser Stelle angeführt.

Die Frage nach allgemein Interessantem oder einem speziellen Interesse, wie „Gefällt mir das Auto, dass ich dort sehe?“, beantwortet unser Gehirn gemäß zeitgemäßer Theorien über gedankliche Abläufe, indem es zunächst das Wahrgenommene mittels Spiegelneuronen auf die sogenannte Theaterebene im Gehirn abbildet. Unser Gehirn macht sich gewissermaßen ein Bild von den Gegebenheiten, gemäß bisheriger Erfahrungen. Dies entspricht der Auslösung und Formulierung eines Wunsches oder Gedankens durch äußere und innere Reize.

Die zurückliegenden Erfahrungen unseres Lebens entsprechen der Implementation von vorhandenem Expertenwissen in unserem FDBMS und den Lernerfolgen des ANN‘s:
- Die Modellbildung und -implementation (Anlegen der Datenbank, Implementierung der Fuzzy-Methoden),
- die Einspeisung bekannten Wissens (Daten und Regeln)
- und das Lernen mit dem Neuronalen Netz.

Die Anfrage wird mit einer Suche im Lösungsraum (Menge aller möglichen Datensätze) realisiert. Die Daten werden vorselektiert (ausgeschlossene Datensätze nicht beachtet)
Die relevanten Datensätze werden nach Zugehörigkeit bewertet („Wenn grün, dann Gefallen ist groß“) und linguistisch oder gedanklich bewertet („Gefällt mir recht gut“).

An dieser Stelle tritt ein kleiner Bruch auf, da „Gefällt mir recht gut“ deutlich unschärfer ist, als der von unserem System gefundene Gesamt-Zugehörigkeitswert des Interesses. Gleichwohl ist die linguistische Formulierung ungleich aussagekräftiger als der scharfe Wert. Im Fuzzy-Sinne entspricht sie einer Zugehörigkeitsfunktion in Form eines Graphen. In unserem Gehirn dagegen ist das Ergebnis ein weitverzeigtes Erregungsmuster in einem komplexen Netz.[12]
Hier stößt man an die Grenzen gegenwärtiger Mensch-Maschine-Schnittstellen. Die Forschung arbeitet aber bereits an intelligenteren Schnittstelle. Für den vorliegenden Fall böte sich beispielsweise ein differenziertes Ergebnis im Sinne einer Erhebung über einer zwei dimensionalen Ebene an, die von subjektivem und rationellem Gefallen aufgespannt sein könnte. Eine solche Projektion eines hochdimensionalen Ergebnisraum, auf eine zweidimensionale Ebene wurde in den 80-Jahren mit bemerkenswerten Ergebnissen von dem Finnen Tuevo Kohonen entwickelt [Ko89].
So ließen sich z.B. Anfragen, wie “Gibt es in dieser Datenmenge etwas, das mich interessiert?“ sinnvoll beantworten.

Aufgrund seines umfassenden Ansatzes mag das System den Titel „Konnektionistisches Modell eines adaptiven Fuzzy-Reglers zur unscharfen Bewertung von Daten“ tragen.

5.1.2. Sequentielle Beschreibung des Systems

Die Beantwortung einer Query (Interest) in der FDBE lässt sich in vier aufeinanderfolgende Prozesse gliedern:

- Vorbereitung (Preprocessing)

Aufbereitung einer eingehenden Query zu einem sequentiellen Prozess, der relevante Datensätze nacheinander an den Regler zur Bewertung anlegt.

- Bewertung (Processing)

Die Zugehörigkeit eines jeden Datensatzes zu dem gegebenen Interesse wird ermittelt.

- Präsentation (Postprocessing)

Das Ergebnis wird in einer bestimmten (vom Client festgelegten) Form aufbereitet und zurückgeliefert.

- Feedback und Lernvorgänge

Das gelieferte Ergebnis wird vom Client bewertet, die Bewertung wird dem System mitgeteilt und dort in einem Lernvorgang verarbeitet.

5.1.2.1. Vorbereitung:

In dem hier vorgeschlagenen System werden aus technischen Gründen alle zu untersuchenden Datensätze zweimal bewertet. Zunächst in der Vorbereitungsphase mit den Mitteln herkömmlicher DBMS’s, dann mit den speziellen Methoden der Fuzzy-Logic in dem AFR.
Diese Zerlegung in ein zweistufiges System macht aus verschiedenen Gründen Sinn:
Die Fuzzy-Methoden in das DBMS zu verlagern wäre wie in „Architekturmodelle“ b) beschrieben mit hohem Implementationsaufwand verbunden.
Andererseits sprechen mehrere Gründe dagegen, die scharfen Bedingungen sämtlich dem AFR zu übertragen.

[12] Gefallen=*recht gut* wird durch ein Erregungsmuster repräsentiert.

Zum einen werden die Datensätze sowieso in einem DBMS vorgehalten. Von dort werden sie gewöhnlich mit SQL-Statements abgefragt. Die Statements enthalten zwingendermaßen eine Where-Clause, deren effiziente Evaluierung mit dem AFR nicht nachzubilden ist.
Mit dem Aufwand, diejenigen Bedingungen aus der Gesamt-Query zu extrahieren, die vom Standard-DBMS evaluiert werden können, gewinnt man den Vorteil, nur die Datensätze mit dem AFR betrachten zu müssen, die relevant sein können, d.h. die möglicherweise eine Zugehörigkeit größer Null zum Interesse aufweisen. Es muss also nicht für alle Datensätze die gesamte Anzahl der Kriterien evaluiert werden, was im Einklang mit dem Ansatz zur Effizienzsteigerung nach 3.3.3 Architekturmodelle c) steht.
Weiterhin ist der AFR nur für Bewertungen auf numerischen Domänen ausgelegt. Hierarchische und Volltext-Domänen hingegen müssen zunächst weiter mit den klassischen Methoden behandelt werden. So ist der AFR beispielsweise nicht in der Lage zu entscheiden, ob ein Datensatz der Kategorie Auto angehört, bzw. wie sehr die Farbe Orange der Bedingung Farbe=Rot genügt.
Die hierfür noch zu entwickelnde Funktionalität wird Module erfordern, die sich möglicherweise zwischen die Datenbank-Query und den AFR schalten ließen. Sie könnten etwa die Ähnlichkeit zwischen „Stephan“ und „Stefan“ ermitteln. Die Aufteilung der Bewertung der Datensätze in mehrere Schritte hat somit auch den Vorteil das System modular zu entwickeln.
Mit dem modularen Ansatz wird also sowohl eine effiziente Berechnung, als auch ein Gewinn an Flexibilität verwirklicht.
Die Beschreibung der Vorbereitungsphase geschieht zweistufig. Zunächst werden Überlegungen angestellt, wie die für den AFR relevanten Datensätze ermittelt werden können. Danach wird beschrieben, wie die so gewonnen Erkenntnisse sich in der Implementierung eines Extraktionsfilters niederschlagen.

Ermittlung der relevanten Datensätze

Die Anfragen an das System liegen gemäß Projekt-Definition in der ersten Normalform vor.
Im Falle einer gewöhnlichen Datenbank-Query sind alle Und-verknüpften Bedingungen (Kriterien einer Alternative) untereinander Ausschluss-Kriterien, d.h. wenn ein Kriterium einer Alternative nicht erfüllt ist, so ist die gesamte Alternative für diesen Datensatz nicht erfüllt.
Dagegen sind die Alternativen untereinander Oder-verknüpft. Hier gilt also, wenn eine Alternative nicht erfüllt ist, so muss deswegen der Datensatz nicht zwingend irrelevant für das gesamte Interesse sein. Alternativen sind untereinander also keine Ausschlusskriterien.

1) Im Falle der Oder-verknüpften Alternativen erhält sich diese Logik beim Übergang aus der Fuzzy-Logic in die boolesche Algebra, u. A. auch unabhängig vom Bewerten der Alternativen mit Gewichtung, da die Leitlinie ist, Datensätze nicht „voreilig“ auszuschließen, nur weil die eine oder andere Alternative nicht erfüllt ist. Weder das Setzen eines globalen Quantifiers, noch das einer globalen Gewichtung auf Ebene der Fuzzy-Logic, können sich hier sinnverkehrend auf die boolesche Logik auswirken, da der Datensatz nur in dem Fall von einer weiteren Betrachtung ausgeschlossen wird, dass alle Alternativen nicht erfüllt sind.
Dies kann in der Fuzzy-Logic nur dann zu einer Gesamt-Zugehörigkeit größer Null führen, wenn man den entarteten Gewichts-Vektor zulässt, dass alle Alternativen eine Gewichtung von Null haben. Dieser lässt sich allerdings auch nicht sinnvoll normieren und trägt auch keine wünschenswerte Bedeutung. Deshalb sei dieser Fall per Definition ausgeschlossen:

$$\Sigma I_i > 0 \qquad \text{(Vorraussetzung 1 a)}$$

Weil alle in dieser Arbeit vorgestellten Ansätze zur Interpretation globaler Gewichtung (Zadeh, Fagin, PIA) spätestens auf der algorithmischen Ebene selbst eine Normalisierung vornehmen, soll Voraussetzung 1 a zu einer Normierung verschärft werden:

$\Sigma I^n_i = 1$, $I^n_j = I_j / \Sigma I_i$,
mit I_j, I_i sind globale Gewichtungen einzelner Alternativen
und I^n_i sind die normierten Werte dieser Gewichtung (Vorraussetzung 1 b)

Diese Normierung muss aber gemäß den in dieser Arbeit vorgestellten Ansätzen für lokale Gewichtung nur dann berechnet werden, wenn nicht gilt:

$I_i = I_j \quad \forall\, i, j$

Andernfalls hat die Normierung keine Bedeutung. Aus Gründen der Performance wird man im Falle gleicher Gewichtung der Alternativen bestrebt sein, ohne Gewichtung zu rechnen.
Im Übrigen ist dieser Vorgang Aufgabe der weiter unten beschriebenen RAE.

2) Im Falle der Und-verknüpften Kriterien muss hingegen mehr Obacht gegeben werden.

Lokaler Quantifier
Zunächst sei der Fall eines gesetzten lokalen Quantifiers besprochen, obwohl das Objektmodell diese Fuzzy-Methode nicht vorsieht. (Beispielsweise „Die meisten der folgenden Bedingungen müssen erfüllt sein: „...“) In diesem Fall muss die Und-Verknüfung der Alternative bei der Abbildung in den booleschen Raum in eine Oder-Verknüpfung umgewandelt werden, da die Quantifier nach Zadeh und OWA unabhängig von Verknüpfungen definiert sind und auf Mengen arbeiten (Siehe Linguistic Quantifier).

Kriterien auf numerischen Domänen
Für den Fall von Kriterien, die sich auf numerische Domänen beziehen, muss zwischen scharfen und unscharfen Bedingungen unterschieden werden[13].
Zur Unterscheidung dieser Bedingungen sieht das Objektmodell im Bereich der Prädikate zur Zeit keine eigene Eigenschaft vor. Die Objekte der Klasse Predicate haben drei Eigenschaften:

arity : int
name : String
description : URL

In erster Näherung könnte man allein den Namen eines Kriteriums als Bedingung dafür nutzen, ob es sich um ein scharfes oder unscharfes Prädikat handelt. So könnten bestimmte Funktionen, wie „>“, „<“, “exists“ u.ä., könnten als scharfe Funktionen deklariert sein.
Im objektorientierten Sinn dagegen wäre die Schärfe eine vierte Objekteigenschaft. Somit ließe sich der gesuchte Extraktionsfilter für relevante Datensätze realisieren, indem er die Prädikate der Kriterien „fragt“, ob ihre Eigenschaft „scharf“ den Wert „true“ hat. Dies entspricht einer vierten Objekteigenschaft von predicate:

crisp : Boolean

Der Extraktionsfilter würde in diesem vorläufigen Modell also die Alternativen der Query auf ihre scharfen Bedingungen reduzieren. In dem Fall, dass eine der Alternativen keine scharfen Bedingungen enthält (z.B. „Ich möchte etwas billiges“), ergäbe sich eine Wildcard „*“, d.h. alle Datensätze müssen betrachtet werden.

Kriterien auf hierarchischen und Volltext Domänen
Der eben beschriebene Fall einer Wildcard wird allerdings nur selten auftreten, da zu den Kriterien auf numerischen Domänen in der Regel noch die Kriterien der Volltext- und der hierarchischen Domänen kommen. Weil der Fuzzy-Zugriff auf die Werte dieser Domänen noch

[13] Hierfür sei zunächst der Fall gesetzter lokaler Importance außer acht gelassen.

nicht realisiert ist, wird es sich bei diesen Zugriffen zunächst immer um scharfe Kriterien handeln. So etwa bei:

„Ich möchte ein Auto das rot ist."

Der von der Datenbank durch die SQL-Query gefundene Zugehörigkeitswert wird hierbei, wie auch bei den scharfen numerischen Bedingungen, immer den Wert Eins oder Null haben.

Extraktionsfilter

Diese Vorüberlegungen in Betracht soll nun der Extraktionsfilter entworfen werden.
Ohne die theoretischen Überlegungen unnötig auf die Spitze treiben zu wollen, scheint es sinnvoll vier Fälle zu betrachten:

- Queries ohne lokale und globale Gewichtung
- Queries ohne lokale, aber mit globaler Gewichtung
- Queries mit globaler Gewichtung und lokaler Gewichtung auf numerischen Domänen
- Queries mit globaler Gewichtung und lokaler Gewichtung auf nichtnumerischen Domänen

Von globalen und lokalen Quantifiern soll hier der Einfachheit halber abgesehen werden.

Queries ohne lokale und globale Gewichtung

Extrahiert man alle scharfen Bedingungen einer Fuzzy-Query und verknüpft diese Kriterien wieder so, wie in der Ausgangsquery, so erhält man bei der Evaluierung der Abfrage eine Tabelle, die alle relevanten Datensätze enthält.
Das Nichterfülltsein einzelner Kriterien ist nämlich auch für Fuzzy-Queries ein Ausschlusskriterium innerhalb einer Alternative, wenn für die Interpretation des Und gilt:

$\forall \mu_2 : \mu_1=0 \rightarrow \mu_1 \text{ UND } \mu_2 = 0$ (Vorraussetzung 2)

Diese Bedingung erfüllen insbesondere die Multiplikation und die Minimumbildung, nicht aber das geometrische Mittel, das besonders bei Theoretikern als nicht zu rigide Form des Und beliebt ist.
Innerhalb einer Alternative, die neben diesem scharfen Kriterium noch unscharfe Kriterien enthält, wird die Zugehörigkeit des Datensatzes zu dieser Altenative durch die unscharfen Attribute beschrieben, wenn analog zu Vorraussetzung 2 für die Interpretation des Und's gefordert wird:

$\forall \mu_2 : \mu_1=1 \rightarrow \mu_1 \text{ UND } \mu_2 = \mu_2$ (Vorraussetzung 3)

Diese Vorraussetzung erfüllen wie bei Vorraussetzung 2 die Multiplikation und die Minimumbildung.
Als Vorteil für die Implementation ergibt sich daraus, dass die Bewertung der SQL-Query dieses Kriteriums mit dem Zugehörigkeitswert Eins nicht mit in den AFR übertragen werden muss. Der AFR muss so nur die unscharfen Kriterien berechnen.

Man muss nun bedenken, wie die boolesche Information der Datenbank mit dem SFR wechselwirkt. Die Tabelle mit den relevanten Datensätzen stellt zwar alle für die Ermittlung des Gesamt-MDs wichtigen Informationen dar, lässt aber bereits gefundene Zugehörigkeiten des booleschen Raumes wieder fallen, indem sie einzig den kumulierten Wert Eins für die Gesamtzugehörigkeit der scharfen Bedingungen wiedergibt.
So geht durch die Oder-Verknüpfung die Information verloren, welche der Alternativen erfüllt war. Da man diese Information nicht hat, kann man nicht darauf schließen, welche der Alternativen vollständig erfüllt wurden.
In diesem ersten Fall scheint es daher angebracht, auf den ermittelten relevanten Datensätzen eine weitere Query zu evaluieren, die die einzelnen Alternativen als Bedingung hat und ihre Erfülltheit in je einer zusätzlichen Spalte vorhält (Nachevaluierung). Dieser Wert stellt dann

dem AFR die Information zur Verfügung, ob alle scharfen numerischen Bedingungen und die Volltext und hierarchischen Bedingungen der jeweiligen Alternative für diesen Datensatz erfüllt sind oder nicht.
Wie der AFR diese Information verarbeiten kann wird im nächsten Abschnitt erläutert.

