Masterthesis

Regelbasierte Mustererkennung mit Fuzzy Logik

Frank Volkmer
Friedrich-Ebert-Str. 132
48153 Münster
frank.volkmer@fh-muenster.de

13. Oktober 2008

Betreuer:
Prof. Dr. rer. nat. Nikolaus Wulff
Dipl.-Ing. Christian Koers

Für:

Erwin Volkmer

Anni Volkmer

Maria Tschage

Danksagung

An dieser Stelle möchte ich die Gelegenheit nutzen, mich bei allen zu bedanken, die mich während meines gesamten Studiums im Allgemeinen und bei der Erstellung dieser Arbeit im Speziellen in vielfacher Art und Weise unterstützt haben.

Herrn Professor Dr. Nikolaus Wulff danke ich für die Unterstützung und die theoretischen Hinweise, die sich auch aus dem JEFIS-Projekt ergeben haben, ohne die die Realisierung dieser Arbeit nicht möglich gewesen wären.

Der Firma Prognost möchte ich danken, dass es möglich war dieses spannende Projekt in den letzten eineinhalb Jahren zu bearbeiten und erfolgreich anzuschließen.

Zudem gilt mein Dank Herrn Dipl. System Wiss. Thorsten Bojer, der die Implementierung des PnPatFS tatkräftig unterstützt hat und wertvolle Hinweise zur Arbeit gegeben hat.

Julia Wolthaus und Timo Reilmann danke ich für die Unterstützung bei der Ausmerzung der vielen kleinen Fehler die sich bei der Erstellung dieser Thesis eingeschlichen haben.

Ein besonderer Dank gilt meiner Schwester Corinna, die mich in der Großzeit meines Studiums ertragen hat und ohne die ich sicherlich manchmal im Chaos versunken wäre.

Eidesstaatliche Erklärung

Hiermit versichere ich, die vorliegende Arbeit selbstständig und unter ausschließlicher Verwendung der angegebenen Literatur und Hilfsmittel erstellt zu haben.

Die Arbeit wurde bisher in gleicher oder ähnlicher Form keiner anderen Prüfungsbehörde vorgelegt und auch nicht veröffentlicht.

Münster, den 13. Oktober 2008

Frank Volkmer

Inhaltsverzeichnis

1 Einleitung

Diese Masterthesis beschreibt die Entwicklung eines regelbasierten Fuzzy Systems zur Erkennung von Schadensfällen an Kolbenverdichtern und -pumpen. Dieses System wurde im Rahmen eines eineinhalbjährigen Projektes, in Zusammenarbeit mit dem Labor für Informatik der Fachhochschule Münster, bei der Firma Prognost Systems GmbH in Rheine entwickelt.

Die bisherige Mustererkennung der Firma Prognost basiert noch auf einer Musterdatenbank und einem Vergleich dieser Muster mit dem Onlinesignal der Maschine. Mit der Hilfe der Fuzzy Logik wird versucht, die Grenze zwischen den Aussagen "Schaden vorhanden" und "Schaden nicht vorhanden" mit der Unschärfe der Fuzzy Logik aufzuweichen.

Das neue Fuzzy System soll durch diese größrere Bandbreite an Information noch genauere Aussagen über den aktuellen Zustand der Maschine liefern, was den Betrieb der Maschine sicherer macht. Ebenso können diese Informationen helfen, Schäden an Mensch, Maschine und Umwelt zu verhindern, die Wartung feiner auf den Zustand der Maschine anzupassen und den Wirkungsgrad der Maschine und des zu bearbeitenden Prozesses zu erhöhen.

Zunächst wird in Kapitel 2 eine Einführung in die Fuzzy Logik gegeben. Diese Einführung umfasst die Definition von Fuzzy Sets und Fuzzy Partitionen sowie die nötigen Fuzzy Operatoren, um diese zu verknüpfen. Am Ende des Kapitels wird auf Fuzzy Systeme eingegangen, welche mit unscharfen Werten und speziellen Fuzzy Regeln rechnen um neue Informationen zu gewinnen.

Das darauffolgende Kapitel befasst sich mit einer speziellen Klasse von Fuzzy Operatoren, den OWA-Operatoren. Einige besondere Ausprägungen werden vorgestellt und deren mathematischen Eigenschaften erläutert. Diese Operatoren werden später genauer untersucht, um bestimmte Eigenschaften des Fuzzy Systems zu optimieren, oder um andere Anforderungsprofile an die regelbasierte Schadenserkennung zu erfüllen.

Danach wird in Kapitel 4 die Firma Prognost Systems GmbH und das System zur Online- Überwachung und Maschinendiagnose PROGNOST®-NT vorgestellt. Dies bein-

haltet eine Übersicht über das Vorgehen bei der Maschinenüberwachung und eine Übersicht über die bereits darauf aufbauende Mustererkennung zur Diagnose von Schäden.

In Kapitel 5 werden die mathematischen Eigenschaften und die Arbeitsweise des neu entwickelten Fuzzy Systems anhand der Vorgaben aus dem vorherigen Kapitel erläutert.

Im folgenden beschreibt das Kapitel PnPatFS das entwickelte Fuzzy System, dessen Architektur und dessen Integration in das PROGNOST®-NT System.

In Kapitel 7 wird dann die Toleranz von Fuzzy Regeln untersucht, indem einzelne Prämissen aus der Regel entfernt werden. Die Auswertung dieser gekürzten Fuzzy Regeln wird anschließend auf Ähnlichkeit zum Ergebnis der originalen Fuzzy Regel überprüft. Verschiedene Fuzzy Operatoren werden auf kompensatorische Eigenschaften bezüglich der Kürzung einzelner Prämissen hin untersucht.

Ein abschließendes Kapitel gibt dann eine Übersicht, welche Probleme bei der Entwicklung des Fuzzy Systems aufgetreten sind und gibt einen Ausblick über mögliche zukünftige Weiterentwicklungen des Projektes.

2 Fuzzy Logik

Dieses Kapitel gibt einen kurzen Einblick in die wichtigsten Begriffe der Fuzzy Logik und soll helfen, den Aufbau und die grundlegenden Prinzipien der Fuzzy Systeme zu verstehen. Fuzzy Systeme nutzen, ähnlich wie Systeme, die auf klassischer Logik basieren, Regeln, um aus unscharfen Informationen neue Informationen zu gewinnen. Diese neuen Informationen können unscharf sein und wieder als Input für ein weiteres Fuzzy System dienen oder gegebenenfalls in scharfe Informationen umgewandelt werden.

2.1 Fuzzy Sets

In der klassischen Logik bestimmt eine charakteristische Funktion $\mu_A : X \longrightarrow \{0, 1\}$, ob ein Element $x \in X$ zu einer Teilmenge $A \subset X$ gehört oder nicht. $\mu_A(x) = 1$ bedeutet dabei, dass $x \in A$ ist, $\mu_A(x) = 0$ das Gegenteil.

Lotfi Zadeh verallgemeinerte[2] 1965 diese Definition der Zugehörigkeit eines Elementes von klassischen Mengen[1] auf eine Funktion

$$\mu_{\mathcal{A}} : X \longrightarrow [0, 1] \ , \tag{2.1}$$

die jedem Element $x \in X$ eine reelle Zahl aus dem Intervall $[0, 1]$ zuordnet. $\mu_{\mathcal{A}}(x) = 0$ steht dabei für keine Zugehörigkeit und $\mu_{\mathcal{A}}(x) = 1$ für volle Zugehörigkeit. Man nennt $\mu_{\mathcal{A}}(x)$ den "Grad", zu dem das Element x zur beschriebenen unscharfen Menge $\mathcal{A}$ gehört. Zunächst betrachtete Zadeh eine Fuzzy Menge $\mathcal{A}$ als eine Menge von geordneten Paaren

$$\mathcal{A} = \{(x, \mu_{\mathcal{A}}(x)) \mid x \in X\} \tag{2.2}$$

Wenn X eine endliche Menge von n Elementen ist,

$$\mathcal{A} = \{(x_1, \mu_1), (x_2, \mu_2), \ldots, (x_n, \mu_n)\} \tag{2.3}$$

[1] engl. Sets

so führte Zadeh die folgende Schreibweise ein:

$$\mathcal{A} = \mu_1/x_1 + \mu_2/x_2 + \cdots + \mu_n/x_n = \sum_{i=1}^{n} \mu_i/x_i \ . \tag{2.4}$$

Dabei ist darauf zu achten, dass dies nur eine Schreibweise ist, welche formal mathematisch nicht ganz korrekt ist. Weder findet eine Division von μ_i durch x_i statt noch ist das Ergebnis der Summation eine reelle Zahl. Nur deshalb ist auch eine Erweiterung der Schreibweise auf unendliche Grundmengen, beispielsweise die reellen Zahlen, möglich:

$$\mathcal{A} = \int \mu_{\mathcal{A}}(x)/x \ . \tag{2.5}$$

Auch hier ist darauf zu achten, dass es sich um eine von Zadeh neu eingeführte Schreibweise handelt, die nichts mit der Integration im eigentlichen Sinne zu tun hat.

Die Fuzzy Logik erweitert also die klassische Logik von der Menge $\{0, 1\}$ oder $\{wahr, falsch\}$ auf das Einheitsintervall $[0, 1]$. Dadurch ist es möglich, auch Zwischenzustände zwischen $wahr$ und $falsch$ zu beschreiben. Die Fuzzy Logik ist also eine Verallgemeinerung der klassischen Logik, da sich mit passenden Fuzzy Sets die klassische Logik abbilden lässt. Abbildung 2.1 verdeutlicht den Unterschied zwischen einer klassischen Menge und einer Fuzzy Menge.

Jedoch gelten für Fuzzy Sets nicht alle Regeln der klassichen Logik wie zum Beispiel das Gesetz des ausgeschlossenen Dritten, welches in der klassischen Logik besagt, dass sich für eine beliebige Aussage P die Aussage $P \vee \neg P$ herleiten lässt. Für alle $x \in \mathcal{A}$ für die $\mu_{\mathcal{A}} \in]0, 1[$ gilt, ist x nicht nur zu einem gewissen Grad in $\mathcal{A}$, sondern auch zu einem gewissen Grad in dessen Komplement $\overline{\mathcal{A}}$ enthalten.

An die Grundmenge X werden in der Literatur in der Regel nur sehr wenige Anforderungen gestellt. Diese Arbeit beschränkt sich darauf X, also das Urbild von $\mu(x)$, auf die reellen Zahlen $\mathbb{R}$ einzuschränken. Sämtliche Beispiele in dieser Masterthesis arbeiten mit reellen Werten, die Messwerte echter Sensoren mit beschränkten Messbereichen repräsentieren. Deshalb schränken wir die hier betrachteten Fuzzy Partitionen und ihre enthaltenen Fuzzy Sets auf ein reelles Intervall $[a, b]$ mit $-\infty < a < b < \infty$ ein.

Dies stellt sicher, dass Vereinigungen und Durchschnitte von Fuzzy Sets ebenfalls innerhalb eines reellen Intervalls liegen mit der Einschränkung, dass bei den Fuzzy Operationen nur eine bestimmte Art von Normen benutzt wird, siehe dazu Kapitel 2.6. Mit

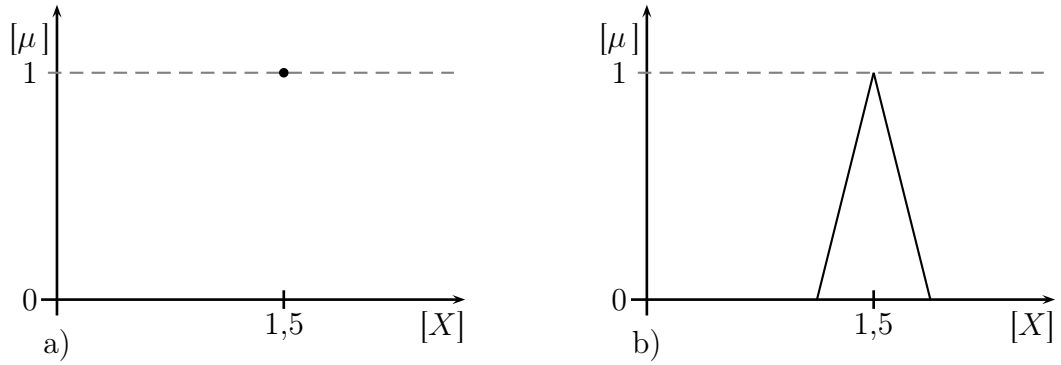

Abbildung 2.1: a) Die scharfe Punktmenge 1,5 und b) die unscharfe Fuzzy Menge "ungefähr 1,5"

der Einschränkung auf diese Operationen sind deren Resultate endlich integrierbar und dies erleichtert das Defuzzyfizieren mit bestimmten Methoden.

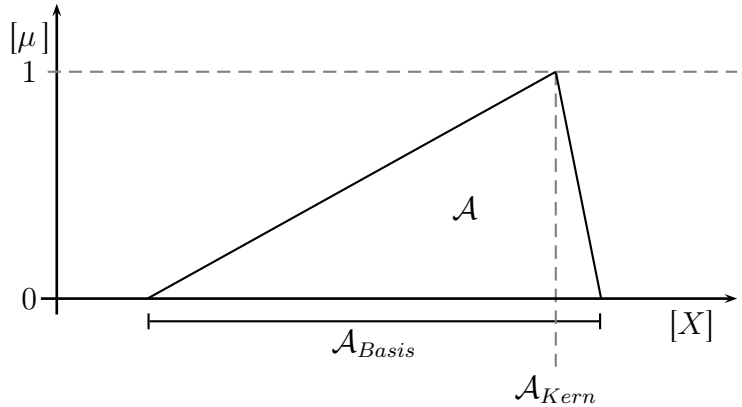

Abbildung 2.2: Basis $\mathcal{A}_{Basis}$ und Kern $\mathcal{A}_{Kern}$ des Fuzzy Sets $\mathcal{A}$.

Die Basis eines Fuzzy Sets $\mathcal{A}$ sei in diesem Kontext die Teilmenge $\mathcal{A}_{Basis} \subset X$ für die gilt $\mu_A(x) > 0$, siehe Abbildung 2.2. Damit lässt sich die Basis in ein Intervall $[a, b]$ einschließen, so dass aus $\forall x, \mu_{\mathcal{A}}(x) > 0$ folgt $x \in [a, b]$. Der Kern[2] eines Fuzzy Sets sei die Teilmenge von $\mathcal{A}_{Kern} \subset \mathcal{A}$, für die gilt $\mu_{\mathcal{A}}(x) = 1$.

[2]Weder muss ein Fuzzy Set einen Kern besitzen noch muss dieser Kern zwingend aus nur einem Element bestehen.

2.1.1 Fuzzy Alpha Schnitt

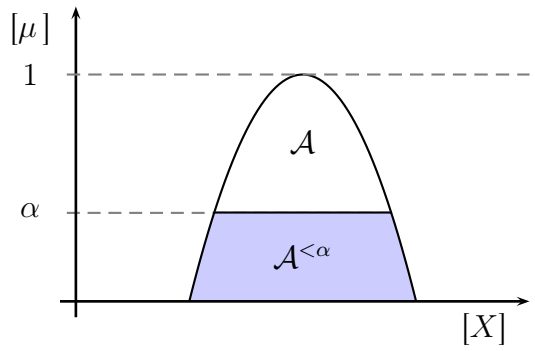

Abbildung 2.3: Fuzzy Alpha Schnitt mit $\alpha = 0,39$.

Ein Alpha Schnitt eines Fuzzy Sets beschreibt die Teilmenge $\mathcal{A}^{<\alpha} \subset \mathcal{A}$ für die, mit einem $\alpha \in [0,1]$, gilt:

$$\mathcal{A}^{<\alpha} = \{(x, \mu_{\mathcal{A}^{<\alpha}}(x)) \mid x \in \mathcal{A} \wedge \mu_{\mathcal{A}^{<\alpha}}(x) = min(\alpha, \mu_{\mathcal{A}}(x))\} \ . \tag{2.6}$$

Die neue Fuzzy Menge $\mathcal{A}^{<\alpha}$ lässt sich durch die neue Zugehörigkeitsfunktion $\mu_{\mathcal{A}^{<\alpha}}(x)$ beschreiben.

Fuzzy Alpha Schnitte werden benutzt, um die Zugehörigkeit eines scharfen Wertes x zu einem Fuzzy Set $\mathcal{A}$ zu visualisieren. Die Fuzzy Sets, die im Ausgang eines Fuzzy Systems erstellt werden, sind oft Kompositionen einzelner Fuzzy Alpha Schnitte, deren α-Werte von der Fuzzy Inferenzeinheit bestimmt werden. Abbildung 2.3 zeigt den Fuzzy Alpha Schnitt eines Fuzzy Sets.

2.2 Fuzzy Partitionen

Eine Fuzzy Partition $\mathbf{P}$ besteht aus einer Menge von Fuzzy Sets. Fuzzy Partitionen werden dazu eingesetzt, um bestimmte Größen in Fuzzy Systemen zu repräsentieren. Dazu wird das Konzept der linguistischen Variablen benutzt, welche linguistische Werte annehmen.

Ein Beispiel für eine solche linguistische Variable könnte die Temperatur eines Raumes sein. Die dazu passenden linguistischen Werte, die diese Variable annehmen kann, könnten *zu kalt*, *angenehm* und *zu warm* sein. Jeder linguistische Wert wird durch ein Fuzzy

Set repräsentiert. Eine mögliche Fuzzy Partition, welche die Raumtemperatur mit den drei linguistischen Variablen repräsentieren könnte, ist in Abbildung 2.4 veranschaulicht. Die Modellierung der einzelnen Fuzzy Sets ist abhängig von der jeweiligen Anwendung.

