# Lernen nutzerspezifischer Gewichte innerhalb einer logikbasierten Anfragesprache

Ingo Schmitt, David Zellhöfer
Brandenburgische Technische Universität Cottbus
Institut für Informatik, Postfach 10 13 44
D-03013 Cottbus, Germany
{schmitt|david.zellhoefer}@tu-cottbus.de

**Abstract:** Unpräzise Anfragen, wie solche nach Ähnlichkeit, stellen traditionelle Datenbanken vor zwei Herausforderungen: die Integration unpräziser Prädikate und die Gewichtung dieser innerhalb ihrer Anfragelogik. Letzteres wird besonders durch das subjektive Ähnlichkeitsempfinden des Nutzers notwendig. Die vorliegende Arbeit stellt ein theoretisches Framework vor, welches Termgewichte in eine quantenlogikbasierte Anfragesprache, die den Gesetzen einer Booleschen Algebra gehorcht, integriert.

Zur Formulierung von Gewichtungen wird nutzerseitig auf das Konzept der Präferenz zurückgegriffen. Intern jedoch werden numerische Gewichtswerte verwendet, um eine definierte Reihenenfolge von Ergebnisobjekten zu erzeugen. Im Rahmen eines Relevance-Feedback-Prozesses finden Verfahren des maschinellen Lernes Einsatz, um den Nutzer bei der Gewichtssetzung entsprechend seiner Anfrageintention zu unterstützen. Experimentelle Ergebnisse unterstreichen die Effektivität des Ansatzes und zeigen deutlich, dass maschinenbasierte Lernverfahren im Rahmen des Relevance-Feedbacks dem Nutzer bei der Recherche helfen.

## 1 Einleitung

Bezüglich einer Anfrage entscheiden traditionelle, auf Boolescher Logik basierende Datenbanksysteme pro Datenbankobjekt, ob es zum Ergebnis gehört oder nicht. Diese binäre Entscheidung genügt allerdings häufig nicht den Anforderungen seitens des Nutzers. Am Beispiel der Suche nach einer komfortablen, günstigen Kamera wird deutlich, dass sich die Kategorien „komfortabel“ und „günstig“ durch ihre Vagheit kaum durch Boolesche Werte abbilden lassen. Vielmehr sucht der Nutzer hier nach Produkten, die seinen Vorstellungen hinreichend gut entsprechen, so dass sich eine spezifische Reihenfolge dieser ergibt[1]. Weitere Beispiele für derartige Anfragen finden sich z.B. in [ACDG03].

Die Integration von vagen oder unpräzisen Prädikaten in logikbasierte Anfragesprachen stellt ein aktuelles Problem der Forschung dar [CW08, Wei07, LSDJ06, RJ05, CRW05]. Klassische Ansätze bedienen sich zumeist der Fuzzy-Logik [SS04, CMPT00, BP95, Zad88]. Der in dieser Arbeit vorgestellte CQQL-Ansatz basiert hingegen auf Gesetzen der Quantenlogik [Sch08, BN36]. Diese werden verwendet, um verschiedene Anfrageparadigmen,

[1] Hierbei kann es sich z.B. um ein Ranking anhand der Ähnlichkeit der Objekte bzgl. der Nutzeranfrage handeln.

wie das im Information-Retrieval(IR) häufig verwendete Vektorraummodell mit der Booleschen Logik, zu kombinieren.

Die Einbindung unpräziser Prädikate in Anfragen führt zu einem weiteren Problem: der Gewichtung. Untersuchungen im Information-Retrieval belegen, dass eine vom Anwender beeinflusste Gewichtung einzelner Terme die Nutzerzufriedenheit mit dem Anfrageergebnis erhöht [SFW83]. Psychologische Experimente stützen diese Ergebnisse und stellen das subjektive Empfinden von Ähnlichkeit heraus [BG93, Sel59]. Folglich ermöglicht erst die Angabe von Termgewichten die Anpassung einer Anfrage an die subjektive Ähnlichkeitswahrnehmung, der Präferenz, des Nutzers.

Die Möglichkeit der Präferenzangabe unterscheidet sich in zwei alternative Klassen: die qualitativen Ansätze [Cho03, Kie02] und die quantitativen, zu denen der in der Arbeit vorgestellte Ansatz gehört. Qualitative Ansätze basieren auf der Angabe von Halbordnungen, in welchen der Nutzer seine Präferenzen bezüglich einer Datenbank formuliert. Hierbei muss angemerkt werden, dass dies voraussetzt, dass der Nutzer bereits zum Zeitpunkt der Anfrageformulierung die Präferenzen kennt[2]. Der Skyline-Operator [KRR02, BKS01] kann ebenfalls dieser Gruppe zugeordnet werden. In Abgrenzung dazu wird beim quantitativen Ansatz eine totale Ordnung der Ergebnisobjekte auf der Grundlage von numerischen Score-Werten erzeugt, wie dies im Information-Retrieval üblich ist. Unterstützer der qualitativen Ansätze betrachten deshalb quantitative Verfahren als Spezialfall der qualitativen Verfahren[3]. Bei der Anfrageformulierung im vorgestellten, quantitativen Ansatz müssen Gewichte numerisch angegeben werden, was gerade bei komplexen Anfragen eine Hürde für den Nutzer darstellt. Durch geeignete Verfahren des maschinellen Lernens kann die Angabe konkreter Gewichte in unserem Ansatz jedoch verborgen werden und stellt so keine zusätzliche Beeinträchtigung der Nutzerinteraktion dar. Beide Klassen von Ansätzen haben jeweils ihre Berechtigung in unterschiedlichen Einsatzszenarien. Folgendes Beispiel soll den Vorteil des quantitativen Ansatzes in einem speziellen Szenario verdeutlichen. Kamera $A$ sei in einer Bewertungskategorie wesentlich besser als Kamera $B$. Gleichzeitig sei Kamera $B$ nur geringfügig besser als $A$ in einer anderen Kategorie. Qualitative Ansätze ermöglichen keine Unterscheidung zwischen „wesentlich“ und „geringfügig besser“, so dass die Ergebnismenge weniger ausdifferenziert wird.

Zur Einbindung von Gewichten in Scoring-Verfahren existiert u.a. ein Vorschlag von Fagin und Wimmers [FW00], dessen arithmetische Berechnung jedoch nicht in eine Anfragelogik eingebettet ist. Der vorliegende Gewichtungsansatz kann als dessen Verallgemeinerung gesehen werden, ist allerdings mächtiger, da er komplett in die Logik integriert ist und den Gesetzen einer booleschen Algebra gehorcht. Dies ermöglicht die weitergehende Nutzung logikbasierter Optimierungstechniken innerhalb von DBS.

Wie bereits vorgestellt, stellt das Setzen von konkreten Gewichten ein wesentliches Problem während der Nutzerinteraktion dar. Folgendes Anwendungsszenario soll dies verdeutlichen.

---

[2] Dass diese Präferenzen in jedem Fall angeben werden können, kann bezweifelt werden. So ist es z.B. beim Image-Retrieval kaum denkbar, dass ein Laie konkret angeben kann, wie wichtig ihm einzelne Feature-Werte sind.

[3] Da jede Totalordnung eine Halbordnung ist. Dem kann eingewandt werden, dass andererseits jede Halbordnung mit der Dushnik-Miller-Dimension $d$ durch den Schnitt von $d$ Totalordnungen ausgedrückt werden kann.

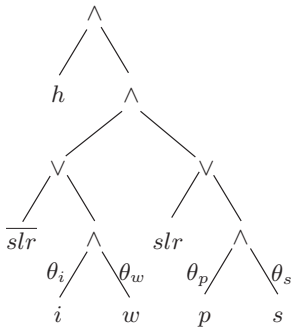



Abbildung 1: Gewichtete Anfrage

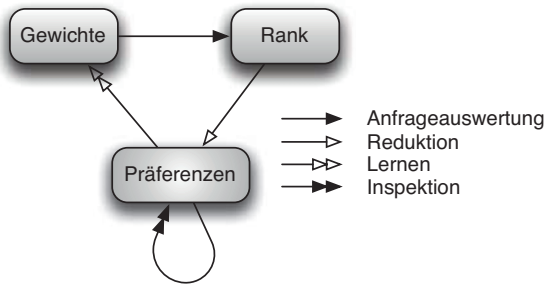



Abbildung 2: Zustandsdiagramm der Datenstrukturen

**Beispiel 1.1** *Ein Anwender sucht nach einer passenden Kamera. Diese soll gut bedienbar sein ($h$). Außerdem fordert er bei einer Spiegelreflexkamera (SLR) gute Bildqualität ($i$) und geringes Gewicht ($w$). Findet sich keine SLR, dann muss das Gerät preisgünstig ($p$) sein und eine geringe Auslöseverzögerung ($s$) bieten. Der relative Einfluss der Bedingungen $i, w, p$ und $s$ wird durch entsprechende Gewichtsvariablen ausgedrückt. Abb. 1 (links) zeigt den entsprechenden Bedingungsbaum.*

Des Weiteren wird angenommen, dass es sich bei $q^\Theta$ um eine gewichtete CQQL-Anfrage mit den Gewichtsvariablen $\Theta = \{\theta_i\}$ handelt. Die Anfragesprache CQQL wird in [Sch08] vorgestellt und in Abschnitt 2 kurz beschrieben. Sie erweitert den relationalen Bereichskalkül um die Behandlung von unscharfen Bedingungen und gewichteten Operanden von Konjunktionen und Disjunktionen. Eine Funktion $w$ ordnet in CQQL jeder Gewichtsvariable $\theta_i \in \Theta$ einen Wert aus dem Intervall $[0, 1]$ zu. Die initiale Gewichtungsfunktion $w^I$ setzt in diesem Fall alle Variablen auf den konstanten Wert 1.