Queries mit globaler Gewichtung
In diesem Fall kann wie unter 1. verfahren werden, da die für die Gewichtung wichtigen Zugehörigkeitswerte durch diese Verfahren für alle gewichteten Größen bekannt sind.

Queries mit globaler Gewichtung und lokaler Gewichtung auf numerischen Domänen
Die Aussage lokaler Gewichtung auf numerischen Domänen möge bezeichnen, dass die Bedingungen der nichtnumerischen Domänen unter allen Gewichten den höchsten Wert haben. Insbesondere weisen alle Kriterien, die nicht durch ein Gewicht abgeschwächt werden vor der Normierung intuitiv den Wert Eins als Gewicht auf. Mathematisch stellt aber auch jeder andere höchste Wert die gleiche Funktionalität dar.
Für die Berechnungen mit der lokalen Gewichtung, müssen zusätzlich zu den bisher gefundenen Zugehörigkeitswerten MD_{gesamt} und $MD_{alternatives}$ noch die der einzelnen Kriterien berechnet werden, um sie gewichten zu können.
Die Randbedingungen dieses dritten Falls legen nahe, die Zugehörigkeitswerte der scharfen wie unscharfen Kriterien auf numerischen Domänen im Fuzzy-Regler zu evaluieren, während die Zugehörigkeitswerte der nichtnumerischen Domänen mit der Nachevaluierung ermittelt werden. Die Query für den Extraktionsfilter darf dieses mal aber nur mit den Kriterien initialisiert werden, die den höchsten Gewichtswert innerhalb einer Alternative haben. Durch die Randbedingungen ist gewährleistet, dass damit alle Kriterien auf nichtnumerischen Domänen ermittelt werden können und alle relevanten Datensätze betrachtet werden.
Sowohl in der Interpretation lokaler Gewichtung nach Fagin, als auch in der Interpretation nach PIA, ist nämlich die Nichterfülltheit eines der Kriterien mit der höchsten Gewichtung innerhalb einer Alternative ein Ausschlusskriterium, wie sich leicht beweisen lässt.
In diesem dritten Fall lässt sich die Erfülltheit eines Alternative also noch aus der Datenbank-Query berechnen, wenn man auf den numerischen Domänen auch scharfe Kriterium im AFR realisiert.

Queries mit globaler Gewichtung und lokaler Gewichtung auf nichtnumerischen Domänen
In diesem Fall müssen die Zugehörigkeitswerte eines jeden nichtnumerischen Kriteriums explizit vom Standard-DBMS ermittelt werden.
Hierzu werden zunächst alle Kriterien mit einer Gewichtung von Null aus der Query eliminiert. Dann werden die einzelnen Kriterien Oder-verknüpft. Dadurch wird einzig der entartete Fall ausgeschlossen, dass alle Gewichte gleich Null sind, der bereits in Vorraussetzung 1a) ausgeschlossen wurde. In der Nachevaluation muss jetzt wenigstens für alle nichtnumerischen Kriterien die Zugehörigkeit ermittelt und im Regler gewichtet werden.
Gegebenenfalls könnte man diesen rechnerischen Aufwand unterdrücken, indem man für die Kriterien dieser Domänen keine Gewichtung zulässt.

5.1.2.2. Bewertung (AFR)

Nach der Ermittlung aller relevanten Datensätze in der Vorbereitungsphase werden die Daten zur Bewertung an einen Regler als Eingangssignal angelegt.
Der Regler wird in Anlehnung an Abbildung 3-6: Max-Min-Inferenz-Methode graphisch entworfen:

In Abbildung 5-3 werden durch unterschiedliche Bearbeitungsrichtungen unterschiedliche Funktionen realisiert:

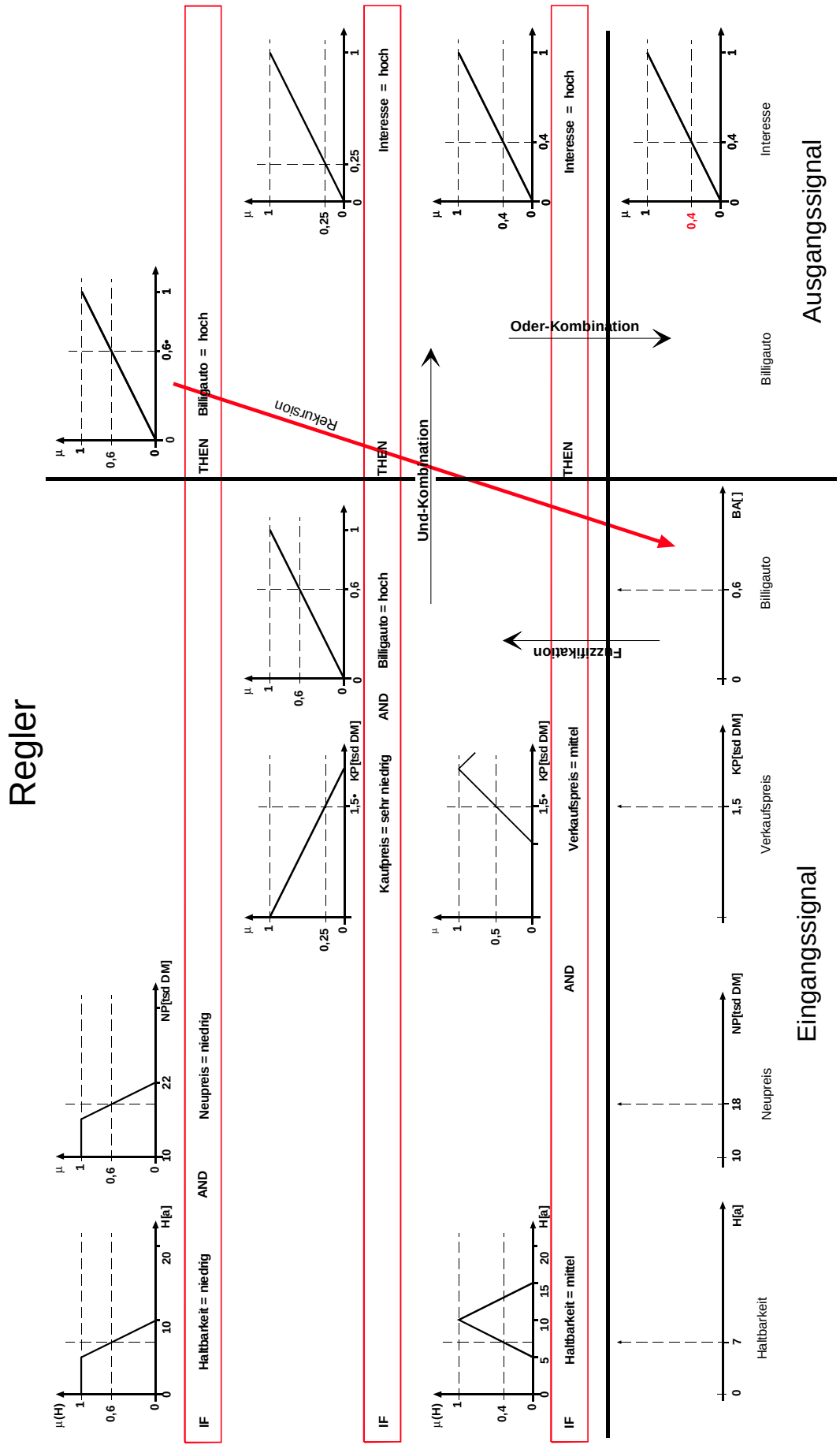

Abbildung 5-3: Signalverarbeitung AFR

In Aufwärtsrichtung (1) findet die Fuzzifikation statt.
In waagerechter Richtung (2) wird die Und-Komposition vollzogen.
In Abwärtsrichtung (3) wird die Oder-Komposition durchgeführt.
Und in diagonaler Richtung (4) finden rekursive Prozesse statt, für auf Basis von Und-Kompositionen zusammengesetzte Fuzzy-Terme.
Defuzzifikation ist nicht notwendig, da die realisierten Verknüpfungen keine Möglichkeitsverteilungen ergeben, sondern diskrete Werte. Dies liegt unter anderem an der Zugehörigkeitsfunktion $\mu_{interessant}$.
Um die Übersichtlichkeit zu wahren wurden nicht alle Vorgänge in ihrer vollen Ausprägung dargestellt.

- So geschieht die Fuzzifikation (1) eigentlich über die Transformation der Domäne auf das Einheitsintervall.
- Die Gewichtung der Kriterien und Alternativen wurde nicht berücksichtigt.
- Außerdem beruht der dargestellte Regler auf einer „Festverdrahtung“ und ist in dieser Form nicht von einer RAE steuerbar.

Die Prozesse seien nun im Einzelnen besprochen. Um die Elemente der Zeichnung benennen zu können, werden die Graphen der Zugehörigkeitsfunktionen wie in einer 4x6-Matrix referenziert:

$E_{i,j}$ ist das j-te Element in der i-ten Zeile.

Dabei sind leere Felder als Nullwerte zu sehen. Auch diese werden mitgezählt.

Die Bewertung muss so ausgelegt werden, dass mit ihr folgende Konzepte realisiert werden können:

Konzept	**Beispiel**
Existenz	Is known
Fuzzy Value	Is large
Fuzzy Relation	Is larger than
Extended Fuzzy Relation	Is between
Gewichtung (lokal und global)	Very important
Complement	Not
1. Normalform	(C_1 And C_2) Or ... Or (C_{n-1} And C_n)

(1) FUZZIFIKATION

Der Vorgang der Fuzzifikation bedeutet frei übersetzt Verunschärfung. In Wirklichkeit ist aber die Abbildung von Eingangswerten auf linguistische Variablen gemeint. Es handelt sich also nicht um die Verringerung des Informationsgehalts eines Wertes, wie der Name nahe legt, sondern vielmehr um eine Erhöhung. Dieses rührt daher, dass das Wissen eines Experten um die Interpretation des Grundbereichs des Wertes in den Wert hinein projiziert wird. So kann indirekt die Zugehörigkeitsfunktion des Wertes zu verschiedenen Linguistischen Variablen ermittelt werden.
Als Beispiel seien die Elemente $E_{3,1}$ und $E_{4,1}$ betrachtet. Es handelt sich hierbei insofern um eine vereinfachte Darstellung, als dass nur die Projektion auf den relevanten Wert einer linguistischen Variable dargestellt ist (KP ist *hoch*). Außerdem scheint es, als wäre die Linguistische Variable (LV) *hoch* auf dem Grundbereich KP definiert, während sie sich tatsächlich auf das Einheitsintervall bezieht.

(1.a) Projektion auf das Einheitsintervall

Die Projektion auf das Einheitsintervall hat zwei Ausprägungen unterschiedlicher Komplexität:

- Projektion zur Evaluierung von Fuzzy Values:

In diesem Fall müssen die anliegenden Eingangssignale durch eine einfache lineare Transformation auf das Einheitsintervall projiziert werden. Für diese Normierung werden die Parameter cUL (common upper limit) und cLL (common lower limit) verwendet, die dem Grundbereich des Wertes zugeordnet sind.

- Projektion zur Evaluierung von (extended) Fuzzy Relations

Um Fuzzy Relations bewerten zu können reicht diese einfache Projektion nicht, da die Normierung vom zweiten (und dritten) Parameter der Relation abhängig ist und von dessen/deren Grundbereich (Siehe FFA: Fuzzy Relation). Da innerhalb eines Interesses mehrere Kriterien Relationen mit unterschiedlichen Parametern auf dem gleichen Datum beschreiben können, muss für jede dieser Relationen ein Eingang am Regler vorgesehen werden, an dem als Signal die spezifische Projektion für diese spezielle Relation anliegt.

Diesen Darstellungen folgend, muss jedes Datum eines Datensatzes einmal in seiner Eigenschaft als Grundlage für die Fuzzy Values normiert werden. Außerdem ist eine Normierung vorzusehen für jede im Interesse angeführte Relation, es sei denn der sekundäre bzw. sekundäre und tertiäre Parameter (erweiterte Relation) mehrerer Relationen ist identisch; dann werden diese Relationen von einer Projektion befriedigt.

Im vorrausgegangenen Abschnitt „Vorbereitung“ wurde für die noch zu entwickelnde Fuzzy-Funktionalität für hierarchische und Volltextdomänen die Einführung unabhängiger Module vorgeschlagen, die zwischen das Preprocessing mit der Datenbank und die Bewertung durch den AFR zu schalten wären. Diesen Gedanken fortsetzend kann auch die Normalisierung als eingangsseitiges Modul verstanden werden.
Um hier eine Einheitlichkeit zu schaffen, könnte man noch einen Schritt weiter gehen und sagen:

Definition 5-1: AFR-Normierungs-Modul:

> Zwischen die sequentielle Bereitstellung relevanter Datensätze aus einer Datenbank und die Bewertung dieser Daten durch Linguistische Variablen mit dem AFR, werden AFR-Normierungs-Module zur Projektion der Daten auf numerische Einheitsintervalle geschaltet.

Diese Normierungsmodule haben also einerseits die Aufgabe numerische Daten zu normalisieren. Für den Fall vorhandener Fuzzy-Relationen in einer Query müssen sie entsprechend viele Eingänge generieren.
Andererseits können sie die Projektion numerischer und Volltext-Domänen auf eine numerische Skala realisieren. So könnte etwa ein Modul „Textähnlichkeit“ definiert werden, indem Distanzmaße im Textraum auf das Einheitsintervall abgebildet werden. Nach der Abbildung ließe sich das Datum dann vom AFR bewerten (Bsp.: Relation „Textähnlichkeit (Content,‘Suchstring‘) ist *hoch*“). Auch hier müssten Relationen befähigt sein eigene Eingänge zu kreieren.
Die Projektion des aktuellen Datensatzes liefert also einen Eingangsvektor **x** an den Regler mit

$$\mathbf{x} \in [0,1]^{l+m}, \text{ mit} \qquad \textbf{(1)}$$

l ist die Anzahl aller Daten eines Datensatzes
und m ist die Anzahl aller Relationen des Interesses mit unterschiedlichen sekundären und tertiären Parametern.

(1.b) Fuzzyfikation der Werte
In Einklang mit der Anforderungsdefinition kann von scharfen Eingangswerten ausgegangen werden. Nach der Normalisierung werden sie mit den Werten einer standardisierten Linguistischen Variablen „Bewertung“ fuzzifiziert. Im einfachsten Fall hat diese Variable folgende Gestalt:

Linguistische Variable	"Bewertung"	
Linguistische Werte (Terme)	Primärterm	: hoch
	Duchschnittswert	: mittel
	Antonym	: niedrig
Modifikator		: not, sehr, kaum

Abbildung 5-4: LV „Bewertung“ (vollständiger AFR)

An diesem Punkt wird man bei der Wahl der Implementation eine strategische Entscheidung treffen müssen.
Die Abbildung eines spezifischen Interesses in den AFR ist laut Entwurf der RAE übertragen worden.
Man kann nun einerseits die Zahl der Berechnungen minimieren wollen, mit dem Preis, für jedes Interesse einen ganz spezifischen Regler initialisieren zu müssen. Dieser Ansatz ist in Bild Abbildung 5-3 dargestellt. Diese Variante wird im Folgenden **spezialisierter AFR** genannt.
Oder aber man versucht den Regler zu standardisieren mit dem Preis möglicherweise überflüssige Berechnungen durchzuführen (**vollständiger AFR**). Der Vorteil des zweiten Ansatzes liegt in einem Quasi-linearen Berechnungsalgorithmus wie er im folgenden beschrieben ist, etwa in der Form[14]:

$$MD_{Interest} = OR_I(\mathbf{I}^T_{glob} \cdot \mathbf{AND}_I (\mathbf{I_{lok}}, \mathbf{mod}, \mathbf{LV(x)})))$$

Beim spezialisierten AFR würde der Eingangsvektor nur gemäß des spezifischen Interesses fuzzifiziert, d.h. die Fuzzifizierung wäre ein Vorgang, der von dem spezifischen Interesse abhängt, wie zuvor schon die Normalisierung in Bezug auf die Abbildung der Relationen. In diesem Fall lässt sich die Abbildung durch geschickte Wahl der standardisierten Linguistischen Variablen vereinfachen. So wurden In Abbildung 5-5 die Modifikatoren in den Vorgang der Fuzzifizierung einbezogen:

Linguistische Variable	"Bewertung"	
Linguistische Werte (Terme)	Primärterm	: hoch
	mod. Primärt.	: sehr hoch
	Duchschnittswert	: mittel
	Antonym	: niedrig
	mod. Antonym	: sehr niedrig

Abbildung 5-5: LV „Bewertung“ (spezialisierter AFR)

[14] Die Größen der Formel werden noch einzeln beschreiben.

