Kuş, M.; Varol, A.; Oğurol, Y.; Varol, Y.: "Verarbeitung von unsiherem Wissen mit Fuzzy-Prolog", Second Turkish-German Joint Computer Application Days, 15-16 October, 1998, Konya, Proceedings, pp. 243-260

2.6. VERARBEITUNG VON UNSICHEREM WISSEN MIT FUZZY-PROLOG

Zusammenfassung

Die Komplexität unserer Umwelt mit ihren zahlreich vemetzen Subsystemen verlangt für ihre adäquate Beschreibung einen interdisziplinären, integrierten Ansatz. Die Informatik stellt hierzu unterschiedlichste Methoden und Werkzeuge der Wissensverarbeitung zur Verfügung. Diese Komplexität resultiert vor allem aus der Dominanz von unsicherem Wissen in der Umwelt. Klassische Methoden der Wissensrepräsentation und-verarbeitung ermöglichen bisher nicht den adäquaten Umgang mit unsicherem Wissen. Es existiert aber die Notwendigkeit, dieses problematische Wissen mit Hilfe von KI-Methoden angemessen zu beschreiben, um die Komplexität der entsprechenden Domäne zu vereinfachen und damit das jeweilige Anwendungsgebiet besser beherrschen zu können

Die Programmiersprache Prolog (Programming in Logic) zählt zu den zentralen Sprachen auf dem Gebiet der Wissensrepräsentation und -Verarbeitung. Daher liegt es sehr nahe, diese Sprache als Grundlage für die Erweiterung um Konstrukte zur Verarbeitung von unsicherem Wissen zu verwenden. Prolog basiert auf dem Prädikatenkalkül der ersten Stufe und arbeitet auf HÖRN-Klauseln. In dieser Arbeit wurde ein Kalkül basierend auf der Kombination der Fuzzy-Set- und der Plausibilitätstheorie verwendet. Nach unseren Erfahrungen läßt sich feststellen, daß diese Kombination eine gute Lösung zur Repräsentation und -Verarbeitung von unsicherem Wissen ist.

Einführung

Kuş, M.; Varol, A.; Oğurol, Y.; Varol, Y.: “Verarbeitung von unsiherem Wissen mit Fuzzy-Prolog”, Second Turkish-German Joint Computer Application Days, 15-16 October, 1998, Konya, Proceedings, pp. 243-260

Es existieren eine Reihe von Methoden und Mechanismen zur Formalisienmg und Repräsentation von Wissen, wie z.B. Prädikatenlogik, Objekte/Frames, semantische Netze etc. Die für die Verarbeitung des Wissens notwendige Softwarekomponente, die sogenannte *Inferenzmaschine,* ist ein Steuerprogramm, das nach einer geeigneten Strategie und gegebenenfalls unter Verwendung von Metawissen aus den bereits in der Wissensbasis gespeicherten Daten, sowie im Dialog oder anderswie erhobenen Falldaten möglichst gezielt Schlußfolgerungen "produziert" [1]. Wissensrepräsentationsformalismen können, verbunden mit ihren Ableitungsstrategien, als *Kalküle* aufgefaßt werden. Eine Inferenzmaschine sollte daher ein korrektes und vollständiges Kalkül implementieren, das theoretisch fundiert ist.

Ein solches Kalkül stellt z.B. die Prädikatenlogik erster Ordnung dar. Sie ist eine theoretisch fundierte Wissensrepräsentation [2], Auf ihr basieren eine Reihe von Programmiersprachen, deren bekanntester Vertreter Prolog ist [3, 4]. Prolog selbst ist ein mächtiges Werkzeug zur Wissensverarbeitung, welches sich auf Hornklauseln - eine Untermenge der Prädikatenlogik erster Ordnungstützt. Ein Prolog-System arbeitet die Anfragen zielorientiert ab, d.h. ausgehend von einer Anfrage wird in der Regelbasis nach gespeichertem Wissen gesucht, welches diese Anfrage bestätigt.

Häufig können Fakten und Regeln in einem Wissensbereich nur "unscharf angegeben werden. Die adäquate Verarbeitung dieses Wissens erfordert die Entwicklung neuer Konzepte. Die Methoden der unsicheren Wissens Verarbeitung ermöglichen einen Umgang mit diesen "unscharfen" Fakten und Regeln. Es stellt sich die Frage, wie man derartige Unsicherheiten modelliert und wie man deren Verarbeitung, z.B. in ein Prolog-System integriert. Die Modellierung und Verarbeitung unsicheren

Wissens erfordert aufgrund deren Komplexität eine formale mathematische Vorgehensweise. Das in dieser Arbeit verwendete Kalkül basiert auf einer Kombination der *Fuzzy-Set*-und *Plausibilitätstheorie (Dempster-Shafer-Theorie) [5,* 6,7].