Die Anfrage wird auf einer Menge von Anfrageobjekten $O$ bestehend aus $n$ Elementen ausgeführt. Die Evaluierung von $q^\Theta$ geschieht durch die Funktion $eval(q^\Theta, o, w)$, welche für jedes Objekt $o \in O$ einen Score aus $[0, 1]$ berechnet[4]. Die Ausführung von $eval(q^\Theta, o, w^I)$ über alle $o$ ergibt einen Rank (Permutation) $p(w^I) : \{1, \ldots, n\} \rightarrow O$, welche eine totale Ordnung der Objekte bezüglich ihrer Scores gegenüber der Anfrage ergibt. Des Weiteren wird angenommen, dass es eine Ziel-Gewichtungsfunktion $w^t$ für $q^\Theta$ gibt, welche einen Rank erzeugt, der genau dem Ähnlichkeitsempfinden des Nutzers entspricht.

Zum Zeitpunkt der initialen Anfrageformulierung ist $w^t$ in der Regel unbekannt. Dies führt dazu, dass die initiale Gewichtung $w^I$ häufig zu einem Rank führt, der dem subjektivem Empfinden des Nutzers widerspricht. Ziel ist es deshalb, sich dem Ziel-Rank, welcher durch $w^t$ erzeugt wird, sukzessive anzunähern (dies entspricht den im Information-Retrieval gebräuchlichen Relevance-Feedback [SB88, Roc71]). Hierbei ist festzuhalten, dass unterschiedliche Gewichte einen starken Einfluss auf das Ergebnis einer Anfrage haben können.

[4]Der Wert 1 gibt die maximale und der Wert 0 die minimale Erfüllung der Bedingung an.

Da das direkte Setzen der Gewichte, gerade bei komplexeren Anfragen, eine erhebliche Hürde für den Anwender darstellt, muss zur Verbesserung der Nutzerinteraktion eine Alternative bereitgestellt werden. Der vorliegende Ansatz greift hierbei auf das intuitiv verständliche Konzept der Präferenz zurück: Während der Interaktion mit dem System wird es dem Anwender mittels Präferenzen ermöglicht, die Reihenfolge der gerankten Ergebnisobjekte zu verändern. Die Angabe einer Präferenz geschieht dabei paarweise und bedeutet, dass ein konkretes Objekt $A$ vor einem Objekt $B$ im neuen Rank erscheinen soll. Anhand dieser Präferenzen lernt das vorgeschlagene System Gewichtswerte. Durch die wiederholte Erzeugung neuer Ranks anhand neu gelernter Gewichtswerte kann gezeigt werden, dass sich die Ergebnisobjekte immer mehr denen des Ziel-Ranks annähern.

Als Folgeproblem ergibt sich die Fragestellung nach der Präsentation der aktuellen Gewichte für den Nutzer. Wir schlagen hier die automatische Ableitung (Reduktion, siehe Abb. 2) von Präferenzen aus den ersten $k$ Objekten des erzeugten Rank vor. Diese müssen nach Rückumwandlung in Gewichte denselben Top-$k$-Rank erzeugen wie die ursprünglichen Gewichte. Diese abgeleiteten Präferenzen unterscheiden sich in der Regel von den vom Anwender bestätigten Präferenzen, da diese meist eine Änderung der ersten $k$ Objekte im neuen Rank bewirken: Objekte verschwinden aus dem Top-$k$-Rank und werden durch neue ersetzt. Bezüglich der Generierung des Top-$k$-Ranks sind beide Präferenzmengen jedoch äquivalent. Die abgeleiteten Präferenzen können nun in einer erneuten Nutzerinteraktion bei Bedarf modifiziert werden. Können alle abgeleiteten Präferenzen bestätigt werden, terminiert der Suchprozess. Voraussetzung einer Terminierung ist dabei die Konsistenz der spezifizierten Präferenzen und die Existenz einer Ziel-Gewichtungsfunktion $w^t$.

Hierbei ist es wichtig anzumerken, dass der Nutzer ausschließlich mit Präferenzen arbeitet und die zu Grunde liegenden Gewichtswerte nicht bemerkt.

### 1.1 Aufbau der Arbeit

Die Arbeit gliedert sich wie folgt. Kapitel 2 umreißt die Anfragesprache CQQL und grenzt diese gegenüber der Fuzzy-Logik ab. Das folgende Kapitel beschreibt die Integration von Gewichten in CQQL. Kapitel 4 erläutert das Prinzip der Präferenz und deren Bedeutung für die zu erlernenden Gewichte. Darauf folgt eine detaillierte Beschreibung der Umwandlung zwischen Präferenzen und Gewichten sowie deren Einbettung in den iterativen Relevance-Feedback-Prozess, welcher in Kapitel 6 diskutiert wird. Den Abschluss der Arbeit bilden Ergebnisse aus Experimenten und deren Bewertungen gefolgt von einem kurzen Ausblick auf kommende Arbeiten.

## 2 CQQL-Bedingungen und ihre Auswertung

CQQL steht für *commuting quantum query language* und basiert auf Ergebnissen der Quantenlogik [Zie05, BN36]. Die Anfragesprache erlaubt die Formulierung von logik-

basierten Anfragen, welche Boolesche und Score-basierte Bedingungen, die im IR üblich sind, vereinen. Score-basierte Bedingungen testen etwa, wie nah ein Attributwert einem Zielwert ist und geben jeweils einen Wert aus dem Intervall $[0, 1]$ zurück. Eine umfassende Darstellung der theoretischen Grundlagen von CQQL findet sich in [Sch08]. Syntaktisch lehnt sich CQQL an den relationalen Bereichskalkül an.

Beispiel 1.1 erfordert eine Anfrage mit Booleschen und Score-basierten Bedingungen auf der Relation

$$cameras(name, handling, SLR, image\_quality, weight, price, shutter\_lag)).$$

Die Bedingung *slr* ist eine klassische DB-Bedingung während die übrigen Score-basiert sind. Die Anfrage wird wie folgt formuliert:

$$\{name | \exists h, slr, i, w, p, s : cameras(name, h, slr, i, w, p, s) \wedge h \approx good \wedge \\ (\neg slr \vee (i \approx high \wedge_{\theta_i, \theta_w} w \approx low)) \wedge (slr \vee (p \approx low \wedge_{\theta_p, \theta_s} s \approx low)\}$$

Wie diese Anfrage beispielhaft zeigt, erweitert CQQL den relationalen Bereichskalkül um die Behandlung von Score-basierten Bedingungen, etwa $i \approx high$, und gewichteten Konjunktionen, etwa $\wedge_{\theta_p, \theta_s}$.

Aufgrund der Score-Werte kann die klassische Boolesche Logik in solch einem Szenario nicht verwendet werden. Auf den ersten Blick bietet sich der Rückgriff auf Fuzzy-Logik [SS04, CMPT00, BP95, Zad88] an. Fuzzy-Logik verfügt jedoch über Nachteile bei der Auswertung von Bedingungen, da sich die Ergebnisse dieser nicht immer mit der Erwartung seitens der Nutzer deckt [SZN08, KN07, Lee94]. Dies bezieht sich insbesondere auf das Verhalten der einzigen distributiven und idempotenten T-Norm (Konjunktion), der *min*-Funktion, deren Ergebnis immer genau einem Eingabeparameter entspricht, währende der andere ignoriert wird (Dominanzproblem). Weiterhin werden Gesetze der Booleschen Algebra, z.B. '$x \wedge \neg x =$ falsch' verletzt: $\min(0.5, 1 - 0.5) \neq 0$. Alternative T-Normen und T-Konormen ohne Dominanzproblem verletzen andererseits Idempotenz und Distributivität.

Die Semantik unseres Sprachvorschlags CQQL baut hingegen auf der Quantenlogik auf. Dies bedeutet, dass Konjunktion, Disjunktion und Negation folgend den Gesetzen der Quantenlogik ausgewertet werden. Konzeptionell wird dabei jeder Tupel als ein normierter Vektor in einem geeignet gewählten Hilbertraum modelliert, während eine Bedingung einem Vektorunterraum entspricht. Alle Vektorunterräume bilden zusammen mit der Halbordnung $\subseteq$ einen orthomodularen Verband. Dieser Verband liefert uns die Konjunktion, Disjunktion und die Negation. Basieren alle Vektorunterräume auf einer gemeinsamen Orthonormalbasis (die Projektoren der Vektorunterräume *kommutieren* also paarweise: $p_i p_j = p_j p_i$), dann erfüllt der entstehende Unterverband alle Gesetze der Booleschen Algebra. Bei der Auswertung einer Bedingung gegenüber einem Tupel wird das quadrierte innere Produkt zwischen dem Vektor und dem Vektorunterraum gemessen.

Sowohl Score-basierte als auch klassische Datenbankbedingungen lassen sich in diesem Modell nachbilden. Es kann sogar gezeigt werden, dass CQQL eine Verallgemeinerung des relationalen Bereichskalküls darstellt sowie den Gesetzen einer Booleschen Algebra genügt [Sch08].

Bevor eine CQQL-Bedingung algorithmisch ausgewertet werden kann, muss diese in eine spezielle syntaktische Form *normalisiert* werden. Der Normalisierungsalgorithmus basiert auf Booleschen Transformationsregeln und wird detailliert in [Sch08] diskutiert. Normalisierungsschritte sind Überführung in die Prenex-Normalform sowie die Anwendung des Distributivgesetzes, um gemeinsame Bedingungen in den Operanden einer Konjunktion oder Disjunktion zu isolieren: $(\varphi \wedge \varphi_1) \vee (\varphi \wedge \varphi_2) \Rightarrow \varphi \wedge (\varphi_1 \vee \varphi_2)$.