Beim vollständigen AFR wäre für jeden Wert eines anliegenden Datensatzes zunächst ein Sympathie-Vektor zu ermitteln, etwa gemäß Abbildung 5-4 :

Leistung(100PS) = hohe Leistung/0.8 + mittlere Leistung/0.5 + niedrige Leistung/0.1.

Sei der Eingangsvektor ein n-zeiliger Vektor, dann ergibt die Fuzzifizierung einen 3·n-zeiligen Sympathie-Vektor, aus dem sich alle, ggf. auch mit Modifikatoren belegten Kriterien berechnen lassen. Daher wäre in diesem Ansatz die Abbildung unabhängig vom spezifischen Interesse und bräuchte nicht von der RAE gesteuert zu werden. Dies kann als Vorteil bei der Implementierung gelten.
Man könnte aber auch auf den Ansatz aus Abbildung 5-5 zurückgreifen. Dann müsste man allerdings einen entsprechend größeren, in diesem Fall 5·n-zeiligen Sympathie-Vektor in Kauf nehmen.

Für den vollständigen AFR wird im weiteren Verlauf der Erläuterungen eine kompakte Formel für die Ermittlung der Zugehörigkeit zum Interesse hergeleitet. Den ersten Baustein hierfür liefert:

LV(x) : $[0,1]^{l+m}$ **à** $[0,1]^{(l+m)\cdot n}$, mit **(2)**
n ist die Quantisierung der Linguistischen Variablen,
m ist Zahl der Attribute
und l ist Zahl der (ungleichen) Relationen

LV(x) stellt die Bewertung des aktuellen Datensatzes mit den Werten der standardisierten Linguistischen Variablen dar. Dieser Vektor soll als **linguistisch bewerteter Eingangswert** bezeichnet werden.

(2) UND-KOMPOSITION

Zunächst sei der einfache, in Abbildung 5-3 dargestellte Fall besprochen. Dazu werden die Elemente $E_{3,1}$, $E_{3,3}$ und $E_{3,6}$ betrachtet. Sie stellen die Inferenz (logischer Schluss)

IF Haltbarkeit = mittel AND Kaufpreis = niedrig THEN Interesse = hoch

für die Daten (Haltbarkeit=7 Jahre) und (Kaufpreis=1500 DM) dar und ergeben ein Interesse von 0,4.
Dieser Fall kann als einfach gelten da er keine Gewichtung mit Gewichtung, keine Modifikatoren und keine Abbildung auf Zwischenwerte enthält.

Die Und-Komposition ist, anders als die graphische Darstellung vermuten lässt, keine Verknüpfung von Flächen, sondern eine Verknüpfung von einzelnen numerischen Werten, nämlich den MDs der einzelnen Kriterien. Im einfachsten Fall handelt es sich dabei um die Minimumbildung je zweier Werte in einem iterativen Prozess, wenn wir uns auf das harte Und beschränken. Die Elemente werden solange paarweise ausgewertet, bis sich Implikation und Prämisse zu einer Folgerung reduziert haben (Approximatives Schließen mit dem Modus Ponens). Die hierbei geltende Kommutativität und Assoziativität gilt für einige Interpretationen des Und nicht, etwa im Falle des geometrischen Mittelwerts

$\prod(a_i)^{1/n}$.

Eine weitere Beschränkung der paarweisen Interpretation ergibt sich, wenn die Gewichtung der Kriterien mit Gewichtung eingeführt wird.
Hier ist nach den drei Interpretationen gemäß Zadeh, Fagin und PIA zu unterscheiden (siehe 3.1.10 Linguistic Quantifier):

Zadeh ignoriert den Verknüpfungsoperator Und, verlangt das Setzen eines Linguistic Quantifiers, ist in dieser Eigenschaft nicht Teil der Anforderungsdefinition und wird deshalb nicht besprochen.
Fagin verlangt in der Form wie seine Interpretation in dieser Arbeit verwendet wird die Sortierung der Kriterien nach abfallender Gewichtung. Man stelle sich die zu verknüpfenden Elemente also entsprechend sortiert vor. Dann bildet er iterativ die Verknüpfung Und über die jeweils ersten j Elemente, gewichtet sie mit der Differenz der aktuellen und der nachfolgenden Gewichtung ($i_j - i_{j+1}$) und multipliziert diesen Wert mit i:

$$\begin{aligned} AND_{Fagin}(\mathbf{i},\mathbf{x}) = {} & (i_1-i_2)\cdot AND(x_1)+ \\ & 2\cdot(i_2-i_3)\cdot AND(x_1, x_2)+ \\ & 3\cdot(i_3-i_4)\cdot AND(x_1, x_2, x_3)+...+ \\ & k\cdot i_k\cdot AND(x_1, ... , x_k) , \end{aligned}$$

mit $\mathbf{i} = (i_j) \in [0,1]^k$ als Vektor der lokalen Gewichtung[15],
$AND: [0,1]^k \rightarrow [0,1]$ (z.B. Minimumbildung),
und k als Komplexität der Und-Komposition.

Dieser Vorgang lässt sich zwar nicht graphisch veranschaulichen, aber es lässt sich denken, dass er in waagerechter Richtung unseres symbolischen Ablaufs in Abbildung 5-3 durchgeführt wird und eine Abbildung
$AND_{Fagin}(\mathbf{i},\mathbf{x}): [0,1]^k \rightarrow [0,1]$ bewirkt.

Nach dem PIA-Ansatz werden die Kriterien vor der Und-Komposition einfach durch die Funktion

$$\begin{aligned} x_j' &= x_j - x_j\cdot i_j + i_j \\ &= (1-i_j)\cdot x_j + i_j \end{aligned}$$

bewertet. Dies lässt sich graphisch als Stauchung bzw. Streckung der Zugehörigkeitsfunktionen interpretieren. Hierbei bleibt die Möglichkeit des paarweise Approximativen Schließens erhalten. Dies ist im Sinne eines einfachen Algorithmus' vorteilhaft.

Abgebildet wird der so ermittelte Wert im Allgemeinen auf eine fiktive Variable namens „Interesse", die im Objektmodell (2.2 Das PIA-Objektmodell) der Eigenschaft score der Klasse Alternative entspricht (alternative.score). In einem speziellen Fall kann die Abbildung aber auch auf eine Zwischenvariable stattfinden (siehe rekursive Prozesse).

Es folgen drei Ansätze zur Implementation von Modifikatoren im Falle des vollständigen AFR's. Zur Beschreibung der Abbildung wird die Definition einer Funktion $*_e$ benötigt, mit der sich gleichformatige Matrizen elementweise miteinander multiplizieren lassen.

Definition 5-2: $*_e$:

$*_e(\mathbf{V},\mathbf{W}): \mathbf{R}^{r\times s}\cdot \rightarrow \mathbf{R}^{r\times s}$, mit $\mathbf{V} = (v_{jk})$, $\mathbf{W} = (w_{jk}) \in \mathbf{R}^{r\times s}$ und $*_e(\mathbf{V},\mathbf{W}) = \mathbf{V} *_e \mathbf{W} = (v_{jk} \cdot w_{jk})$

Im vorangegangenen Abschnitt (1) Fuzzifikation wurde dargestellt, dass es im Falle des vollständigen AFR's nützlich sein könnte die Evaluierung der Modifikatoren nicht schon bei der Fuzzifizierung der Eingangswerte vorzunehmen um nicht unnötig viele linguistisch bewertete Eingangswerte vorzuhalten
Wenn man die Modifikatoren in den AFR verlegt und definiert, dass in einer Alternative nicht gleichzeitig verschieden modifizierte Linguistische Werte als Bedingung genannt werden

[15] Anstelle des θ in der Definition wird hier ein i verwendet.

dürfen (Bsp.: Ein Auto, das klein und sehr klein ist), so erhält man zur Realisierung der Modifikatoren die nachfolgend beschriebenen Varianten a) und b).
Dagegen beschreibt c) die Und-Komposition, wenn die Modifikatoren während der Fuzzifikation eingebunden wurden.

a) Berechnung von Aternative.score mit Modifikatoren auf Ebene der lokalen Gewichtung (explizit):
Die Modifikation ist unabhängig von der lokalen Gewichtung, muss allerdings in der Reihenfolge zuerst ausgeführt werden:

> $\mathbf{mod}(\mathbf{LV(x)}) \in \mathbf{F}_m^{\,p \times (l+m) \cdot n}$: $[0,1]^{(l+m) \cdot n}$ **à** $[0,1]^{p \times (l+m) \cdot n}$ ist eine Matrix von Modifizierungsfunktionen mit
> p ist Komplexität der Oder-Komposition von Interest,
> und $\mathbf{F}_m$ ist der Raum aller definierten Modifizierungsfunktionen.

Jede Alternative ordnet jedem Element des linguistisch bewerteten Eingangswert eine Modifikation zu.
Im Falle dass kein Modifikator anzuwenden ist wird die Identität verwendetet: $mod_{dent}(x) = x$.
Dagegen kann der Modifikator *sehr* durch $mod_{sehr}(x) = x^2$ repräsentiert werden.

Nach den Modifikatoren wird die lokale Gewichtung berücksichtigt. Nach Fagin's Interpretation (3a) muss dies im Zusammenspiel mit der Und-Funktion geschehen. Die PIA-Interpretation (3b) lässt dagegen ein sequentielles Vorgehen zu.

> $\mathbf{MD_{Alternative}} = \mathbf{AND}_{Fagin}(\mathbf{I}_{lok}, \mathbf{mod}(\mathbf{LV(x)}))$, nach Fagin **(3a)**
> $\mathbf{MD_{Alternative}} = \mathbf{AND}(\mathbf{I}_{lok} *_{\mathbf{e}} \mathbf{mod}(\mathbf{LV(x)}))$, nach PIA **(3b)**
> mit
> $\mathbf{I}_{lok} \in [0,1]^{p \times (l+m) \cdot n}$ als Matrix der lokalen Gewichtung,
> in der analog zu **mod**() den Elementen des linguistisch bewerteten Eingangswerts Gewichtung zugeordnet wird (jede Alternative ordnet jedem Element des linguistisch bewerteten Eingangswert eine Gewichtung zu),
> **AND**(), $\mathbf{AND}_{Fagin}$(): $[0,1]^{p \times (l+m) \cdot n}$ **à** $[0,1]^p$ als Realisation des Und's,
> und $\mathbf{MD}_{Alternative} \in [0,1]^p$ ist Zugehörigkeitsvektor über alle Alternativen ($Aternative.score_i$).

Diese kompakte Darstellung der Bewertung erfordert, dass die Redundanzen im linguistisch bewerteten Eingangswert durch die Zuordnung von nullwertigen lokalen Gewichtungen eliminiert werden. Dies lässt sich etwa mit „Das Auto ist zu 0.4 rot, aber das ist unwichtig" interpretieren.
Um sich für die weiteren Betrachtungen nicht unnötig auf eine der beiden Interpretationen (Fagin oder PIA) festlegen zu müssen, werden (3a) und (3b) o.B.d.A. zusammengefasst:

> $\mathbf{MD_{Alternative}} = \mathbf{AND}_I(\mathbf{I_{lok}}, \mathbf{mod}(\mathbf{LV(x)}))$, allgemeine Form **(3)**
> mit $\mathbf{AND}_I$() als Gewichtung berücksichtigender Interpretation einer Und-Komposition.

b) Berechnung von Aternative.score mit Modifikatoren auf Ebene der lokalen Gewichtung (implizit):
Die lokale Gewichtung und die Modifikatoren ähneln sich in ihrer Funktionalität. Sie geben einzelnen Kriterien eine Wertung. Das gleichzeitige zur Verfügung stellen beider Methoden kann etwa zu einer Bedingung

> „dass der Preis sehr niedrig ist, ist sehr wichtig" = $1\,(\mu_{niedrig}(Preis))^2$,
> mit i(sehr wichtig)=1 als Repräsentation von *sehr wichtig*
> und $mod_{sehr}(x)$: x **à** x^2 als Repräsentation des Modifikators *sehr*

führen. Die beiden Methoden widersprechen sich nicht, sondern ergänzen sich. Der Unterschied ist in erster Linie darin zu sehen, dass die Gewichtung darauf beschränkt ist Zugehörigkeitswerte mit einem Faktor zu bewerten, während die Modifikatoren beliebige Funktionen mod: [0,1] **à** [0,1] darstellen können. So liegt der Gedanke nahe, die beiden Methoden zu einer zu verknüpfen.

> **Bew()** = $\mathbf{I}_{lok}$ ***e*** **mod() :** $\in \mathbf{F}_b^{\,p \times (l+m) \cdot n}$: $[0,1]^{(l+m) \cdot n}$ **à** $[0,1]^{p \times (l+m) \cdot n}$ ist eine Matrix von Bewertungsfunktionen mit
> $\mathbf{F}_b = \mathbf{F}_m$ ist der Raum aller Bewertungsfunktionen, der dem Raum aller erlaubten Modifizierungsfunktionen gleich ist.

Dieses Verfahren kann nur nach der PIA-Interpretation von Gewichtung angewendet werden, da in Fagin's Formel bei der Interpretation des Und eine getrennte Behandlung von Modifikatoren und Gewichtung erfolgen muss. Somit ergäbe sich eine in solcher Weise komplexe Abbildung **AND**(**Bew**()), dass sie nicht gewinnbringend erscheint.
Für den PIA-Ansatz hingegen bedeutet sie einen erheblichen Vorteil gegenüber Variante a). **Bew**() wird nämlich direkt von der RAE erzeugt und müsste für Version a) extra in $\mathbf{I}_{lok}$ und **mod()** getrennt werden.

$$MD_{\mathbf{Alternative}} = AND\,(Bew(\,LV(x)\,)) \qquad (4)$$

c) Berechnung von Alternative.score mit Modifikatoren in der Fuzzifikation
Da die Behandlung der Modifikatoren innerhalb des AFR's einigen Aufwand bedeutet soll noch die Bereitstellung aller linguistisch bewerteten **und modifizierten** Eingangswerte betrachtet werden. Diese werden dann während der Fuzzifizierung berechnet (vgl. Abbildung 5-3: Signalverarbeitung AFR $e_{2,3}$).
Damit erhöht die Zahl der zu betrachtenden Werte zunächst um den Faktor 2 für die Bereitstellung aller Komplemente und zusätzlich noch um den Faktor der Anzahl aller weiteren Modifikatoren.:

> $\mathbf{LV(x)} : [0,1]^{l+m}$ **à** $[0,1]^{(l+m) \cdot n \cdot 2 \cdot (Dim(mod) - 1)}$;
> mit Dim(mod) ist die Anzahl aller definierten Modifikatoren.

Dies kann je nach erlaubter Komplexität der einzelnen Kriterien und der Anfrage einen erheblichen Rechenaufwand vor allen Dingen auch in der Weiterbehandlung der Werte bedeuten.
Die Berechnung von Alternative.score vereinfacht sich damit im Gegenzug zu:

$\mathbf{MD_{Alternative}} = \mathbf{AND}_{Fagin}\,(\mathbf{I}_{lok}\,,\,\mathbf{LV(x)}\,)$**,** nach Fagin **(5a)**
$\mathbf{MD_{Alternative}} = \mathbf{AND_I}\,(\mathbf{I_{lok}}\,,\,\mathbf{LV}(x)\,)$, allgemeine Form **(5)**

Im Falle der PIA-Interpretation lässt sich zudem die Realisation der lokalen Gewichtung in die Fuzzifizierung vorverlegen, so dass sich für Alternative.score ergibt:

$$MD_{\mathbf{Alternative}} = AND\,(\,LV(x)\,),\ \text{nach PIA} \qquad \mathbf{(5b)}$$

Im konkreten Fall muss die Interpretation des Und's gekennzeichnet werden (etwa $\mathbf{AND}_{Fagin,\,min}$).

(3) ODER-KOMPOSITION
Die Oder-Komposition verhält sich ganz ähnlich, wie die Und-Komposition.

Es entfällt aber die Problematik, die durch die „verschleppten" Modifikatoren entstand.
Im Umkehrschluß bedeutet das, dass es keine Möglichkeit gibt einer Bedingung (Alternative) Nachdruck zu verleihen, indem man sie mit *sehr* belegt, respektive das *sehr* lässt sich nicht als beliebige Modifizierungsfunktion deuten, sondern nur als multiplikative Gewichtung.

Die Standarddeutung des Oder's ist der Max-Operator.
An die Stelle des geometrischen Durchschnitts tritt der algebraische Durchschnitt.
Zadeh wird aus den gleichen Gründen wie in obiger Diskussion auch hier nicht betrachtet.