Wesentliches Verarbeitungsprinzip eines Prolog-Systems ist die *Unifikation,* die in dieser Arbeit zur Verarbeitung von unsicherem Wissen erweitert werden muß [8]. Die klassische Unifikation ist ein rein syntaktisches Verfahren, das versucht, Terme (syntaktische Einheiten) durch die Ersetzung von Variablen identisch zu machen. Die Ergänzung der klassischen Unifikation um eine *semantische Unifikation* ist ein möglicher Schritt bei der Erweiterung eines Prolog-Systems, um unsicheres Wissen verarbeiten zu können [9]. Die semantische Unifikation bezeichnet dabei den Vorgang, Terme mit ähnlicher Bedeutung (Semantik) zu unifizieren.

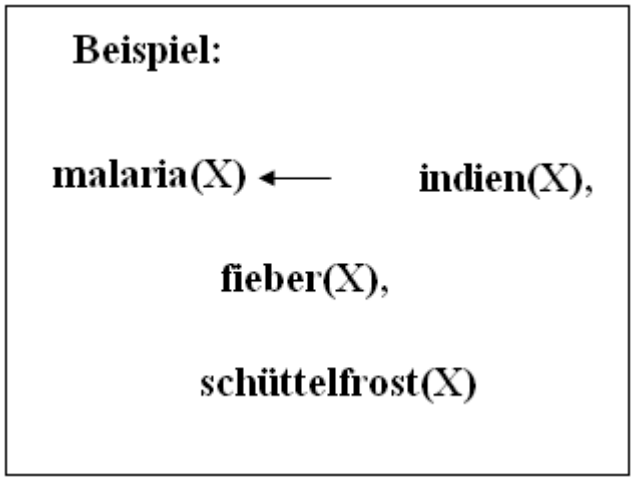
Die Zuordnung von Wertintervallen zu Fakten und Regeln bietet dabei eine Möglichkeit zur Beschreibung von Unsicherheiten in regelbasierten Systemen. Diese Wertintervalle können als ein Maß für die Unsicherheit interpretiert werden, wobei die untere Grenze des Intervalls ein notwendiges Kriterium und die obere Grenze ein mögliches Kriterium für die Erfüllbarkeit darstellt [10].

Konzept

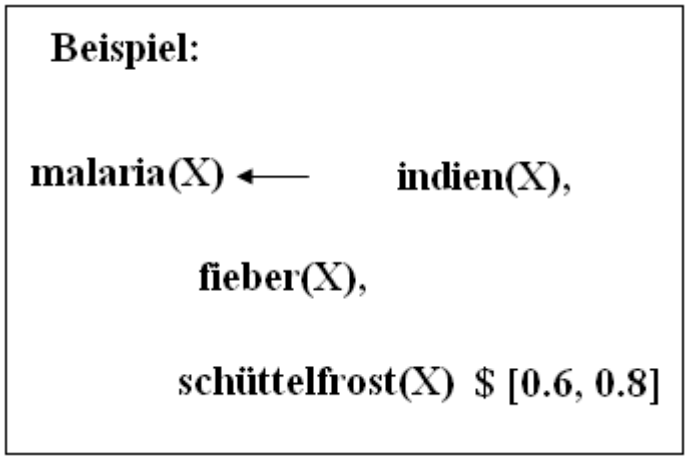
Um unsichere Regeln und Fakten modellieren zu können, muß die klassische Sprache Prolog entsprechend erweitert werden. Es stellt sich zwangsläufig die Frage, in welcher Form und Struktur sich die Erweiterung der Sprache realisieren läßt, d.h. welche Theorien hierzu herangezogen werden können. Aus den unterschiedlichsten Ansätzen der unsicheren Wissens-verarbeitung und -modellienzng wurden Ideen aus der *Fuzzy-Set-Theorie* und der *Plausibilitätstheorie {Evidenztheorie)* aufgegriffen. Hierbei liefert die Fuzzy-Set-Theorie die Idee der *linguistischen Variablen* zur

Modellierung von linguistischen Unschärfen, welche sich in der menschlichen Sprache stark widerspiegelt. Beispiele hierzu sind Formulierungen der Art "hohe Emissions- werte", "schmutzige Gewässer" oder "starke ÜV-Imissionen".