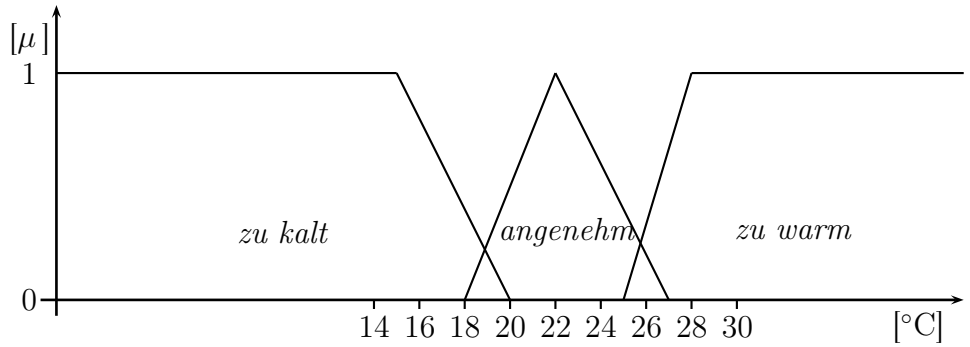

Abbildung 2.4: Mögliche Fuzzy Partition für die Raumtemperatur mit den Fuzzy Sets *zu kalt*, *angenehm* und *zu warm*.

2.3 Fuzzyfizierung

Um nun einen scharfen Wert x' in einen unscharfen Wert zu transformieren, bedient man sich der Fuzzyfizierung. Dabei wird mit Hilfe der Zugehörigkeitsfunktion $\mu(x)$ die Zugehörigkeit α bestimmt.

Die Fuzzyfizierung einer Fuzzy Partition $\mathbf{P}$ mit n Fuzzy Sets führt dazu, dass nicht nur ein α-Wert bestimmt wird, sondern, dass das Ergebnis eines solchen Vorgangs ein Vektor $\vec{\alpha}$ der Länge n ist. Dieser Vektor wird aus den Zugehörigkeiten α_i der einzelnen Fuzzy Sets $\mathcal{A}_i$ der Fuzzy Partition $\mathbf{P}$ gebildet.

In Abbildung 2.5 kann man sehen, wie die einzelnen Zugehörigkeiten durch Fuzzy Alpha Schnitte dargestellt werden und den resultierenden Vektor $\vec{\alpha}(x) = \{\alpha_1, \alpha_2, \alpha_3\}$ mit $\alpha_1 = 0{,}36$, $\alpha_2 = 0{,}75$ und $\alpha_3 = 0$ bilden.

Diese Werte können dann an eine Inferenzeinheit weiter geleitet werden, welche dann versucht, anhand von verschiedenen Fuzzy Regeln, ein Ergebnis abzuleiten.

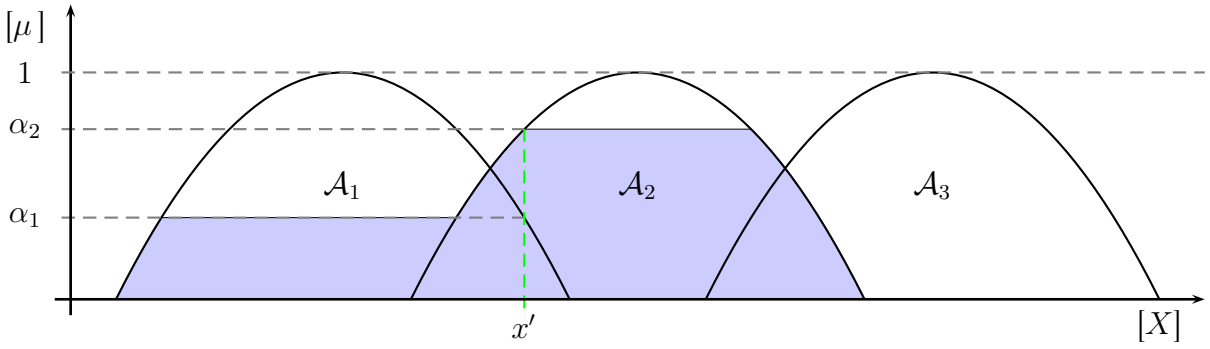

Abbildung 2.5: *Fuzzyfizierung* des Wertes x' innerhalb der Fuzzy Partition $\mathbf{P}$ bestehend aus den Fuzzy Sets $\mathcal{A}_1$, $\mathcal{A}_2$ und $\mathcal{A}_3$. Aus den Zugehörigkeitswerten wird der Alphavektor $\vec{\alpha_{\mathbf{P}}}(x')$ gebildet.

2.4 Defuzzyfizierung

Scharfe Werte lassen sich kanonisch in Fuzzy Sets einbetten, umgekehrt ist das jedoch nicht trivial. Um einem Fuzzy Set $\mathcal{B}$ wieder einen scharfen Wert x_{out} zuzuordnen, gibt es mehrere Verfahren.

Das wohl einfachste Defuzzyfizierungsverfahren ist die Anwendung des Maximum-Kriteriums. Hierbei wird ein beliebiger Punkt b_{out} der Basis von $\mathcal{B}$ ausgewählt, für den $\mu_{\mathcal{B}}(b_{out})$ einen maximalen Zugehörigkeitswert annimmt. Das Problem dabei ist, dass die Menge der Werte x für die $\mu_{\mathcal{B}}(x)$ maximal ist, nicht zwangsweise aus nur einem Element zu bestehen braucht. Wählt man nun einen Wert zufällig aus dieser Menge aus, so führt das dazu, dass ein Fuzzy Regler, dessen Output defuzzyfiziert wird, ein nichtdeterministisches Verhalten bekommt.

Das Maximum Kriterium kann dahingehend beschränkt werden, dass ein bestimmter Punkt x aus der Menge der maximalen $\mu_{\mathcal{B}}(x)$ ausgewählt wird. Zum Beispiel der kleinste, der größte oder der Mittelwert dieser beiden Werte. Auch diese Einschränkungen haben immer noch den Nachteil, dass sie gewisse Unstetigkeiten des Ergebnisses zulassen.

Wählt man zum Beispiel den Mittelpunkt der Maximumsmenge, betrachtet das Beispiel aus Abbildung 2.6 und ändert die Zugehörigkeit des Fuzzy Sets $\mathcal{A}_1$ auf einen größeren Wert als α_2, so werden alle drei eingezeichneten Werte sprunghaft in das erste Fuzzy Set $\mathcal{A}_1$ wechseln.

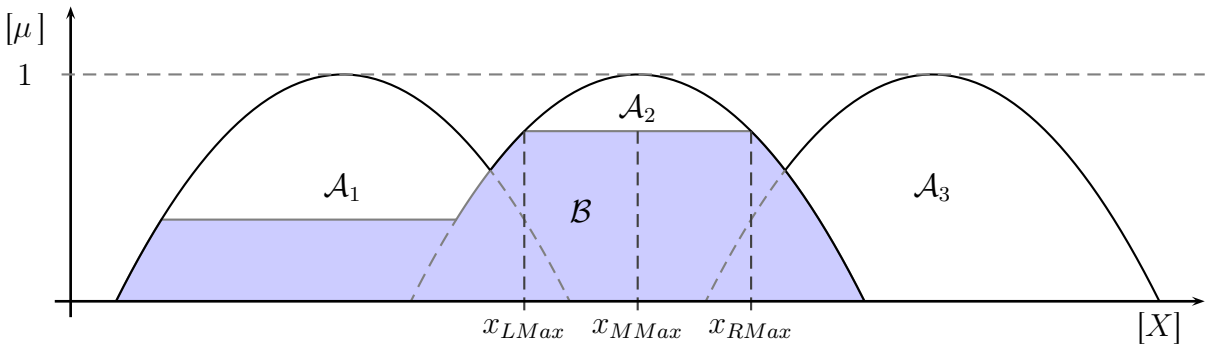

Abbildung 2.6: *Defuzzyfizierung* des Fuzzy Sets $\mathcal{B}$ gebildet aus vereinigten Fuzzy Alpha Schnitten der Fuzzy Sets $\mathcal{A}_1$ und $\mathcal{A}_2$. Die einzelnen Defuzzyfizierungswerte der linksseitigen x_{LMax}, der rechtsseitigen x_{RMax} und der gemittelten Maximumsmethode x_{MMax} sind eingezeichnet.

Eine wesentlich aufwendigere Methode ist die Defuzzyfizierung mittels des Massenschwerpunktes, in der englischen Literatur Center of Gravity Methode genannt. Diese Methode erlaubt es, dass b_{out} als Funktion der Eingangsgrößen in der Regel stetig ist.

Diese Methode setzt voraus, dass die Basis von $\mathcal{B}$ ein Intervall ist, damit die folgenden Integrale über dem zu betrachtenden Universum X bestimmbar sind:

$$x_{CoG} = \frac{\int_X x * \mu(x) \mathrm{d}x}{\int_X \mu(x) \mathrm{d}x} \tag{2.7}$$

Fasst man ein Fuzzy Set als eine Wahrscheinlichkeitsverteilung auf, so entspricht x_{CoG} dem normierten Erwartungswert $E(X)$ dieser Verteilung. Da für Wahrscheinlichkeitsverteilungen immer gilt, dass das Integral über der Grundmenge gleich 1 ist, müssen bei der Übertragung dieser Berechnungen auf Fuzzy Sets sämtliche zu berechnenden Integrale mit ihrer Fläche normiert werden.

Durch die Anwendung der verschiedenen t- und s-Normen (siehe 2.6) kann es möglich sein, dass sich $\mu_{\mathcal{B}}(x)$ nicht mehr in einer trivialen Form angeben lässt und die Integration nicht mehr oder nur noch sehr schwer analytisch lösbar ist. Dies führt dazu, dass numerische Verfahren, wie die Sehnentrapezformel, die Kepplersche Fassregel oder randomisierte Monte-Carlo-Algorithmen zur Integration eingesetzt werden müssen, die den rechnerischen Aufwand der Center of Gravity Methode erheblich erhöhen.

Um eine Gewichtung vorzunehmen, bei der die Fuzzy Sets einer Ausgangs Fuzzy Partition mit unterschiedlichen Gewichten in die Defuzzyfizierung einfließen, kann man folgende, von Kosko in [1] benutzte, Methode anwenden:

$$x_{wC} = \frac{\sum_i w_i * F_i * c_i * \mu_{\mathcal{B}_i}(x)}{\sum_i w_i * F_i * \mu_{\mathcal{B}_i}(x)} \tag{2.8}$$

Hierbei ist w_i das Gewicht des i-ten Fuzzy Sets $\mathcal{B}_i$ der Ausgangspartition, c_i dessen Center of Gravity und F_i dessen Fläche. Diese Methode hat zudem noch den Vorteil, dass sie ähnliche Ergebnisse wie die Center of Gravity Methode liefert, aber rechnerisch deutlich weniger komplex ist, da keine Integrale berechnet werden müssen.

2.5 Verschiedene Fuzzy Sets

An die Zugehörigkeitsfunktion $\mu(x)$ werden in der Regel nur sehr wenige Anforderungen gestellt. Dadurch können Fuzzy Sets die unterschiedlichsten Formen und Ausprägungen annehmen. Durch die Modellierung eines Problems mit Fuzzy Logik nimmt man systematische Ungenauigkeiten in Kauf, weshalb in den meisten Anwendungen häufig rechnerisch einfach zu handhabende Fuzzy Sets verwendet werden, um den Aufwand gering zu halten.

2.5.1 Fuzzy Polygon

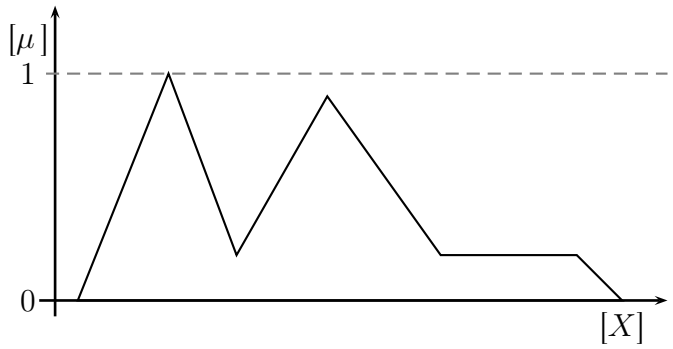

Abbildung 2.7: Fuzzy Polygon, bestehend aus einzelnen Gradenabschnitten.

Ein Fuzzy Polygon ist ein Fuzzy Set, welches durch eine Menge von Punkten beschrieben wird. Diese Punkte sind durch Geraden verbunden, die sich, zusammen mit den

Koordinaten der beiden einschließenden Punkten, als lineare Funktion darstellen lassen. Für n nach ihrer x-Komponente aufsteigend sortierte Punkte gilt:

$$\mu_{Polygon}(x) = m_i * x + b_i \quad \text{mit } 1 \leq i \leq n-1 \tag{2.9}$$

mit der Steigung

$$m_i = \frac{y_{i+1} - y_i}{x_{i+1} - x_i} \tag{2.10}$$

und dem Achsenabschnitt:

$$b_i = y_i - m_i * x_i \tag{2.11}$$

Mit einem solchen Fuzzy Polygon lassen sich die meisten Anwendungen mit ausreichender Genauigkeit und Modellierungstiefe bearbeiten, ohne dass man einen zu hohen rechnerischen Aufwand in Kauf nehmen muss.

Für einzelne Abschnitte eines Fuzzy Polygons ist es möglich, dass innerhalb eines Intervalls die Steigung $m = 0$ ist, wenn die zwei benachbarten Punkte auf der selben Höhe liegen. In diesen Abschnitten ist bei der Modellierung des Fuzzy Sets zu beachten, dass eine Änderung von x in diesem Intervall keine Änderung von $\mu_{Polygon}(x)$ nach sich zieht.

2.5.2 Fuzzy Dreieck

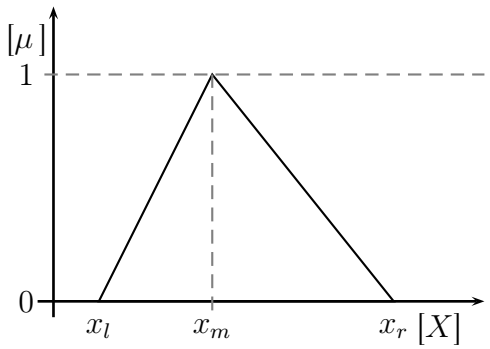

Abbildung 2.8: Fuzzy Dreieck, beschrieben durch die 3 Punkte x_l, x_m und x_r.

Fuzzy Dreiecke sind spezielle Fuzzy Polgone, die nur aus genau drei Punkten x_l, x_m und x_r bestehen. Von diesen drei Punkten hat der innere Punkt x_m die Zugehörigkeit 1 und die äußeren beiden Punkte x_l und x_r die Zugehörigkeit 0.

Gerade für Fuzzy Dreiecke ist es sehr einfach, diese zu defuzzyfizieren. Das Ergebnis jeder Form der Maximumsmethode ist immer x_m. Für symmetrische Fuzzy Dreiecke liegt auch der Center of Gravity immer bei x_m. Für unsymmetrische Fuzzy Dreiecke lässt sich der Center of Gravity mit Hilfe der Gleichung 2.7 zu

$$x_{CoG} = \frac{x_l + x_m + x_r}{3} \tag{2.12}$$

auflösen.

2.6 Fuzzy Set Operationen

Um Fuzzy Sets miteinander zu verknüpfen, entwickelte Zadeh eine Erweiterung der Basisoperatoren der klassischen Logik, der Vereinigung, des Durchschnitts und der Negation. Das Gesetz vom Ausschluß des Dritten, kann dabei jedoch nicht erhalten bleiben, siehe 2.1.

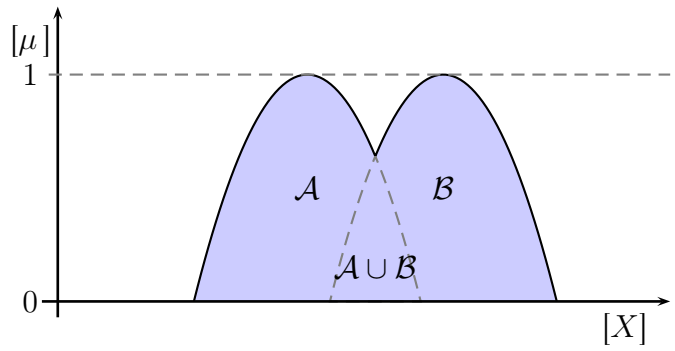

Abbildung 2.9: Fuzzy Vereinigung $\mathcal{A} \cup \mathcal{B}$ der Fuzzy Sets $\mathcal{A}$ und $\mathcal{B}$.