Fagin's Formel lautet hier

$$OR_{Fagin}(\mathbf{i},\mathbf{x}) = (i_1-i_2)\cdot OR(x_1)+ 2\cdot(i_2-i_3)\cdot OR(x_1, x_2)+ 3\cdot(i_3-i_4)\cdot OR(x_1, x_2, x_3)+...+ k\cdot i_k\cdot OR(x_1, ... , x_k) ,$$

mit $\mathbf{i} = (i_j) \in [0,1]^k$ als Vektor der Gewichtung,
OR: $[0,1]^k$ **à** [0,1] (z.B. Maximumbildung),
und k als Komplexität der Oder-Komposition.

Die PIA-Interpretation der Gewichtung liefert

$$x_j' = i_j * x_j$$

und bildet dann das Maximum über alle x_j'.

Aus der Fülle mittlerweile kombinierbarer Fallunterscheidungen wurden folgende zur kompakten Darstellung des vollständigen AFR's ausgewählt:
(6a) Interest.score mit Faginscher Interpretation bei impliziten Modifikatoren
(6b) Dito, mit PIA-Interpretation. Man beachte, dass es sich hierbei um eine quasi lineare Verkettung handelt.
(6) Interest.score o.B.d.A.

$$MD_{Interest,\ Fagin} = OR_{Fagin}(\ \mathbf{I}^T_{glob}\ ,\ \mathbf{AND}_{Fagin}(\mathbf{I_{lok}}\ ,\ \mathbf{LV(x)})\ ) \qquad \mathbf{(6a)}$$

$$MD_{Interest,\ PIA} = OR(\ \mathbf{I}^T_{glob}\ \mathbf{*_e}\ \mathbf{AND}\ (\mathbf{I}_{lok}\ \mathbf{*_e}\ \mathbf{LV(x)})\ )\ ) \qquad \mathbf{(6b)}$$

$$MD_{Interest} = OR_I(\ \mathbf{I}^T_{glob}\cdot \mathbf{AND}_I\ (\mathbf{I_{lok}}\ ,\ \mathbf{mod},\ \mathbf{LV(x)})\ )\ ) \qquad \mathbf{(6)}$$

mit
- $\mathbf{I}_{glob} \in [0,1]^p$ als Vektor der globalen Gewichtung, mit der den einzelnen Alternativen Gewichtung zugeordnet wird,
- OR(): $[0,1]^p$ **à** [0,1] als Realisation des Oder's,
- $OR_I()$: $[0,1]^p$ **à** [0,1] als Importance-fähige Realisation des Oder's,
- und $MD_{Interest}$ = Interest.score $\in$ [0,1] ist Zugehörigkeitswert des Interesses.

Auch hier ist darauf zu achten, dass im konkreten Fall die Interpretation des Oder's gekennzeichnet werden muss (etwa $OR_{Fagin,\ max}$).

(4) REKURSIVE PROZESSE

Für den Fall, dass man einen aus einer Verknüpfung gewonnenen Wert als Attribute nutzen möchte (Beispiel: Ich möchte ein Billigauto, wenn der Kaufpreis sehr niedrig ist), benötigt man rekursive Prozesse. In diesen kann man zunächst den Wert verknüpfter Attribute ermitteln und diesen dann wiederum als Eingangssignal des Reglers nutzen (Siehe Abbildung 5-3: Signalverarbeitung AFR). Dabei ist der Unterschied zum bisherigen Vorgehen, dass man den Wert eines alternative.score nicht direkt für interest.score weiterverarbeitet, sondern ihn als Eingangssignal verarbeitet.
Um diesen Vorgang zu ermöglichen, muss von der bisherigen Linearität des Signalwegs abgewichen werden. Das Ergebnis findet sich durch einen iterativen Prozess, der mehrere Durchläufe umfassen kann.

5.1.2.3. Präsentation des Ergebnisses

Die als interest.score ermittelten Zugehörigkeiten werden dem anliegenden Datensatz als neues Attribut hinzugefügt. Zur Anzeige kommen dann die in der Ausgangsquery mittels select-Clause angeforderten Attribute, interest.score, sowie ggf. zusätzlich angeforderte Zwischenergebnisse, wie alternative.score, criterion.score oder unscharfe Attribute aus rekursiven Prozessen.
Datensätze, deren interest.score-Wert die cut-off-Grenze unterschreiten, werden nicht mit einbezogen.
Je nach Implementierung werden die Datensätze iterativ, also sofort nach der Evaluierung übergeben, oder in einer Tabelle zwischengespeichert.
Die Tabelle wird gemäß der User-Präferenzen sortiert. Diesen Sortierkriterien wird der ermittelte Gesamt-MD als vorrangiges Sortierkriterium übergeordnet.

5.1.2.4. ANN , Personalisierung und RAE

RAE

Um den Regler abfragespezifisch zu steuern werden die Aktivierungsgrade der Regeln von einem speziellen Modul gesteuert, der Regleraktivierungseinheit (RAE). So lässt sich die für eine Anfrage „Ich möchte ein Billigauto“ spezifische Regel aktivieren (weight=1), während das für eine Anfrage „Ich möchte ein gelbes Auto“ spezifische Kriterium unterdrückt wird (weight=0).
Die RAE erhält ihre Informationen aus der Fuzzy-Query. Wenn ein vollständiger AFR (siehe 5.1.2.2 Bewertung (AFR)) zu steuern ist, so kann die RAE entweder mit einer linearen Abbildung oder mit einem künstlichen Neuronalen Netz (s.u.) ein Eingangsmuster zu einem Erregungsmuster umsetzen.
Im Fall eines spezialisierten AFR’s kommt der RAE die Erzeugung der festverdrahteten Regeln zu.

feedback

Die Steuerung eines vollständigen AFR’s mit einem neuronalen Netz bietet den besonderen Vorteil ein lernfähiges System darzustellen, das zudem noch hohe Individualität in der Personalisierung zulässt. Dieser Entwurf erlaubt es beispielsweise auch feine Nuancen eines Empfindens der Farbe Rot abzubilden.
Die Aktivierung würde über die Zuweisung von Gewichtung erfolgen. In einer Query nicht gestellte Bedingungen erhielten die Gewichtung 0, was etwa interpretiert werden könnte als „interessiert mich nicht“. Es ist zweckmäßig für diesen Fall von Erregungsmustern zu sprechen, wie es im Bereich Neuronaler Netze üblich ist.
Ein am Regler anliegender Datensatz erregt dann, wie oben beschrieben, zunächst ein spezifisches Muster von Neuronen, die als Matrix von Sympathievektoren darstellbar sind. Die meisten (alle nicht interessierenden) Neuronenausgänge werden von dem steuernden ANN gehemmt, gemäß des ihm anliegenden Fuzzysignals. Das ANN wiederum wird von den individuellen Mustern der Personalisierung variiert.
Die von dem Künstlichen Neuronalen Netz modifizierbaren Parameter wären in der vollen Ausprägung eines solchen Systems die Projektionsparameter der (numerischen) Domänen auf das Einheitsintervall, der Verlauf der Zugehörigkeitswerte der Linguistischen Variablen, die Neuronen (das Abbildungsverhalten) der RAE und die Neuronen (Initialisierung des RAE) der Personalisierungseinheit.
Ein in dieser Weise mehrschichtiges, komplexes Neuronales Netz kann durch ein geeignetes Fehlersignal lernen, indem das Signal durch alle Schichten hindurch zurück propagiert wird.

Gleichzeitig lässt sich wegen der ihm innewohnenden Fuzzy-Funktionalität eine geeignete Anfangsinitialisierung implementieren, die Abbild des Wissens von Spezialisten ist. Auf diese Weise umgeht man mehrere typische Phänomene Neuronaler Netze:

- Schwierige Anfangsinitialisierung,
- Nichtinterpretierbarkeit,
- Verfangen in lokalen Minima.

Trotzdem die Erforschung Neuronaler Netze längst den Kinderschuhen entwachsen ist, sollte man nicht übersehen, dass dieser Forschungszweig immer noch mit vielen ungelösten Problemen kämpft, von denen eines in unserem Fall besonders schwer wiegen dürfte, das der mangelhaften Performance. Wenn diese Technik im PIA angewandt werden sollte, so müsste man von der effizienten Form der Trapeze als Zugehörigkeitsfunktionen abkommen und stattdessen beispielsweise Gauß-Glocken oder sigmoide Zugehörigkeitsfunktionen verwenden, die mit erheblich mehr Rechenaufwand verbunden wären. Nur mit ihrer Hilfe lassen sich geeignete Fehlersignale erzeugen.
Wirklich performant könnte ein solches Verfahren derzeit wohl nur sein, wenn es gelänge, den Regler in seiner derzeitigen Form mit trapezförmigen Zugehörigkeitsfunktionen zu belassen und die Funktion der Neuronalen Netze nur im ANN und der Personalisierung zu realisieren. Mittels der Gewichtung können ja auch im gewöhnlichen Regler inhibitorische Reize ausgeübt werden. Ein schwieriges Problem dürfte dabei aber die Nichtlinearität der Abbildungen im Regler sein (Trapezfunktionen), da es nicht einfach ist, durch diese Abbildung hindurch das Fehlersignal zu propagieren. Der Lernalgorithmus könnte in diesem Fall als nebenläufiger Prozess realisiert werden, der die Evaluierung einer Query nicht behindert.

Trotz der angesprochenen Performance Probleme, vertreten im Bereich der Neuronalen Netze und der Fuzzy-Logic viele Forscher die Meinung, dass die eine Technik ohne die andere nicht so richtig Sinn mache. So beklagen die einen die Schwierigkeit in der Praxis stimmige Zugehörigkeitsfunktionen zu finden und die anderen ihre Netze seien semantisch eine Blackbox, da niemand sagen könne, was ein Neuronales Netz genau macht und wieso es funktioniert. Der Erfinder der Fuzzy Logic Lotfi Zadeh gründete unter anderem aus dieser Misere heraus die neue Disziplin „Soft Computing“ in der er, neben den eben benannten Themen, noch genetische Algorithmen, Agenten-Theorie und ähnliche Disziplinen zusammenfaßt, damit sie zusammen einen Sinn ergäben.
Agenten, sei abschließend erwähnt, könnten auch in unserem Fall einen Sinn machen, indem sie unvollständige Information selbständig durch Zusatzinformationen aus anderen Informationsquellen ergänzen. Sie könnten beispielsweise herausfinden, dass es sich bei einem Passat um ein Auto der Marke Volkswagen handelt.

Mit dem vorgestellten Entwurf werden alle Anforderungen der PIA-Systems erfüllt. Einzig die Berechnung ternärer Funktionen bedarf noch einiger Erläuterungen.
Der Regler verwirklicht gemäß der bisher in PIA festgelegten Syntax für numerische Domänen folgende Konzepte:

- Existenz
- Fuzzy Value
- Fuzzy Relation
- Extended Fuzzy Relation
- Importance
- Complement
- 1. Normalform

5.1.2.5. Ternäre Funktionen (Extended Fuzzy Relation)

Schließlich müssen für den Ansatz von PIA, Abfragen der Form „T liegt ungefähr zwischen TB und TC“ zuzulassen, die vorgenannten Konzepte noch auf ternäre Funktionen erweitert werden:

Außerdem ist bei höherwertigen Funktionen (arity>1) zu bedenken, dass sie, trotzdem sie scharfe Funktionen sind, möglicherweise nicht als SQL-Clause implementiert sind und im Fuzzy-Regler realisiert werden müssen. Man denke beispielsweise an die Funktion

„x liegt zwischen a und b“

oder

„x liegt zwischen a und b oder zwischen c und d“.

Zur Lösung dieses Problems lassen sich zwei Ansätze finden. Zum einen könnte ein eigenes mathematisches Konzept entwickelt werden, wie im Kapitel 5. Lösungsentwurf angeführt, zum anderen ließe sich die ternäre Funktion mit Verknüpfungsoperatoren in zwei binäre Funktionen zerlegen (siehe hierzu auch Particularization and Conditional Possibility Distributions in [KaZe84] S.91):

(T ist größer als TB) und (T ist kleiner als TC).

Auch dieser Ansatz erfordert im Sinne eine theoretischen Fundierung eine weiterführende Diskussion, nämlich über die Funktion des „Und“.

Diese Funktionen lassen sich dennoch (mit gewissem Aufwand) als Ausschluss-Kriterien nutzen, wenn sie geschickt umformuliert werden:

[[x>a] and [x<b]] , bzw. [x>a] and [x<d]] .

5.2. Interface

Anschließend werden die Schnittstellen des Systems entworfen, nämlich einerseits die Query-Schnittstelle, mit der aus PIA heraus auf die FDBMS zugegriffen werden kann und andererseits die Administrationsschnittstelle, mit der die FDBMS verwaltet werden kann.

5.2.1. Query-Schnittstelle

In dem vorliegenden Entwurf ruft die Schnittstelle des PIA-Frontends eingangsseitig zwecks Evaluierung einer Query ein Objekt der Klasse Fuzzy-Application auf und übergibt diesem die Query. Diese Query setzt sich gemäß erweitertem Objektmodell (siehe 5.3 Erweiterung des PIA-Objektmodells) aus einer Condition und einer Presentation zusammen. Damit ist die zu untersuchende Datenmenge gegeben und die darauf anzuwendende Bedingung, gemäß derer die Datensätze zu bewerten sind. Der Aufruf des Fuzzy-Application-Objekts initialisiert also die Ablaufsteuerung, die ihrerseits die nötigen Schritte in die Wege leitet, insbesondere die Parameter an den Interpreter übergibt.
Dem Objekt Fuzzy-Application ist zu diesem Zweck eine Methode process_query(QueryParameters: queryparameters) zuzuordnen, die als Schnittstelle zum Frontend fungiert. Die Parameter werden in der Form übergeben, wie sie das Objektmodell vorsieht, nämlich vom Typ Condition und Presentation. Intern werden die Parameter dann zeitweise als SQL-String für Datenbankabfragen interpretiert.
Ausgangsseitig wird ein Objekt von Typ Offer erzeugt, das die relevanten Datensätze repräsentiert. Das Fuzzy-Application-Objekt antwortet auf die Methode process_query() mit einem Offer, wobei noch nicht festgelegt ist, wer für die Struktur der Offer zuständig ist. Diese

Frage kann erst bei der Implementation entschieden werden, in Abhängigkeit von einer zweckmäßigen Umsetzung. Gemäß des bisherigen Modell-Entwurfs liegt eine unsortierte Liste von Items in der Offer vor, die von der GUI-Application als präsentierendem Part noch für die Ausgabe aufbereitet werden muss. In dieser Version enthielte Offer alle Daten der Items und es müssten noch die relevanten Datenfelder vom Frontend ausgewählt und die Items nach MD und weiteren Kriterien sortiert werden.
Im Sinne einer verteilten Anwendung bieten sich zur Kommunikation der Application Objekte CGI-Interfaces nach dem Beispiel der Internet-Architekturen an, wie sie in der prototypischen Realisation beschrieben sind, oder aber andere Architekturen für verteilte Anwendungen, wie CORBA, COM oder OLE.
Die bisher vorgesehen Form einer Condition sei mit folgendem Beispiel repräsentiert; es handelt sich um eine einfache Folge Und-verknüpfter Kriterien, denen eine importance zugeordnet ist:
{(product category, is, car, normal),(performance, is very high, normal),(comfort, is not too low, normal),(lifetime, is high, important)}

5.2.2. Administrationsschnittstelle

Die Administrationsschnittstelle bezieht sich gemäß dem PIA-Objektmodell auf den Maintainer und den Provider. Dem Provider muss wegen der verteilten Struktur eine externe Schnittstelle zur Verfügung gestellt werden, um seinen Datenbestand pflegen zu können. Hier bietet sich wiederum die Einrichtung eines CGI-Interfaces an, mit dem der Provider auf seine Daten zugreifen kann. (Hinzufügen, Entfernen und Editieren).
Dem Maintainer dagegen wird eine einfaches Interface innerhalb der Applikation genügen, da i.A. davon ausgegangen werden kann, dass er ohne Einschränkung etwaiger Sicherheitsanforderungen an das System einen direkten Zugriff auf das System haben kann. Ihm obliegt mit den gegebenen Erweiterungen insbesondere auch die Pflege der Fuzzy-Zugehörigkeitsfunktionen und ggf. auch der Steuerungsparameter der Neuronalen Netze.

5.3. Erweiterung des PIA-Objektmodells

Um die vorgesehenen Fuzzy-Funktionalitäten zu ermöglichen muss das bestehende PIA Objekt-Modell noch um einige Klassen erweitert werden.