Wissen wird in den regelbasierten Sprachen durch *Wenn ... Dann ...* Regeln formuliert. Die klassische Logik liefert nicht die Möglichkeit, einer Implikationen einen Grad der Gewichtung, z.B. durch ein Wertintervall, anzufügen. Betrachtet man z.B. folgendes logisches Programm;



Wertet man das obige Beispiel nach der klassischen Logik aus, hat eine Person *X* die Krankheit Malaria, wenn Person *X* in Indien ist, Fieber und Schüttelfrost hat. Ein Grad der Gewichtung ist hierbei nicht möglich. Betrachtet man nun aber das folgende logische Programm in Form einer Implikation, an der ein Grad an Gewichtung angelügt ist:



Wertet man diese Implikation mit einem Grad an Gewichtung, in Form eines logischen Programms aus, würde man folgende Interpretation erhalten: Die Person *X* hat zu *60%-80%* die Krankheit Malaria, wenn *X* in Indien ist, Fieber und Schüttelfrost hat.

Interpretiert man das Beispiel ohne Gewichtung, hat Person *X* zu 700% Malaria, wenn die JPrämissen der Implikation erfüllt sind. Im letztgenannten Beispiel dagegen, hat *X* zu *60^>-80%* Malaria, wenn die Prämissen erfüllt sind. Sie kann aber auch zu *20%-40%* eine andere Krankheit, z.B. eine Erkaltung, haben. Das Ergebnis ist also mit einer gewissen Unsicherheit behaftet, was für den zu modellierenden Sachverhalt angemessener erscheint.

Um diesen Grad der Gewichtung realisieren zu können, wird ein Konzept aus der Plausibilitätstheorie aufgegriffen. Das bietet die Möglichkeit, Regeln und Fakten durch ein *Wertintervall* zu gewichten. Die Ideen der Fuzzy-Set Theorie und Evidenztheorie lassen sich verknüpfen, weil die Possibilitätsmaßespezielle Plausibilitätsmaße sind und Fuzzy-Mengen als Possibiütäten interpretiert werden können. Um die Ideen der linguistischen Variablen und Wertintervalle einzubeziehen, um unsicheres Wissen repräsentieren und verarbeiten zu können, wird zunächst ein Kalkül

aufgestellt werden, welches diese Ideen aufgreift. Dieses Kalkül wird im folgendem als *Support-Logik- Kalkül* und die Wertintervalle als *Support-Intervalle* bezeichnet. Das Support- Logik-Kalkül basiert also auf den Ideen der Fuzzy-Set Theorie und der Evidenz-Theorie (Abb. l) [9].

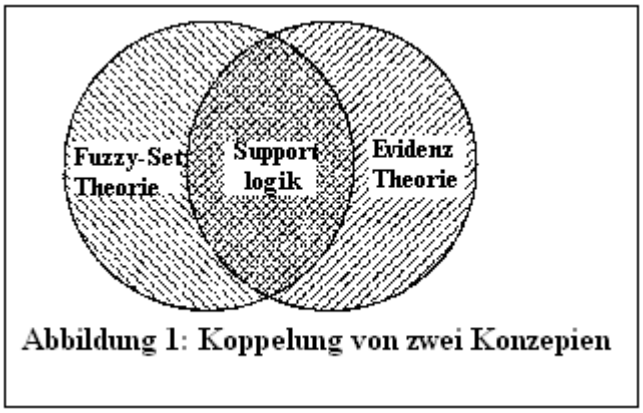


Abbildung 1: Koppelung von zwei Konzepien

Im unserem Fuzzy-Prolog sollen diskrete und kontinuierliche Fuzzy-Mengen erlaubt sein. Diskrete Fuzzy-Mengen lassen sich durch eine Aufzählung der *Singletons* beschreiben. Kontinuierliche Puzzy-Mengen werden auch durch diskrete Fuzzy-Mengen beschrieben, die jedoch linear interpoliert

werden. Bei der Deklaration muß der Name der Fuzzy-Menge (bzw. des Fuzzy-Terms) und der Name der zugehörigen Gründmenge (bzw. Linguistische Variable) mit angegeben werden.

Beispiel:

Deklaration von. .zwei kontinuierlichen Fuzzy-Mengen (durch einSystemprädikat):

Fuzzy_set(durchschnitt,körpergröße # [0:160, l: 170, l: 180,0:190]).

füzzy_set(groß, k0rpergr0ße# [0:180,1:190, 0:200]) (Abb. 2).