Für die Fuzzy Sets $\mathcal{A}, \mathcal{B}, \mathcal{C}$ und $\mathcal{D}$ über X sei $a = \mu_{\mathcal{A}}(x)$, $b = \mu_{\mathcal{B}}(x), c = \mu_{\mathcal{C}}(x)$ und $d = \mu_{\mathcal{D}}(x)$. Mit $\mu : X \to [0,1]\,; x \in X$ folgt dann $a, b, c, d \in [0,1]$. Eine t-Norm ist eine Funktion

$$t : [0,1] \times [0,1] \to [0,1] \tag{2.13}$$

die den folgenden fünf Eigenschaften genügt:

$$(1) \quad t(0,a) = 0 \qquad \text{Nullelment} \tag{2.14}$$

$$(2) \quad t(a,1) = a \qquad \text{Neutrales Element} \tag{2.15}$$

$$(3) \quad a \leq b \wedge c \leq d \Rightarrow t(a,c) \leq t(b,d) \qquad \text{Monotonie} \tag{2.16}$$

$$(4) \quad t(a,b) = t(b,a) \qquad \text{Kommutativität} \tag{2.17}$$

$$(5) \quad t(a,t(b,c)) = t(t(a,b),c). \qquad \text{Assoziativität.} \tag{2.18}$$

Eine s-Norm, auch t-Conorm genannt, ist eine Funktion

$$s : [0,1] \times [0,1] \rightarrow [0,1] \tag{2.19}$$

mit

$$(1) \quad s(a,1) = 1 \qquad \text{Nullelement} \tag{2.20}$$

$$(2) \quad s(0,a) = a \qquad \text{Neutrales Element} \tag{2.21}$$

$$(3) \quad a < b \wedge c < d \Rightarrow s(a,c) < s(b,d) \qquad \text{Monotonie} \tag{2.22}$$

$$(4) \quad s(a,b) = s(b,a) \qquad \text{Kommutativität} \tag{2.23}$$

$$(5) \quad s(a,s(b,c)) = s(s(a,b),c) \qquad \text{Assoziativität.} \tag{2.24}$$

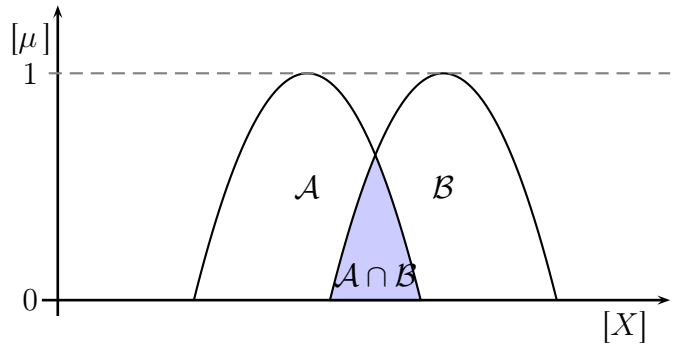

Abbildung 2.10: Fuzzy Durchschnitt $\mathcal{A} \cap \mathcal{B}$ der Fuzzy Sets $\mathcal{A}$ und $\mathcal{B}$.

Durch die Assoziativität der t- und s-Norm lassen sich diese auch auf längere Argumentlisten anwenden, indem man die Liste rekursiv abarbeitet.

Die Abbildungen 2.9 und 2.10 verdeutlichen die Anwendung einer t- und einer s-Norm als Vereinigungs- beziehungsweise als Durchschnittsoperator für Fuzzy Mengen. Die Zugehörigkeitsfunktion der neuen vereinigten oder durchschnittenen Fuzzy Menge lässt sich durch Anwenden der entsprechenden Norm angeben:

$$\mu_{\mathcal{A}\cup\mathcal{B}}(x) = s(\mu_{\mathcal{A}}(x), \mu_{\mathcal{B}}(x)) \tag{2.25}$$

$$\mu_{\mathcal{A}\cap\mathcal{B}}(x) = t(\mu_{\mathcal{A}}(x), \mu_{\mathcal{B}}(x)) \ . \tag{2.26}$$

Die ursprünglich von Zadeh vorgeschlagenen t- und s-Normen sind nicht die einzigen, welche diese Bedingungen erfüllen. Im Laufe der Jahre hat die Erforschung der Fuzzy Logik viele verschiedene Normen hervorgebracht, die diesen Kriterien genügen (siehe Tabelle 2.1).

Name	t-Norm ($\cap$)	s-norm ($\cup$)
Zadeh	min(a, b)	max(a,b)
Algebraic	ab	$a+b-ab$
Einstein	$\frac{ab}{1+(1-a)(1-b)}$	$\frac{a+b}{1+ab}$
Drastic	$\begin{cases} \min(a,b) & \text{falls } \max(a,b)=1 \\ 0 & \text{sonst} \end{cases}$	$\begin{cases} \max(a,b) & \text{falls } \min(a,b)=0 \\ 1 & \text{sonst} \end{cases}$
Hamacher	$\begin{cases} \frac{ab}{a+b-ab} & \text{falls} a,b \neq 0 \\ 0 & \text{sonst} \end{cases}$	$\begin{cases} \frac{a+b-2ab}{1-ab} & \text{falls} a,b \neq 1 \\ 1 & \text{sonst} \end{cases}$

Tabelle 2.1: Eine kurze Übersicht über verschiedene t- und s-Normen, siehe auch [3].

Die zu einer t-Norm zugehörige s-Norm lässt sich mit Hilfe der Standardnegation und den de Morganschen Gesetzen ermitteln, über welche diese Normen miteinander verknüpft sind. Deshalb ist es möglich, die eine Norm durch die andere und umgekehrt darzustellen. Es gilt

$$s(a,b) = 1 - t(1-a, 1-b) \tag{2.27}$$

$$t(a,b) = 1 - s(1-a, 1-b). \tag{2.28}$$

Die Standardnegation bildet die Komplementmenge $\overline{\mathcal{A}}$ (siehe Abbildung 2.11) einer Fuzzy Menge $\mathcal{A}$, welche dann die folgender Zugehörigkeitsfuntkion hat:

$$\mu_{\overline{\mathcal{A}}}(x) = 1 - \mu_{\mathcal{A}}(x) \ . \tag{2.29}$$

Durch die Anwendung bestimmter Normen zur Verknüpfung von Fuzzy Sets ist es danach nicht mehr so einfach, die Zugehörigkeitsfunktion des neuen Fuzzy Sets in einer geschlossenen Form anzugeben.

Die Verknüpfung zweier Fuzzy Polygone mit der min-Norm oder der max-Norm lässt sich wieder als Fuzzy Polygon angeben. Andere Normen hingegen, wie die von Hamacher und Einstein, bilden aus Fuzzy Polygonen ein Fuzzy Set, welches aus Abschnitten gebrochenrationaler Funktionen besteht. Für diese lässt sich generisch nur sehr schwer eine geschlossene Form zur Bildung des Integrals angeben. Deshalb ist es bei solchen Normen auch einfacher, auf numerische Verfahren zur Integration zurückzugreifen, welche allerdings den Aufwand deutlich erhöhen.

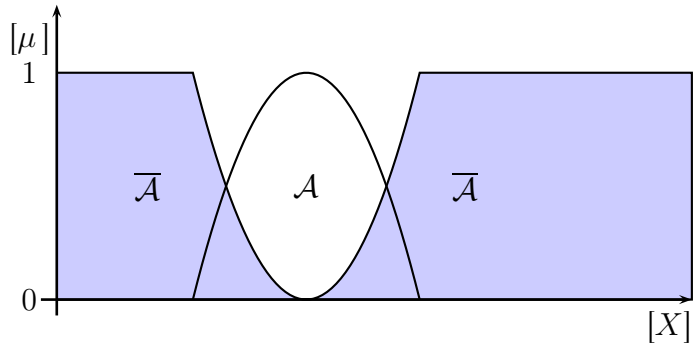

Abbildung 2.11: Fuzzy Negation $\overline{\mathcal{A}}$ des Fuzzy Sets $\mathcal{A}$.

2.7 Bildung von Fuzzy Sets

Bei der Modellierung von Fuzzy Systemen und damit auch bei der Modellierung der einzelnen Fuzzy Sets muss in der Regel auf das Wissen von Experten zurückgegriffen werden, welche sich mit dem zu modellierenden System auskennen. Dabei stellen die Experten verschiedene Fuzzy Regeln auf, um zu beschreiben, wie sich das System in bestimmten Situationen verhält. Dazu fassen sie die Eingangs- und Ausgangsgrößen des

Fuzzy Systems als linguistische Variablen auf und definieren dafür die passenden Fuzzy Sets, welche die linguistischen Werte repräsentieren.

Eine andere Möglichkeit einzelne Fuzzy Sets zu beschreiben ist es, bestimmte statistische Eigenschaften einer Systemgröße zu ermitteln. Wenn ausreichend Daten über ein bestimtes Verhalten des Systems vorliegen, so dass sich der Definitionsbereich einer Systemgröße in Bereiche aufteilen lässt zu denen man Vorkommenswahrscheinlichkeiten angeben kann, so kann beispielsweise durch eine Berechnung des Mittelwerts und dessen Standardabweichung recht einfach ein Fuzzy Set beschrieben werden.

Ebenso kann auch direkt die Verteilungsfunktion der Zufallsgröße als Zugehörigkeitsfunktion eines Fuzzy Sets dienen. Oft sind diese Verteilungsfunktionen aber nicht praktisch, da der rechnerische Aufwand bei der Integration oder der Anwendung von Fuzzy Operatoren zu hoch ist. Man kann diese Zugehörigkeitsfunktionen zum Beispiel durch approximierende Polygonzüge ersetzen und nimmt dabei einen gewissen systematischen Fehler in Kauf.

2.8 Fuzzy System

Ein Fuzzy System ist im klassischen Sinn eine Regel- oder Steuereinheit, welche Fuzzy Logik benutzt, um einen Prozess zu regeln oder zu steuern. Der schematische Aufbau eines solchen Fuzzy Systems ist in Abbildung 2.12 zu sehen.

Die Schnittstelle an einen zu steuernden Prozess bilden der Fuzzyfizierer und der Defuzzyfizierer. Der Fuzzyfizierer wandelt die scharfen Eingangsdaten in $\vec{\alpha}$-Vektoren um, indem diese festen Eingangsdaten mit dazu passenden Fuzzy Partitionen fuzzyfiziert werden. Der Defuzzyfizierer wandelt das Ergebnis der Inferenzeinheit wieder in scharfe Werte um und leitet diese an den zu steuernden Prozess weiter.

Ein Fuzzy System, auch Fuzzy Controller genannt, arbeitet ähnlich einer klassischen Inferenzeinheit intern mit Regeln. Diese werden in einer Regelbasis abgelegt. Nach E. Mamdani und S. Assilian [6] werden spezielle Fuzzy Regeln der Form

$$\textbf{Wenn } x_1 = \mathbf{A}_1 \textbf{ und } \ldots \textbf{ und } x_n = \mathbf{A}_n \textbf{ dann } y = \mathbf{B}_k \tag{2.30}$$

verwendet. Dabei sind die $x_1, \ldots, x_n$ die Eingangsvariablen dieser Fuzzy Regel und y ist die Ausgangsvariable. Die $\mathbf{A}_i$ sind einzelne Fuzzy Sets der jeweiligen zur Eingangsvariablen x_i gehörenden Fuzzy Patitionen und $\mathbf{B}_k$ ist ein Fuzzy Set der Ausgangspartition.

$x_i = \mathbf{A}_i$ bezeichnet man als eine Prämisse der Fuzzy Regel. Die Prämisse wird ausgewertet, indem der Erfüllungsgrad $\alpha_{\mathbf{A}_i}$ des Fuzzy Sets $\mathbf{A}_i$ mittels Fuzzyfizierung von x_i berechnet wird. Die eigentliche Mamdani-Inferenz besteht nun darin, die $\alpha_{\mathbf{A}_i}$ der Prämissen über das Minimum zu verknüpfen und daraus die Zugehörigkeit des Fuzzy Sets $\mathbf{B}_k$ für die Ausgangspartition zu bestimmen:

$$\alpha_{\mathbf{B}_k} = min_i(\alpha_{\mathbf{A}_i}) \ . \tag{2.31}$$

Dann werden alle Fuzzy Sets der Ausgangspartiton $\mathbf{B}$ an den Defuzzfizierer geschickt, welcher diese erst vereinigt und dann defuzzyfiziert.

Es genügt, die verwendeten Fuzzy Regeln auf eine Konklusion zu beschränken, um die Übersichtlichkeit der Regelbasis zu verbessern. Dies ist keine echte Einschränkung, da sich jede Regel mit n unterschiedlichen Konklusionen in n unterschiedliche Regeln mit den selben Prämissen und unterschiedlichen Konklusionen zerlegen lässt.

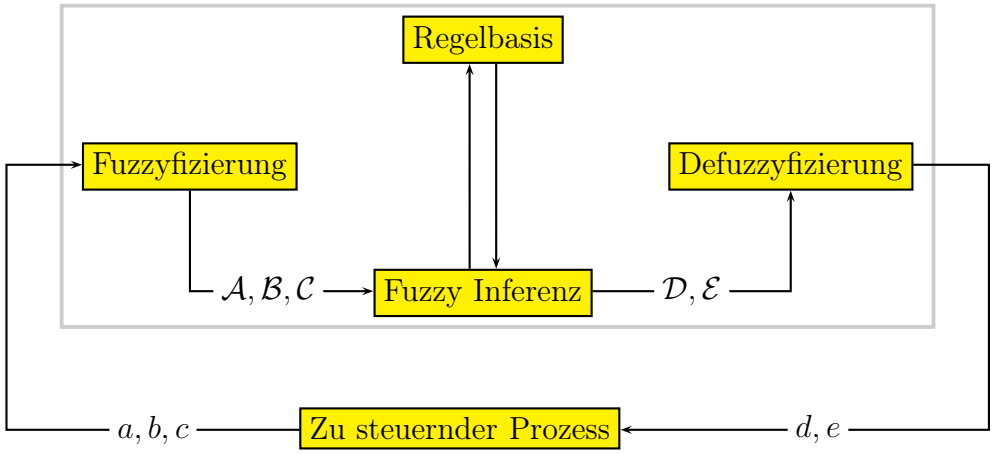

Abbildung 2.12: Schematischer Aufbau eines Fuzzy Systems, welches einen externen Prozess steuert.

3 Fuzzy Operatoren

Fuzzy Operatoren erweitern das Konzept der zweiwertigen t- und s-Normen auf n-stellige Funktionen. Da Fuzzy Normen assoziativ sind, lassen sie sich auch rekursiv auf Argumentlisten anwenden, Fuzzy Operatoren sind in der Regel jedoch nicht mehr assoziativ.

In diesem Kapitel sollen nun einige unterschiedliche Operatorfamilien und deren Eigenschaften vorgestellt werden.

3.1 OWA-Operatoren

Yager beschrieb in [5] die OWA[1]-Operatoren. Ein OWA-Operator der Dimension n ist eine Abbildung

$$F : \mathbb{R}^n \longrightarrow \mathbb{R} \tag{3.1}$$

die einen dazugehörigen Gewichtsvektor $\vec{w}$ mit $w_j \in [0, 1]$ besitzt. Für diesen Gewichtsvektor gilt zudem die folgende Normierung

$$\sum_{i=1}^{n} w_i = 1 \ . \tag{3.2}$$

Ein Vektor $\vec{x}$ der Dimension n wird dann folgendermaßen zusammengefasst:

$$F(x_1, x_2, \ldots, x_n) = \sum_{i=1}^{n} w_i * y_i \tag{3.3}$$

Hierbei ist y_i das i-te Element der absteigend nach ihrer Größe sortiereten x-Elemente.

Je nachdem wie $\vec{w}$ gewählt wird, lassen sich mit einem OWA-Operator viele unterschiedliche Operatoren abbilden. Für $\vec{w^*} = (1, 0, \ldots, 0)$ ist $F(\vec{x}) = max(x_1, x_2, \ldots, x_n)$, für $\vec{w_*} = (0, \ldots, 0, 1)$ ist $F(\vec{x}) = min(x_1, x_2, \ldots, x_n)$ und für $\vec{w_{avg}} = (\frac{1}{n}, \frac{1}{n}, \ldots, \frac{1}{n})$ ist $F(\vec{x}) = mean(x_1, x_2, \ldots, x_n)$, also das arithmetische Mittel von $\vec{x}$.

[1] Ordered Weighted Average

Ein OWA-Operator ist also ausschließlich über seinen Gewichtsvektor $\vec{w}$ definiert, über den auch zwei Operatoren miteinander verglichen werden können. Yager schlug dazu die folgenden zwei Maße vor.

$$H_{\vec{w}} = -\sum_{i=1}^{n} w_i \ln w_i \tag{3.4}$$

Die Streuung oder Entropie $H_{\vec{w}}$ beschreibt wie stark die Informationen der einzelnen Komponenten von $\vec{x}$ in das Endergebnis einfließen. Die Entropie eines OWA-Operators liegt im Intervall $[0, \ln n]$, wobei das Minimum angenommen wird, wenn $\exists i : w_i = 1 \Leftrightarrow H_{\vec{w}} = 0$ und das Maximum wird angenommen, wenn $w_i = \frac{1}{n} \forall i \Leftrightarrow H_{\vec{w}} = \ln n$, also wenn es sich um das arithmetische Mittel handelt.

Um die Entropie von Operatoren unterschiedlicher Dimension miteinander zu vergleichen macht es Sinn diese normalisiert anzugeben:

$$\overline{H}_{\vec{w}} = -\frac{1}{\ln n} \sum_{i=1}^{n} w_i \ln w_i \ . \tag{3.5}$$

OWA-Operatoren sind mittelnde Operatoren und können die Gesamte Bandbreite zwischen den t- und s-Normen abdecken. Für jeden mittelnden Operator $M(\vec{x})$ gilt deswegen

$$min(x_1, \ldots, x_n) \leq \ldots M(x_1, \ldots, x_n) \cdots \leq max(x_1, \ldots, x_n) \tag{3.6}$$

Um nun die Tendenz eines OWA-Operators anzugeben, bietet sich das folgende Maß an:

$$\alpha(\vec{w}) = \sum_{i=1}^{n} \frac{n-i}{n-1} w_i \ . \tag{3.7}$$

$\alpha(\vec{w})$ nennt man die *orness* eines OWA-Operators. Für w_* lässt sich zeigen, dass $\alpha(\vec{w_*})$ immer 0 ist. Ebenso ist $\alpha(\vec{w^*})$ immer gleich 1. Dies ist sofort klar, wenn man sich ins Gedächtnis ruft, dass der *max*-Operator einer s-Norm entspricht und damit einer Veroderung. Der *min*-Operator besitzt die *orness* 0 und ist damit das genaue Gegenteil einer Veroderung, nämlich eine Verundung. Analog zur *orness* eines OWA-Operators lässt sich auch dessen *andness* β durch die folgende Beziehung angeben:

$$\beta(\vec{w}) = 1 - \alpha(\vec{w}) \ . \tag{3.8}$$

Für den mittelnden Operator, welcher durch $w_{avg}^{\vec{}}$ beschrieben ist, gilt immer $\alpha(w_{avg}^{\vec{}}) = 0.5$.