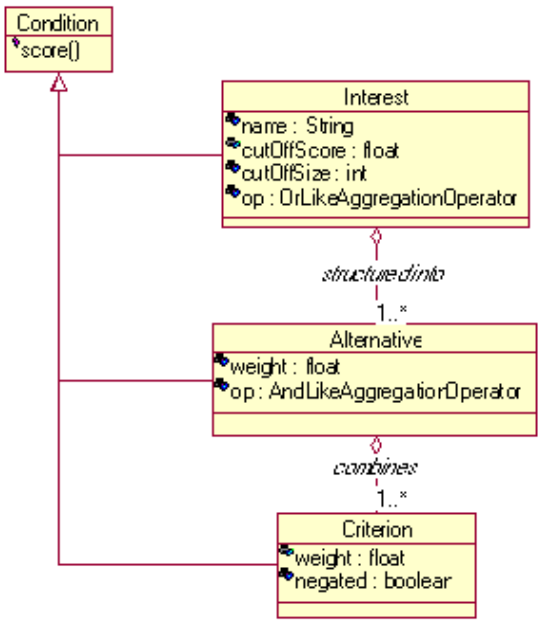

Abbildung 5-6: Objekt „Condition“

Zunächst werden die drei Klassen Interest, Alternative und Criterion unter einer Oberklasse Condition zusammengefasst. Von dieser erben sie die Methode score. Der Grund für dieses Vorgehen ist zum einen die Vereinheitlichung der Methode score und zum anderen ist es für die noch einzuführenden Klasse Fuzzy-Application unbedeutend, welche Form die Bedingung hat, die sie bearbeiten soll.
Auf der anderen Seite des Ausgangs-Modells wird Catalog-Collection als Generalisation von Catalog eingeführt, um Katalog-Verbünde zu ermöglichen, d.h. die Suche über verteilte Angebote. Der vorgestellte Lösungsentwurf erlaubt mit seiner temporären Tabelle relevanter Datensätze das Zusammenfassen solcher Verbünde, bevor sie abschließend bewertet werden.
Die Klassen Catalog.Collection, Catalog und Item werden in der neuen Oberklasse Presentation zusammengefaßt. Ihre gemeinsame Eigenschaft ist es Suchmenge einer Suchanfrage sein zu können.

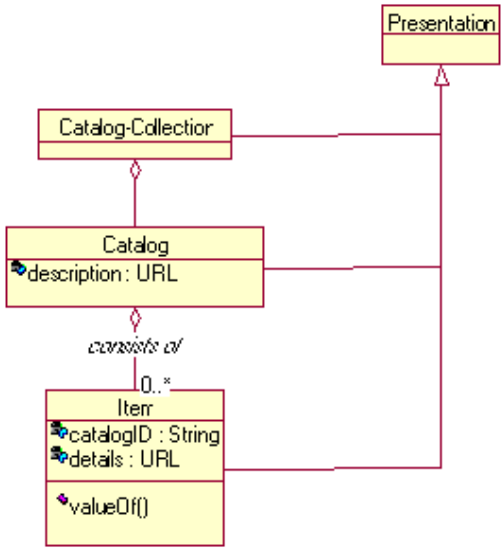

Abbildung 5-7: Objekt „Presentation“

Schließlich wird, aufsetzend auf Condition und Presentation die Klasse Fuzzy-Application eingeführt, die von dem bestehenden PIA-Frontend, einem Objekt der Klasse GUI-Application, angesprochen werden kann und eine Offer generiert.

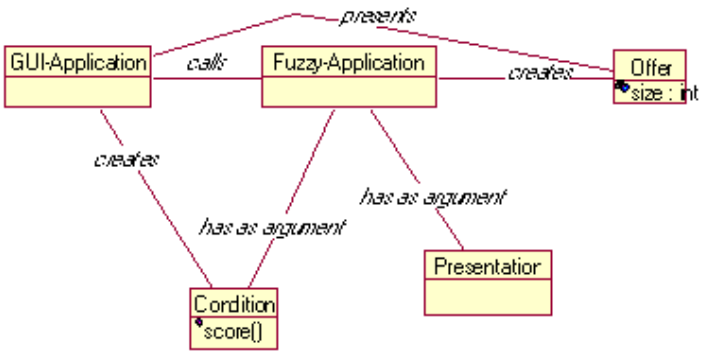

Abbildung 5-8: Objekt „Fuzzy-Application“

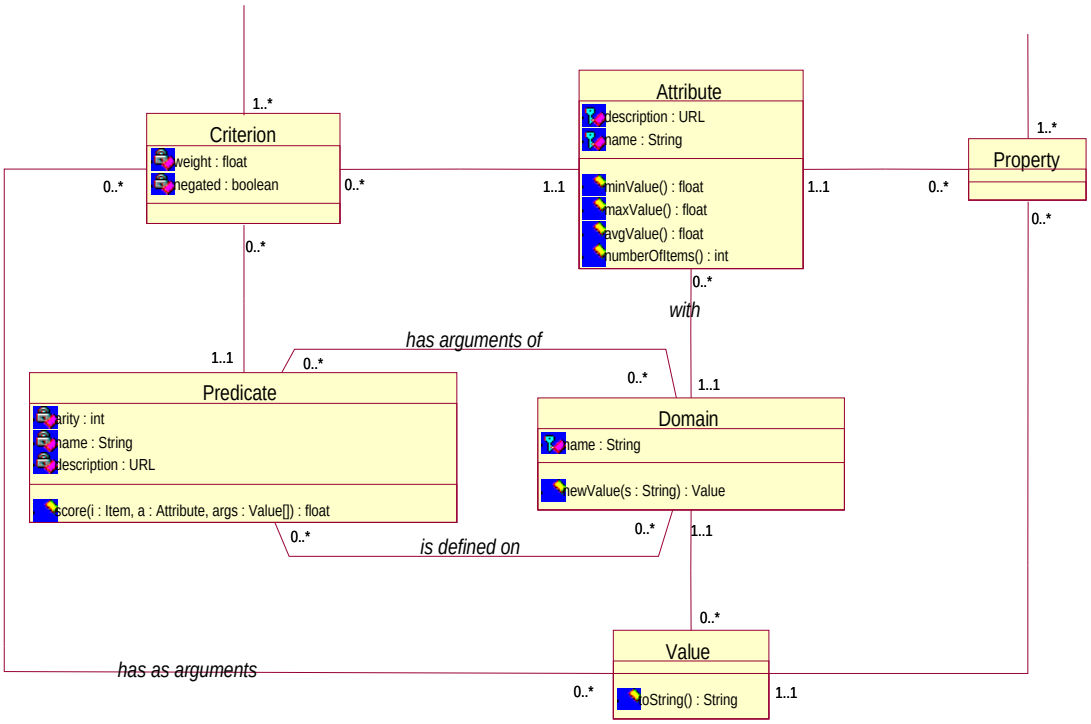

Abbildung 5-9: Das PIA-Domänenmodell

Abschließend sei noch eine Anregung zur Erweiterung der Klasse Predicate gegeben.
Wie bereits erwähnt entstehen bei der bisherigen Definition der Klasse semantische Defizite.[16]

Derzeit ist wird die Methode score() der Klasse Predicate beschrieben durch
Score(i:item, a:Attribute, Value[]): float
So ist beispielsweise das Prädikat teuer auf der Domäne Geldwert definiert und hat als unäre Funktion keine Argumente aus dieser Domäne.
Ein Kriterium benutzt dieses Prädikat z.B. für die Bedingung (Kaufpreis = teuer). teuer.score(Beispielauto X, Kaufpreis, -)

Bei der Ermittlung des score werden die Methoden minValue(), maxValue(), avgValue() und numberOfItems() herangezogen.
Hierbei muss sichergestellt werden, dass diese Methoden ihr Ergebnis in Abhängigkeit von bestimmten Eigenschaften des zu bewertenden Items ermitteln. Wenigstens die Eigenschaft, dass es sich um eine Preisbewertung bzgl. der Kategorie Auto handelt muss beachtet werden. Um auch noch zu berücksichtigen, dass es sich vielleicht um einen neueren Jaguar handelt und dieser bei einem angenommenen Preis von 5.000,- € relativ günstig ist, muss konzeptionelle Arbeit in den Begriff „relativ“ gesteckt werden.
Score(i:item, a:Attribute, Value[]): float
Stellt die Übergabe der Attribute des Items sicher.
minValue(), maxValue(), avgValue() und numberOfItems() müssen als Parameter diese Attribute mit übergeben bekommen, beispielsweise
Kaufpreis.minValue(Marke=“Jaguar“, Baujahr=“1998“)

[16] Zur Anschaulichkeit ist die Abbildung 2-2: Das PIA-Domänenmodell hier och einmal angeführt als Abbildung 5-9.

Kapitel 6 - Ergebnis

Für die prototypische Realisierung der Fuzzy-Datenbank Anfragen wurde die Software FfA modifiziert. (Siehe Kapitel 4: FfA)
Das erzeugte System besteht aus mehreren Teilen und ist im Sinne des eben beschriebenen Objektmodells eine Realisation der Methode „score“ des Objekts „condition“. Neben dieser Schnittstelle (FfA/PIA) ist die Zweite die relationale Datenbank, die derzeit in PIA die angestrebte objektorientierte Datenbank simuliert.

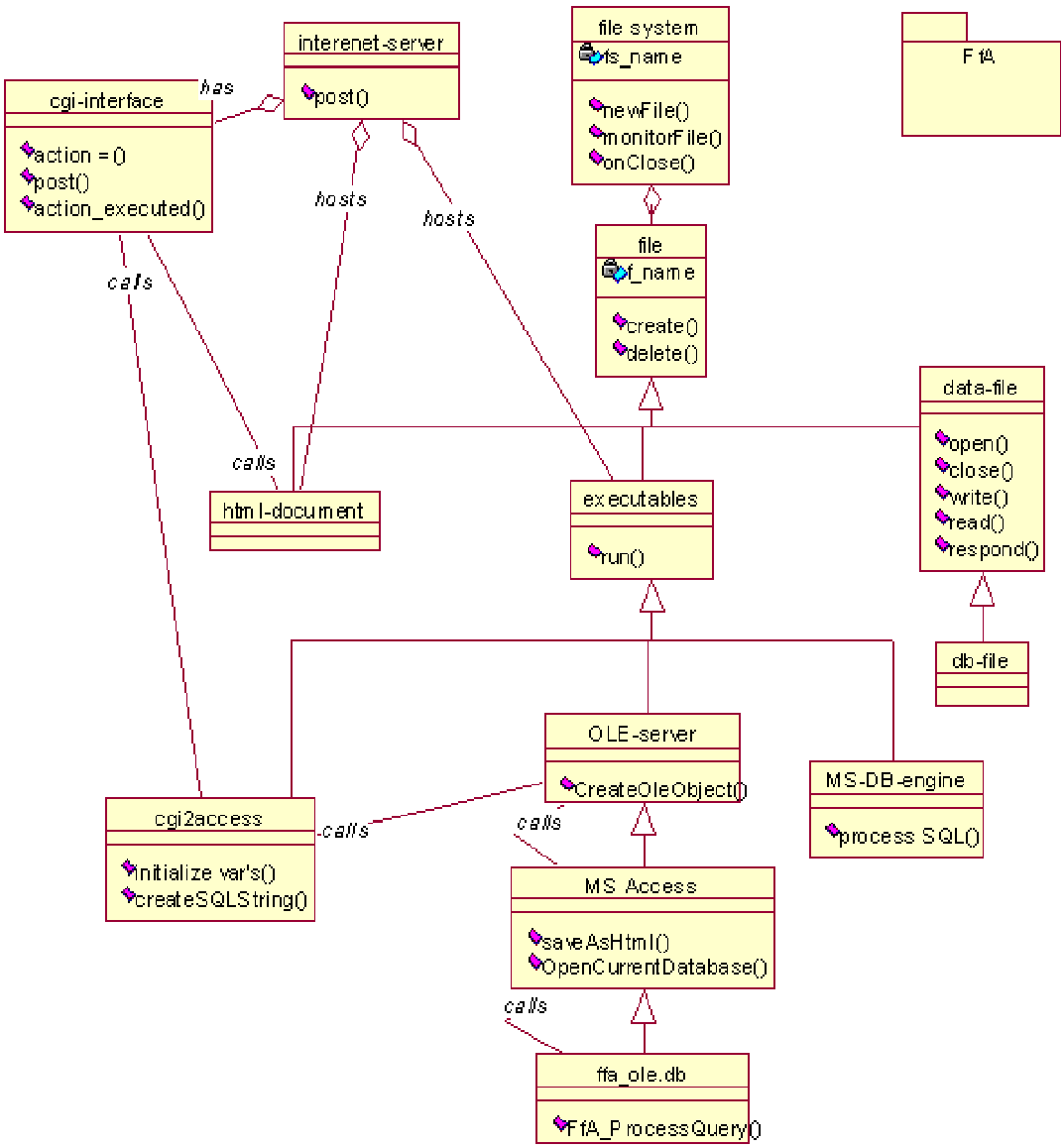

Abbildung 6-1: main class diagramm

Client-Abfragen werden in einer dreischichtigen Architektur bearbeitet:

1. Ein Webbrowser ermöglicht den komfortablen Entwurf einer Abfrage und dient der Darstellung des Ergebnisses.

2. Ein in Delphi geschriebenes CGI-Programm bereitet die Daten auf und vermittelt zwischen einem Windows-Webserver und der eigentlichen Applikation.
3. Die in VBA geschriebene Access-Applikation FfA wird per OLE von dem CGI-Programm gesteuert und ermittelt die Ergebnismenge.

Das System wird mittels verschiedener UML-Diagramme beschrieben. Zunächst sieht man in einem übergeordneten Klassendiagramm (Abbildung 6-1: main class diagramm) die Zusammenhänge der verschiedenen Klassen im Gesamtprozess. Man erkennt hier zunächst als wesentliche Funktionalität den Weg des Call‘s eines Html-Dokuments an ffa_ole.db.FfA_ProcessQuery. Aus Gründen der Vereinfachung wurde die Rolle des Internet-Clients hier vernachlässigt. Die ausgelösten Prozesse werden in Abbildung 6-2: start_query sequence diagramm veranschaulicht, wobei der die Berechnung durchführende Prozess die Nummer 26 FfA_ProcessQuery ist, der ungefähr in der Mitte des Diagramms steht.