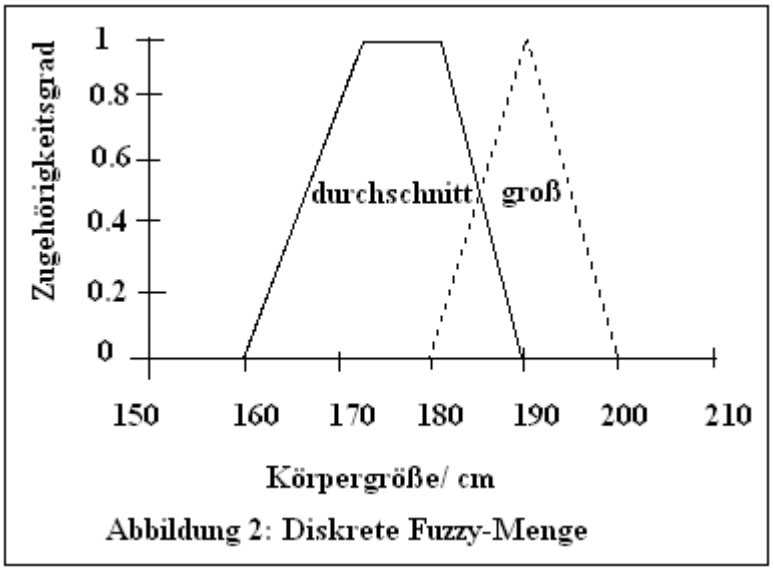


Abbildung 2: Diskrete Fuzzy-Menge

Deklaration einer diskreten Fuzzy-Menge (durch ein Systemprädikat):

fuzzy_set(warni,temperatur : [0.5:21, 0,75:22, 1:23. 0.75:24, 0.5:25]) (Abb. 3).

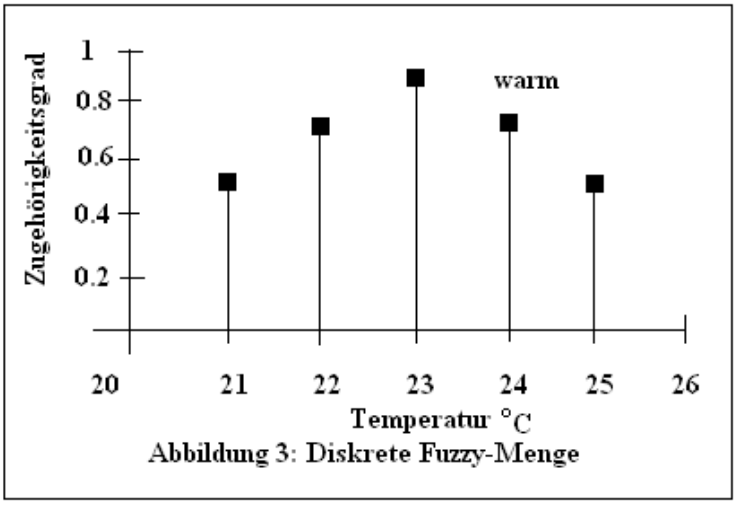


Abbildung 3: Diskrete Fuzzy-Menge

Diskrete Fuzzy-Mengen verwendet man in da" Regel, wenn Zwischenwerte nicht erfaßt werden können oder nicht existieren. Aus einer kontinuierlichen Fuzzy-Menge kann eine diskrete Fuzzy-Menge durch Diskretisierung der Grundmenge gewonnen werden oder umgekehrt durch Inter polation von Zwischenwerten in eine kontinuierliche Fuzzy-Menge verwandelt

werden.

Die Einführung von Puzzy-Mengen in Prolog macht es notwendig, die klassische Unifikation zu erweitern. Diese Erweiterung zur semantischen Unifikation ist Bestandteil des Support-Logik-Kalküls. Fakten und Regeln sollen in Prolog durch Support-Intervalle gewichtet werden. Hierzu wird die Syntax der Sprache Prolog um Support-Intervalle erweitert. Sind Fakten und Regeln ohne Support-Intervall angegeben, wird ihnen automatisch das Einheitsintervall *[1,1]* zugeordnet.

Support Logik Kalkül

Die Gewichtung von Regeln und Fakten im Fuzzy-Prolog erfolgt durch sogenannte Support-Intervalle. Ein Support-Intervall besitzt die Form *['Nee, Pos],* wobei der erste Wert des Intervalls *(Nee)* den notwendigen Support (engl. necessity) und der zweite Wert *(Pos)* den möglichen Support (engl. possibility) für die Auswahl einer Hornklausel angibt. Jede Klausel wird also mit einem Support-Intervall belegt und erhält dadurch eine Gewichtung. Die Syntax der gewichteten Hornklausel ist wie folgt definiert:

Ist eine Klausel ohne Support-Intervall angegeben, so wird sie automatisch mit dem Intervall *[1,1]* belegt. L_0 ist der Kopf (engl. *head)* und L_1, ... ,L_k der Rumpf (engl. *body)* der gewichteten Hornklausel. Einer gewichteten Hornklauseln läßt sich folgende prozedurale Semantik zuordnen ([Bald86]); Sind die Prämissen L_1, ... ,L_k erfüllt, so ist die Konklusion *lq* mit dem Grad *Nee* und -*lq* mit dem Grad *(1-Pos)* erfüllt.