Aus den Gesetzen der Quantenlogik und Quantenmechanik lassen sich für jede normalisierte Konjunktion und Disjunktion eine Auswertung mit einfachen arithmetischen Operationen ableiten. Seien $eval(\varphi, o) \in [0; 1]$ die Auswertung der Bedingung $\varphi$ von Objekt $o$ und $\varphi_1 \wedge \varphi_2, \varphi_1 \vee \varphi_2, \neg\varphi$ *normalisierte*[5] Bedingungen, dann wird die CQQL-Evaluierung durch rekursive Anwendung der folgenden Vorschriften erreicht:

$$
\begin{array}{ll}
eval(\varphi_1 \wedge \varphi_2, o) = & eval(\varphi_1, o) * eval(\varphi_2, o) \\
eval(\varphi_1 \vee \varphi_2, o) = & eval(\varphi_1, o) + eval(\varphi_2, o) - eval(\varphi_1, o) * eval(\varphi_2, o) \\
 & \text{wenn } \varphi_1 \text{ und } \varphi_2 \text{ nicht exklusiv} \\
eval(\varphi_1 \vee \varphi_2, o) = & eval(\varphi_1, o) + eval(\varphi_2, o) \\
 & \text{wenn } \varphi_1 \text{ and } \varphi_2 \text{ exklusiv} \\
eval(\neg\varphi, o) = & 1 - eval(\varphi, o).
\end{array}
$$

Eine Disjunktion ist *exklusiv*, wenn sie die Form $(\varphi \wedge \ldots) \vee (\neg\varphi \wedge \ldots)$ hat. Jede Evaluierung einer komplexen Anfragebedingung kann durch eine Summe von Produkten (analog zur DNF) atomarer Bedingungsauswertungen auf Objektattributen ausgedrückt werden.

## 3 Gewichtung von Anfragebedingungen

Die Integration von Gewichten in CQQL gestaltet sich einfach. Die Operanden von Konjunktion und Disjunktionen einer Bedingung können mit Gewichtsvariablen $\theta_i$ ausgestattet werden. Vor der Auswertung der Bedingung werden die Gewichtsvariablen mittels der Gewichtungsfunktion $w$ auf Werte aus dem Intervall $[0, 1]$ abgebildet. Die Werte von Gewichtsvariable $\theta_i$ kontrollieren den Einfluss der Bedingung $\varphi_i$ in der gewichteten Konjunktion $\varphi_1 \wedge_{\theta_1,\theta_2} \varphi_2$ und der Disjunktion $\varphi_1 \wedge_{\theta_1,\theta_2} \varphi_2$.

Die Kernidee des vorgestellten Ansatzes besteht in der Anwendung einer syntaktischen Ersetzungsregel. Diese Transformation wandelt eine gewichtete Konjunktion oder Disjunktion in einen logischen Ausdruck ohne Gewichte um, wobei Gewichtswerte in konstante Score-Werte überführt werden:

$\varphi_1 \wedge_{\theta_1,\theta_2} \varphi_2 := (\varphi_1 \vee \neg\theta_1) \wedge (\varphi_2 \vee \neg\theta_2)$ und $\varphi_1 \vee_{\theta_1,\theta_2} \varphi_2 := (\varphi_1 \wedge \theta_1) \vee (\varphi_2 \wedge \theta_2)$

Abb. 3 illustriert den Transformationsschritt für Beispiel 1.1. Als Anfrage $q^{\Theta}$ erhalten wir

$$
\begin{array}{lll}
\{name & | & \exists h, slr, i, w, p, s : cameras(name, h, slr, i, w, p, s) \wedge h \approx good \wedge \\
 & & (\neg slr \vee ((i \approx high \vee \neg\theta_i) \wedge (w \approx low \vee \neg\theta_w))) \wedge \\
 & & (slr \vee ((p \approx low \vee \neg\theta_p) \wedge (s \approx low \vee \neg\theta_s)))\}.
\end{array}
$$

[5] Aufgrund der Forderung nach Normalisierung kann die Bedingung $x \wedge \neg x = \text{falsch}$ in CQQL gar nicht *direkt* ausgewertet werden. Tatsächlich wird sie innerhalb der Normalisierung aufgelöst.

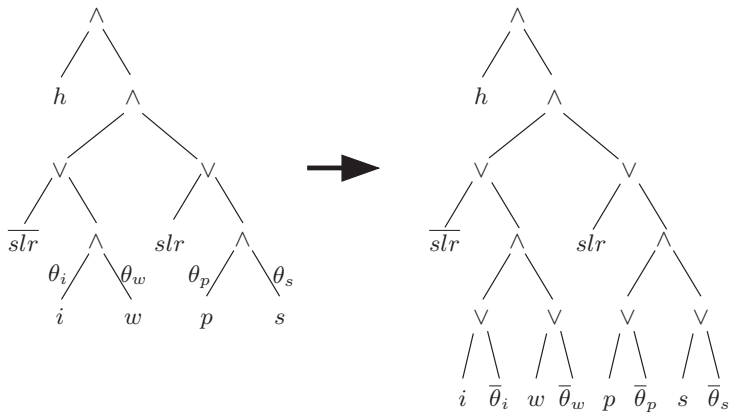



Abbildung 3: Transformation der Beispielanfrage in ihr logisches Äquivalent

Die arithmetische Auswertung der Anfrage nach der Normalisierung und der Anwendung der Regeln aus dem vorherigen Abschnitt ergibt

$$\begin{aligned} eval(q^{\Theta}, o) &= h(slr * iw + \overline{slr} * ps) \text{ mit} \\ iw &= (i + \overline{\theta}_i - i\overline{\theta}_i)(w + \overline{\theta}_w - w\overline{\theta}_w) \\ ps &= (p + \overline{\theta}_p - p\overline{\theta}_p)(s + \overline{\theta}_s - s\overline{\theta}_s), \end{aligned}$$

wobei $\overline{\theta}_x = 1 - \theta_x$ gilt, und die Werte $h, i, w, p, s$ die Score-Werte von Objekt $o$ bzgl. der atomaren Bedingungen sind.

Tabelle 1 gibt beispielhaft die Score-Werte für die atomaren Beispielbedingungen für vier Objekte an. In Abhängigkeit von den Gewichtswerten werden, wie in Tabelle 2 dargestellt, unterschiedliche Gesamt-Score-Werte für die vier Objekte entsprechend der Anfrageauswertung berechnet. Dabei zeigt sich, dass Objekt $o_2$ immer schlechter als $o_1$ bewertet wird. Dies ist nicht verwunderlich, da alle atomaren Score-Werte von $o_2$ nicht besser als die von $o_1$ sind. Durch keine Wertebelegung der Gewichte lässt sich daher erreichen, dass $o_2$ besser als $o_1$ bewertet wird.

Objekt $o_3$ wird bei den ersten Gewichtswerten schlechter als Objekt $o_4$ ausgewertet. Dies lässt sich umkehren, wenn die Bedingung $p$ durch ein Gewicht ausgeschaltet wird.

Insgesamt werden die Spiegelreflexkameras $o_1$ und $o_2$ schlechter als die einfachen Kameras $o_3$ und $o_4$ ausgewertet. Möchte man dieses Ergebnis verändern, so ist es nur nötig, die

| $o_i$ | $slr$ | $h$ | $i$ | $p$ | $s$ | $w$ |
|---|---|---|---|---|---|---|
| $o_1$ | 1 | 0,8 | 0,7 | 0,6 | 0,8 | 0,3 |
| $o_2$ | 1 | 0,4 | 0,5 | 0,5 | 0,6 | 0,2 |
| $o_3$ | 0 | 0,8 | 0,3 | 0,4 | 0,7 | 0,5 |
| $o_4$ | 0 | 0,9 | 0,6 | 0,8 | 0,6 | 0,6 |

Tabelle 1: Beispielhafte Score-Werte für atomare Bedingungen

| $\theta_i$ | $\theta_w$ | $\theta_p$ | $\theta_s$ | $o_1$ | $o_2$ | $o_3$ | $o_4$ |
|---|---|---|---|---|---|---|---|
| 1,0 | 1,0 | 1,0 | 1,0 | 0,168 | 0,04 | 0,224 | 0,432 |
| 1,0 | 1,0 | 0,0 | 1,0 | 0,168 | 0,04 | 0,56 | 0,54 |
| 0,5 | 0,5 | 1,0 | 1,0 | 0,442 | 0,18 | 0,224 | 0,432 |

Tabelle 2: Gesamt-Score-Werte in Abhängigkeit von Gewichten

| Name | Formel |
|---|---|
| Gew. Summe (gs) | $\theta_1 * x + \theta_2 * y$ |
| Gew. Disj. (gd) | $(x \wedge \theta_1) \vee (y \wedge \theta_2)$ |
| Gew. Konj. (gk) | $\left(x \vee \overline{\theta}_1\right) \wedge \left(y \vee \overline{\theta}_2\right)$ |

| Name | Arithm. Auswertung |
|---|---|
| Gew. Disj. (gd) | $\theta_1 * x + \theta_2 * y -$ $\theta_1 * x * \theta_2 * y$ |
| Gew. Konj. (gk) | $(\theta_1 + x - \theta_1 * x) *$ $(\theta_2 + y - \theta_2 * y)$ |

| x | y | $\theta_1$ | $\theta_2$ | gs | gd | gk |
|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 | 0 | 0 | 0 |
| 0 | 1 | 0 | 0 | 0 | 0 | 1 |
| 0 | 1 | 0 | 1 | 1 | 1 | 1 |
| 0 | 1 | 1 | 0 | 0 | 0 | 0 |
| 0 | 1 | 1 | 1 | 1 | 1 | 0 |
| 1 | 0 | 0 | 0 | 0 | 0 | 1 |
| 1 | 0 | 0 | 1 | 0 | 0 | 0 |
| 1 | 0 | 1 | 0 | 1 | 1 | 1 |
| 1 | 0 | 1 | 1 | 1 | 1 | 0 |
| 1 | 1 | 0 | 0 | 0 | 0 | 1 |
| 1 | 1 | 0 | 1 | 1 | 1 | 1 |
| 1 | 1 | 1 | 0 | 1 | 1 | 1 |
| 1 | 1 | 1 | 1 | 2 | 1 | 1 |

Abbildung 4: Vergleich verschiedener Gewichtungsformeln

Gewichtswerte für die Spiegelreflexbedingungen $i$ und $w$ gemeinsam herunterzusetzen. Dies bewirkt ein Anstieg der SLR-Kameras, da die Bedingungen $i$ und $w$ in der Konjunktion ihren Einfluss auf die Gesamt-Score-Werte verlieren.