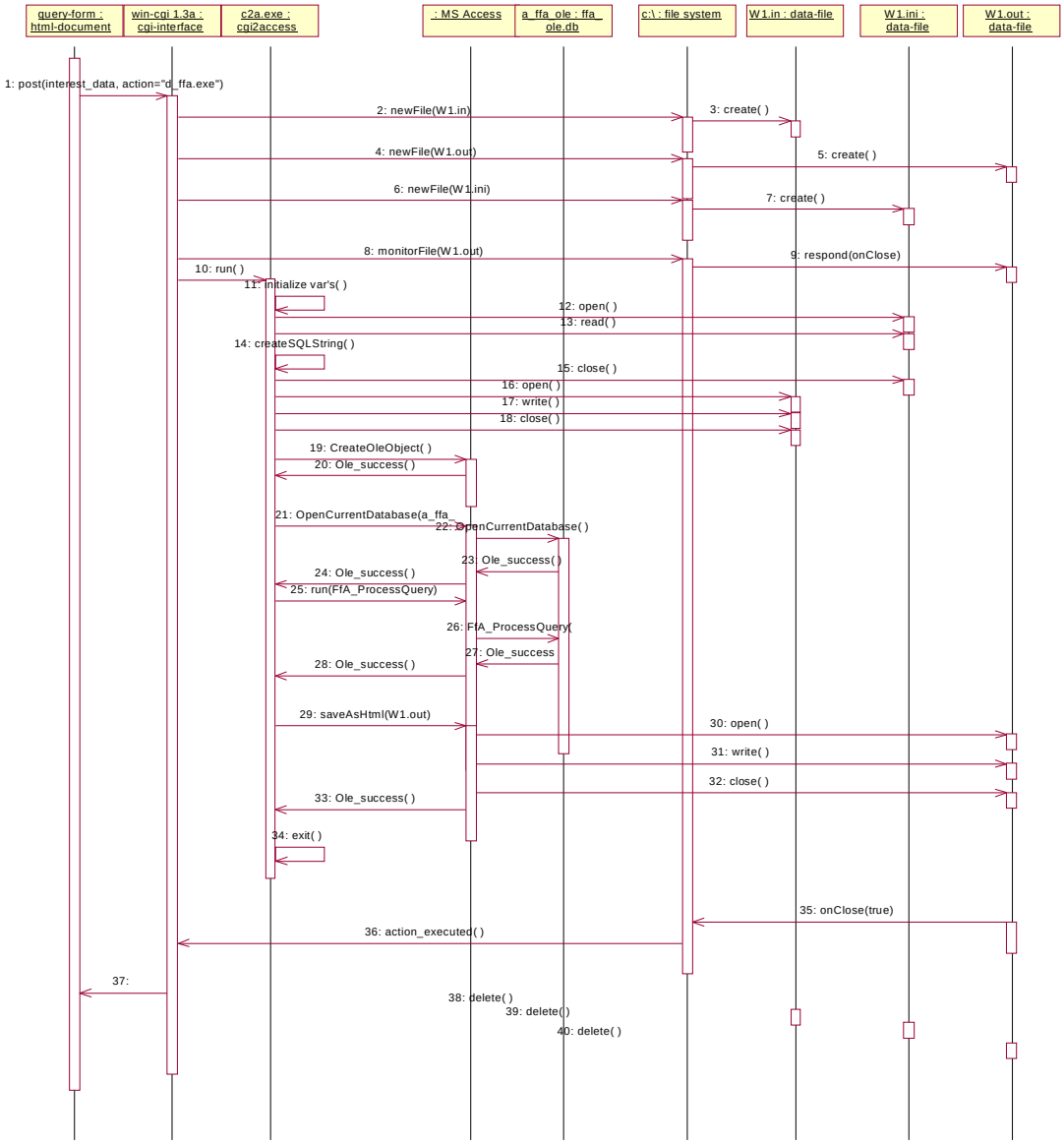

Abbildung 6-2: start_query sequence diagramm

Der Ablauf sei kurz beschrieben:
Das auszuwertende Interesse wird zu Demonstrationszwecken mit einem Formular in Form eines Html-Dokuments nachgebildet (siehe Abbildung 6-3: Html-Query). Später wird es durch das PIA-Frontend abgelöst. Es lässt sich in seiner gegenwärtigen Form nur von Microsofts Internet Explorer öffnen. Dies rührt daher, das es komplett auf Javascript basiert, welches in unterschiedlichen Browsern unterschiedlich implementiert ist. Es enthält keine einzige Zeile rudimentären Html-Codes, da der gesamte Inhalt des Dokuments per Write()-Befehlen geschrieben wird. Der Vorteil, der sich daraus ergibt, ist, dass es sich besonders dynamisch den User-Preferenzen anpassen kann, bzgl. gewünschter Komplexität und Interpretation der Abfrage.
Die Einträge der Felder werden mit action="/cgi-win/c2a.EXE", method="post" an den Web-Server gesandt, der die Applikation in seinen CGI-Pfaden sucht. In diesem Fall findet er sie im win-cgi-Verzeichnis. Im Falle des Shareware Webservers Sambar V4.1 wird daraufhin das Windows CGI 1.3a Interface genutzt, welches verschiedene Workarounds für fehlende Features von Windows-Webservern im Vergleich zu Unix-Servern bietet. Speziell das fehlende stdin und stdout werden durch temporäre Dateien ersetzt. Beim Aufruf eines CGI-Files werden drei Dateien erzeugt (*i* steht für einen fortlaufenden Integer, der Überschneidungen verschiedener Prozesse verhindert):
W*i*.txt enthält die Formulardaten als wären sie per Methode „get" gesandt worden als einen String.
W*i*.ini enthält die gesamten Umgebungsvariablen inklusive der Formularfelder mit Values in Form der Windows typischen Ini-Files.
W*i*.out ist eine Datei, die für die Ausgabe gedacht ist. Wenn sie geschlossen wird, übernimmt der Web-Server sie und sendet sie zum anfragenden Client.
Diesen Effekt macht sich die CGI-Applikation c2a.EXE zunutze (siehe 6.2 Die Schnittstelle C2a.exe). Es wurde in der Entwicklungsumgebung Delphi 2.0 in Object-Pascal geschrieben und hat die Aufgabe zwischen der WIN-CGI 3.1 Schnittstelle (siehe www.sambar.com) und der FfA-Applikation in Access zu vermitteln.
Es übersetzt die in W*i*.ini enthaltenen Formulardaten in gültige FfA-SQL-Syntax und steuert die Prozeduren des Moduls Query und die Speicherfunktionen von Access mittels OLE 2.0.
Zunächst startet es einen OLE-Server Prozess auf Access. Dieser wird im zweiten Schritt dazu angehalten eine Replikation der Datenbank zu öffnen, in der sich das Modul query befindet, dessen Funktion FfA_processQuery die Interpretation der Query übernehmen soll. Dann wird die Prozedur mit den aktuellen Parametern gestartet. Access kompiliert den aus FfA 2.0 extrahierten VBA-Code und berechnet in der Access-Umgebung das Ergebnis der Anfrage. Der Prozess endet mit einer Erfolgsmeldung an den OLE-Server, der dies an den Delphi-Client meldet. Der ruft wiederum einen Access-Prozeß auf, um die Ergebnisdatei in Html-Form unter dem Namen der temporären Datei W*i*.out zu speichern, die vom WIN-CGI Interface beobachtet wird (Monitoring). Das Speichern dieser Datei löst ein OnClose-Ereignis aus, woraufhin sie von WIN-CGI an den Client, der die Anfrage gestellt hat, zurückgesandt wird und im Browser-Fenster erscheint.
Da mehrere OLE-Server-Prozesse gleichzeitig auf Access gestartet werden können und mit Replikationen der Original Datenbank arbeiten ist das System bzgl. Abfragen Multi-User fähig.

6.1. Das Query-Formular

Das Formular (siehe Abbildung 6-3: Html-Query) enthält die Standard-Schalter Submit und Reset und besteht außerdem aus einer Reihe von Auswahlfeldern, mit denen die Anfrage formuliert werden kann.

Die Auswahlfelder strukturieren sich nach
- Parameter der Query (Interest)
- Parameter der Subcondition (Alternative)
- Parameter der Condition (Criterion)

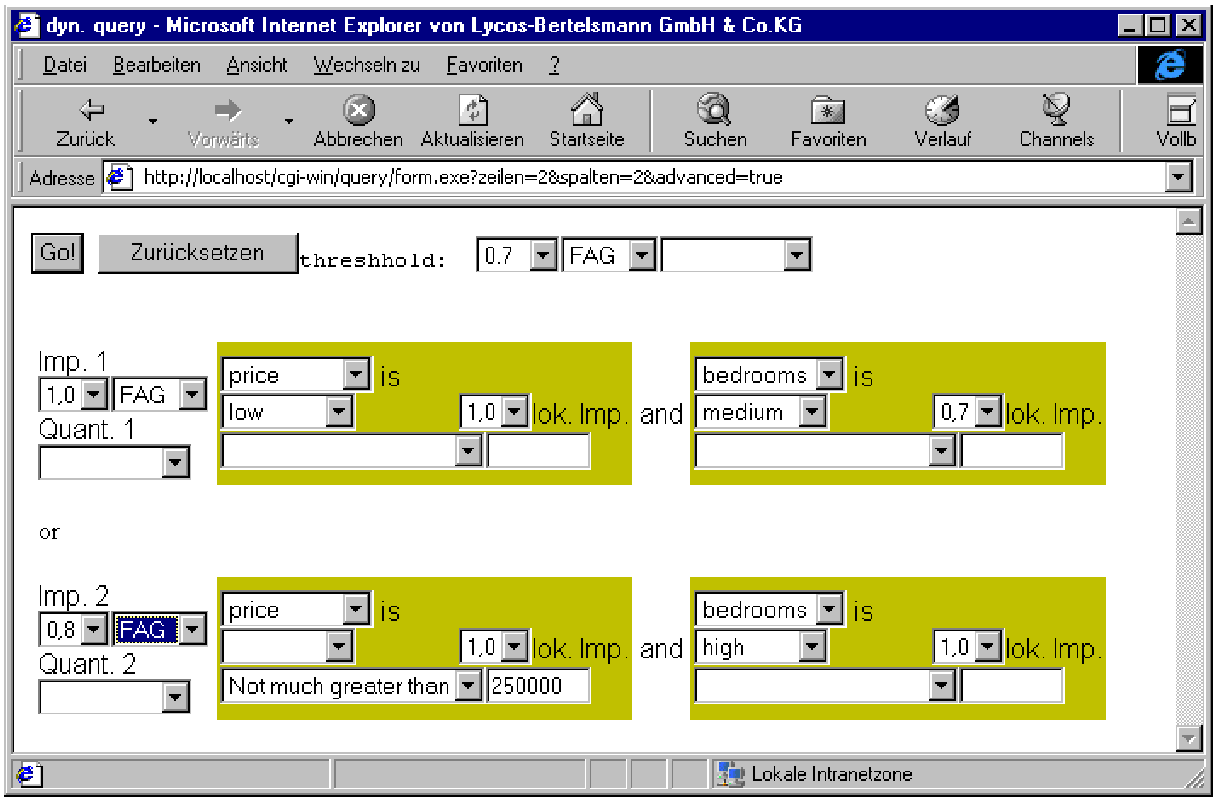

Abbildung 6-3: Html-Query

1. Parameter der Query (Interest)

Threshhold

Grenzwert, den der Matching Degree eines jeden Datensatzes überschreiten muss, damit er als der Query zugehörig interpretiert wird.

Interpretation der Globalen Importance

Methode mit der die Gewichtung der Subconditions interpretiert werden sollen. (nach Fagin, OWA oder PIA)

Fuzzy Quantifier der Query

Modifikator für den Verknüpfungsoperator der Subconditions

2. Parameter der Subcondition (Alternative)

Globale Importance

Zuordnung einer Bedeutung (weight) zu einer Subcondition

Interpretation der Lokalen Importance

Methode mit der die Gewichtung der Conditions interpretiert werden sollen.

Fuzzy Quantifier der Subcondition

Modifikator für den Verknüpfungsoperator der Conditions

3. Parameter der Condition (Criterion)

Attributname
Angabe, welcher Wert eines Datensatzes für eine Bedingung herangezogen werden soll.

Globale Importance
Zuordnung einer Bedeutung (weight) zu einer Condition

Fuzzy Value
Möglichkeit das Attribut mit einer unären Funktion zu bewerten

Fuzzy Relation
Möglichkeit das Attribut mit einer binären Funktion zu bewerten

Vergleichswert der Fuzzy Relation
Parameter für den Fall einer gesetzten binären Bedingung

Mit dem aus diesen Elementen gebildete Formular wird die Query formuliert. Wenigstens eine Bedingung muß gesetzt sein. Sowohl den Atomen, als auch den Subconditions lassen sich Gewichtungen zuschreiben, deren Interpretation sich einstellen lässt. Voreingestellt ist die Interpretation nach Fagin. Die Evaluierung wird über den Go-Button ausgelöst.
Die Datei lässt sich als /cgi-win/query/form.exe aufrufen. Optional können die Parameter „zeilen“, „spalten“ und „advanced“ per get-Parameter übergeben werden, etwa

```
http://localhost/cgi-win/query/form.exe?zeilen=3&spalten=4&advanced=true.
```

Das Formular kann damit auf bis zu 10 Zeilen und bis zu 10 Spalten erweitert werden. Wenn „advanced=true“ gesetzt wird, erscheinen die erweiterten Eingabeoptionen, mit denen sich beispielsweise die Interpretation der Gewichtung wählen lässt. Ohne diesen Parameter erscheint das Formular nur mit den wichtigsten Eingabeoptionen. Per Voreinstellung besteht die Query aus 2 Zeilen und 2 Spalten.
Der erzeugte Html-Code lässt sich mit der Datei query.cfg manipulieren. Hier kann man beispielweise einstellen, welche Fuzzy-Values man zur Auswahl stellen möchte.

6.2. Die Schnittstelle C2a.exe

Der Name der Schnittstelle steht für „CGI to Access“. Grundidee dieser Delphi-Applikation ist es die FfA Software in ihrer MS-Access-Umgebung zugänglich zu machen.
Der Webserver Sambar bekommt die Query in Form von Post-Parametern. Diese werden gemäß der WIN-CGI Spezifikation in einer temporären Ini-Dateien abgelegt. Daraufhin wird c2a.exe mit drei Kommandozeilen-Parametern aufgerufen, die Verweise auf temporäre Dateien sind. Dies sind die erwähnte Ini-Datei, eine Datei mit den Parametern im Get-Format und eine Datei die zur Ergebnisausgabe vorgesehen ist. Diese wird vom Webserver beobachtet. Tritt ein Schließen-Ereignis auf ihr auf, so wird sie als Ergebnis an den fragenden Client gesandt.
Um mit den beschriebenen Konzepten auf eine Formularanfrage mit einer html-Seite zu reagieren müssen verschiedene Funktionen implementiert werden

- Übernahme der Kommandozeilen-Parameter
- Auslesen der Ini-Datei
- Übersetzung der Formularparameter in das FfA eigene SQL (FQUERY)
- Ergebnistabelle gemäß Query errechnen
- Ergebnistabelle als html-Datei formatieren

- Ergebnis unter dem spezifischen Dateinamen speichern

Das Access eigene Visual Basic erweist sich für diese Aufgabe als ungeeignet. Es kann beispielsweise keine Kommandozeilen-Parameter übernehmen und enthält auch keine Methoden, die der Syntax von Windows-Ini-Dateien Rechnung tragen. Zur Steuerung der Abfrage wurde daher Delphi gewählt. Das FfA-Access-Modul wird als OLE-Prozess angesprochen.

Das Programm besteht aus Initialisierung (Query und Umgebungsvariablen in interne Variablen einlesen), erzeugen eines entsprechenden FQUERY-Strings und Steuerung der Access-Datenbank über OLE (Ergebnis berechnen und als Html-Datei abspeichern). Dabei werden typische Fehlerfälle abgefangen und gegebenenfalls in der Ergebnisdatei als Fehler benannt.
Im Falle einer Fagin-Interpretation von Gewichtung wird die notwendige Sortierung und Normierung vorgenommen. Die Anwendung wird lässt sich über eine Datei d2a.ini steuern, die im gleichen Verzeichnis wie die Applikation selbst liegt.
Die Datei enthält folgende Parameter:

```
[config]
database=\dbms\ffa_ole.mdb
default_in=\tmp\debug\default_ffa.ini
default_query=\tmp\debug\default_ffa.txt
default_out=\tmp\debug\default_ffa.out
Access_OLE =Access.Application.8
```

Die Pfadangaben sind relativ zur Wurzelverzeichnis des Webservers. „database“ ist die Datenbank, die durchsucht werden soll. Die anderen drei Dateien dienen Debugzwecken. Zu Entwicklungszwecken arbeitet das Programm auch ohne Kommandozeilenparameter. In diesem Fall werden die drei genannten Dateien verwendet.
Der Wert von „Access_OLE” findet sich in der Windows-Registry unter “HKEY_CLASSES_ROOT\Access.Application\CurVer”. Es handelt sich um den internen (OLE-)Namen der Access-Version. Er muss gegebenenfalls maschinenabhängig angepasst werden.

6.3. Ffa-Ole als Realisation der FDBMS

Die ursprüngliche Version von FfA 2.0 wurde für die Zwecke von PIA umgeschrieben.
Der Original-Code besteht aus insgesamt fünf Modulen, die zu einem zusammengefasst wurden. Der Quellcode reduzierte sich so von ehedem 3000 Zeilen auf ca. 1000. Dieser Effekt ist hauptsächlich darauf zurückzuführen, dass die Funktionalität der Software auf zwei Anwendungsformen aufgeteilt wurde. Die Funktion Client-Abfragen beantworten zu können wird mit dieser Version realisiert. Da diese Version ausschließlich von der CGI-Anwendung per OLE gesteuert wird konnte auf die aufwendigen Window-Handels verzichtet werden. Die Administrationsfunktionen (Maintainer) hingegen werden von einer gesonderten Version realisiert, die in 6.4 Administrationsinterface beschrieben wird. Die Administrationsfunktionen (Provider) werden derzeit noch von dieser Schnittstelle miterledigt. Wie im Lösungsentwurf beschrieben ist aber eine CGI-Schnittstelle zu den Tabellen der Datenbank wünschenswert, um eine Fernwartung der Daten zu gewährleisten.
Die Trennung gilt der Modularisierung der Software. Die aufwendigen Editierfunktionen machen die Client-Version langsam und unübersichtlich. Da sie nur gelegentlich benötigt werden, wurden sie ausgegliedert.
Das verbleibende Modul trägt den Namen „Query“ und enthält die Objekte, die im Klassen-Diagramm Abbildung 6-4 dargestellt sind. Ihr Zusammenwirken wird in Abbildung 6-5: process_query Sequnence Diagramm Query dargestellt. Zu sehen sind MS Access als Hauptklasse die sich in Module unterteilt, die Prozeduren enthalten, die ihrerseits in funktionale Unterklassen gegliedert wurden, die als Pseudo-Sub bezeichnet wurden.