Die Supports müssen jeweils die folgende Bedingung erfüllen: *Nec + (l -Pos)≤.l*

Ist der Rumpf einer gewichteten Hornklausel leer, so ist die Syntax der Form: L_0 *$[Nec, Pos]*

Der Fakt *lq* ist mit dem Grad *Nee* und die Negation — L_0 mit dem Grad *(1-Pos)* erfüllt, für jede beliebige Variableninstanziierung.

Bei den Support-Intervallen handelt es sich nietet um Wahrscheinlichkeitsmaße im Sinne der klassischen Wahrscheinlichkeitstheorie, sondern das Support-Intervall beschreibt die Grenzen, in dem die unbekannte "Wahrscheinlichkeit" liegt. Ein Support-Intervall von *[0,1]* beschreibt die totale Unsicherheit bzw. Unwissenheit über eine Aussage.

Die Verrechnung der Support-Intervalle erfolgt durch die Definition von kontextsensitiven Operatoren. Diese Operatoren entsprechen Fuzzy-Operatoren, die um Support-Intervalle erweitert sind. Die Modellierung der Operatoren ist im Kalkül nicht fest vorgegeben. In dieser Arbeit wird das *Multiplikations-Modell* [11] zur Beschreibung der Negation-, Konjunktion, Diskjunktion und Inferenzoperatoren verwendet, daß mit der *Dempster-Shafer-Theorie* konform ist. Ein Alternatives Modell ist z.B. die Beschreibung durch die klassischen Min-Max-Operatoren der Fuzzy-Set-Theorie.

Fuzzy-Mengen

Eine Gewichtung von Regeln und Fakten mit Support-Intervallen reicht nicht aus, um linguistische Unschärfe zu modellieren. Die Repräsentation von Fuzzy-Mengen ist daher sinnvoll. Fuzzy-Mengen stellen ähnlich wie Konstanten atomare Objekte dar und müssen vor der ersten Benutzung definiert werden. Für die Definition der Fuzzy-Menge, müssen der Name des Terms, der Name der zugehörigen Grundmenge (entspricht

der linguistischen Variablen) und die Fuzzy-Menge in Form einer Liste angegeben werden.

Beispiel:

Eine Fuzzy-Menge *warm.* die auf der Domäne *Temperatur* definiert ist, wird z.B. durch ein Prädikat *fuzzy_set* definiert. Die Liste beinhaltet Singeltons, die linear interpoliert werden können.

füzzy_set(warm, temperatur # [....]).

Zwei Arten von Fuzzy-Mengen werden im Support-Logik-Kalkül zugelassen. Die erste Form erlaubt die Verweildung diskreter, unscharfer Mengen und die zweite Form erlaubt die Verwendung linear interpolierter, diskreter, unscharfer Mengen, die wie folgt definiert sind:

Definition: Diskrete'Fuzzy-Menge im Support-Logik-Kalkül

Eine endliche diskrete Fuzzy-Menge ist im Support-Logik-Kalkül ein 3- Tupel *(T,M,FS),* wobei *T* der Name der Fuzzy-Menge, *M* der Name der Grundmenge *G* und *FS* eine endliche Fuzzy-Menge auf *G* ist. Die endliche Fuzzy-Menge erhält die Beschreibungsform *FS*

:= *{x/μ(x)* l *x ∈ G}. μ(x)* ist ein Wert aus dem Intervall *[0,1].*

Definition: Kontinuierliche Fuzzy-Menge im Support-Logik-Kalkül Eine kontinuierliche Fuzzy-Menge wird im Support-Logik-Kalkül ist eine diskrete Fuzzy-Menge mit linear interpolierten Elementen

Im Rahmen des Support-Logik-Kalküls werden nur Fuzzy-Mengen zugelassen, welche die Normalisiertheitseigenschaft besitzen. Eine Einführung von Fuzzy-Mengen beeinflußt die klassische (syntaktische)

Unifikation. Die Erweiterung der klassischen Unifikation .zur semantischen Unifikation ist Gegenstand des nächsten Abschnitts.

Erweiterte Unifikation

Eine Erweiterung der logischen Programmiersprache Prolog um Fuzzy-Konzepte erfordert eine angemessene Modifizierung der klassischen Unifika- tionsoperation. Die rein auf der syntaktischen Ebene arbeitende Opera-tion wird um eine semantische Ebene erweitert. Die semantische Unifikation liefert, zusätzlich zur Substitutionsmenge, ein Support-Intervall als Ergebnis. Das Sup- port-Intervall beschreibt den Grad der Ähnlichkeit der zu unifizierenden Terme.