Die Betrachtung der folgenden Extremfälle erleichtert das Verständnis der Ersetzungsregel. Ein *Nullgewicht ($\theta_i = 0$)* führt dazu, dass die mit Null gewichtete Bedingung keinen Einfluss auf das Gesamtergebnis hat. *Gleiche Gewichtswerte ($\theta_1 = \theta_2 = 1$)* hingegen führen zu einem Gesamtergebnis, welches sich so verhält, als ob keine Gewichte vorhanden wären.

Fagin und Wimmers fordern in [FW00] die Linearität der Gewichtung. Aufgrund der Linearität der CQQL-Auswertungen (Summe von Produkten) bewegt sich die Evaluierung bezogen auf eine Gewichtsvariable zwischen diesen beiden Fällen in CQQL ebenfalls linear. Abb. 4 stellt die gewichtete Konjunktion und Disjunktion der einfachen, gewichteten Summe, als Beispiel einer einfachen Heuristik, gegenüber und demonstriert das Verhalten der Gewichtung auf Booleschen Werten[6]. Damit soll auch gezeigt werden, dass unsere Gewichtung mittels einer syntaktischen Ersetzungsregel nicht auf die Sprache CQQL beschränkt ist, sondern etwa auch innerhalb der Booleschen Logik funktioniert.

Bereits existierende Gewichtungsansätze, wie z.B. von Fagin und Wimmers [FW00], führen arithmetische Auswertungen oberhalb einer logischen Bedingung aus. Bezüglich der Optimierung innerhalb eines DBS ist dieses Vorgehen nicht zu empfehlen, da so eine logische Optimierung erschwert bzw. unmöglich wird. Beispielsweise kann die Verletzung der Assoziativität und Distributivität gezeigt werden [SS03]. Fagin und Wimmers schränken die Freiheit der Gewichtung zudem gegenüber unserem Verfahren ein, da sie voraussetzen, dass für alle Gewichte $\theta_i \in [0, 1]$ einer gewichteten Konjunktion oder Disjunktion immer $\sum_i \theta_i = 1$ gilt.

Die innovative Idee unsers Gewichtungsansatzes besteht darin, dass sich eine Gewichtung *ausschließlich* mit den Mitteln einer Logik realisieren lässt.

[6] 0 für 'falsch' und 1 für 'wahr'

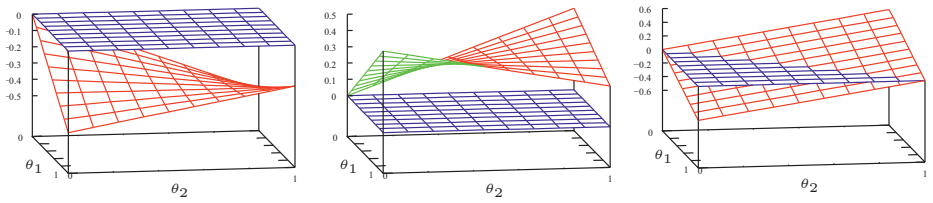



Abbildung 5: Fälle der Nützlichkeit der Präferenz $o_1 \geq o_2$ bezogen auf die 0-Hyperebene

# 4 Präferenzen

Die Angabe von Präferenzen, also ob ein Objekt eine Anfrage besser als ein anderes erfüllt, ist in vielen Fällen einfacher für den Nutzer als das explizite Setzen von Gewichten, um das subjektive Empfinden von Ähnlichkeit bezüglich einer Anfrage auszudrücken. Da binäre Präferenzen zwischen zwei Objekten intuitiv verständlich sind, greift der vorgestellte Ansatz auf diese im Rahmen der Benutzerschnittstelle zurück. Im Gegensatz zu anderen Verfahren [Cho07, Kie02] verbergen Präferenzen in dieser Arbeit konkrete Gewichte, welche intern Ranks erzeugen (siehe Abb. 2). Hierdurch wird eine feinere Ausdifferenzierung der Ergebnisobjekte in Form einer totalen Ordnung erzielt (siehe Kap. 1).

Präferenzen bezüglich einer gewichteten Anfrage $q^{\Theta}$ werden als $o_1 \geq o_2$ angegeben, wobei $o_1, o_2 \in O$ gilt. Eine Gewichtungsfunktion $w$ erfüllt eine Präferenz, wenn Folgendes gilt: $eval(q^{\Theta}, o_1, w) - eval(q^{\Theta}, o_2, w) \geq 0$. Eine Menge von definierten Präferenzen $P$ für $q^{\Theta}$ heißt *konsistent*, wenn ihr transitiver und reflexiver Abschluss eine Halbordnung darstellt. D.h., dass Zyklen wie $o_1 \geq o_2$ und $o_2 \geq o_1$ mit $o_1 \neq o_2$ nicht erlaubt sind. Im Folgenden wird angenommen, das $P$ konsistent ist und bereits bezüglich Reflexivität und Transitivität reduziert ist.

Abb. 5 illustriert das Konzept der Präferenz bezogen auf den (Hyper-)Einheitswürfel, der durch zwei Gewichte, also die beiden Achsen $\theta_1$ und $\theta_2$, aufgespannt wird. Die Präferenz bezüglich einer Anfrage kann dabei als Hyperebene über dem Einheitswürfel (im vorliegenden Fall über der Einheitesebene) gesehen werden. Je nach Gewichtungswerten und maximalem oder minimalem Funktionswert $\max$ bzw. $\min$ fällt jede Präferenz dabei in eine der folgenden Kategorien:

*Nicht erfüllbar*: Eine Präferenz ist nicht erfüllbar, wenn $max < 0$ gilt. D.h. die Präferenz kann durch keine Gewichtungwerte von $q^{\Theta}$ erfüllt werden (siehe Abb. 5 (links)).

*Nutzlos*: Eine Präferenz ist zum Lernen von Gewichten nutzlos, wenn $min \geq 0$ gilt. In diesem Fall stellt die Präferenz keine Einschränkung für die Gewichtswerte von $q^{\Theta}$ dar (siehe Abb. 5 (Mitte)).

*Nützlich*: In den übrigen Fällen ist die Präferenz nützlich (siehe Abb. 5 (rechts)).

Ein Beispiel für eine nicht erfüllbare Präferenz ist $o_2 \geq o_1$, wobei $o_1$ und $o_2$ die Objekte aus Tabelle 1 darstellen. Wenn man diese Präferenz umkehrt, also $o_1 \geq o_2$ fordert, erhält man eine nutzlose Präferenz.

Im folgenden Abschnitt wird gezeigt, wie nicht erfüllbare oder nutzlose Präferenzen erkannt und bereits während der Nutzerinteraktion unterbunden können. Des Weiteren wird deshalb angenommen, dass die Menge aller Präferenzen $P$ nur nützliche Elemente enthält.

### 4.1 $\theta$-Unabhängigkeit

Um Präferenzen anhand maximaler und minimaler Differenzen effektiv klassifizieren zu können, ist es notwendig zu wissen, ob eine Gewichtsvariable von anderen abhängt.