An Prozeduren wurden nur die beiden wichtigsten aufgeführt: FfA_ProcessQuery und FfA_MD.
FfA_ProcessQuery ist ein vorbereitender Vorgang, in dem der Pseudo-SQL Code von FfA in gültige Access Syntax transferiert wird. Am Ende dieses Parse-Vorgangs wird das entstandene SQL-Statement evaluiert. In ihm enthalten ist ein Prozeduraufruf FfA_MD, quasi eine Access Stored Procedure. Sie führt die Berechnung des MDs über alle Datensätze aus und erzeugt das Neue Attribut MD. Schließlich wird eine temporäre Tabelle erzeugt, die alle Datensätze enthält deren MD die Threshhold-Schwelle überschritten haben. Per OLE wird diese von d2a.exe als Html in der gemonitorten Datei gespeichert und vom Server an den Client weitergegeben.

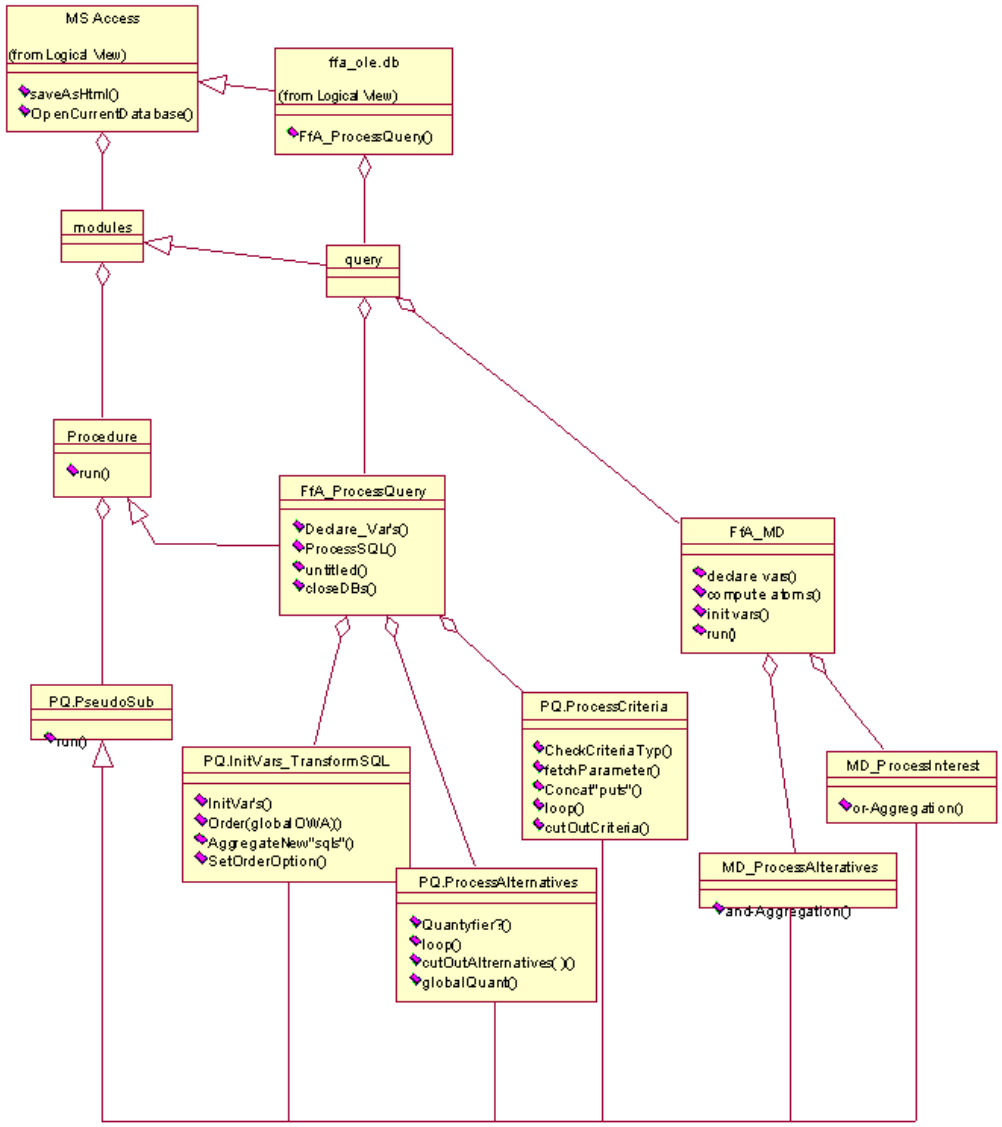

Abbildung 6-4: ffa Class Diagramm Query

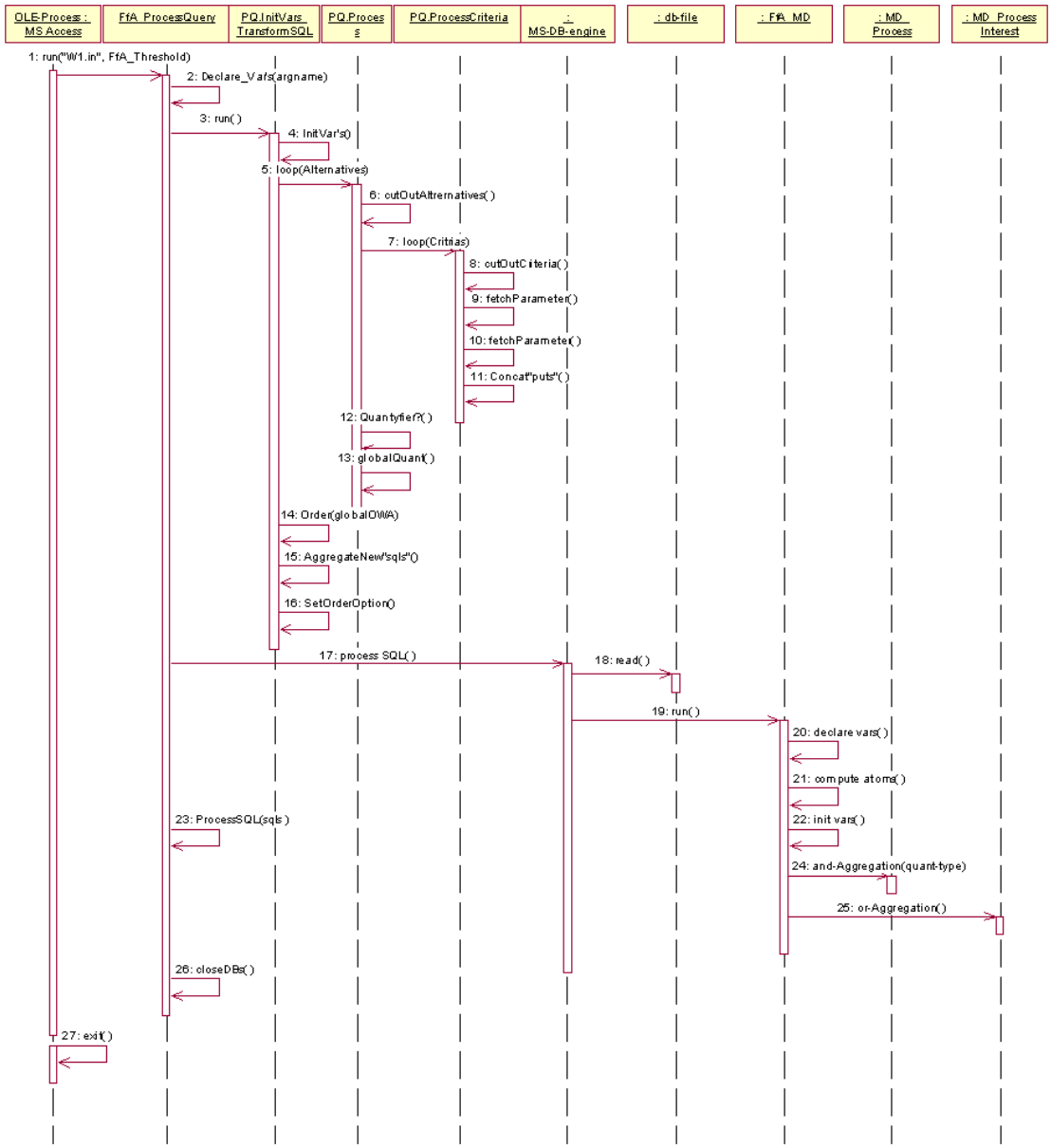

Abbildung 6-5: process_query Sequnence Diagramm Query

Die Hauptprozedur heißt

```
FfA_ProcessQuery(
sql_str As String,
FfA_Threshold As Double,
st_FAG As String)
```

Der übergebene Parameter sql_str ist ein String der den Namen der Datei angibt, in der der auszuführende String in FfA-SQL-Syntax steht. Auf diese Weise lässt sich das Modul sowohl aus der Datenbank-Umgebung heraus leicht testen als auch vom CGI-File per OLE steuern. Test werden mittels des Go!-Buttons des Formulars FfA-Toolbar1 durchgeführt. Im Falle des direkten Aufrufs aus Access heraus (Debug-Level 1) wird der SQL-String ausgeführt, der

beispielsweise in der Abfrage "C:\Programme\sambar41\tmp\bakw3.txt" editierbar ist. Der Ort dieser Datei wird im Modul selbst angegeben (in der Funktion FfA_RunQuery()).
Die Funktion von FfA_ProcessQuery lässt sich folgendermaßen zusammenfassen:
Zunächst wird die Syntax auf Konformität geprüft. Insbesondere wird kontrolliert, ob die SELECT-FROM-WHERE-Form eingehalten wurde und ob keiner der Operatoren IMP, EQV, XOR oder OR verwendet wurde.
Dann wird der Bedingungsteil der Query freigestellt und in „wheres“ gespeichert.
Daraus werden die Alternativen extrahiert und in „ands“ nacheinander zwischengespeichert, während „wheres“ jeweils um „ands“ verkleinert wird.
Aus den „ands“ werden die einzelnen Kriterien extrahiert und im Vektor „Atoms“ abgelegt. Mit jeder Extraktion wird ein neues Objekt von Typ Atom_Type erzeugt und mit Werten belegt.
Der Typ hat folgende Definition:

```
Type Atom_Type
type As Integer          ' Fuzzy value (=FfA_FVType) OR fuzzy
                         relation (=FfA_FRType) OR non-fuzzy atomic
                         condition (=FfA_CrispType)
attr(2) As Double        ' range of the attribute copied from the
                         table FFA_ATTRIBUTES
attr_value As Variant    'Place for the values of the attribute (or,
                         rather corresponding field) appearing in
                         given atomic condition in subsequent
                         records. Set during querying process. In the
                         case of FfA_CrispType  contains logical
                         value corresponding to fulfilment of this
                         atomic condition.
secop(2) As Double       ' as above for the second operand appearing
                         in a given atomic condition
secop_value As Variant
fv(4) As Double           set once before querying
                         Parameter des Attributs (auge)
attrname As String
fvname As String
secopname As String
impor As Single
```

Die einzelnen Attribute der Atome sind weitestgehend selbsterklärend. Zu „impor“ sei jedoch angemerkt, dass dieser Wert nicht zur Verwendung in einzelnen Kriterien vorgesehen ist, sondern lediglich zur Bewertung der einzelnen Interessen, was sich nicht mit dem Objekt-Ansatz von PIA deckt. Hier können auch die Kriterien eine Gewichtung (weight) tragen.
Um diesen Mangel zu überkommen wurde der zusätzliche Parameter st_FAG eingeführt. Er übergibt in Form eines Strings die lokalen Gewichtungen (auf Ebene der Kriterien einer Alternative) an FfA_ProcessQuery. Sie sind durch Leerzeichen getrennte Gleitkommazahlen. Falls dieser String leer ist, wird eine lokale Gewichtung von Eins für jedes Element angenommen.
Die Funktion des Attributs impor in obiger Typdefinition ist verwirrend. Um einer Alternative eine (globale) Gewichtung zuzuschreiben wird ihr eine zusätzliches beliebiges Attribut hinzugefügt, das mit der gewünschten Gewichtung belegt wird (Bsp. (HOUSES.BATHROOMS)=[FfA_FI 0.5])). Um dieser Verwirrung zumindest ein wenig entgegenzuwirken setzt die Delphi-Schnittstelle das Pseudo-Attribut „IgnoreMe“ ein: (HOUSES.ignoreMe)=[FfA_FI 0.5].

6.4. Administrationsinterface

Wie oben angeführt, wurde aus Performancegründen und wegen der gegebenen Funktionalität, die Administrationsoberfläche von FfA97WWW als Realisierungsgrundlage für die Verwaltung der Fuzzy-Funktionen verwendet. Da auf den Quellcode dieser Version nicht zugegriffen werden konnte, greift die für PIA modifizierte Version von FfA20 auf die Tabellen dieses Add-

In’s zu, anstatt, wie es performanter gewesen wäre, FfA97 auf die Tabellen von FfA20 zugreifen zu lassen.
Um die Editierfunktionen zu nutzen, wird FfA97 als Access-DB geöffnet. Wenn man nun das Formular Toolbar1 auswählt, stehen einem die in 4.3 FfA-Toolbar beschriebenen Möglichkeiten zur Verfügung um die Parameter der verschiedenen Fuzzy-Funktionen zu editieren.

6.5. Zusammenfassung

Das hier dargestellte Ergebnis der prototypischen Implementation ermöglich mit Hilfe des Webservers, des Web-Formulars, der Applikation c2a.exe, der OLE-fähigen Datenbank und des AddIns den unscharfen Zugriff auf numerische Datenbanken und das Editieren der steuernden Parameter.
Um den Prototyp zu nutzen wird lediglich die dieser Arbeit beigefügte CD benötigt, sowie MS Access in der Version 97 oder 2000. Alle Konfigurationsdateien lassen sich mit gewöhnlichen Texteditoren bearbeiten. Die Applikation lässt sich für andere Datenbanken mittels Access anpassen. Lediglich für den Fall funktionaler Änderungen der Schnittstelle c2a.exe wird zusätzlich eine Version von Borlands Delphi benötigt.

Kapitel 7 - Zusammenfassung und Ausblick

In dieser Arbeit wurden für PIA die Möglichkeiten des Einsatzes der Fuzzy-Logic untersucht, um die Zugehörigkeit von Datensätzen zu ganz oder teilweise unscharfen Bedingungen zu ermitteln. Die Bedingungen mussten der ersten Normalform genügen und einzelne Bedingungen mussten sich mittels einer Gewichtung "Importance" abschwächen lassen.
Nach einer einführenden Betrachtung zu PIA , die die Randbedingungen dieser Arbeit beschrieb, wurden in knapper aber umfassender Form die Prinzipien, Stärken und Schwächen der Fuzzy-Logic, insbesondere in Hinblick auf Datenbankabfragen erläutert.. Anschließend ergab ein ausführlicher Einblick in die Software FfA sowohl tiefere Einblicke in die praktischen Probleme des Einsatzes der Fuzzy-Logic in Datenbanken, als auch die Grundlagen zum Verständnis der prototypischen Realisierung, die auf dieser Software basiert.

Nachdem solchermaßen die Grundlagen des Problems erläutert waren, wurden die beiden weiteren Ergebnisse dieser Arbeit präsentiert: Der Lösungsentwurf und die prototypische Realisierung.
Der Lösungsentwurf gründet auf dem Entwurf einer neuen, selbstentwickelten Systemarchitektur, die den Einsatz der Fuzzy-Methodik als Modul zulässt, dass in die Kommunikation zwischen DBMS und DBE geschaltet wird. Dieser modulare Ansatz lässt einerseits die Integration in bestehende Systeme zu und bietet andererseits die Möglichkeit einzelne Funktionalitäten (Zugriff auf beispielsweise numerische Felder in Datenbanken vs. unscharfe Abfragen in Information Retrival Systemen) unabhängig voneinander zu realisieren.

Während der Erläuterungen zu dem System wurde aufgezeigt unter welchen Bedingungen sich die Berechnung eines Zugehörigkeitswertes auf einfache Matrizen Operationen beschränkt, womit dem "Need for Speed" Rechnung getragen wurde, der für kommerzielle Datenbank-Anwendungen i. A. und im Falle umfangreicher Bewertungsfunktionen wie dieser im Besonderen, als Nadelöhr zu einer breiten Akzeptanz gesehen werden kann.
Ein weiteres wichtiges Augenmerk wurde darauf gelegt, dass das System grundsätzlich lernfähig im Sinne der Theorie der Neuronalen Netze ist. Das Problem des rückwirkenden Fehlersignals kann zwar mit dem vorgestellten System nicht als gelöst angesehen werden, bietet aber eine gute Grundlage für die Entwicklung eines solchen Systems. Dabei wird sich das Performance Problem aber noch einmal deutlich verschärfen. Zur Lösung dieses Problems wird daher ein Offline Lernen in einem zweiten parallel laufenden System vorgeschlagen.