Beispiel:

Gebeben sei ein kleines Prolog-Programm mit einem einzigen Fakt p(ca,,25°C). Eine Anfrage der Form ?- p(warm) würde im klassischen Prolog fehlschlagen, obwohl die Terme eine ähnliche Bedeutung besitzen können. Sind die Terme ca_25^0C und *warm* Beispielsweise als Fuzzy-Mengen über einer gleichen Grundmenge definiert, so läßt sich die Anfrage, aufgrund der sematischen Unifikation in einem Fuzzy-Prolog-System, reduzieren.

Fuzzy -Patter n-Matching

Die klassische Unifikation läßt sich als einfaches Pattern-Matching, verknüpft mit Variablenbindungen, auffassen. Unabhängig von der Bedeutung der zu unifizierenden Terme wird der Test auf Gleichheit rein syntaktisch durchgeführt. Die Unifikation schlägt fehl, falls sich die beiden Terme nicht durch mögliche Variablenbindungen entsprechen. Symbol für Symbol werden zwei Tßrrne verglichen und durch eventuelle Variablenbindung identisch gemacht [12].

Die beiden Ausdrücke "warm" und "ca_25°C" beschreiben z.B. die Temperatur in einem Raum. Ein rein syntaktischer Vergleich wurde

fehlschlagen, obwohl beide Ausdrücke eine ähnliche Bedeutung besitzen. An dieser Stelle setzt das Konzept der partiellen Übereinstimmung (Fuzzy-Pattem-Matching) an. Die linguistischen Unschärfen lassen sich in der Fuzzy-Set-Theorie durch Fuzzy-Mengen charakterisieren und auch als Possibilitäts- funktionen interpretieren.

Gegeben sei ein atomares unscharfes Konzept *P,* dargestellt durch eine Possibilitätsfunktion. Für das unscharfe Konzept "warm" bedeutet dies z.B. die Möglichkeit, daß die Raumtemperatur Werte zwischen *18°C* und *26°C* annehmen kann (Abb. 4).

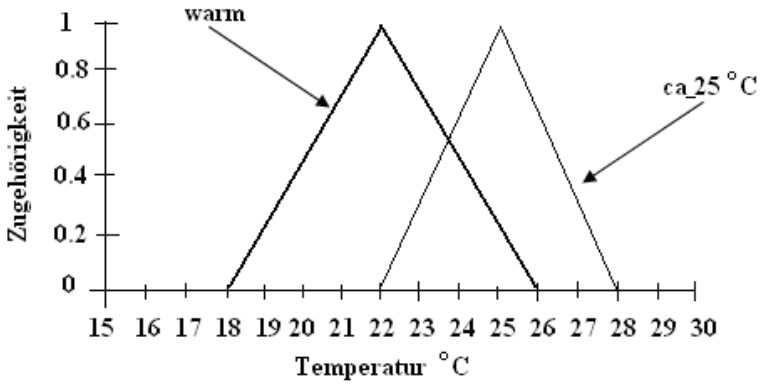


Abbildung 4: Unscharfe Konzepte "warm" und "ca 25 °C

Ist *P* als Datum gegeben, und soll es mit einem ähnlichen Konzept *D* als Muster verglichen werden, z.B. dem Konzept "ca_25°C", so will man wissen, zu welchem Grad es möglich ist, daß ein Raum mit einer Temperatur von "ca_25°C", Werte annehmen kann, die durch die Temperaturen "warm" beschrieben sind. *Des* größtmögliche Wert ist das Möglichkeitsmaß, gegeben

durch *m(P\D).* Dieses optimistische Maß bildet die obere Schranke der Übereinstimmung der beiden Konzepte.

Definition: *(P\D)*

Sind *Wp* und ###^) Zugehörigkeitsfunktionen über ### für *P* und *D,* so berechnet sich dieses Maß durch ([13]):

$$###(PID = \sup_{x \in \Omega} \min(### p(x), ### D(x)).$$

Die Notwendigkeit .ßder auch Sicherheit, mit der Werte, die dem gegebenen Konzept *D* entsprechen, dem zum testenden Konzept *P* angehören, wird durch folgende Definition bestimmt.

Definition: N(PID)

Sind $###_p$ und $###_D$ Zugetlörigkeitsfimlitionen. über ### für *P* und D, so berechnet sich dieses Maß durch ([13]);

N(P\D) =1- ###('###PID)

Hier wird die Möglichkeit betrachtet, daß *P* nicht eintritt, wenn *D* gegeben ist. Möglichkeit und Notwendigkeit verhalten sich nach dieser Definition dual zueinander. Dies bedeutet, daß sich die Notwendigkeit eines Ereignisses aus der Unmöglichkeit des komplementären Ereignisses bestimmen läßt. *N(P\D)* betrachtet also die Schnittmenge -P $\cap$ *D* [12].