3.2 Gewichtsverteilungen

Für einen OWA-Operator bieten sich mehrere mögliche Verteilungen der Gewichte an. Zum Beispiel die Potenzverteilung, bei der die Gewichte zusammen mit einem Parameter $p \in \mathbb{R}$ nach folgender Formel berechnet werden:

$$w(j) = \frac{1}{j^p} \ . \tag{3.9}$$

Über den Paramter p lässt sich die orness des Operators einstellen. Offensichtlich handelt es sich für $p = 0$ um das arithmetische Mittel.

Für eine Polynomialverteilung gilt, mit einem Parameter $p \in \mathbb{R}$, für die Gewichte

$$w(j) = q^j \ . \tag{3.10}$$

Bei beiden Verteilungen ist darauf zu achten, dass die Gewichte mit $\sum_i w_i$ zu normalisieren sind, damit diese Formel 3.2 genügen.

3.3 OWA-Operator als Quantifizierer

Mit der passenden Anordnung der Gewichte kann ein OWA-Operator als ein Quantifizierer verwendet werden, der alle x oberhalb eines Schwellwertes a stark gewichtet und alle darunter sehr schwach. Der Quantifizierer nutzt eine Grenzfunktion $Q(x)$ um den Übergang um den Schwellwert zu modelieren. Unter anderem bietet sich dafür die Sigmoidfunktion an:

$$Q(x) = \frac{1}{1 + e^{\frac{-(x-a)}{c}}} \ . \tag{3.11}$$

a ist hierbei der Quantifizierungswert und c die Transitionsbreite. Für die Gewichte gilt dann:

$$w(x) = \frac{dQ}{dx} = Q(x) * (1 - Q(x)) \ . \tag{3.12}$$

Dabei ist darauf zu achten, dass auch diese Gewichte nicht normalisiert sind und erst mit $\sum_i w_i$ normalisiert werden müssen bevor sie in dem OWA-Operator verwendet werden können.

4 Prognost

Die Prognost Systems GmbH (siehe [12]), ist eine Ausgründung der Kötter Consulting GmbH und vertreibt die Hardware und die spezialisierte Software PROGNOST®-NT , um oszillierenden Kolbenmaschinen, im Speziellen Kolbenverdichter und Pumpen, zu überwachen.

Dieses Kapitel gibt einen Überblick über die Überwachung einer Maschine und erläutert die Vorraussetzungen für eine automatische Schadenserkennung.

4.1 Prognost Systems GmbH

Das Hauptprodukt der Firma ist das PROGNOST®-NT System, welches zur Zustandsüberwachung von Kolbenmaschinen eingesetzt wird. Prognost Systems vertreibt die Software in unterschiedlichen Konfigurationen zusammen mit speziellen Sensorpaketen und eigens angefertigten Schaltschränken. In diesen arbeiten spezielle Module, welche die Sensoren auslesen und die empfangenen Daten digitalisieren, verarbeiten und dann per Netzwerk verteilen. Die Firma hat sich mit diesem System in den letzten fünfzehn Jahren zum Marktführer für die Überwachung von oszillierenden Kolbenmaschinen hervor gearbeitet.

Das PROGNOST®-NT System ist zeigt dem Maschinenführer den aktuellen Zustand der Maschine unter zur Hilfenahme einer grafischen Oberfläche detailliert an. Dazu gehören alle Signale und Analysen der Maschine sowie der Verschleißstatus einzelner Bauteile. Innerhalb dieser grafischen Oberfläche kann der Maschinenführer den gespeichtern Verlauf einzelner Analysen betrachten oder ein Bauteileregister der Maschine anzeigen lassen.

Durch diese Art der Überwachung kann eine lange Verfügbarkeit und der sichere Betrieb einer Maschine innerhalb eines Prozesses gewährleistet werden. Wenn eine genaue Übersicht über den Zustand und den Verschleiß einzelner Bauteile einer Maschine vorhanden sind, werden anstehende Wartungen oder Reparaturen besser planbar. Dies hilft, die Betriebskosten zu reduzieren.

Zusätzlich bietet die Firma mehrere Serviceverträge und -dienstleistungen an, um von PROGNOST®-NT überwachte Maschinen per Ferndiagnose regelmäßig von Experten im eigenen Haus überprüfen zu lassen. Die regelmäßige Ferndiagnose erlaubt es, die Erkenntnis über das Verhalten und mögliche Schäden an den überwachten Maschinen im eigenen Haus zu vergrößern.

4.2 Sensoren und Analysen

Bei der Installierung eines PROGNOST®-NT Systems werden verschiedene Sensoren an der zu überwachenden Maschine angebracht. Diese erfassen hochauflösend einzelne physikalische Paramter der Maschine wie den Druck innerhalb der Verdichtungskammern, die Temperaturen an den Ventilen, die Lage der Kolbenstange und verschiedene Schwingungswerte an den Zylindern und den Laufbahnen der Kolben.

Diese Daten werden dann an einen speziellen Schaltschrank der Firma Prognost in der Leitwarte des Prozesses übermittelt. Dort werden diese Daten aufbereitet und digitalisiert, um dann zusammen mit den Daten des Prozessleitsystems an den Erfassungsrechner weiter geleitet zu werden. Der Erfassungsrechner generiert aus den Informationen der Sensoren bestimmte Analysen. Diese Analysen liefern zum Beispiel durch Differenzierung, Integration oder Bildung von statistischen Werten wie dem Mittelwert oder dem Absolutwert, weitere Informationen über den Zustand der Maschine.

Die Verknüpfung der einzelnen Werte über mathematische Operatoren kann neue Analysen erzeugen, wenn diese Verknüpfungen physikalischen Formeln entsprechen, die aus den bestehenden Sensoren neue physikalische Zustandsgrößen der Maschine berechnen, die nicht von Sensoren erfasst sind.

4.3 Warnschwellen

Um eine Maschinenüberwachung zu gewährleisten, müssen die einzelnen Analysen überwacht werden. Dies kann geschehen, indem sie mit Warnschwellen versehen werden. Eine Warnschwelle teilt den Wertebereich einer Analyse in zwei Abschnitte. Wenn sich das Signal in dem Abschnitt aufhält der den normalen Betriebszustand einer Maschine beschreibt, so bezeichnet man diese Warnschwelle als unverletzt. Sollte das Signal der Analyse diesen Bereich verlassen und über die Warnschwelle hinweg in den anderen Abschnitt wechseln, so bezeichnet man diese Warnschwelle als verletzt.

Eine Verletzung einer Warnschwelle kann ein Indiz dafür sein, dass eine Maschine defekt ist. Eine solche Verletzung ist aber noch kein klarer Beweis für einen Schaden, da alle Signale der Maschine Schwankungen unterliegen, die durch äußere Einflüsse, wie beispielsweise tägliche oder saisonale Temperaturunterschiede, hervorgerufen werden.

4.3.1 Warnschwellensetzung

Da eine manuelle Einstellung oder Nachstellung der Warnschwellen im Allgemeinen zu ungenau und bei vielen Hundert Analysen viel zu aufwändig ist, werden diese durch das PROGNOST®-NT System automatisch gesetzt. Der automatischen Warnschwellensetzung müssen dazu Informationen über einen "Gutzustand", also einen schadensfreien Zustand eines Betriebszustandes für den die Warnschwellen berechnet werden sollen, zur Verfügung gestellt werden. Diese Daten müssen einen ausreichend langen Zeitraum abdecken, damit die Warnschwellensetzung möglichst genau berechnet werden kann.

Das PROGNOST®-NT System speichert sämtliche Betriebsdaten einer Maschine in einer Datenbank, damit diese Informationen zur späteren Analyse heran gezogen werden können. Die Messwerte der Analysen werden in den Trenddaten nicht in voller Auflösung abgespeichert, da dies zu aufwändig wäre und zu viel Speicherplatz verbrauchen würde. Die Trenddaten stehen in der Regel nur in einer Auflösung auf Minutenebene zur Verfügung und werden als Mittelwerte gespeichert. Zu diesen Minutenmittelwerten steht auch noch das absolute Maximum und das absolute Minimum des gemittelten Intervalls zur Verfügung.

Bei der automatischen Warnschwellensetzung gibt es zwei mögliche Verfahren. Bei dem einfacheren Verfahren wird das Maximum der Maxima und das Minimum der Minima der angenommen Trendwerte ermittelt, und um diese Werte werden dann mit einem gewissen Faktor die Warnschwellen gelegt. Dieses Verfahren hat den Nachteil, dass es die Dynamik eines Systems zwischen diesen Schranken fast nicht berücksichtigt, zudem könnten einzelne Ausreisser die Warnschwellensetzung verschlechtern, indem diese dadurch zu weit außen gesetzt werden.

Sämtliche Informationen der Sensoren und damit auch die daraus generierten Informationen der Analysen unterliegen Schwankungen. Diese sind im Allgemeinen statistischer Art, zum Beispiel durch Messungenauigkeiten oder das Rauschen in den Signalleitungen. Das komplexere Verfahren zur Berechnung der Warnschwellen setzt vorraus, dass Informationen in der Trenddatenbank vorhanden sind, in denen sich das System in einem

"Gutzustand" befindet. Nur unter dieser Vorraussetzung kann davon ausgegangen werden, dass die Schwankungen in diesen Messwerten des Datensatzes entweder Messfehler sind, oder die zu messende Größe bei der analogen Übertragung vom Sensor verrauscht wurde. Durch das Berechnen bestimmter Analysen schleichen sich systematische Fehler ein, welche starke Schwankungen in den Analyseninformationen hervor rufen können, wenn sich die individuellen Fehler der einzelnen Sensoren gegenseitig verstärken.

Nur unter dieser Annahme, dass die Trenddaten einen "Gutzustand" ohne einen Schaden beschreiben, kann davon ausgegangen werden, dass die Schwankungen der Messwerte normalverteilt sind. Für die Maxima und die Minima wird dann jeweils eine statistische Analyse durchgeführt, um die jeweiligen Mittelwerte μ_{min} und μ_{max} sowie die Standardabweichungen σ_{min} und σ_{max} zu berechnen, mit denen die Normalverteilung der Minima und Maxima beschrieben werden kann.

Die Standardabweichungen jeder Analyse werden auf einen Mindestwert von 1% des Messbereichs der Analyse überprüft und gegebenenfalls auf diesen Mindestwert korrigiert.

Die erste obere Warnschwelle berechnet sich dann aus dem Mittelwert der Maxima plus der Standardabweichung. Um die zweite obere Warnschwelle zu berechnen, wird der positive Abstand der ersten oberen Warnschwelle zum Mittelwert der Trendwerte verdoppelt und auf den Mittelwert der Trendwerte aufaddiert. Eine mögliche dritte Warnschwelle berechnet sich aus dem doppelten positiven Abstand der ersten und zweiten Warnschwelle plus der ersten Warnschwelle. Die Berechnung der unteren Warnschwellen funktioniert analog. In Abbildung 4.1 ist die Lage der einzelnen Warnschwellen und Referenzpunkte für eine Analyse X verdeutlicht.

Die Mittelwerte μ_{min} und μ_{max} bilden dabei die Referenzpunkte, die bei der dynamischen Fuzzyfizierung, siehe 5.2, benötigt werden. Diese Referenzpunkte schließen ein Intervall ein, dass man als den "Gutbereich" der Analyse betrachten kann, in dem sich das Signal im Normalfall aufhält, wenn an der Maschine keine Schäden zu verzeichnen sind.

4.4 Experten

Als Bestandteil eines komplexeren Prozesses wird eine Maschine im Allgemeinen von Experten bedient und kontrolliert. Diese kennen zum einen den Prozess und dessen

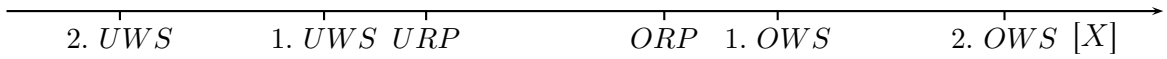

Abbildung 4.1: Anordnung der unteren und oberen Warnschwellen und deren Referenzpunkte für eine Analyse X.

Eigenarten und Besonderheiten und zum andern die unterschiedlichen Betriebszustände der Maschine.

Ein solcher Experte ist in de Regel auch mit einem umfangreichen Wissen über die möglichen Fehlerquellen und Schäden ausgestattet, die an einer solchen Maschine auftreten können. Dieses Wissen einerseits beruht auf Erfahrungswerten im Umgang mit der Maschine und stammt andererseits aus den Handbüchern und den Spezifikationen des Herstellers, welche die Laufzeiten und das Verschleißverhalten einzelner Bauteile beschreiben.

4.5 Schäden und Schadensmuster

Wenn ein Schaden an einer Maschine vorliegt, so äußert sich das in der Regel durch eine Verletzung von mehr als einer Warnschwelle. Eine solche Kombination von Warnschwellenverletzungen wird ein Muster eines Schadens genannt.

Wenn die Warnschwellen, wie oben beschrieben, mit Hilfe der statistischen Eigenschaften einer Analyse berechnet werden, so kann eine Verletzung einer Warnschwelle nur bedeuten, dass das Signal der Warnschwelle den Bereich des "Gutzustandes" deutlich verlassen hat, und sich ein solches Verlassen nicht mehr mit den natürlichen Schwankungen des Signals erklären lässt, da diese bei der Berechnung der Warnschwelle berücksichtigt wurden.

Die Schadensmuster sind unabhänging von der jeweiligen Setzung der Warnschwellen und lassen sich dadurch auf ähnliche, andere Maschinen übertragen, auch wenn dort mit anderen Leistungen und Parametern gearbeitet wird.

Einmal aufgetreten kann ein neuer Schaden von einem Experten klassifiziert werden und zusammen mit dem zugehörigen Muster in einer Musterdatenbank abgespeichert werden.

4.6 Entwicklung der Schadenserkennung in PROGNOST®-NT

Die einfachste Schadenserkennung, die zunächst in PROGNOST®-NT realisiert wurde, umfasste den direkten Vergleich einer aktuellen Konfiguration von Warnschwellenverletzungen mit abgespeicherten Schadensmustern aus einer Datenbank. Diese Datenbank ist im Laufe der Zeit mit der zunehmenden Erfahrung der Maschinenführer und Experten bei diagnostizierten und klassifizierten Schäden gewachsen.

Dieses System der Schadenserkennung hat jedoch den entscheidenden Nachteil, dass bei einem direkten Vergleich der Muster eine exakte Übereinstimmung der aktuellen Konfiguration mit einem abgespeicherten Schadensmuster vorliegen muss, um den Schaden überhaupt zu erkennen. Wenn in der aktuellen Konfiguration der Warnschwellenverletzungen eine Warnschwelle zu viel oder zu wenig vorhanden ist, kann mit diesem System kein Schaden erkannt werden.

4.6.1 LVQ

Zu einem späteren Zeitpunkt wurde deshalb der direkte Vergleich der Muster mit der aktuellen Konfiguration durch einen Vergleich mit einem neuronalen Netz abgelöst, welches in der Form eines DGRLVQ[1], siehe [7], realisiert wurde. Ein solcher lernender Vektorquantisierer ist in der Lage, die Muster aus einer Datenbank anhand ihrer Schadensklassen in Clustern zusammenzufassen und diese durch einen berechneten Repräsentanten zu beschreiben. Liegt nun eine Konfiguration von Warnschwellenverletzungen vor, so berechnet das neuronale Netz anhand einer speziellen Metrik die Distanz zu allen Repräsentanten aller Schadensklassen und liefert die Schadensklasse mit dem geringsten Abstand als wahrscheinlichsten Schaden zurück. Die Entwicklung dieses Systems der Mustererkennung in PROGNOST®-NT ist in [8] beschrieben.

Durch die Einführung eines Abstandes ist das neuronale Netz in der Lage, auch Warnschwellenverletzungen, die “ähnlich” zu einem gespeicherten Muster sind, zu erkennen. Der Ähnlichkeitsgrad oder die Nähe der aktuellen Konfiguration der Verletzungen zu ei-

[1] Dynamic Generalized Relevance Learning Vector Quantization

nem gespeicherten Muster sagt dabei etwas darüber aus, wie sicher dieses Muster erkannt wurde.

4.6.2 Probleme des LVQ

Diese Version der Mustererkennung hat, ebenso wie der vorherige direkte Mustervergleich, den Nachteil, dass sie auf Warnschwellenverletzungen angewiesen ist. Es muss also mindestens ein Signal den "Gutbereich" schon verlassen haben. Eine Früherkennung eines Schadens oder die Diagnose der Entwicklung eines Schadens ist mit diesem System alleine nicht möglich.

Die überwachten Maschinen werden in der Regel in unterschiedlichen Betriebszuständen gefahren, da oft unterschiedliche Lasten oder Durchsätze geregelt werden müssen. Bei oszillierenden Maschinen kann das bedeuten, dass sich durch unterschiedliche Betriebszustände und deren unterschiedlichen Belastungen einige Ereignisse in Abhängigkeit von der Umdrehung verschieben.

Wenn die Signale pro Umdrehung ausgewertet werden, und einige Analysen hinsichtlich der Umdrehung segmentiert sind, bedeutet das auch, dass das Verschieben bestimmter Ereignisse zu unterschiedlichen Schadensmustern führt, wenn bestimmte, den Schaden beschreibende, Warnschwellenverletzungen in unterschiedlichen Segmenten auftreten.

Dies erzeugt eine Menge neuer Muster derselben Schadensklasse. Dadurch können die Schäden in dieser Klasse nie mit voller Sicherheit erkannt werden, da durch die vielen Muster der Raum der zu erkennenden Muster wächst und der Repräsentant eine größere Menge weiter verstreuter Muster beschreiben muss.