Mit dem vorgestellten System wird eine Lösung für alle gestellten Anforderungen bereitgestellt:
- Existenz
- Fuzzy Value
- Fuzzy Relation
- Extended Fuzzy Relation
- Importance
- Complement
- 1. Normalform

Mit der prototypischen Realisierung wird den Entwicklern von PIA ein Werkzeug an die Hand gegeben, um die verschiedenen angeforderten Methoden in unterschiedlichen Interpretationen zu evaluieren, da auch der Prototyp allen obigen Anforderungen genügt.
Das System erlaubt es über ein Webinterface, das derzeit stellvertretend für die PIA-Oberfläche steht, von einem beliebigen Rechner aus auf alle ODBC-fähigen, mit dem Rechner verbundenen Datenbanken zuzugreifen.

Von einem Prototyp muss indes trotzdem gesprochen werden, da es sich keineswegs um eine saubere Implementation im Sinne des Software-Engineerings handelt, sondern vielmehr um die "zurechtgebogene" Version einer über Jahre gewachsenen Software, die technologisch auf unzeitgemäßen und hohen Ansprüchen nicht genügenden Systemen aufsetzt.
So kann vom Einsatz des modifizierten Access-Add-Ins auf einem produktiven Webserver nur abgeraten werden, da die Fehlerbehandlungsroutinen nicht nur unvollständig, sondern auch nicht Web-gerecht implementiert sind.
Dem Anspruch eines Testsystems indes schienen die vorgenommenen Modifikationen zu genügen, so dass die Implementation als Lösung der gestellten Aufgabe gelten kann. Statt auf die Stabilität wurde das Augenmerk auf die Erweiterung der Funktionsvielfalt gelegt.

Die Praxistauglichkeit des Systems wird sich zunächst im Zusammenspiel mit PIA erweisen müssen. Bei Erfolg im Zusammenspiel mit dem Gesamtsystem wird die nächste Klippe auf dem Weg zum praktischen Einsatz die Performance in einem realen Einsatz von PIA liegen. Es steht zu vermuten, dass die gewachsene Struktur des Prototyps diesen Anforderungen nicht gerecht wird, so dass eine Implementierung des offenen Modellentwurfs erforderlich sein könnte.
Dieser Schritt scheint auch angeraten, für den Fall, dass PIA zu einem lernfähigen System ausgebaut wird. Der Einsatz Neuronaler Netzen mit Error-Backpropagation-Funktionalität erfordert in jedem Fall ein sehr viel klarer strukturiertes System. Eine Personalisierung der Fuzzy-Zugehörigkeitsfunktionen hingegen ließe sich wohl auch mit diesem System noch leisten.

Für zukünftige Erweiterungen des Systems könnte weiter der Einsatz von Fuzzy Thesauri in Betracht gezogen werden. Darin wird einem Begriff wie "Schrank" explizit eine gewisse Ähnlichkeit zu Kommode zugeordnet. Diese Methodik könnte sich auch für Fuzzy-Ähnlichkeiten in hierarchischen Domänen als zweckdienlich erweisen.
Eine weitere Methodik, die gegebenenfalls eigenständige Untersuchungen rechtfertigen könnte, wäre der Einsatz einer Fuzzy-phonetischen Ähnlichkeitssuche, die heute schon in einigen kommerziellen Systemen zum Einsatz kommt (Pauschalreisedatenbank Bistro von IFF). Zeichenketten werden hierbei nach komplizierten Algorithmen in Phonem- (Laut-) Folgen übersetzt. Dann wird die Ähnlichkeit der Lautfolgen ermittelt, die im Allgemeinen sehr viel aussagekräftiger ist als die der Zeichenfolgen.

Anhang

A1. Unabhängigkeitserklärung

Hiermit erkläre ich, dass ich diese Arbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe.

Hamburg, den 08.08.2002

A2. Inhalt der CD

In der Anlage zu dieser Diplomarbeit befindet sich eine CD. Sie enthält die Dateien des Prototyps und das vorliegende Dokument in PDF-Form.

Inhalt:

`DA_EAW0802.pdf`	PDF-Datei dieser Arbeit
`install.txt`	Installationsanleitung
`readme.txt`	Datei mit den letzten Änderungen
`sambar`	Verzeichnis des Webservers inkl.:
`bin`	Verzeichnis der Server-Binaries
`cgi-win`	cgi-Verzeichnis
`ffa`	Verzeichnis der c2a-Applikation
`c2a.dpr; shell_u.pas`	Projekt-Quellcode
`c2a.exe`	c2a-Applikation
`query`	Verzeichnis des Query-Formulars
`form.dpr; form_u.pas`	Projekt-Quellcode
`form.exe`	Form-Applikation
`config`	Verzeichnis der Konfigurationsdateien
`dbms`	Verzeichnis der Datenbanken
`ffa_ole97.mdb`	Datenbank mit Tabelle „houses“, den Parametern für die Fuzzy-Funktionen und dem VBA-Code
`docs`	Dokumenten-Verzeichnis des Servers
`index.html`	enthält Redirect auf form.exe
`log`	Verzeichnis der Log-Dateien
`search`	Verzeichnis der Such-Indizes des Servers (ohne Bedeutung)
`tmp`	Arbeitsverzeichnis von win-CGI
`debug`	Hilfsdateien der c2a-Applikation

A3. Literaturverzeichnis

[Bo97], Bosc P., Some Appproaches for Processing SQLf-nested Queries,To appear,Journal of Intelligent Systems

[Bo87], Bocklisch S.F, Prozeßanalyse mit unscharfen Verfahren, 1987, Berlin, VEB Verlag Technik

[Bo95], Bosc P, et al., Quantified Statements and Database Fuzzy Querying“ in Fuzzyness in Database Management Systems, 1995, Heidelberg, Physica Verlag, eds. P. Bosc and J. Kacprzyk

[BoPi93], Bosc P., Pivert O., On the Evaluation of Simple Fuzzy Relational Queries: Principles and Measures“ in Fuzzy-Logic: State of the Art, 1993, Boston, Kluwer Acadenic Publishers (eds. R. Lowen and M. Roubens)

[BoRu99], G. Booch, J. Rumbaugh, James, I. Jacobson, Das UML-Benutzerhandbuch, 1999, Bonn, Addison-Wesley, 3-8273-1486-0

[Ca01], Celio Carreto, Prototypische Realisierung einer interaktiven Benutzeroberfläche für heterogene Kataloge, Juli 2001, Technische Universität Hamburg-Harburg, Arbeitsbereich Softwaresysteme, Germany, Diplomarbeit

[DeDo90], Deyi, L.,Dongbo, L.,A Fuzzy Prolog Database System, 1990, Taunton, England,Research Studies Press Ltd.,0-86380-102-1

[Du91],Dutta, S. „Approximate Reasoning by Analogy to Answer Null Queries.“ International Journal of approximate Reasoning, Vol. 5 (1991), pp. 373-398.

[FaWi97], Fagin R., Wimmers E., Incorating User Preferences in Multimedia Queries, 1997, Proc. 6th International Conference on Database Theory, Springer-Verlag Lecture Notes i Computer Science 1186

[KaYa84a], Kacprzyk J., Yager, R. R., Linguistic Quantifier and belief Qualification in multi Criteria and multistage decision Making, 1984, Control and Cybernetics 13

[KaYa84b], Kacprzyk J., Yager, R. R., Softer Optimisation and Control Models via Fuzzy Linguistic Quantifiers, 1984, Information Sciences 34

[KaZa97], Kacprzyk J., Zadrozny S., Fuzzy Queries in Microsoft Access V.2“ in Fuzzy Information Engineering, 1997, New York, John Wiley & Sons, eds. Dubois D., Prade H., Yager R.

[KaZa97b]], Kacprzyk J., Zadrozny S., Multistage Fuzzy Control : A Model-Based Approach to Fuzzy Control and Decision Making, 1997, New York, John Wiley & Sons,

[KaZe84], Kandel A., Zermankova-Leech M., Fuzzy Relational Databases- A Key to Expert Systems, 1984, Verlag TÜV Rheinland,

[Ko89], Kohonen T., Self-Organization and Associative Memory (3rd ed.), 1989, Berlin - Heidelberg, Springer Verlag

[Ku93], Kurzweill, R., Das Zeitalter der künstlichen Intelligenz, 1993, München, Carl Hauser Verlag, 3-446-17375-7

[MaSt99], Florian Matthes, Ulrike Steffens, PIA - A Generic Model and System for Interactive Product and Service Catalogs, April 1999, Technische Universität Hamburg-Harburg, Germany,,Technical report, Arbeitsbereich Softwaresysteme, Accepted for Digital Libraries 99

[McFr94],McNeill, D.,Freiberger, P.,Die „unscharfe“
Logik erobert die Technik, 1994, München, Droemer Knaur, 3-426-26583-4

[St97], Ulrike Steffens, Wechselseitige Integration von Information Retrieval und Datenbanken, December 1997, Universität Hamburg, Germany, Diplomarbeit, Arbeitsbereich Informatik

[Wi60], Wittgenstein, L., Tractatus logico-philosophicus. Tagebücher 1914-1916, 1960, Frankfurt/Main

[Ya83], Yager, R. R.,,Robot Planning with Fuzzy Sets, 1983, Robotica 1

[Ya88], Yager, R. R., On ordered weighted average aggrgation operators in multi-media desision making, 1988, Man and Cybernetics, SMC-18, IEEE Transactions on Systems,

[Za65], Lotfi A. Zadeh:, Fuzzy Sets, Information and Control,1965
[Za83], A computational approach to fuzzy quantifier in natural languages, 1983, Computers and Mathematics with Applications
[Ch96], Davis Chapman, Building Internet Applikations with Delphi 2, 1996, Indianapolis, Que, 0-7897-0732-2
[Zh98], Lan Zhang., Objektorientierte Analyse und Entwurf eines Internet-Produkt-Informationssystems., Juli 1998, Universität Hamburg, Germany, Studienarbeit, Arbeitsbereich Informatik

A4. Abbildungsverzeichnis

A5. FfA97 Erweiterungen

Die in 4.4 FfA97 Erweiterungen angeführten, in dieser Arbeit nicht verwendeten, funktionalen Erweiterungen von FfA97 werden in der Online-Hilfe wie folgt in englischer Sprache beschrieben:

A2.1. Fuzzy set constants

Fuzzy set constant represents in a query the user's requirement as to the value of a single-valued attribute or a multi-valued attribute. Its use may be exemplified in the following atomic conditions:

1. 1.COUNTRY In 1.0/Bulgaria
2. COUNTRY In 1.0/Belarus + 1.0/Russia + 1.0/Ukraine
3. 3. COUNTRY In 1.0/CzechRepublic + 1.0/Hungary + 1.0/Poland + 1.0/Slovakia + 0.8/Belarus + ... + 0.8/Ukraine + ...

where COUNTRY is a field from a table.

The user, looking for a customer from Bulgaria only, will employ the first condition. If a few countries are relevant, the second condition may be relevant. Finally, if the choice of a customer's country of origin refers to a vague concept like, e.g., the Central Europe or the "developing countries", the third form should be employed.
The way a fuzzy set constant used along with a multi-valued attribute takes part in the calculation of matching degree of an atomic condition depends on the compatibility operator employed.

Parameters naming convention

1. [FfA_FS fuzzy set constant name]
   when used along with a single-valued attribute
   Example: COUNTRY = [FfA_FS Central Europe]
2. [FfA_FC compatibility operator name|FfA_FS fuzzy set constant name]
   when used along with a multi-valued attribute
   Example: PROFILE = [FfA_FC Jaccard|FfA_FS Exact sciences]

A2.2. Compatibility operator

Compatibility operators make it possible to express a relation that should be met by a single-valued attribute or a multi-valued attribute and a fuzzy set constant in an atomic condition. In the case of a single-valued attribute only one compatibility operator is applicable, namely the IN operator.
The matching degree of an atomic condition involving a single-valued attribute (at) and a fuzzy set (FS) is calculated as equal to , where is the value of the attribute at in a given record R.
In case of a multi-valued attribute, the following operators may be employed by the user:

```
NAME
QUERY PARAMETER

the degree of possibility of matching
[FfA_FC Possibility|FfA_FS fuzzy set name]
```

```
the degree of necessity of matching
[FfA_FC Necessity<|FfA_FS fuzzy set name]
[FfA_FC Necessity>|FfA_FS fuzzy set name]

the generalized Jaccard coefficient
[FfA_FC Jaccard|FfA_FS fuzzy set name]
```

Formally, a compatibility operator is a binary operation on fuzzy sets. Let FS (in a query) and D (in a database record) be two fuzzy sets defined in the same universe of discourse U, i.e., . The compatibility operators mentioned above may be defined as follows. Let md(FS,D) denote the matching degree of a simple condition referring to the compatibility operator to be defined. Then the following definitions are employed:

1. Degree of possibility of matching
   For the general case when both FS and D are fuzzy, we have
   while in case when both these sets are crisp, we obtain
   and, finally, if is a single-element crisp set, then
   This compatibility operator is proper in situations where just non-emptiness of the intersection of the value of an attribute and a corresponding pattern in a query is satisfactory. In the case of the "Main_products_purchased" attribute, it is adequate while looking for a customer who is mostly interested in at least one product mentioned in a query. A possible fuzziness of both the sets FS and D allows to account for different levels of relevance of particular products.
2. Degree of necessity of matching
   For the general case of fuzzy FS and D, we have
   while in case when both these sets are crisp, we obtain
   and, finally, if is a single element crisp set, then
   Obviously this operator is not symmetrical, hence we can define two different operators corresponding to md(FS,D) and md(D,FS), respectively.
   These compatibility operators are proper when (strong) inclusion of the set representing the value of an attribute in the set corresponding to a pattern given in a query, or vice versa, is required. In a customers database example, when using a compatibility operator along with the "Main_products_purchased" attribute, the first operator is adequate while looking for a customer particularly interested in all products specified in a query (and maybe even some more). On the other hand, the second operator of this pair is adequate for the search of a customer particularly interested only in products specified in the query, not necessarily all.
3. Generalized Jaccard coefficient
   For the general case of fuzzy and/or crisp FS and D, we have
   and this is one of most popular operators used in the classical, crisp framework.
   Again, in the case of the Main_products_purchased attribute, the Jaccard operator is proper while looking for a customer particularly interested in products specified in a query and not many other ones.
   Parameters naming convention
   [FfA_FC compatibility operator name|FfA_FS fuzzy set constant name]
   Example: [FfA_FM Profile] = [FfA_FC Jaccard|FfA_FS Exact sciences]

A2.3. Single- and multi-valued attributes

Both types of attributes may be used along with a fuzzy set constant in a query.
A single-valued attribute may be considered as a special case of a multi-valued attribute. Namely, only a special case of the compatibility operators (IN) is meaningful in the case of the former one, while various types of compatibility operators may be employed in the case of the latter. Both types of attributes may be exemplified by Country and Main_products_purchased

fields in a fictitious database of a trading company, respectively. In case of the former one, a fuzzy set may be used in a corresponding atomic condition of a query as, e.g., in
Find customers from Central Europe
what may be written using FQUERYs for Access extended SQL syntax as follows:
COUNTRY = [FfA_FS Central Europe]
assuming a fuzzy set constant Central Europe is defined by the user.
In case of the latter one a fuzzy set may be used in a query as well as in a record as a value of the attribute. The value of such an attribute will be a list of relevant products. The direct use of such types of data is inconsistent with the relational database paradigm. Still, such an attribute may exist in the user's view of the database, even if the real arrangement of the data is different. In the original database scheme, a list of fields corresponds to such an virtual multi-valued attribute. Each of this fields is of logical or real type, corresponding to the characteristic function of a crisp set or the membership function of a fuzzy set, respectively.
The matching degree for an atomic condition involving a single-valued attribute (AT) and a fuzzy set (FS) is calculated for each record in a straightforward manner as the value of the membership function of the fuzzy set FS for the element being the value of the attribute AT in a given record.
The calculation of matching degree for multi-valued attributes is discussed here.
Parameters naming convention
Single-valued attribute is referred directly using the name of a field in a table.
For multi-valued attribute use:
[FfA_FM multi-valued attribute name]
Example: [FfA_FM Profile] = [FfA_FC Jaccard|FfA_FS Exact sciences]