Beispiel: Die Berechnung der Übereinstimmung der in der Abb. 4 beschriebenen Konzepte wird durch die Abb. 5 veranschaulicht.

Π(warm\ca.25°C) $\approx$ *0,57* und

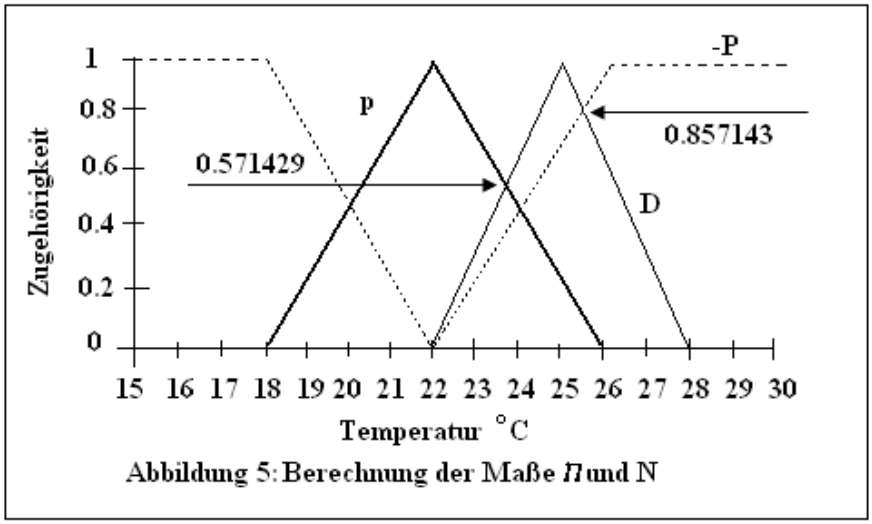


Abbildung 5: Berechnung der Maße Π und N

N(warmi\ca.25°C) = l- ΠI(nicht warm\ca.25°C) =1 - *0,86 == 0,14*

Das Ergebnis wird in Form eines Support-Intervalls *[0.14, 0.57]* notiert.)ieses Support-Intervall wird in [14] als *possibilistic support pair* bezeichnet. -olgende Eigenschaften sind bei diesen Maßen festzustellen [12].

Sind *P* und *D* scharfe Konzepte, gilt *N(P,D) = l ### P ###* D, d.h. die Bedeutung von *D* ist in der Bedeutung von *P* enthalten

Ist *D* ein scharfer Wert x_0 und soll die Übereinstimmung eines unscharfen Wertes *P* zu x_0 bestimmt werden, gilt *###m(p\x$_0$) = N(P\x$_0$) =* ### P(x)

###Im umgekehrten Fall gilt *m(x$_0$\P) =* ###p(x) und N*(x$_0$\x$_0$) = 0*

Es gilt das Sicherheitsmaß *###(P\D) =###(D\P)*

Für das Notwendigkeitsmaß gilt *N(P\D) ### N(D\P),* weil *N(D\P)* ein Maß für *P ### D* ist und im allgemeinen ###P ###D ### ###D###P gilt.

Während ein scharfes Konzept A mit sich selbst stets zu *100%* Übereinstimmt, gilt dies im unscharfen Fall nicht vollständig. Zwar gilt ### *(A\A) = l* aber *N(A\A) ### 0.5*. Hier wird oftmals vergessen, daß A aur unscharf ist und der genaue Wert, den A annimmt, nicht vollständig bekannt ist. Diese Unsicherheit muß beim pessimistischen Vergleich mit berücksichtigt werden. Fordert man *N(A\A) = l, so* gilt 1- ###(###AIA) = 1, also ###A ### A == ###. Dies gilt nur für scharfe Konzepte.

Esgilt###(P\D),)###N(P\D).

Erweiterter Resolutionsprozeß

Die semantische Unifikation liefert als Ergebnis einer erfolgreichen Unifikationen eine Substitution und einen Support-Intervall. Die Grundidee der erweiterten Resolution liegt darin, die klassische SLD-Resolution um entsprechende Konzepte zu erweitern.

Die Verarbeitung eines füzzy-logischen Programms läßt sich in 4 Schritten beschreiben:

1. Berechne eine Ableitung für ein Ziel ohne Berücksichtigung der Support-Intervalle analog zur SLD-Ableitung.
2. Berechne das Ergebnis für die erfolgreiche Ableitung durch Berücksichtigung der Support-Intervalle
3. Wiederhole Schritt l und 2 für alle möglichen Ableitungspfade;.
4. Bewerte bzw.' kombiniere die Ergebnisse aus den unterschiedlichen Ableitungspfaden nach einer bestimmten Methode, die weiter unten beschrieben wird.