Ein weiteres Problem dieser Art der Mustererkennung ist, dass dieser Vektorquantisierer explizit nicht vorhandene Warnschwellenverletzungen nicht prüfen kann, wenn diese benötigt werden, um bestimmte Schadensklassen voneinander zu trennen.

4.7 Expertenwissen

Aus Erfahrungen der Servicemitarbeiter der Firma Prognost wurde eine Liste verschiedener Regeln erstellt, nach denen Schäden diagnostiziert werden können. Diese Regeln sind ähnlich der Regel aus Listing 4.1 aufgebaut. Als Prämissen gehen also einzelne

```
IF
Analyse 1 ist 1. untere WS
AND
{
    Analyse 2 ist 1. obere WS
    OR
    Analyse 3 ist 1. obere WS
}
AND
Analyse 4 ist 1. obere WS
AND
{
    Analyse 5 ist 1. untere WS
    OR
    Analyse 6 ist 1. untere WS
}
AND
Analyse 7 ist 1. obere WS
THEN
Schaden
```

Listing 4.1: Expertenwissen liefert solche oder ähnliche einfache Regeln, um Schadensfälle zu beschreiben.

Analysen und eine Verletzung einer Warnschwelle dieser Analyse in die Regel ein. Die Konklusion ist dann eine Aussage über die Ausprägung des Schadens.

In Abbildung 4.2 ist die semantische Struktur der Regel noch einmal deutlich zu erkennen. Diese Regel entspricht der folgenden Formel in aussagenlogischer Schreibweise[2]:

$$\begin{aligned} Schaden = Analyse_1 \otimes (Analyse_2 \oplus Analyse_3) \otimes \\ Analyse_4 \otimes (Analyse_5 \oplus Analyse_6) \otimes Analyse_7 \end{aligned} \tag{4.1}$$

Die Informationen dieser Regel können benutzt werden, um daraus neue Muster für die Musterdatenbank zu generieren. Die Muster der Musterdatenbank zeichnen sich dadurch aus, dass es sich um so genannte *Minterme* handelt, siehe [9]. Bei diesen Termen sind alle

[2]Der Übersicht wegen wurden die zu betrachtenden Warnschwellen nicht dargestellt, da es nur die Struktur der Verknüpfungen wichtig ist.

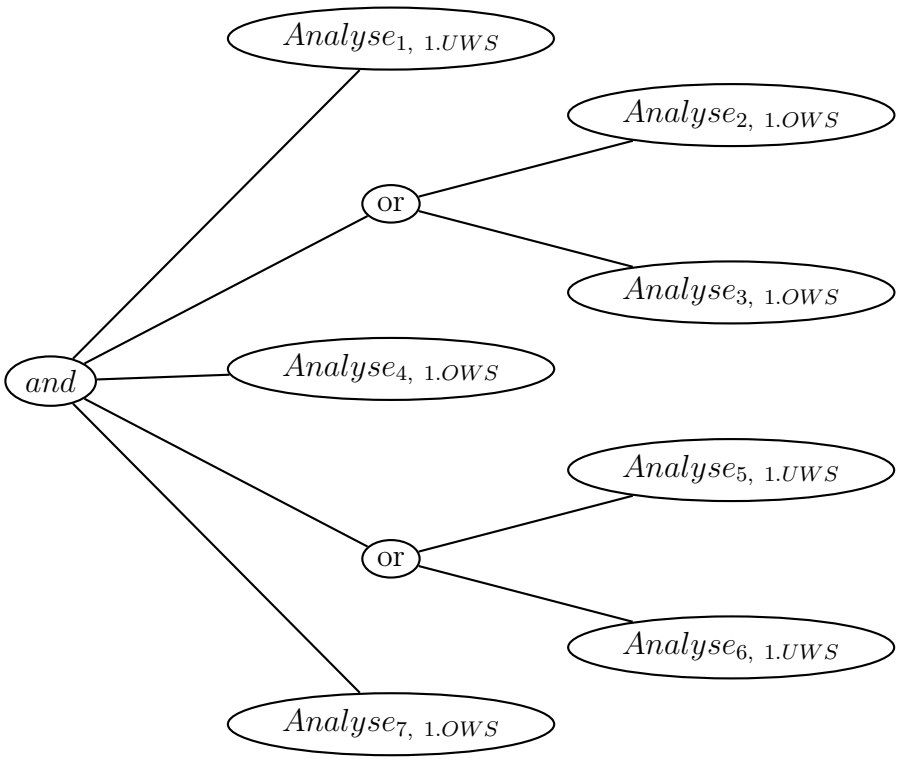

Abbildung 4.2: Semantische Struktur des Regelbaums aus Listing 4.1.

Aussagen ausschließlich über ein logisches Und ($\wedge$) verknüpft. Durch ausmultiplizieren der aussagenlogischen Formel 4.1 lässt sich die folgende *Disjunktive Normalform* bilden:

$$Schaden = Analyse_{12457} \oplus Analyse_{13457} \oplus Analyse_{12467} \oplus Analyse_{13467} \tag{4.2}$$

In Formel 4.2 lässt sich erkennen, dass jeder Minterm oder jeder "Summand" dieser Regel mit einem logischen Oder ($\vee$) verknüpft ist. Jeder Minterm kann als eine Schadenskonfiguration aufgefasst werden und so als ein neues Muster in die Datenbank übernommen werden.

4.8 Abstraktion

Die in diesem Kapitel anhand eines Beispiels der Firma Prognost, vorgestellten Ideen und Möglichkeiten eine Kolbenmaschine zu überwachen, lassen sich auf die Überwachung von vielen anderen Maschinen verallgemeinern.

Dazu muss eine solche Maschine ebenfalls mit Sensoren überwacht werden welche die wichtigsten physikalischen Größen der Maschine und des bearbeiteten Prozesses überwachen. Aus diesen Sensorinformationen lassen sich dann, mit der Hilfe von Experten, ebenfalls weitere Analysen generieren, die mit Warnschwellen überwacht werden können. Die Berechnung der Warnschwellen und Klassifizierung von Schadensmustern kann mit der Hilfe von Experten, welche die jeweilige Maschine kennen, erfolgen.

Mit der Definition von Regeln, die einen Schaden anhand von Warnschwellenverletzungen beschreiben, ist es dann möglich, das in Kapitel 5 beschriebene System ähnlich zu verwenden.

4.9 JEFIS

Im Rahmen eines einjährigen Masterprojektes, siehe [11], wurde im Sommersemester 2007 und im anschließenden Wintersemeter 2007/2008 das bereits bestehende JEFIS[3] weiterentwickelt und ausgebaut.

JEFIS beinhaltet unter anderem eine einfache Regelmaschine, die mit klassischer Logik arbeitet, und die Regeln mit einem Feed Forward Algorithmus verarbeitet. Ziel des Masterprojektes war es, diese klassische Regelmaschine mit einem Fuzzy System zu verbinden. Dadurch soll es möglich sein, sowohl scharfe als auch unscharfe Fakten aus einer Faktenbasis mit einer passenden Regelbasis zu verarbeiten. Dazu können einige unterschiedliche Fuzzy Controller verwendet werden, die intern eine Vielzahl unterschiedlicher implementierter Fuzzy Normen und Operatoren verwenden können, um die Fuzzy Regeln auszuwerten.

Parallel zur Entwicklung der auf Fuzzy Logik basierenden Mustererkennung bei der Firma Prognost wurde die Infrastruktur von JEFIS genutzt, um bestimmte Theorien zu verifizieren und zu überprüfen, bevor diese im Projekt bei Prognost implementiert wurden.

[3] Java Expert Fuzzy Inference System

5 Schadenserkennung mit einem regelbasierten Fuzzy System

Dieses Kapitel gibt eine Übersicht über ein regelbasiertes Fuzzy System, welches in der Lage ist, Expertenregeln, ähnlich denen aus Kapitel 4.7, zu benutzen, um Schadensfälle an einer Maschine zu erkennen. Ziel des zu entwickelnden Systems soll es sein, Schäden zu diagnostizieren, die sich noch in der Entwicklung befinden und eine Warnschwelle noch nicht verletzt haben.

5.1 Fuzzy Regeln

In Kapitel 4.7 wurde Expertenwissen in der Form von Regeln vorgestellt, mit dessen Hilfe sich Schäden an Maschinen beschreiben lassen. Diese Regeln lassen sich, wie gezeigt, mit den Mitteln der Aussagenlogik in eine Disjunktive Normalform überführen, um einzelne Schadensmuster zu generieren.

Zunächst müssen aus diesen Expertenregeln Fuzzy Regeln erzeugt werden. Diese sind nach Formel 2.30 der Form

$$\textbf{Wenn } x_1 = \mathbf{A}_1 \textbf{ und } \ldots \textbf{ und } x_n = \mathbf{A}_n \textbf{ dann } y = \mathbf{B}_k \ . \tag{5.1}$$

In einem Fuzzy Controller lassen sich die Fuzzy Regeln in einem Hyperwürfel anordnen, bei dem jede Dimension einer Eingangsvariablen entspricht. Die einzelnen Fuzzy Sets der Fuzzy Variablen teilen den Hyperwürfel in Zellen ein, wobei in jeder Zelle die Konklusion der Fuzzy Regel steht und die Und-Verknüpfung der Fuzzy Sets einer solchen Zelle die Prämisse der Fuzzy Regel darstellen.

Wenn jede Zelle eines solchen Hyperwürfels mit einer Konklusion versehen ist, spricht man von einer voll besetzten Regelbasis. Das muss nicht immer der Fall sein, so beispielsweise dann, wenn Fuzzy Logik eingesetzt wird, um einen Prozess zu steuern. Einige Regeln machen dann möglicherweise keinen Sinn, weil sich einige Parameter des Prozesses schon ausserhalb kontrollierbarer Grenzen aufhalten.

5.1.1 Natural Logic Controller

Aceves-Lopez und Aguilar-Martin beschreiben in [4] einen spezialisierten, vereinfachten Mamdani Controller, der mit einer minimalen Regelbasis auskommt, den *Natural Logic Controller*. Dieser Fuzzy Controller kommt mit nur zwei Fuzzy Sets zur Fuzzyfizierung der Eingänge aus und benutzt nur zwei Fuzzy Regeln, um einen Prozess zu steuern.

In diesem speziellen Anwendungsfall liefern die Expertenregeln aus 4.7 nur eine einzige Regel für die Regelbasis bei k unterschiedlichen Analysen:

$$\textbf{IF } Analyse_1 = \mathcal{V} \; and \; \ldots \; and \; Analyse_k = \mathcal{V} \textbf{ THEN } Schaden = \mathcal{V}_{out} \; . \tag{5.2}$$

Da die einzelnen Mengen der Prämissen der jeweiligen Konjunktionsterme einer Regel nicht gleich sind, spannt jeder dieser Konjunktionsterme einen eigenen Hyperwürfel mit nur einer Regel auf. Wenn die zu untersuchende Analyse mit nur einem einzigen Fuzzy Set repräsentiert wird, ist der Hyperwürfel zwar voll besetzt, jedoch kann man in diesem Fall nicht mehr von einem Fuzzy System sprechen.

Fuzzy Systeme zeichnen sich vor allem dadurch aus, dass verschiedene Fuzzy Regeln ihre Ergebnisse gegenseitig kompensieren, um das Ergebnis zu ermitteln. Es wird also mindestens eine weitere Fuzzy Regel benötigt, um die Verletzungen der Analysen im Ausgang zu kompensieren. Als einzige Regel bietet sich die genaue gegenteilige Regel zu der aus Formel 5.2 an:

$$\textbf{IF } Analyse_1 = \mathcal{N} \; and \; \ldots \; and \; Analyse_k = \mathcal{N} \textbf{ THEN } Schaden = \mathcal{N}_{out} \; . \tag{5.3}$$

Es wäre zwar auch möglich, mehrere Fuzzy Sets in der Fuzzy Partition P_X zu benutzen, um die Analysen zu fuzzyfizieren. Das Problem dabei ist allerdings, dass die einzig sinnvollen zusätzlichen j neuen Fuzzy Regeln der folgenden Form sind:

$$\textbf{IF } Analyse_1 = \mathcal{G}_i \; and \; \ldots \; and \; Analyse_k = \mathcal{G}_i \textbf{ THEN } Schaden = \mathcal{G}_{i\,out} \; , \tag{5.4}$$

wenn die Fuzzy Sets zwischen den Sets $\mathcal{N}$ und $\mathcal{V}$ mit $\mathcal{G}_1$ bis $\mathcal{G}_j$ bezeichnet werden für $i = 1 \ldots j$.

Mit dem linearen Anstieg der Fuzzy Sets in den Fuzzy Partitionen zur Fuzzyfizierung der Analysen steigt die Anzahl der möglichen Fuzzy Regeln in dem Hyperwürfel exponentiell mit der Anzahl der Analysen. Mit einer steigenden Anzahl an Fuzzy Sets ist

der Hyperwürfel immer dünner besetzt, was die Fähigkeit des Fuzzy Systems, extreme Ergebnisse einer Fuzzy Regel zu kompensieren, weiter sinken lässt.

Der Natural Logic Controller ist gerade deshalb auf genau zwei Fuzzy Sets und zwei Fuzzy Regeln beschränkt.

5.2 Fuzzyfizierung

Um eine Analyse X unscharf zu betrachten, muss diese zunächst fuzzyfiziert werden. Dazu wird eine Fuzzy Partition P_X benutzt, die abhängig von X, der zu betrachtenden Warnschwelle WS und dem zugehörigen Referenzpunkt RP konstruiert wird. Abbildung 5.1 zeigt eine solche Fuzzy Partition für eine obere Warnschwelle.

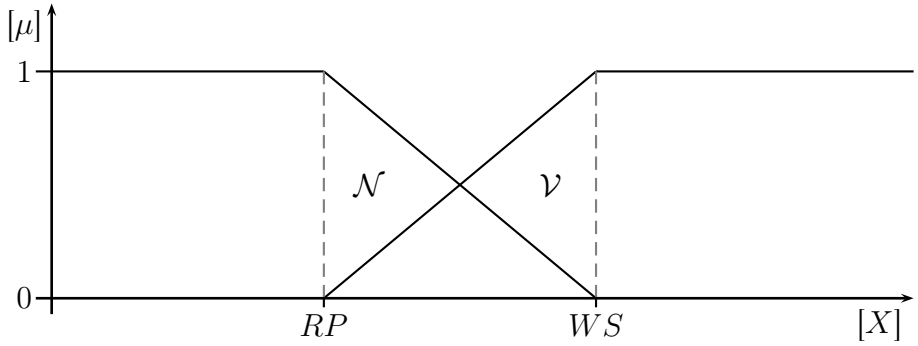

Abbildung 5.1: Fuzzy Partition P_X mit den zwei Fuzzy Sets $\mathcal{N}$ und $\mathcal{V}$ zur Fuzzyfizierung einer Analyse X.

Das Fuzzy Set $\mathcal{V}$ beschreibt, wie stark die Warnschwelle WS, die es zu betrachten gilt, schon verletzt ist. Das Fuzzy Set beginnt beim Referenzpunkt RP, der zur Berechnung der Warnschwelle WS benutzt wurde, und steigt dann linear bis zur Warnschwelle WS an. Dadurch wird die scharfe Verletzung der Warnschwelle WS aufgeweicht. Die Zugehörigkeitsfunktion $\mu_{\mathcal{V}}(x)$ ist folgendermaßen definiert:

$$\mu_{\mathcal{V}}(x) = \begin{cases} 0 & \text{für } x < RP \\ \frac{x-RP}{WS-RP} & \text{für } RP \leq x \leq WS \\ 1 & \text{für } x > WS \end{cases} . \tag{5.5}$$

Das Fuzzy Set $\mathcal{N}$ ist das Gegenteil des Fuzzy Sets $\mathcal{V}$ und beschreibt den Grad zu dem die Analyse die Warnschwelle nicht verletzt ist und sich noch im Normalbereich aufhält. Die Zugehörigkeitsfunktion von $\mathcal{N}$ lässt sich damit leicht folgendermaßen ausdrücken:

$$\mu_{\mathcal{N}}(x) = 1 - \mu_{\mathcal{V}}(x) \tag{5.6}$$

Die Fuzzyfizierung einer Analyse mit einer solchen, auf den Referenzpunkt und die Warnschwelle angepasste Fuzzy Partition, weicht die Verletzung einer Warnschwelle auf.

Sollte, wie in diesem Beispiel angedeutet, keine obere Warnschwelle, sondern eine untere Warnschwelle fuzzyfiziert werden, muss lediglich die Zugehörigkeitsfunktion für das Fuzzy Set $\mathcal{V}$ angepasst werden. Es gilt dann

$$\mu_{\mathcal{V}}(x) = \begin{cases} 1 & \text{für } x < WS \\ \frac{x-RP}{RP-WS} & \text{für } WS \leq x \leq RP \\ 0 & \text{für } x > RP \end{cases} . \tag{5.7}$$

Die Zugehörigkeitsfunktion $\mu_{\mathcal{N}}(x)$ aus Formel 5.6 muss für eine untere Warnschwelle nicht angepasst werden, da diese sich nur auf $\mu_{\mathcal{V}}(x)$ bezieht.

5.3 Implikation

Als Operator für die Implikation wurde nicht das Minimum gewählt, da dieses einen entscheidenden Nachteil hat, wenn auch nur eine der Prämissen bei 0 liegt. Dann wird das komplette Ergebnis des Minterms, der von dieser Fuzzy Regel ausgewertet wird, auf 0 gesetzt, völlig unabhängig davon, wie stark oder schwach die anderen Prämissen verletzt sind. Deshalb wird zu Aggregation der Prämissen das arithmetische Mittel

$$x_{aM} = \frac{1}{n} \sum_{i=1}^{n} x_i \tag{5.8}$$

eingesetzt, um diesem Problem zu entgehen.