**Definition 4.1** *Folgende Funktion sei für eine gewichtete CQQL-Anfrage $q^{\Theta}$ definiert:*

$$\mathit{diff}(o_1, o_2, \theta, v, w) = eval(q^{\Theta}, o_1, w_{\theta,v}) - eval(q^{\Theta}, o_2, w_{\theta,v}).$$

*Dabei sind $o_1, o_2$ Datenbankobjekte, $\theta \in \Theta$ eine Gewichtsvariable, $v \in [0,1]$ ein konkreter Wert für $\theta$, $w_{\theta,v}$ eine Gewichtungsfunktion*[7] *und $eval()$ eine CQQL-Auswertungsfunktion. Eine Anfrage $q^{\Theta}$ heißt für ein $\theta$* $\theta$-abhängig *von anderen Gewichtsvariablen, wenn zwei beliebige Objekte $o_1, o_2$, zwei Gewichtswerte $v_1, v_2 \in [0,1]$ und zwei Gewichtungsfunktionen $w_1, w_2$ existieren, so dass*

$$\begin{aligned} &\mathit{diff}(o_1, o_2, \theta, v_1, w_1) \geq \mathit{diff}(o_1, o_2, \theta, v_2, w_1) \qquad \wedge \\ &\mathit{diff}(o_1, o_2, \theta, v_1, w_2) \leq \mathit{diff}(o_1, o_2, \theta, v_2, w_2) \end{aligned}$$

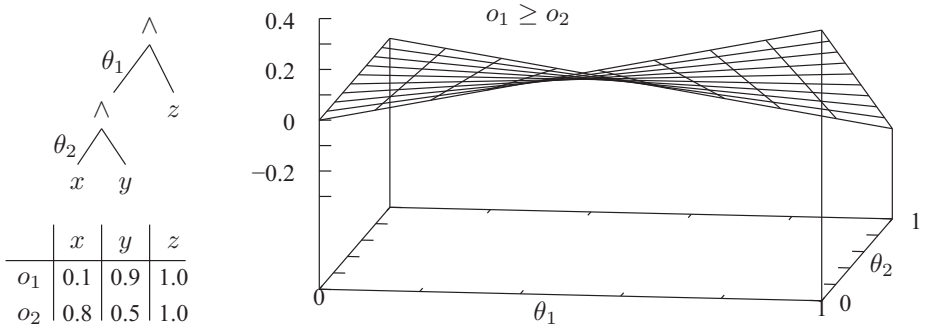
*gilt. Eine Anfrage $q^{\Theta}$ heißt* $\theta$-unabhängig*, wenn sie für kein $\theta$* $\theta$-abhängig *ist.*

$\theta$-Unabhängigkeit bedeutet also, dass die Monotonie von *diff* bezüglich unterschiedlicher Werte für das Gewicht $\theta$ unabhängig von anderen Gewichtsvariablen ist. Die tatsächliche Überprüfung der $\theta$-Unabhängigkeit einer Anfrage nach Def. 4.1 ist nicht praktikabel. Deshalb bietet die folgende Beobachtung die Möglichkeit eines effizienten Tests.

> Eine CQQL-Anfrage $q^{\Theta}$ ist $\theta$-unabhängig, wenn für jede gewichtete Konjunktion $w^{\wedge}_{\theta_1,\theta_2}(\varphi_1, \varphi_2)$ oder Disjunktion $w^{\vee}_{\theta_1,\theta_2}(\varphi_1, \varphi_2)$ keine Unterbedingung $\varphi_1$ oder $\varphi_2$ weitere gewichtete Konjunktionen oder Disjunktionen enthält.

Demzufolge garantiert der Verzicht auf verschachtelte, gewichtete Bedingungen $\theta$-Unabhängigkeit, da ja dann Gewichte in voneinander unabhängigen Teilbedingungen wirken. Abschnitt 4.2 diskutiert Besonderheiten bei verschachtelten, gewichteten Bedingungen. Abb. 6 illustriert das Verhalten einer $\theta$-abhängigen Anfrage mit geschachtelter Gewichtung. Die Anfragebedingung besteht aus zwei ineinander geschachtelten und gewichteten Konjunktionen. Die Score-Werte der Bedingungen $x, y, z$ für die Objekte $o_1$ und $o_2$ sind in der Tabelle angegeben. Die Präferenz $o_1 \geq o_2$ erzeugt als Auswertungsdifferenz die in der Abbildung dargestellte Hyperfläche. Dort nimmt die Different bei steigendem $\theta_1$ für $\theta_2 = 0$ zu und für $\theta_2 = 1$ ab.

[7] $w_{\theta,v} = w \setminus \{(\theta, w(\theta))\} \cup \{(\theta, v)\}$, d.h. alle Gewichte sind entsprechend der Gewichtungsfunktion $w$ gesetzt wobei $\theta$ auf $v$ gesetzt ist.




| | $x$ | $y$ | $z$ |
|---|---|---|---|
| $o_1$ | 0.1 | 0.9 | 1.0 |
| $o_2$ | 0.8 | 0.5 | 1.0 |

Abbildung 6: $\theta$-Abhängigkeit aufgrund geschachtelter Gewichtung für $o_1 \geq o_2$

Aufgrund der Monotonie ist es nun einfach, für eine Präferenz und eine $\theta$-unabhängige Anfrage das Maximum und Minimum von *diff* zu ermitteln. Das Maximum kann sich nur in einer Ecke $\{0,1\}^n$ des $n$-dimensionalen Einheitshyperwürfels, der durch $n$ Gewichte aufgespannt wird, befinden. Dieser gegenüber befindet sich das Minimum.

Der Algorithmus zum Finden des Maximums und zur Kategorisierung einer ungeschachtelten Präferenz findet sich in Abb. 7. Für jede Gewichtsvariable $\theta$ und ein beliebiges $w$ überprüft man lediglich *diff*$(o_1, o_2, \theta, 0, w) \geq$ *diff*$(o_1, o_2, \theta, 1, w)$. Ergibt der Test *wahr*, dann ist das $\theta$-spezifische Maximum $w_{max} = 0$ und Minimum $w_{min} = 1$ bzw. umgekehrt. Nachdem die entsprechenden Ecken gefunden worden sind, kann die maximale bzw. minimale Differenz $max$ und $min$ berechnet werden, um die Präferenz in die Kategorien *nicht erfüllbar, Nutzlos* oder *Nützlich* einzuordnen.

```
for each θ ∈ Θ do
  if diff(o1, o2, θ, 0, w) ≥ diff(o1, o2, θ, 1, w) then
    w_max(θ) = 0
    w_min(θ) = 1
  else
    w_max(θ) = 1
    w_min(θ) = 0
max = diff(o1, o2, θ, w_max(θ), w_max)
min = diff(o1, o2, θ, w_min(θ), w_min)
if max < 0 or min ≥ 0 then
  reject preference o1 ≥ o2
display (w_max)
```

Abbildung 7: Algorithmus zur Kategorisierung einer Präferenz $o_1 \geq o_2$

Im Falle einer nützlichen Präferenz enthält $w_{max}$ wichtige Informationen für den Anwender. Basierend auf einer Präferenz versucht der Lernalgorithmus die Gewichte in Richtung des Maximums zu verschieben. Demnach charakterisiert die Lage des Maximums

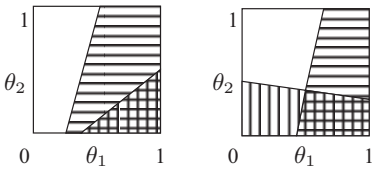



Abbildung 8: Implikation (links) and Überlappung (rechts) von Präferenzen

die Präferenz. Mehrere Präferenzen sind daher ausdrucksmächtiger, wenn deren Maxima in unterschiedlichen Ecken liegen. Die Präferenz $o_3 \geq o_4$ der Objekte aus Tabelle 1 bzgl. unserer Beispielanfrage führt zu einer Bevorzugung der Bedingung $s$ auf Kosten der Bedingung $p$. Die Gewichte streben also zu dem Extrem $\theta_s = 1$ und $\theta_p = 0$.

Jede nützliche Präferenz definiert durch die Forderung *diff* $> 0$ außerdem eine Region des Gewichtshyperwürfels, die nur die Präferenz erfüllende Gewichtswerte enthält. Wird mehr als eine Präferenz angegeben, dann entspricht die Region gültiger Gewichtswerte dem Schnitt der durch die Präferenzen definierten Bereiche. Abb. 8 verdeutlicht dies. Eine Implikation bedeutet hier, dass eine Präferenz in einer anderen (räumlich) enthalten ist. D.h. die Entfernung der enthaltenen Präferenz ändert also die Beschränkung der Gewichtswerte nicht. Hauptziel ist es, die minimale Anzahl von Präferenzen zu ermitteln, die einen kleinstmöglichen aber nicht leeren Schnitt aufweisen, um Gewichtswerte möglichst eindeutig zu definieren. Eine geringe Anzahl an Präferenzen bedeutet vor allem weniger Last für den Nutzer, was detailliert in Kap. 6 begründet wird, und eine schnellere Ausführung des Lernalgorithmus bewirkt (siehe Kap. 7).

### 4.2 Verschachtelte Gewichtung

Aufgrund der $\theta$-Abhängigkeit in geschachtelten, gewichteten Anfragen gestaltet sich das Finden der Maxima und Minima schwieriger als in ungeschachtelten Anfragen. In diesem Fall ist es nicht mehr möglich, die Gewichte isoliert zu testen. Für die Lösung dieses nichtlinearen Optimierungsproblems bieten sich u.a. Hill-Climbing-Algorithmen, wie der im Folgenden verwendete Algorithmus [NM65], an. Aufgrund der Eigenschaft von CQQL-Evaluierungen, dass diese als Summe von Produkten von atomaren Bedingungen und Gewichten eines Objekts (Exponenten sind jeweils 1) ausgedrückt werden können, läuft man jedoch nicht Gefahr in lokalen Maxima oder Minima stecken zu bleiben.

Abb. 9 zeigt anhand eines Beispiels, dass in bestimmten Situationen eine geschachtelt gewichtete Anfrage durch Einführung neuer Gewichte in eine ungeschachtelte Anfrage umgewandelt werden kann. Die Transformationen ergeben sich durch die Ersetzungsregel für die gewichtete Konjunktion sowie durch Gesetze der Booleschen Algebra. Diese Umwandlung ist immer dann möglich, wenn eine gewichtete Konjunktion auf einer gewichteten Konjunktion folgt. Dies gilt analog für zwei aufeinander folgende Disjunktionen.