Die 'Bewertung bzw. Kombination aus unterschiedlichen Ableitungspfaden kann nach unterschiedlichen Methoden realisiert werden, wie z.B. die allgemeine Zuweisungsmethode (general assignment method) nach [11], Max-Support-Intervall Methode nach [12] oder die modifizierte Dempster- Shafer Kombinationsmethode [141. Die Wahl der Methoden ist vom Anwendungsgebiet abhängig und ist daher nicht Bestandteil des Kalküls.

Schlußbemerkung

Die semantische Unifikation ist ein mögliches Verfahren zur integration von Fuzzy-Mengen in ein Prolog-System. Ein Nachteil der semantischen Unifikation ist aber ihre Nichtkommutativität. Es gilt in der Regel:

$$P \approx_u D = [N(P/D), \prod(P/D)] \neq [N(D/P), \prod(D/P)] = P \approx_u D$$

Der Beweis der Nichtkommutativität ist durch ein (Gegen-) Beispiel erbracht. Die semantische Unifikation der Fuzzy-Terme "warm" und "ca_25°C" liefert einen Support-Intervall von *[0.14, 0.57],* hingegen die Unifikation von "ca_25°C" mit "warm" einen Support-Intervall von *[0, 0.57]* als Ergebnis hat. Die Nichtkommutativität der semantischen Unifikation bedeutet, daß der semantische Unifikator nicht eindeutig ist. Die Beweisführung in der SLD-Resolution ist gerichtet, d.h. es wird immer versucht ein Ziel mit einer Programmklausel zu reduzieren. Daher kann die Eigenschaft der Kommutativität in einem Prolog-System vernächläßigt werden. Die Nichtkommutativität muß bei der Entwicklung von Applikationen berücksic-htigt werden.

Um eine Strategie zur Verarbeitung von gewichteten Klauseln anzugeben, ist die Modifizierung der SLD-Resolution eine Möglichkeit. Die Entscheidung die SLD-Resolution zu modifizieren beruht darauf, daß diese

Strategie in den meisten Prolog-Systemen implementiert ist. Weiterhin können zwei heterogene Suchstrategien im selben Prolog-System zu Problemen rühren. Beim Support-Logik-Kalkül müssen alle Lösungen einer Anfrage (mittels der SLD-Ableitung) gefunden und in einer geordneten Weise ausgegeben werden. Ein Problem ergibt sich im Falle von unendlich vielen Ergebnissen oder unendlich langen Ableitungen, bei denen die Inferenzmaschine nicht terminiert.

Literaturverzeichnis

[1] N.H.C. Thuy, P. Schnupp. *Wissensverarbeitung und Experten-Systeme.* R, Oldenbourg Verlag. München, 1989

[2] F. Puppe. *Einführung in Expertensysteme.* Springer- Verlag. 2.Aufl. Berlin. 1991

[3] A. Colmerauer, et. al. *Ün Systeme de Communication Homme-Machi'ne en Franfais.* Research Report, üniversite Aix-Marseille, 1973

[4] R. Kowalski. *Predicate logic äs programming language.* Information Processing. Seite 569-574. North-Holland, 1974

[5] L. A. Zadeh. *Fuzzy Sets.* Information and Control, 8, Seite 338-353. 1965

[6} A. Dempster. *Upper änd lower probabilities induced by a multivalued mapping.* Ann. Math. Stat, 38, Seite 325-339. 1967

[7] G. Shafer. A *mathematical theory ofevidence.* Princeton University Pess. Princeton, 1976

[8] J.A. Robinson. A *machine-oriented logic based on the resolution principle.* Journal ofthe ACM, 12(1). Seite 23-41. 1965

[9] J.F. Baldwin. A *new approach to approximate reasoning using ßizzy logic.* Fuzzy Sets and Sytems, 2. Seite 193-219. Bristol, 1979

[10] D, Dubois, H.Prade. *Possibility Theory. An Approach to Computerized Processing of Üncertainty.* Plenum Press.New York, 1988

[11] J.F. Baldwin. *Support Logic Programming.* In: AJones, etal., (Eds.)> Fuzzy Sets Theory and Applications. Reidel Dordrecht. Boston, 1986

[12] C, Geiger. *ConFuP- Concept of a parallel logic programming language withßizzy semantics.* Diploma thesis. University of Paderborn, 1993

[13] M. Cayrol, H. Farreny, H. Prade. *Fuzzy Pattern Matching.* In:emetes, Vol 11. Seite 103-166. Thaies Publicaüons Ltd. 1982