Die Verwendung einer klassischen s-Norm zur Veroderung der einzelnen Konjunktionsterme der Fuzzy Regeln bringt einige Nachteile mit sich. Nach Formel 2.20 der Eigenschaften für eine s-Norm gilt $s(1, a) = 1$. Das kann dazu führen, dass sich das Ergebnis nicht wie gewünscht verschiebt. Wie in 4.7 beschrieben, ist jeder Konjunktionsterm einer Disjunktiven Normalform bereits hinreichend, um einen Schaden zu erkennen.

Wenn nun die unterschiedlichen Konjunktionsterm stark unterschiedliche Ergebnisse liefern, führt das dazu, dass bei der Vereinigung mit einer s-Norm das Maximum der einzelnen Fuzzy Sets gebildet wird. Wenn aber in der Ausgangspartition alle Fuzzy Sets maximal gefüllt sind, führt das in diesem speziellen Fall dazu, dass sich das Ergebnis der gesamten Regel bei 0,5 einpendeln wird.
Für die Vereinigung der einzelnen Konjunktionsterme ist eine s-Norm also ungeeignet. Die Informationen der einzelnen Konjunktionsterme können mittels Defuzzyfizierung ermittelt werden. Da jeder der Konjunktionsterme einen Aspekt des Schadens beschreibt, ist es nur logisch anzunehmen, dass das Maximum der Erkennung eines Schadens der Konjunktionsterme das Ergebnis der Fuzzy Regel sein muss. Dazu müssen die einzelnen Ergebnisse der Konjunktionsterme durch Defuzzyfizierung ermittelt werden, um dann das Maximum dieser Werte als das Ergebnis x_{out} der gesamten Regel zu nehmen:

$$x_{out} = max_i(CoG(P_{out\,i})) \ . \tag{5.9}$$

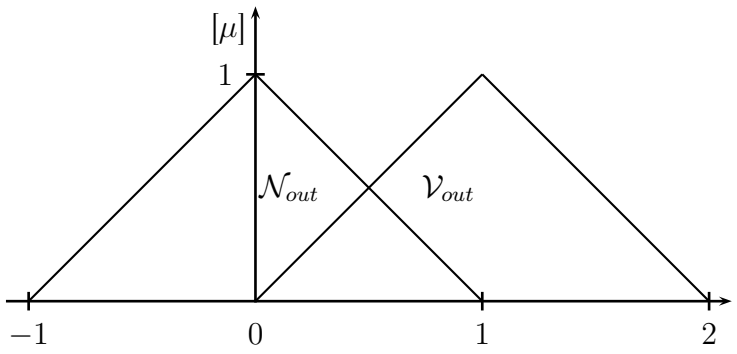

Abbildung 5.2: Fuzzy Partition P_{out} mit den zwei Fuzzy Sets $\mathcal{N}_{out}$ und $\mathcal{V}_{out}$ zur Aggregation der Warnschwellenverletzungen einer Analyse X.

Die Fuzzy Partition P_{out} in welche die beiden Fuzzy Regeln aus Formel 5.2 und 5.3 abbilden, besteht aus zwei Fuzzy Dreiecken(siehe Abbildung 5.2) $\mathcal{N}_{out}$ und $\mathcal{V}_{out}$. Für diese beiden Fuzzy Sets gelten die folgenden Zugehörigkeitsfunktionen:

$$\mu_{\mathcal{V}_{out}}(x) = \begin{cases} x & \text{für } 0 \leq x < 1 \\ -x+2 & \text{für } 1 \leq x \leq 2 \\ 0 & \text{sonst} \end{cases} \tag{5.10}$$

und

$$\mu_{\mathcal{N}_{out}}(x) = \begin{cases} x+1 & \text{für } -1 \leq x < 0 \\ -x+1 & \text{für } 0 \leq x \leq 1 \\ 0 & \text{sonst} \end{cases} \tag{5.11}$$

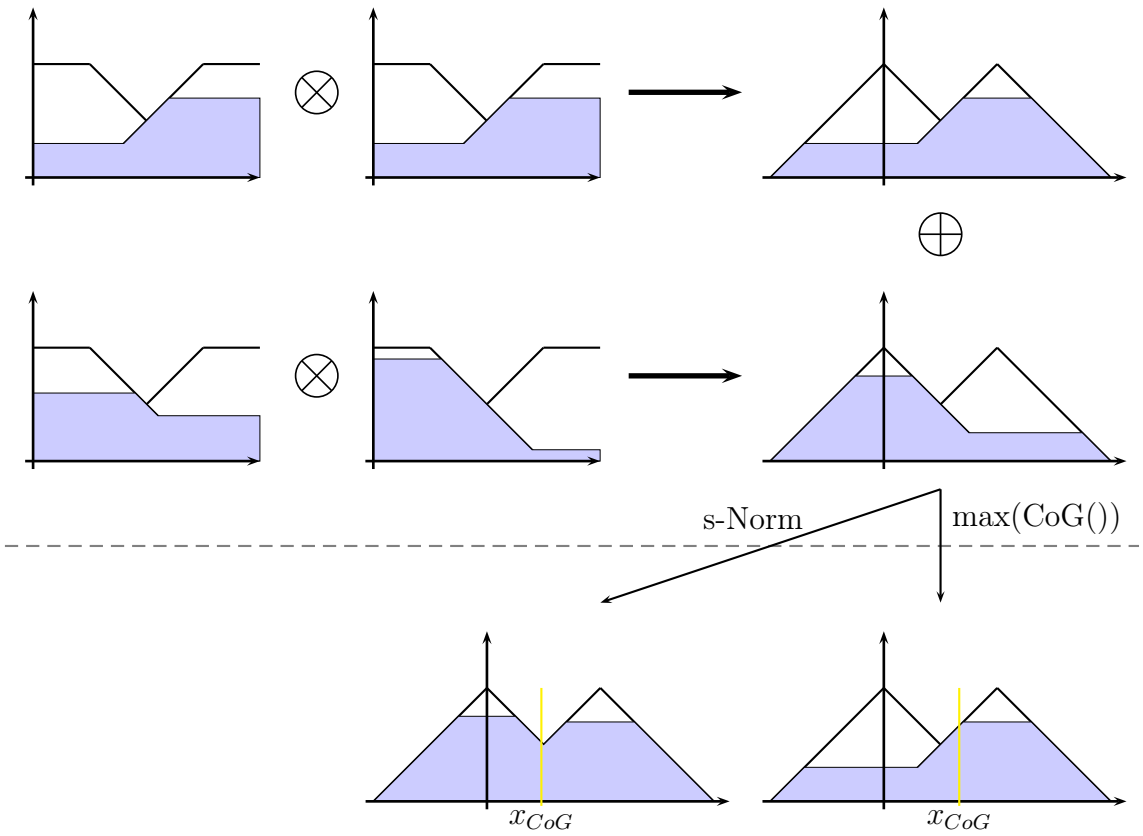

Abbildung 5.3: Veranschaulichung der Wahl des $\oplus$-Operators.

Abbildung 5.3 verdeutlicht, warum die klassische s-Norm ungeeignet ist, um die einzelnen Minterme der Fuzzy Regel miteinander zu verknüpfen. Die s-Norm liefert einen Wert von $0,49$, während die $max_i(CoG(P_{outi})$ den Wert 0.665 liefert. Man kann auch gut erkennen, wie sich die beiden unterschiedlichen Ergebnisse der Minterme in der s-Norm-Implikation verlieren.

5.4 Optimierung der Defuzzyfizierung

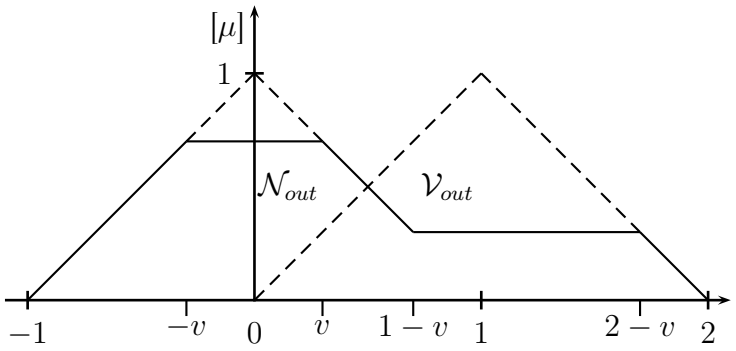

Abbildung 5.4: Defuzzyfizierung von P_{out}.

Grundsätzlich lässt sich das Integral zur Berechnung des Center of Gravity (siehe 2.7) einer Ausgangspartition bei einer Mamdani Inferenz als Funktion der Zugehörigkeiten der einzelnen Fuzzy Setzs der Ausgangspartition angeben. Bei mehr als zwei Fuzzy Sets oder bei komplexeren Fuzzy Sets als einfachen Fuzzy Dreiecken kann diese Funktion aber sehr schnell sehr komplex werden, weil viele Fallunterscheidungen getroffen werden müssen, bei denen die Zugehörigkeitswerte benachbarter Fuzzy Sets verglichen werden.

Durch die spezielle Gestaltung der Fuzzy Regeln und der Fuzzy Operatoren des beschriebenen Fuzzy Systems ergibt sich für die Ausgangspartition P_{out} der Fall, dass für die Zugehörigkeiten der beiden Fuzzy Sets $\mathcal{N}_{out}$ und $\mathcal{V}_{out}$ gilt

$$\mu_{\mathcal{V}_{out}}(x) = 1 - \mu_{\mathcal{V}_{out}}(x) \ . \tag{5.12}$$

Damit hängt der defuzzyfizierte Wert von P_{out} nur noch von der Zugehörigkeit v des Fuzzy Sets $\mathcal{V}_{out}$ ab und es lässt sich der $CoG(P_{out})$ als eine Funktion von v darstellen:

$$CoG(v) = \frac{\int_{-1}^{2} x * \mu(x, v)\mathrm{d}x}{\int_{-1}^{2} \mu(x, v)\mathrm{d}x} \tag{5.13}$$

$$= \frac{\int_{-1}^{-v}(1+x)\mathrm{d}x + (1-v)\int_{-v}^{v}\mathrm{d}x + \int_{1-v}^{v}(1-x)\mathrm{d}x + v\int_{1-v}^{2-v}\mathrm{d}x + \int_{2-v}^{2}(2-x)\mathrm{d}x}{\int_{-1}^{-v}x(1+x)\mathrm{d}x + (1-v)\int_{-v}^{v}x\mathrm{d}x + \int_{1-v}^{v}x(1-x)\mathrm{d}x + v\int_{1-v}^{2-v}x\mathrm{d}x + \int_{2-v}^{2}x(2-x)\mathrm{d}x} \tag{5.14}$$

$$= \frac{(\frac{3}{2} - \frac{v}{2}) * v}{-v^2 + v + 1} \tag{5.15}$$

Da v ein Fuzzy Alpha Schnitt ist, gilt $v \in [0, 1]$. $CoG(v)$ bildet in das Einheitsintervall ab, da mit $v = 1$ das Ergebnis 1 ist und 0 für $v = 0$. Abbildung 5.5 zeigt den Verlauf der Funktion im Einheitsintervall.

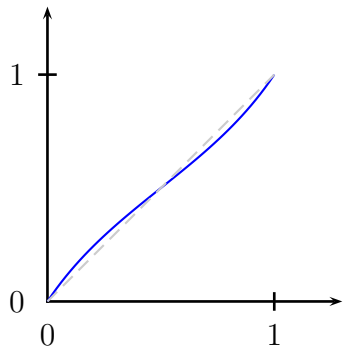

Abbildung 5.5: Der Verlauf von $CoG(v)$ im Intervall $[0, 1]$.

Die Defuzzyfizierung lässt sich weiter vereinfachen, indem man nun die ermittelte Funktion aus Formel 5.15 in Formel 5.9 einsetzt. Wenn nun gezeigt werden kann, dass Formel 5.15 streng monoton steigend ist, lässt sich Formel 5.9 zu

$$x_{out} = CoG(max_i(v_i)) \tag{5.16}$$

vereinfachen.

Die strenge Monotonie von $CoG(v)$ auf dem Intervall $[0, 1]$ lässt sich recht einfach nachrechnen:

$$
\begin{aligned}
& f(a) < f(b) \\
\Leftrightarrow \quad & \frac{(\frac{3}{2} - \frac{a}{2}) * a}{-a^2 + a + 1} < \frac{(\frac{3}{2} - \frac{b}{2}) * b}{-b^2 + b + 1} \\
\Leftrightarrow \quad & ((\frac{3}{2} - \frac{a}{2}) * a) * (-b^2 + b + 1) < ((\frac{3}{2} - \frac{b}{2}) * b) * (-a^2 + a + 1) \\
\Leftrightarrow & -3ab^2 + 3ab + 3a + a^2b^2 - a^2b - a^2 < -3ba^2 + 3ba + 3b + b^2a^2 - b^2a - b^2 \\
\Leftrightarrow \quad & -2ab^2 + 3a - a^2 < -2ba^2 + 3b - b^2
\end{aligned}
$$

da $a < b$ gilt, kann auf der linken Seite $a^2 - 3a$ und auf der rechten Seite $b^2 - 3b$ addiert werden, ohne die Ungleichung zu verändern

$$
\begin{aligned}
\Leftrightarrow \quad & ab^2 > ba^2 \\
\Leftrightarrow \quad & b > a \qquad \text{q.e.d}
\end{aligned}
$$

Da nur Äquivalenzumformungen durchgeführt wurden, ist die Monotonie damit gezeigt und Formel 5.16 kann als Defuzzyfizierung verwendet werden.

6 PnPatFS

Dieses Kapitel gibt eine Übersicht über das entwickelte, regelbasierte Fuzzy System zur Schadenserkennung und über die Integration in das PROGNOST®-NT System.

Zunächst wird die Schnittstelle zum bestehenden System, zusammen mit der internen Datenhaltung des Moduls, erläutert, um dann danach das Parsing der Regelbasis aus einer XML-Datei zu veranschaulichen. Zum Schluss wird dann die Architektur und die Arbeitsweise des Fuzzy Controllers erläutert.

6.1 Architektur und Aufbau

Intern besteht das PnPatFS[1]-Modul aus vier Subsystemen. Das PnPatFS-Modul verbirgt sich hinter einer Fassade, welche in *IPatternFS.h* beschrieben ist. Die Klasse *CMachine* implementiert dieses Interface und enthält damit die eigentlich Fassadenfunktionalität. Die Schnittstelle initialisiert das Modul und steuert die Auswertung der Regeln.

Ein weiteres Subsystem ist die interne Datenhaltung, welche die Informationen über die Analysen, die Warnschwellen und die Messwerte in einer Baumstruktur aufbewahrt, die ihrerseits die Maschinentopologie repräsentiert. Die anderen beiden Subsysteme sind die Regelbasis, welche die geparsten Regeln der XML-Dokumente enthält, und der Fuzzy Controller, der diese Regeln auswertet.

6.2 Schnittstelle zu PROGNOST®-NT

Zunächst wird eine Instanz von *CMachine* mit dem Aufruf von *Init(...)* initialisiert; alle anderen Methodenaufrufe geben ohne diese Initialisierung direkt einen Fehler zurück. An *Init(...)* werden die Informationen über alle auf der Maschine vorhandenen Analysen, zusammen mit deren Position, auf der Maschine übergeben. Diese Informationen enthalten eine eineindeutige Analysennummer, den Wertebereich der Analyse, die Segmentierung, die Art der Analyse und den Sensortyp.

[1] **P**rog**n**ost **Pat**tern **F**uzzy **S**ystem

Die Initialisierung beginnt mit dem Parsing der XML-Dateien. Die geparsten Regeln werden mit den Informationen der an *Init(...)* übergebenen Analyseninformation parametrisiert. Im Erfolgsfall wird das *IsInit*-Flag gesetzt und erst danach kann *InitLimits(...)* aufgerufen werden, um die Informationen über die Warnschwellen zu setzen.

InitLimits(...) speichert die übergebenen Warnschwelleninformationen in den Analyseninformationen der internen Datenhaltung. Diese Informationen beinhalten die eineindeutige Analysennummer und gegebenenfalls eine Segmentnummer, um die Warnschwellen dem passenden Segment der Analyse zuordnen zu können. Desweiteren sind die 4 Werte der möglichen Warnschwellen und ein Bitfeld enthalten, in welchem die Gültigkeit der einzelnen Warnschwellen kodiert ist.

Danach stößt *InitLimits(...)* eine Überprüfung der Regeln auf Vollständigkeit der Warnschwelleninformationen an. Nur wenn für alle Prämissen einer Regel aktuelle Warnschwelleninformationen vorliegen, kann diese auch ausgewertet werden. Sollte die Überprüfung fehlschlagen, so wird die Regel nur als ungültig markiert und nicht gelöscht, da eine spätere Änderung der Warnschwellen eine erneute Aktivierung zur Folge haben kann. Am Ende von *InitLimits(...)* wird das *IsInitLimits*-Flag gesetzt und die Initialisierung der Regelmaschine ist erfolgreich abgeschlossen.

Ab diesem Zeitpunkt kann jederzeit ein Satz Analysenmesswerte an *SetPattern(...)* übergeben werden, um eine Schadenserkennung einzuleiten. Ein Analysenmesswertdatensatz beinhaltet wieder die eineindeutige Analysennummer, um die Informationen in der internen Datenhaltung ablegen zu können. Weiter sind der Messwert selbst und ein Statusbitfeld enthalten, welches Informationen über die Gültigkeit des Messwertes enthält. Nach der Übergabe dieser Informationen an die interne Datenhaltung lässt *SetPattern(...)* sämtliche Regelgruppen auswerten. Wenn die Ergebnisse dieser Auswertungen das jeweilige individuelle oder das globale Treshold überschreiten, so werden die Ergebnisse in speziellen Ergebnisklassen gespeichert. Nach der Auswertung der Regeln löscht *SetPattern(...)* alle erhaltenen Analysenmesswerte, da beim nächsten Aufruf neue zu erwarten sind. Nach dem Aufruf von *SetPattern(...)* kann bis zum erneuten Aufruf von *SetPattern(...)* mittels *GetPatternRecognitionResult(...)* nachgefragt werden, ob Schäden identifiziert wurden. Wenn dies der Fall ist, kann dann eine Liste der identifizierten Schadensklassen, absteigend nach ihrem Erkennungsgrad sortiert, abgerufen mit werden.