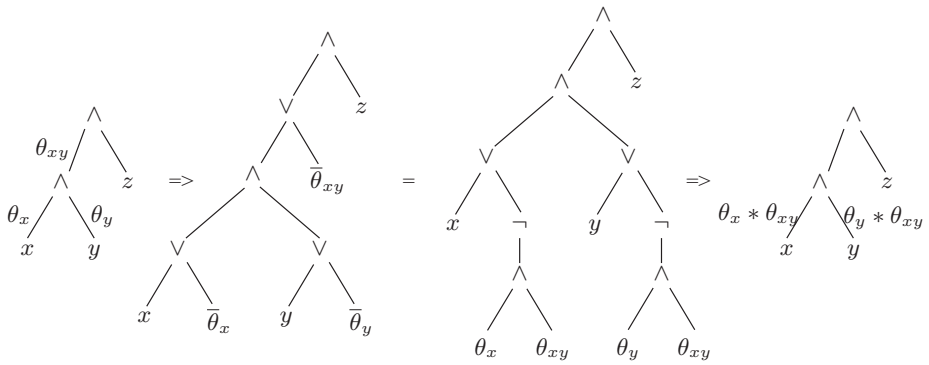



Abbildung 9: Transformation einer geschachtelten Anfrage in eine ungeschachtelte

# 5 Abbildungen zwischen Präferenzen und Gewichten

Die vorliegende Arbeit nutzt Präferenzen zur Nutzerkommunikation und Gewichte zur internen Repräsentation dieser. Mittels dieser Gewichte wird eine totale Ordnung von Ergebnisobjekten (Rank) erzeugt, welche dem Ähnlichkeitsempfinden des Nutzers bzgl. seiner Anfrage entsprechen soll. Im Folgenden wird die Abbildung von Präferenzen in Gewichte und von einem Rank in eine Menge von Präferenzen diskutiert.

## 5.1 Lernen von Gewichten aus Präferenzen

Die Eingabe des verwendeten Lernalgorithmus *prefsToWeight* bildet eine nutzerdefinierte gewichtete Anfrage $q^{\Theta}$ sowie eine konsistente Menge von Präferenzen $P$, die in Gewichte umgewandelt werden sollen (siehe Kap. 4), sowie eine Menge von Anfrageobjekten $O$. Außerdem wird angenommen, dass der Schnitt gültiger Präferenzregionen nicht leer ist.

Jede Präferenz bedeutet eine Einschränkung der Gewichtswerte $w$ durch $eval(q^{\Theta}, o_1, w) - eval(q^{\Theta}, o_2, w) \geq 0$. Mehrere Präferenzen werden wie folgt zusammengefasst:

$$\min_{(o_i \geq o_j) \in P} \left( eval(q^{\Theta}, o_i, w) - eval(q^{\Theta}, o_j, w) \right).$$

Damit die einzelnen Präferenzen gelten, wird gefordert, dass das Ergebnis der Zusammenfassung nicht kleiner als Null werden darf.

Das Lernen der Gewichte kann als ein Optimierungsproblem formuliert werden: das Finden der Gewichtswerte $w$, die obige Zielfunktion maximieren.

Betrachtet man die zu maximierende Funktion, so fällt auf, dass es sich um ein nichtlineares Optimierungsproblem handelt, da eine CQQL-Auswertung als Summe von Produkten atomarer Bedingungen und Gewichten (analog zur ausgezeichneten disjunktiven Normalform) ausgedrückt werden kann. Nichtlinearität führt zu einem schwer berechenbaren Problem. Abschwächend kann jedoch gesagt werden, dass nicht unbedingt das Maximum be-

rechnet werden muss, solange alle Präferenzen respektiert werden. Insofern sind gelernte Gewichte, welche Präferenzen genügen, nicht eindeutig festgelegt.

Nelder und Mead [NM65] schlagen für solche Probleme einen Hill-Climbing-Algorithmus, den *Downhill-Simplex-Algorithmus* vor, welcher das Maximum der Zielfunktion annähert. Um die Güte der Approximation zu verbessern, nutzen wir verschiedene, zufällig gewählte Startpunkte und verwenden das maximale Resultat der Durchläufe. Experimentelle Ergebnisse dieses Ansatzes werden in Kapitel 7 präsentiert.

## 5.2 Reduktion: Ableitungen von Präferenzen aus einem Rank

Die aus der Nutzerinteraktion gewonnene Gewichtungsfunktion $w$ erzeugt einen neuen Rank, welcher dem Anwender erneut präsentiert werden soll. Anstelle der Anzeige sämtlicher Ergebnisobjekte oder der Top-$k$-Objekte kann der Nutzer durch die Präsentation der charakteristischen Präferenzen $P'$, welche den Top-$k$-Rank erzeugen, unterstützt werden. Hierdurch wird das Verständnis des Nutzers für die aktuellen Gewichtswerte der Anfrage verbessert. Die abgeleiteten Präferenzen $P'$ unterscheiden sich in der Regel von den vom Anwender bestätigten Präferenzen, da letztere meist eine Änderung der ersten $k$ Objekte im neuen Rank bewirken, z.B. wenn ein Objekt durch den Nutzer als irrelevant bewertet wird. Diese aus dem Top-$k$-Rank entfernten Objekte werden durch neue ersetzt, welche die Grundlage für die Ableitung bilden. Bezüglich der Generierung des Top-$k$-Ranks sind beide Präferenzmengen jedoch äquivalent. Die abgeleiteten Präferenzen können nun in einer erneuten Nutzerinteraktion bei Bedarf modifiziert werden.

Folgende Anforderungen werden dabei für die abzuleitetende Präferenzmenge $P'$, für Gewichtungsfunktion $w$ und $k$ Ergebnisobjekte aufgestellt:

1. Die gegebene Gewichtungsfunktion $w$ sowie $P'$ erzeugen zwei Ranks $p(w)$ und $p(\textit{prefsToWeight}(P'))$, die $k$-äquivalent sind: $p(w) =_k p(\textit{prefsToWeight}(P'))$[8].

2. $P'$ ist minimal. Es kann also keine Präferenz entfernt werden kann, ohne dass die $k$-Äquivalenz verletzt wird:
   $\neg\exists(o_i, o_j) \in P' : p(\textit{prefsToWeight}(P' \setminus \{(o_i, o_j)\})) =_k p(w)$.

Zur Ableitung der Präferenzen aus einem gegebenen Rank, der durch $w$ erzeugt wurde, findet der folgende Algorithmus[9] Einsatz:

1. Anhand der ersten $k$ Objekte des Ranks werden paarweise Präferenzen $o_i \geq o_{i+1}$ für $i = 1, .., k-1$ abgeleitet. Die entstandene Präferenzmenge $P'$ ist damit bereits um Transitivität und Reflexivität reduziert. Nicht erfüllbare Präferenzen und leere Schnitte von Präferenzregionen treten nicht auf.

2. Nutzlose Präferenzen werden entfernt (siehe Kap. 4).

[8] D.h., ihre Top-$k$ Elemente sind gleich und treten in der gleichen Reihenfolge auf.
[9] Der Aufwand des Algorithmus ist quadratisch bezogen auf $k$ genutzte Objekte.

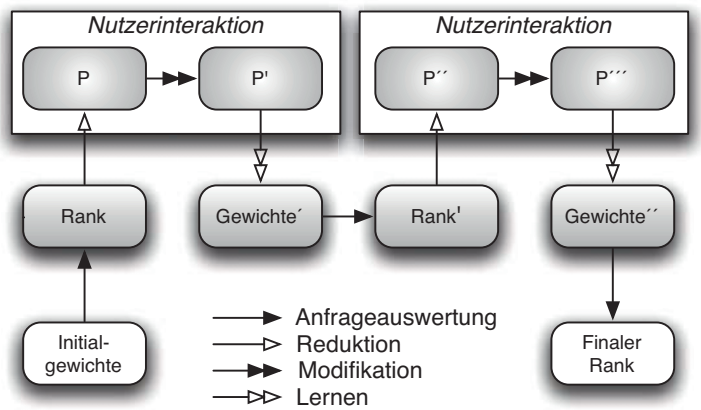



Abbildung 10: Verfeinerung von Gewichten durch Nutzerinteraktion mittels Präferenzen $P$

3. Erzeugung einer minimalen, $k$-äquivalenten Präferenzmenge $P'$:

   (a) Für jede Präferenz $p$ aus $P'$

      i. Teste, ob das Entfernen von $p$ aus $P'$ die $k$-Äquivalenz verletzt.

      ii. Bei Verletzung wird diese Präferenz wieder eingefügt.

   (b) Wiederhole Schritt (a), so lange, bis keine Präferenzen mehr ohne Verletzung der $k$-Äquivalenz entfernt werden können.

## 6 Konzeptioneller Relevance-Feedback-Prozess

Abb. 10 stellt den iterativen Relevance-Feedback-Prozess dar, welcher die vorgestellten Konzepte in sich vereint. Ziel diese Prozesses ist es, Gewichte durch nutzerseitige Interaktion mittels Präferenzen zu modifizieren und zu erlernen, um das subjekte Ähnlichkeitsempfinden bzgl. einer Nutzeranfrage abzubilden.

Während des Relevance-Feedbacks werden dem Nutzer Präferenzen präsentiert, die er entsprechend seines Empfindens verändern kann. Dabei kann aus den folgenden Alternativen gewählt werden:

*Bestätigung*: Eine Präferenz entspricht der Intention des Nutzers und muss nicht modifiziert werden.

*Umkehrung*: Eine Präferenz kann umgekehrt werden, wenn sie der beabsichtigten Bedeutung der Anfrage widerspricht.

*Entfernung*: Kann zwischen zwei Objekten keins bevorzugt werden, so kann die Präferenz entfernt werden.