Mit *SetLimitDataDirty()* ist es PROGNOST®-NT möglich, die Warnschwelleninformationen innerhalb der Datenhaltung als ungültig zu markieren. Erst nach einem er-

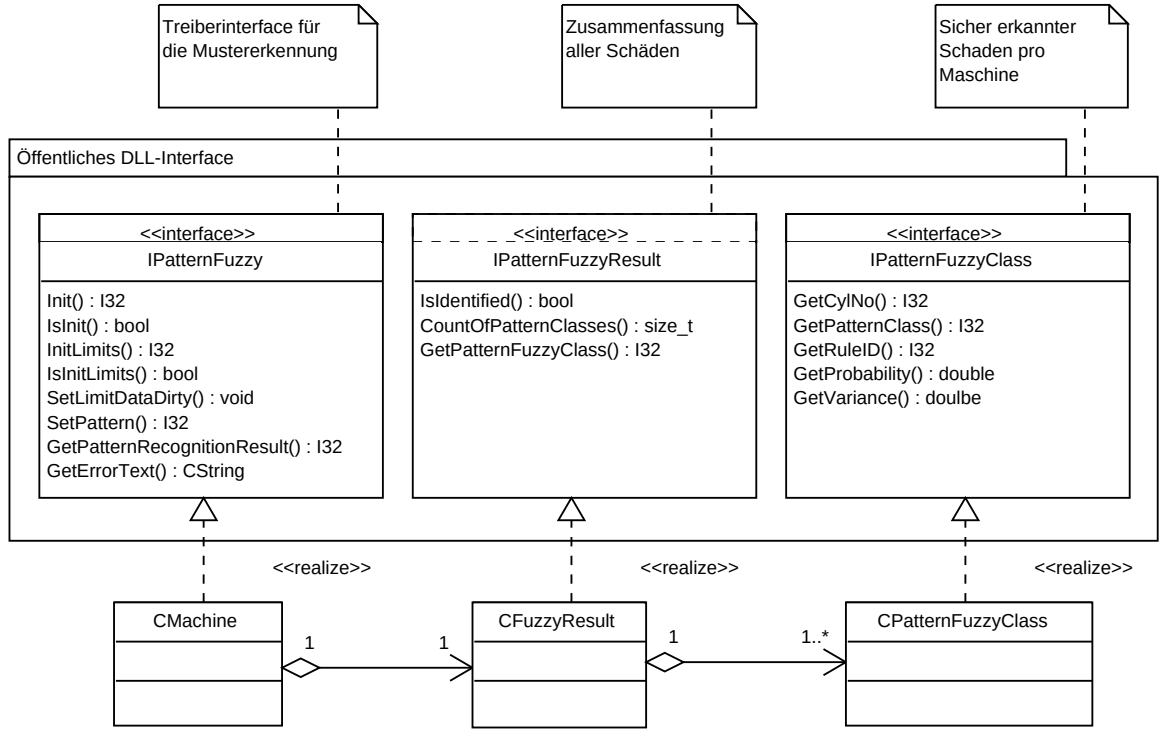

Abbildung 6.1: UML-Klassendiagramm der PnPatFS-Schnittstelle zu PROGNOST®-NT .

neuten Aufruf von *InitLimits(...)* können wieder Schäden diagnostiziert werden. Diese Funktion wurde hinzugefügt, da das Validieren der Warnschwellen und das eigentliche Aufrufen von *SetPattern(...)* an unterschiedlichen Stellen innerhalb von PROGNOST®-NT geschieht.

SetPattern(...) sowie die beiden Initialisierungsfunktionen überprüfen gleich zu Beginn, ob die Größe und die Anzahl des übergebenen Arrays mit den Informationsstrukturen übereinstimmen, um zu vermeiden, dass versucht wird, auf nicht allozierten Speicher zuzugreifen.

Zu sämtlichen Fehlercodes des Moduls, die auftreten können, kann mittels *GetErrorText(...)* ein String abgefragt werden, welcher dem Anwender genauere Informationen über die Art des Fehlers zukommen lässt.

6.3 XML und XSD

Zu Beginn des Projektes lagen die zu verwendenden Expertenregeln als Brainstorming Dokument vor, in welchem die Regeln sowie die Vorbedingungen zur Schadensanalyse definiert waren. Um diese Informationen maschinenlesbar abzulegen, wurde das XML-Format[2] gewählt. XML erlaubt es, komplexe Zusammenhänge in einem Dokument zu modellieren und bleibt dabei bei gutem Design noch gut für Menschen lesbar. Durch die Wahl des XML-Formates ist es einfach, die Regelbasis zu warten und um neue Expertenregeln zu erweitern.

Ein XSD-Dokument[3] definiert die Syntax und die Struktur des XML-Regeldokumentes und erlaubt es dem XML-Parser, welcher das Regeldokument einliest, eine Überprüfung und Validierung der Struktur vorzunehmen. Dadurch können fehlerhafte Dokumente automatisch erkannt werden.

Um eine Abwärtskompatibilität des PROGNOST®-NT Systems bis Windows 2000 sicherzustellen, kommt als XML-Bibliothek MSXML 4.0[4] zum Einsatz. MSXML bietet eine umfangreiche Bibliothek, um XML-Dokumente einzulesen, zu manipulieren und zu speichern.

Sämtliche nötigen Funktionen, um ein XML-Dokument zu laden, zu validieren, Anfragen zu stellen und um Informationen zu Elementen und deren Attributen auszulesen, sind in der Klasse *CXMLParser* gekapselt worden. Die Wrapperklasse kommuniziert mit MSXML über die COM[5]-Schnittstelle von Windows.

Die XML-Regelbasis und das XSD-Dokument werden bei der Kompilierung des Projekts als Ressource mit in die DLL eingebunden, welche die Funktionalitäten des PnPatFS-Moduls bereitstellt.

6.4 Interne Datenhaltung

Intern verwaltet das PnPatFS-Modul sämtliche vorhandenen Analyseninformationen, indem sie hierarchisch in einer Baumstruktur abgespeichert werden, die durch verschach-

```
typedef map<I32, ANALYSIS_INFO>                AnalysisMap;
typedef map<I32, ANALYSIS_INFO>::iterator      AnalysisMapIter;
typedef map<I32, AnalysisMap>                  MpMap;
typedef map<I32, AnalysisMap>::iterator        MpMapIter;
typedef map<I32, MpMap>                        CylMap;
typedef map<I32, MpMap>::iterator              CylMapIter;
typedef map<I32, CylMap>                       CrankMap;
typedef map<I32, CylMap>::iterator             CrankMapIter;
```

Listing 6.1: Verschachtelte HashMaps und deren Iteratoren zur Repräsentation der internen Baumstruktur.

telte HashMaps realisiert ist. Die Struktur der Verschachtelung ist in Listing 6.1 zu sehen.

Logisch gesehen ist die oberste Ebene die Maschinenebene gefolgt von der Kurbelebene. Jede Kurbel kann mehrere Zylinder haben und jedem Zylinder sind Messpunkte zugeordnet. Zu jedem Messpunkt kann es verschiedene Analysen geben, welche in ein, acht oder sechsunddreißig Segment(e) zerlegt sein können.

Sämtliche Analysen, die maschinenbezogen sind, befinden sich auf der Maschinenebene und sind mit der Kurbelnummer -1 indiziert. Jede Kurbel ist unter ihrer Kurbelnummer zu finden. Pro Kurbel kann es mehr als einen Zylinder geben, der mit seiner Zylindernummer indiziert ist. Sämtliche kurbelbezogenen Analysen sind mit der Zylindernummer -1 indiziert. Auf der nächst tieferen Ebene befinden sich die Messpunkte, die mit der Messpunktnummer indiziert sind. Zu jedem Messpunkt gibt es verschiedene Analysen, die per Analysennummer identifiziert sind. Jede Analyse enthält einen Vektor, der die nötigen Informationen für die einzelnen Segmente aufnehmen kann.

Kurbelnummer, Zylindernummer, Messpunktnummer und Analysennummer sind pro Maschine eindeutig. Es wäre auch möglich, die Analysen in einer flacheren, über die Analysennummer indizierten, Datenstruktur abzulegen, jedoch sind für bestimmte Überprüfungen, die beim Regelparsing vorgenommen werden, die zusätzlichen topologischen Informationen der Baumstruktur hilfreich.

Da die übergebenen Daten an *Init(...)*, *InitLimits(...)* und *SetPattern(...)* nicht vorsortiert sind, erlaubt die Baumstruktur ein recht performantes Einfügen neuer Daten beim Aufruf von *Init(...)* und ebenfalls ein performantes Ändern der enthaltenen Informatio-

[2]Extensible Markup Language
[3]XML Schema Definition
[4]Microsoft XML Core Services
[5]Component Object Model

nen beim Aufruf von *InitLimits(...)* und *SetPattern(...)*. Mit drei Aufrufen kann jede Analyse der Maschine erreicht werden, mit einem weiteren können die Informationen für ein spezielles Segment abgefragt werden.

In der Struktur der Analyseninformationen sind Informationen über den Sensortyp und die Art der Analyse hinterlegt, die verwendet werden, um die Analysennummer zu den geparsten Informationen aus der Regel zu finden. In der Struktur ist außerdem ein Zähler für die Anzahl der Segmente der Analyse enthalten. Zusätzlich ist ein Vektor eingebettet, der die Datenfelder für den Messbereich, die Warnschwellen, den Messwert, den Messwertstatus und den Warnschwellenstatus der einzelnen Segmente enthält. Der Messwert und der Messwertstatus werden zu Beginn des Aufrufs von *SetPattern(...)* gesetzt, während die Warnschwelleninformationen von *InitLimits(...)* gesetzt werden. Alle anderen Daten werden bei der Initialisierung von *Init(...)* gesetzt und danach nicht mehr verändert.

6.5 Verarbeiten der Regelbasis

Nachdem die Analyseninformationen während der Initialisierung in der internen Datenhaltung abgelegt wurden, wird das Regelparsing aufgerufen. Dabei wird zunächst das XSD-Dokument eingelesen. Zusammen mit den Informationen des XSD-Dokuments wird dann das XML-Dokument geladen und schon beim Laden vom XML-Parser mit Hilfe der Schemabeschreibung aus dem XSD-Dokument validiert. Wenn also ein ungültiges XML-Dokument hinterlegt wurde, so bricht die Initialisierung direkt mit einer entsprechenden Fehlermeldung ab.

Jede Regel, die vom XML-Parser geparsed wird, wird in eine Instanz der Klasse *CRuleGroup* überführt. Eine solche Regelgruppe verwaltet intern die einzelnen Minterme der disjunktiven Normalform der Regel, die bei der späteren Verarbeitung entstehen.

Für jede Regel ist eine bestimmte Granularität in dem Attribut *ruleType* des *Rule*-Elementes definiert. Hier kann angegeben werden, ob eine Regel pro Maschine, pro Zylinder oder pro Kurbel gültig ist. Je nach Anzahl der Zylinder oder Kurbeln der Maschine wird die Regelgruppe dann für die einzelnen Instanzen dupliziert. Dabei wird geprüft, ob alle Analysen, die als Prämissen in die Regel eingehen, in der aktuellen Granularitätsstufe vorhanden sind. Einige Prämissen können im DataSpecies-Element explizit das Attribut ignorable auf true gesetzt haben. Dann führt eine Abwesenheit

dieser Analyse nicht dazu, dass die komplette Regel für diese Granularität verworfen wird.

6.5.1 Regelgültigkeiten

Nachdem das XML-Dokument eingelesen ist, wird versucht die Regeln aus dem Dokument zu extrahieren. Hierbei wird pro Regel jeweils geprüft, ob diese für die aktuelle Maschine gültig ist. Jede Regel enthält dazu im *Rule*-Element ein *machineType*-Attribut. In diesem Attribut ist kodiert, ob die Regel für den aktuellen Maschinentyp oder den aktuellen Maschinengruppentyp gültig ist. Wenn eine Regel ungültig ist, springt der XML-Parser zur nächsten Regel.

Pro Regel existieren bis zu vier optionalen Elemente, welche abhängig von der Existenz oder der expliziten Nichtexistenz bestimmter Analysen eine Regel validieren. Die Elemente *dependsOnOneOf* und *dependsOnAllOf* enthalten jeweils eine Liste von Analysentypen, von denen im ersten Fall entweder eine oder im zweiten Fall alle in der jeweiligen Granularitätsstufe vorhanden sein müssen, damit die Regel gültig ist.

Mit den Elementen *deniedByOneOf* und *deniedByAllOf* können einzelne Regeln validiert werden, indem explizit die Abwesenheit einer Analyse oder aller Analysen der Listen der beiden Elemente gefordert wird.

6.5.2 Regelparsing

Wenn die Gültigkeiten der jeweiligen Regel geklärt sind, beginnt das Parsen der Prämissen und deren logischer Abhängigkeiten. Eine Regel ist dabei so aufgebaut, dass alle Informationen einer Prämisse in einem *DataSpecies*-Element hinterlegt sind.

Eine Regel enthält entweder als erstes Element ein *and*-Element, ein *or*-Element oder ein *DataSpecies*-Element. Jedes *and*-Element kann beliebig viele *DataSpecies*- und beliebig viele *or*-Elemente als Subelemente enthalten. Ebenso kann ein *or*-Element beliebig viele *DataSpecies*- und beliebig viele *and*-Elemente als Subelemente enthalten.

Da der Fuzzy Controller nur konjunktiv verknüpfte Regeln beherrscht, müssen die oder-Zweige der Regeln aufgelöst und die gesamte Regel in eine disjunktive Normalform umgewandelt werden. Dazu wird die Baumstruktur zunächst in eine lineare Liste überführt, in der sämtliche *DataSpecies*-Element enthalten sind. Der Regelbaum aus dem XML-Dokument wird nach dem preOrder-Verfahren traversiert, die öffnenden *and*-

und *or*-Elemente werden beim Hinabsteigen und die zugehörigen schließenden Elemente beim Aufsteigen zurück zum Wurzelknoten eingefügt.

Nach der Überführung in eine lineare Liste wird diese so lange bearbeitet, bis sämtliche *and*- und *or*-Elemente entfernt sind. Der dazu entwickelte Listenparser beginnt am Anfang der Liste und überprüft sämtliche Einträge der Reihe nach. Zu jeder Datenspecies wird dann der Pointer auf die dazu passende Analysenstruktur in der internen Datenhaltung gespeichert.

Einige bestimmte *DataSpecies*-Elemente haben mehrfache Repräsentationen durch Analysen innerhalb einer Granularitätsstufe, wie zum Beispiel Ventiltemperaturen. Wenn der Listenparser auf eine solches *DataSpecies*-Element trifft, so wird geprüft, wie oft die zugehörigen Analysen vorhanden sind. Wenn sie mehr als einmal vorhanden ist, wird jede mögliche Instanz der an diese Stelle eingefügt.

Wenn der Listenparser auf ein *and*-Element trifft, so sucht er das passende schließende *and*-Element und entfernt beide.

Wenn der Listenparser auf ein *or*-Element trifft, wird das zugehörige schließende *or*-Element gesucht. Dann werden die beiden *or*-Elemente entfernt und die inneren Bestandteile dieser or-Liste zerlegt, indem alle Datenspecies einzeln und alle von *and*-Elementen eingeschlossenen Elemente als Gruppe betrachtet werden. Jedes einzelne dieser zerlegten Elemente wird dann in eine Kopie der Originalliste an der Stelle der or-Elemente eingefügt.

Dies wird solange wiederholt, bis der Regelbaum in eine Liste von Prämissenlisten überführt wurde. Dabei repräsentiert jede Prämissenliste einen Minterm und die Liste der Prämissenlisten die disjunktive Normalform der Regel.

6.5.3 Beispielparsing einer Regel

In Listing 6.2 ist eine Regel in XML-Notation zu finden, anhand derer ein Parsing gezeigt wird. Diese konstruierte Regel enthält alle nötigen Beispiele, um zu zeigen, welche unterschiedlichen Konfigurationen auftreten können und wie diese zu parsen sind. Die Struktur der Regel ist in Figur 6.2 vereinfacht dargestellt. Zunächst wird die Regel mittels Preordertraversierung in eine lineare Liste umgewandelt, siehe Figur 6.3.

```
<Rule>
    <DataSpecies type="1" where="1">
        <Threshold threshold="1.upper" />
    </DataSpecies>
    <or>
        <DataSpecies type="2" where="2">
            <Threshold threshold="1.upper" />
        </DataSpecies>
        <DataSpecies type="3" where="3">
            <Threshold threshold="1.upper" />
        </DataSpecies>
        <and>
            <DataSpecies type="4" where="4">
                <Threshold threshold="1.upper" />
            </DataSpecies>
            <DataSpecies type="5" where="5">
                <Threshold threshold="1.upper" />
            </DataSpecies>
            <or>
                <DataSpecies type="6" where="6">
                    <Threshold threshold="1.upper" />
                </DataSpecies>
                <DataSpecies type="7" where="7">
                    <Threshold threshold="1.upper" />
                </DataSpecies>
            </or>
            <!-- DataSpecies 8 ist doppelt vorhanden (bspw.
                Saugventiltemperatur) -->
            <DataSpecies type="8" where="8">
                <Threshold threshold="1.upper" />
            </DataSpecies>
        </and>
    </or>
    <DataSpecies type="9" where="9">
        <Threshold threshold="1.upper" />
    </DataSpecies>
</Rule>
```

Listing 6.2: Eine Beispielregel, die alle Sonderfälle enthält, welche während des Regelparsings auftreten können.