*Erstellung*: Eine weitere Präferenz kann hinzugefügt werden, wenn diese die Intention des Nutzers besser ausdrückt.

Nach der Modifikation der Präferenzen durch den Nutzer werden diese auf Widersprüche und Nützlichkeit, wie in Kap. 4 und 5 beschrieben, hin untersucht. Um den Nutzer bei der Erstellung neuer Präferenzen zu unterstützen, ist es denkbar, dass ein Algorithmus Präferenzen vorschlägt. Gute Kandidaten sind hierbei solche, welche den Schnitt der Präferenzregionen halbieren oder aber die Gewichte in eine spezifische Richtung drängen.

Die Veränderung der Präferenzen durch den Nutzer führt i. d. R. zum jeweils neuen Rank $Rank'$ (siehe Abb. 10). Dies hat zur Folge, dass alte Top-$k$-Objekte verschwinden und dafür neue Objekte erscheinen. Diese neuen Top-$k$-Objekte dienen dann zur Ableitung neuer Präferenzen $P'$, welche vom Benutzer modifiziert werden können. Diese Schritte bewirken insgesamt ein Entwickeln der Präferenzen, d.h. $P$, $P'$ und $P''$ usw. unterscheiden sich. Greift der Nutzer nicht (weiter) ein, da er entweder zufrieden mit den abgeleiteten Präferenzen ist oder die Suche einfach abbricht, dann erreicht das System einen Fixpunkt. An diesem verändert sich $P^n$ nicht weiter. Ein Vergleich zwischen den verschiedenen Präferenzmengen kann dazu dienen, das Erreichen des Fixpunkts abzuschätzen.

Wesentlichen Einfluss auf die Nutzbarkeit der Verfeinerung der Ergebnismenge hat der Parameter $k$. Bei einem hohen Wert enthält einerseits die aus einem Top-$k$-Rank erzeugte Präferenzmenge viele Elemente, was dazu führt, dass der Nutzer eine Vielzahl dieser bewerten muss. Andererseits ermöglicht eine hohe Anzahl geprüfter Präferenzen ein schnelles und hochwertiges Lernen durch den Algorithmus ohne häufige Iterationsschritte. Demzufolge muss $k$ so gewählt werden, dass der Nutzer nicht überfordert wird, da er prinzipiell wenig Aufwand für die Interaktion mit dem System aufbringen möchte [SP05].

## 7 Experimente

Der diskutierte Ansatz wurde anhand des vorgestellten Beispiels 1.1 experimentell überprüft. Die Experimente wurden auf einem aktuellen Notebook[10] mit 456 Kameradatensätzen durchgeführt. Die Algorithmen sind in Java 5 implementiert und können auf Wunsch zugesandt werden. Grundlage der Experimente bilden jeweils 1000 Durchläufe.

Das erste Experiment analysiert, wieviele automatisch aus einem Rank abgeleitete Präferenzen tatsächlich nützlich sind (siehe Abs. 5.2), wobei alle Gewichte auf 1 gesetzt wurden. Abb. 11 zeigt, dass ca. 62% der Präferenz nützlich sind, so dass eine Reduktion also sinnvoll ist.

Abb. 12 zeigt die Laufzeitentwicklung des Lernalgorithmus. Diese hängt hauptsächlich von zwei Parametern ab: der Anzahl der Präferenzen und den Gewichtsvariablen innerhalb der Anfrage. Bemerkenswert ist hier, dass die Laufzeit für weniger als 10 Gewichtsvariablen und weniger als 40 Präferenzen unter einer Sekunde liegt. Eine höhere Anzahl der Parameter ist aufgrund des Nutzerverhaltens kaum zu erwarten [SP05], erhöht jedoch die Laufzeit nur in einem vernünftigen Rahmen.

[10] Intel Core 2 Duo 2.5 GHz, 4 GB RAM, Mac OS X 10.5.5, Java 5

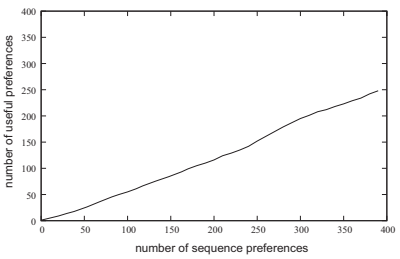



Abbildung 11: Anzahl nützlicher automatisch abgeleiteter Präferenzen

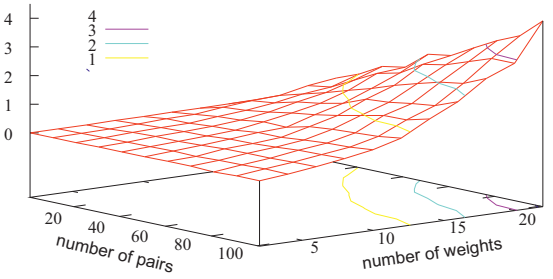



Abbildung 12: Laufzeit [s] in Abhängigkeit der Anzahl von Gewichten ($weights$) und Präferenzen ($pairs$)

Bezüglich der Nutzerinteraktion ist besonders interessant, wie schnell der Nutzer zu einem zufriedenstellenden Ergebnis bzgl. seiner Anfrage gelangt. Ziel des zweiten Experiments ist deshalb die Untersuchung, wieviele Präferenzen der Anwender überprüfen muss bis der Ziel-Rank erreicht ist. Wir gehen hier idealerweise von konsistenten, also widerspruchsfreien Präferenzen aus. Anhand einer zufälligen Gewichtung wurden die Top-$k$-Objekte berechnet und anschließend, ausgehend von einer zufälligen Initialgewichtung, überprüft, wieviele Präferenzen der Nutzer sehen oder korrigieren muss bis sein Ziel-Rank erreicht ist. Hierbei kam das Beispiel 1.1 in zwei Top-$k$-Varianten zum Einsatz. Einmal mit $k = 10$ und einmal $k = 20$.

Bei den Präferenzen wird bei diesem Experiment in gesehene und korrigierte unterschieden, wobei letztere durch den Nutzer umgekehrt wurden. Die Abbildungen zeigen jeweils zwei Kurven. Die obere gibt an, wieviele Zielobjekte bereits gesehen wurden und die untere welche Objekte im aktuellen Durchlauf unter den Top-$k$ liegen. Die Kurven konvergieren in Richtung von $k$, jedoch fällt auf, dass sie nicht streng steigen, was sich auf die Verwendung von Zufallszahlen im Algorithmus zurückführen lässt.

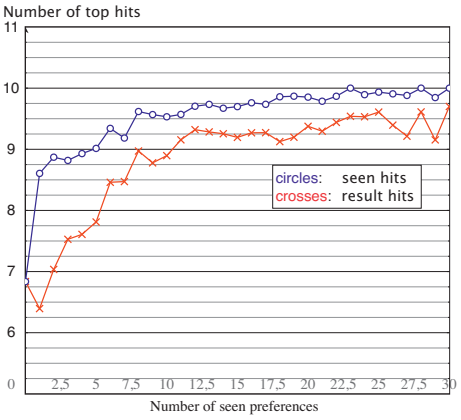



Abbildung 13: Top-10, gesehene Präferenzen

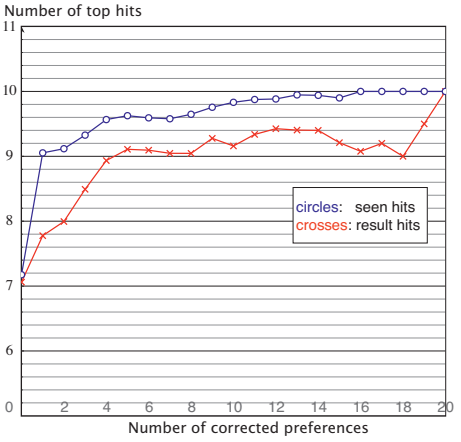



Abbildung 14: Top-10, korrigierte Präferenzen

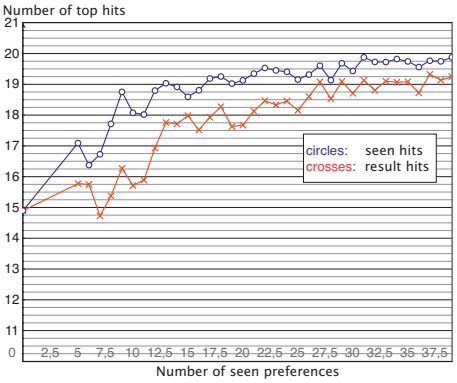



Abbildung 15: Top-20, gesehene Präferenzen

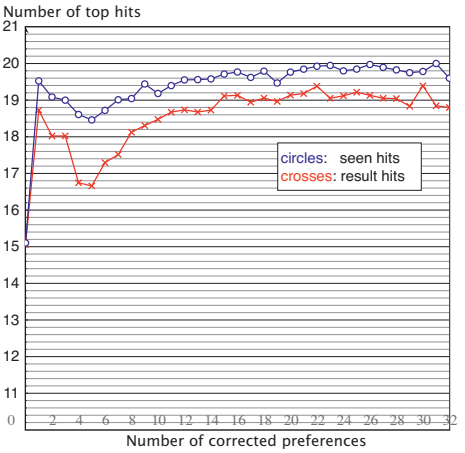



Abbildung 16: Top-20, korrigierte Präferenzen

Es fällt auf, dass in vielen Fällen bessere Resultate erzielt werden, wenn keine Präferenzen anstelle weniger definiert sind (siehe Abb. 13 - 16). Es kann folglich vermutet werden, dass die Angabe weniger Präferenzen die Semantik der Initialgewichte erst zerstört aber noch nicht ausreichend für eine neue Semantik ist. Offenbar muss erst eine kritische Masse an Präferenzen definiert werden, um die Suchqualität zu verbessern. Weniger überraschend ist die Erkenntnis, dass relativ zu einem höheren $k$ mehr Präferenzen benötigt werden, um die Suchqualität zu verbessern.

# 8 Ausblick

Aufbauend auf den vielversprechenden, experimentellen Ergebnissen muss das vorgestellte Verfahren erweitert werden. Im Schwerpunkt der Forschung steht dabei die Entwicklung einer Benutzeroberfläche, die den Anwender insbesondere bei der Verfeinerung seiner Anfrage unterstützt. Grundlage bildet dabei die Ausnutzung der Präferenzregionen und Gewichtsmaxima, um dem Nutzer sinnvolle Empfehlungen bei der Verfeinerung zu geben.

Das Finden einer Menge guter Präferenzen ist dabei mit dem Problem der Generierung von Entscheidungsbäumen verwandt, für das zahlreiche Algorithmen bekannt sind [BA97]. Gute Präferenzen, die dem Nutzer vorgeschlagen werden können, sind solche, welche den Schnitt bereits existierender Präferenzregionen halbieren. Anhand dieser Präferenzen kann der Nutzer seine Anfrage zielgerichtet steuern und so ein Verständnis für das konzeptionelle Modell des Systems gewinnen. Außerdem muss das Verhalten des Verfahrens bei wenig Präferenzen näher untersucht werden.

Zur Evaluierung des Lernens von Gewichten müssen weitere Algorithmen untersucht werden, die eine feinere Regulierung des Lernverfahrens ermöglichen. Denkbar sind hier genetische Algorithmen, deren Selektionskriterien sich an Nutzeranforderungen und Anwen-

dungskontexte anpassen.

Im Rahmen der Entwicklung eines integrierten Prototyps, welcher für Nutzertests unabdingbar ist, werden parallel geeignete Interface-Metaphern und Kommunikationsmechanismen analysiert, um dem Anwender ein vorhersehbar reagierendes, ihn unterstützendes Werkzeug zur Seite zu stellen. Hierbei werden auch Bereiche beachtet, welche die Anfrageformulierung nicht direkt tangieren, sondern einem ganzheitlichen Suchbegriff zuzuordnen sind. Denkbar sind hier unterstützende Recherchewerkzeuge, wie Ablagen, die gefundene Objekte aufnehmen, welche als Nebenprodukt der Hauptsuche anfallen und beispielsweise als Ausgangspunkt einer neuen Suche dienen können.

## Literatur


[ACDG03] Sanjay Agrawal, Surajit Chaudhuri, Gautam Das und Aristides Gionis. Automated Ranking of Database Query Results. In *CIDR*, 2003.

[BA97] Leonard A. Breslow und David W. Aha. Simplifying decision trees: A survey. *Knowl. Eng. Rev.*, 12(1):1–40, 1997.

[BG93] Vicki Bruce und Patrick R. Green. *Visual Perception –physiology, psychology and ecology (2nd ed., reprinted)*. Lawrence Erlbaum Associates, Publishers, Hove and London, UK, 1993.

[BKS01] Stephan Börzsönyi, Donald Kossmann und Konrad Stocker. The Skyline Operator. In *Proceedings of the 17th International Conference on Data Engineering*, Seiten 421–430, Washington, DC, USA, 2001. IEEE Computer Society.

[BN36] G. Birkhoff und J. von Neumann. The Logic of Quantum Mechanics. *Annals of Mathematics*, 37:823–843, 1936.

[BP95] P. Bosc und O. Pivert. SQLf: A Relational Database Language for Fuzzy Querying. *IEEE Transactions on Fuzzy Systems*, 3(1):1–17, Februar 1995.

[Cho03] Jan Chomicki. Preference formulas in relational queries. *ACM Trans. Database Syst.*, 28(4):427–466, 2003.

[Cho07] Jan Chomicki. Database querying under changing preferences. *Ann. Math. Artif. Intell.*, 50(1-2):79–109, 2007.

[CMPT00] Paolo Ciaccia, Danilo Montesi, Wilma Penzo und Alberto Trombetta. Imprecision and User Preferences in Multimedia Queries: A Generic Algebraic Approach. In K.-D. Schewe und B. Thalheim, Hrsg., *FoIKS: Foundations of Information and Knowledge Systems, First International Symposium, FoIKS 2000, Burg, Germany, February 14-17, 2000*, Jgg. 1762 of *Lecture Notes in Computer Science*, Seiten 50–71. Springer, 2000.

[CRW05] Surajit Chaudhuri, Raghu Ramakrishnan und Gerhard Weikum. Integrating DB and IR Technologies: What is the Sound of One Hand Clapping? In *CIDR*, Seiten 1–12, 2005.

[CW08] Claremont Workshop. The Claremont Database Research Self Assessment. Bericht, 2008.

[FW00] R. Fagin und E. L. Wimmers. A Formula for Incorporating Weights into Scoring Rules. *Theoretical Computer Science*, 239(2):309–338, 2000.

[Kie02] W. Kießling. Foundations of Preferences in Database Systems. In *Proc. of the 28th Int. Conf. on Very Large Data Bases, VLDB'02, Hong Kong, China, August, 2002*, Seiten 311–322. Morgan Kaufmann Publishers, 2002.

[KN07] Aljoscha Klose und Andreas Nürnberger. On the Properties of Prototype-based Fuzzy Classifiers. *IEEE Transactions on Systems, Man, and Cybernetics Part B*, 37(4):817–835, 2007.

[KRR02] D. Kossmann, F. Ramsak und S. Rost. Shooting Stars in the Sky: An Online Algorithm for Skyline Queries. In *Proc. of the 28th Int. Conf. on Very Large Data Bases, VLDB'02, Hong Kong, China, August, 2002*, Seiten 275–286. Morgan Kaufmann Publishers, 2002.

[Lee94] Joon Ho Lee. Properties of Extended Boolean Models in Information Retrieval. In SIGIR, Hrsg., *SIGIR '94: Proceedings of the 17th annual international ACM SIGIR conference on Research and development in information retrieval*, Seiten 182–190. Springer–Verlag New York, Inc., 1994.

[LSDJ06] Michael S. Lew, Nicu Sebe, Chabane Djeraba und Ramesh Jain. Content-based multimedia information retrieval: State of the art and challenges. *ACM Trans. Multimedia Comput. Commun. Appl.*, 2(1):1–19, 2006.

[NM65] J. A. Nelder und R. Mead. A Simplex Method for Function Minimization. *Computer Journal*, 7:308–313, 1965.

[RJ05] Lawrence A. Rowe und Ramesh Jain. ACM SIGMM retreat report on future directions in multimedia research. *ACM Trans. Multimedia Comput. Commun. Appl.*, 1(1):3–13, 2005.

[Roc71] J. J. Jr. Rocchio. Relevance Feedback in Information Retrieval. In G. Salton, Hrsg., *The SMART Retrieval System - Experiments in automatic Document Processing*, Kapitel 14, Seiten 313–323. Prentice Hall, Englewood Cliffs, New Jersey, USA, 1971.

[SB88] Gerard Salton und Chris Buckley. Improving Retrieval Performance by Relevance Feedback. Bericht, Ithaca, NY, USA, 1988.

[Sch08] Ingo Schmitt. QQL: A DB&IR Query Language. *The VLDB Journal*, 17(1):39–56, 2008.

[Sel59] O. G. Selfridge. Pandemonium. A paradigm for learning. *The mechanics of thought processes*, 1959.

[SFW83] Gerard Salton, Edward A. Fox und Harry Wu. Extended Boolean Information Retrieval. *Commun. ACM*, 26(11):1022–1036, 1983.

[SP05] Ben Shneiderman und Catherine Plaisant. *Designing the user interface: Strategies for effective human–computer interaction*. Pearson, Boston, 4. ed.. Auflage, 2005.

[SS03] N. Schulz und I. Schmitt. Relevanzwichtung in komplexen Ähnlichkeitsanfragen. In G. Weikum, H. Schöning und E. Rahm, Hrsg., *Datenbanksysteme in Business, Technologie und Web, BTW'03, 10. GI-Fachtagung, Leipzig, Februar 2003*, Lecture Notes in Informatics (LNI) Volume P-26, Seiten 187–196, Bonn, 2003. Gesellschaft für Informatik.

[SS04] I. Schmitt und N. Schulz. Similarity Relational Calculus and its Reduction to a Similarity Algebra. In Dietmar Seipel und J. M. Turull-Torres, Hrsg., *Third Intern. Symposium on Foundations of Information and Knowledge Systems (FoIKS'04), Austria, February 17-20*, Jgg. 2942 of *Lecture Notes in Computer Science*, Seiten 252–272. Springer-Verlag Berlin Heidelberg, 2004.

[SZN08] Ingo Schmitt, David Zellhöfer und Andreas Nürnberger. Towards Quantum Logic Based Multimedia Retrieval. *Annual Meeting of the North American Fuzzy Information Processing Society, 2008*, Seiten 1–6, 2008.

[Wei07] Gerhard Weikum. DB&IR: both sides now. In SIGMOD, Hrsg., *SIGMOD '07: Proceedings of the 2007 ACM SIGMOD international conference on Management of data*, Seiten 25–30. ACM, 2007.

[Zad88] Lofti A. Zadeh. Fuzzy Logic. *IEEE Computer*, 21(4):83–93, April 1988.

[Zie05] Martin Ziegler. Quantum Logic: Order Structures in Quantum Mechanics. Bericht, University Paderborn, Germany, 2005.