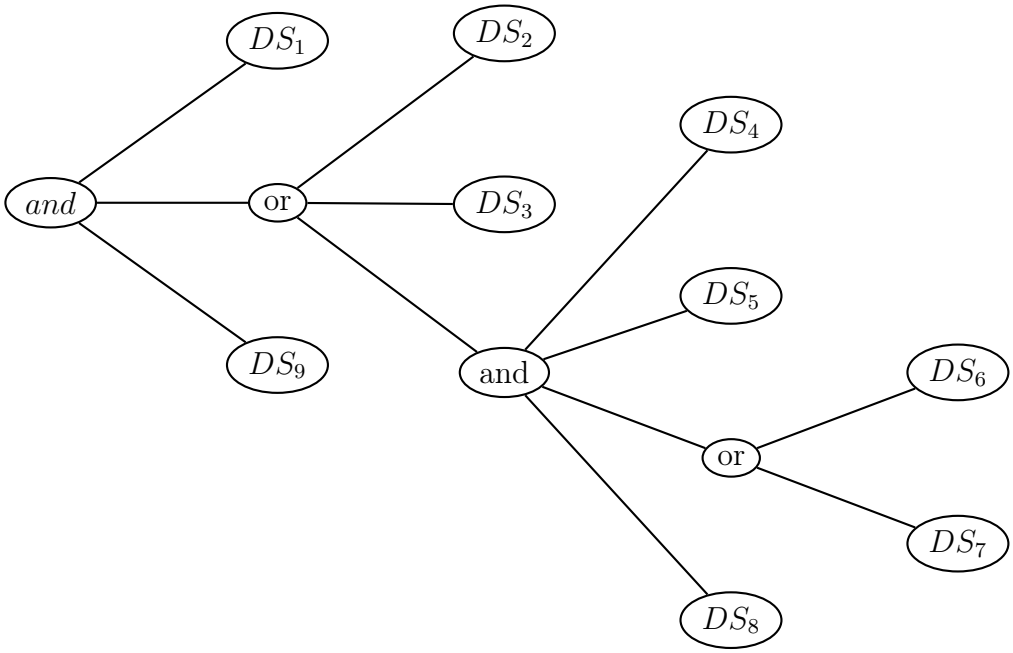

Abbildung 6.2: Semantische Struktur des Regelbaums aus Listing 6.2.

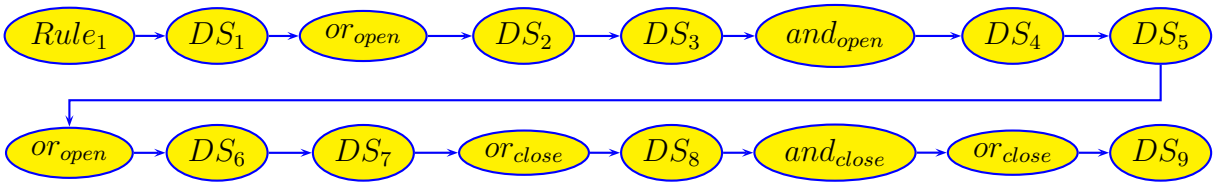

Abbildung 6.3: Die Beispielregel aus Listing 6.2 nach dem Überführen in eine lineare Liste.

Diese Liste wird nun von vorne beginnend durchlaufen. Zum ersten Knoten DS_1 wird der passende Analysenpointer gesucht. Danach wird mit dem nächsten Knoten fortgefahren. Da der nächste Knoten ein or_{open} ist, wird das dazu passende or_{close} gesucht. Diese sind in Abbildung 6.4 grau markiert. Durch Entfernen der einzelnen Verbindungen zwischen den Knoten wird das Innere der or-Liste in einzelne Elemente aufgeteilt. Das sind in diesem Fall die Knoten DS_2, DS_3 und die von den and-Knoten eingeschlossene Substruktur.

Nach der Aufteilung in drei unterschiedliche Regeln, siehe Abbildung 6.5, wird jede Regel erneut einzeln durch den Listenparser verarbeitet. Bei $Rule_1$ und $Rule_2$ wird erst

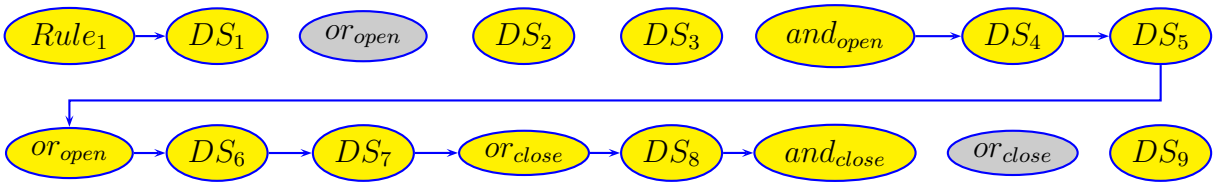

Abbildung 6.4: Identifizierung der or-Elemente und Isolierung der einzelnen Subelemente.

der Knoten DS_1 verarbeitet und dann jeweils die Knoten DS_2 und DS_3. Danach wird in beiden Regeln der Knoten DS_9 verarbeitet, und damit ist das Listenparsing für diese ersten beiden Regeln beendet.

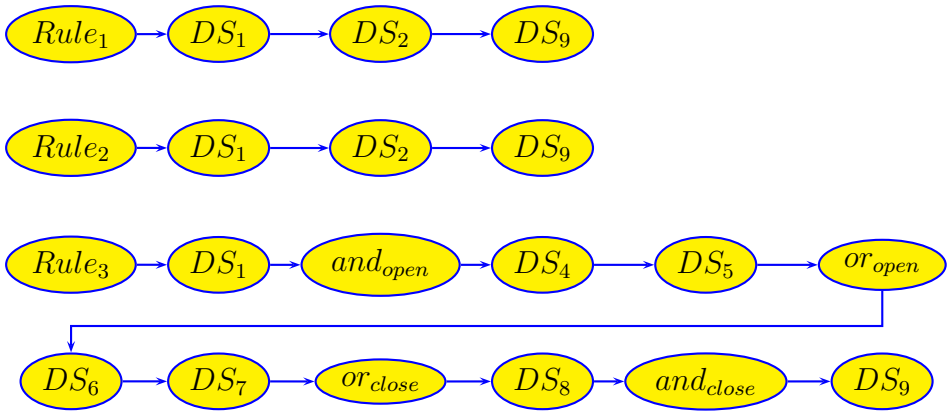

Abbildung 6.5: Nach dem Duplizieren der Regel und Einfügen der isolierten or-Bestandteile.

Bei $Rule_3$ wird zunächst der Knoten DS_1 verarbeitet. Da danach ein and_{open} erkannt wird, wird das dazugehörige and_{close} gesucht und entfernt, siehe Abbildung 6.6.

Danach werden in $Rule_3$ die Knoten DS_4 und DS_5 verarbeitet. Da der nächste Knoten ein or_{open} ist, wird das dazugehörige or_{close} gesucht und die inneren Elemente der or-Liste werden separiert. Siehe Abbildung 6.7.

Die isolierten Knoten DS_6 und DS_7 werden statt der or-Struktur in $Rule_3$ und in die neu erstellte Regel $Rule_4$ eingesetzt, siehe Abbildung 6.8.

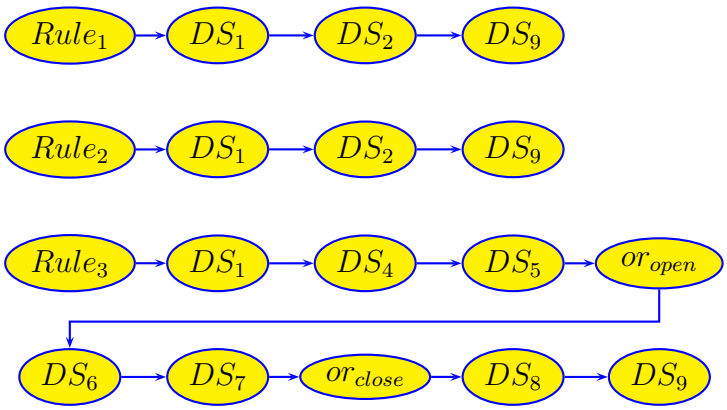

Abbildung 6.6: Nach dem Entfernen der and-Knoten aus $Rule_3$.

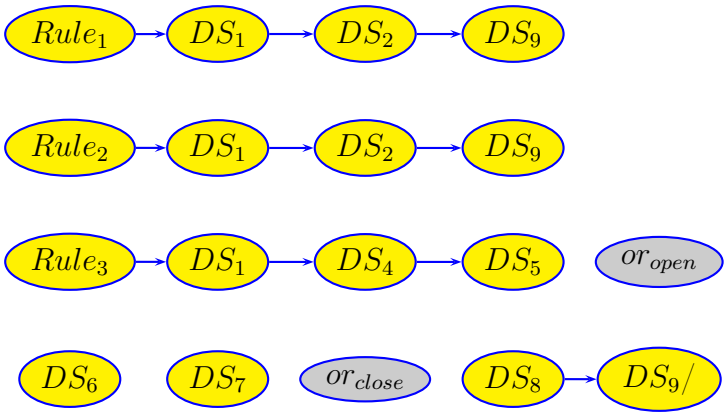

Abbildung 6.7: Identifizierung der or-Elemente und Isolierung der einzelnen Subelemente in $Rule_3$.

Die Regeln $Rule_3$ und $Rule_4$ werden dann vom Listenparser erneut verarbeitet. Dabei werden jeweils die Knoten DS_6 bzw. DS_7 und DS_8 und DS_9 überprüft. Da DS_8, wie in Listing 6.2 angemerkt, doppelt vorhanden ist, beispielsweise eine Saugventiltemperatur, wird dieser Knoten jeweils in $Rule_3$ und in $Rule_4$ durch die beiden Ausprägungen ersetzt. Nach dieser letzten Operation ist das Verarbeiten der Regel aus Listing 6.2 beendet und das Ergebnis ist in Abbildung 6.9 abgebildet.

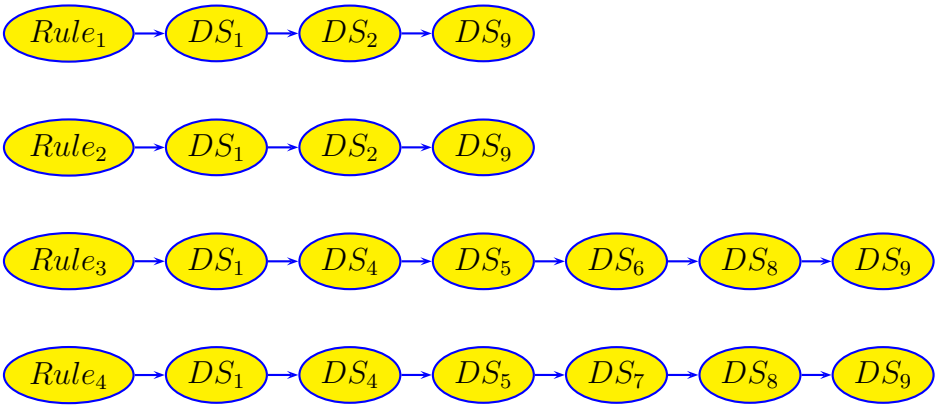

Abbildung 6.8: Nach der Duplizierung der Regel und Einfügen der isolierten or-Bestandteile.

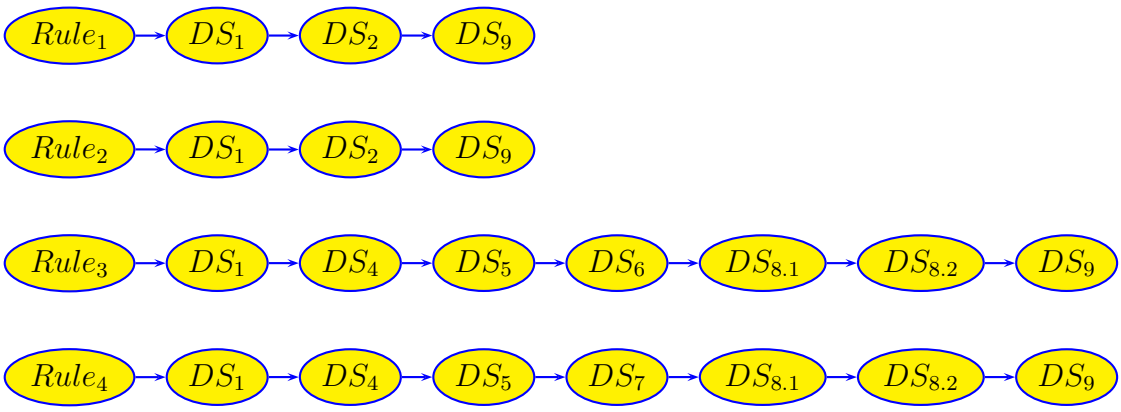

Abbildung 6.9: Finale Liste der geparsten Regeln.

6.6 Regelauswertung

Mit dem Aufruf von *SetPattern(...)* wird ein Auswertungszyklus gestartet. Nach der Überprüfung der übergebenen Daten werden diese an die interne Datenhaltung zur Speicherung übergeben. Wenn noch ein Datensatz über erkannte Schäden aus dem vorhergehenden Auswertungszyklus vorhanden ist, wird dieser gelöscht.

Dann werden die einzelnen, der Fuzzy Maschine bekannten, Regelgruppen durch den Aufruf der *Evaluate(...)* ausgewertet. Der Center of Gravity, siehe 2.4, wird dabei als Value-Return-Paramter übergeben und innerhalb der *Evaluate(...)*-Methode gesetzt. Wenn der errechnete Center of Gravity entweder die in der XML-Regelbasis für diese Regel individuell gesetzten oder die an *StePattern(...)* übergebene Schwelle überschreitet, so

wird das Ergebnis dieser Regel gespeichert. Dazu wird eine neue Instanz der Klasse *CPatternFuzzyResult* erzeugt und mit dem Ergebnis parametrisiert. Sämtliche Ergebnisse des Auswertungszyklus werden in einer Liste gespeichert und können vom PROGNOST®-NT System bis zum nächsten Auswertungszyklus abgerufen werden. Die Ergebnisse werden absteigend nach dem Center of Gravity sortiert, so dass die Schadensklasse mit der höchsten Erkennungsrate an der ersten Stelle steht.

6.7 Fuzzy Implementierung

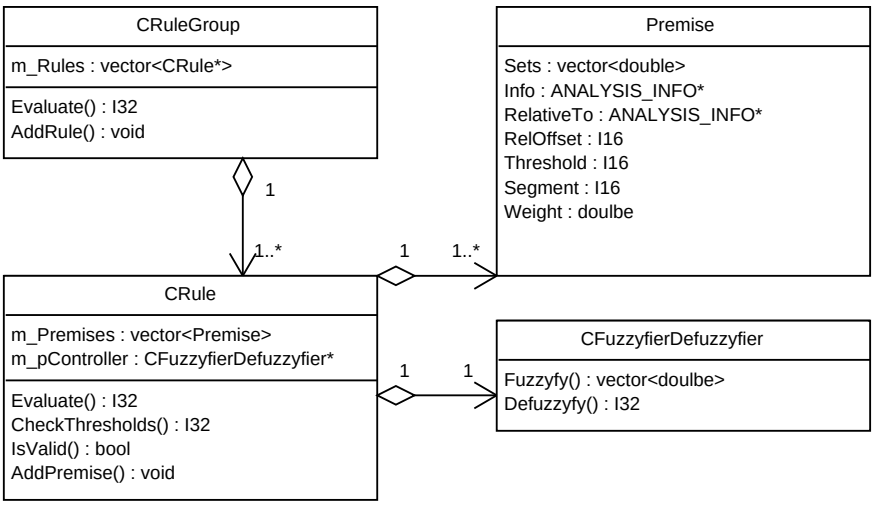

Abbildung 6.10: UML-Diagramm des spezialisierten Fuzzy Controllers.

Durch die spezielle Wahl der Fuzzy Operatoren, der Verwendung der speziellen Fuzzy Partitionen zur Fuzzyfizierung der einzelnen Analysen und der speziellen Fuzzy Inferenz, die in Kapitel 5 erläutert wurde, ist das resultierende Fuzzy System sehr minimalistisch.

Zwar wurde nach dem Vorbild des JEFIS-Projekts (siehe [11]) zunächst ein vollständiges Fuzzy System mit einzelnen Klassen für unterschiedliche Fuzzy Sets und Fuzzy Operatoren realisiert, die tatsächlich benötigte Funktionalität lässt sich jedoch ebenso an anderen Stellen des Systems unterbringen.

Jede Fuzzy Regel ist als eine Instanz der Klasse *CRuleGroup* vorhanden. In jeder dieser Klassen befindet sich eine Liste von *CRule*-Objekten, die einem Minterm der DNF der Expertenregel entsprechen. Jedes *CRule*-Objekt ist in der Lage, mittels der *Evaluate(...)*-Methode, die internen Prämissen auszuwerten.

Dazu benutzt *CRule* die Klasse *CFuzzyfierDefuzzyfier*, welche mit der Methode *Fuzzyfy(...)* eine Prämisse fuzzyfiziert. Die Methode benutzt die Informationen einer übergebenen Analysenstruktur und erzeugt die Fuzzy Sets $\mathcal{V}$ und $\mathcal{N}$, wie in 5.2 beschrieben, um die zu betrachtende Warnschwelle und den zugehörigen Referenzpunkt herum. Die ermittelten Zugehörigkeiten werden zurück gegeben.

In der *Evaluate(...)*-Methode von *CRule* ist der Fuzzy Operator implementiert, welcher die einzelnen Prämissen miteinander verknüpft. Wie in 5.3 beschrieben, handelt es sich um das arithmetische Mittel. Es ist möglich, an einzelnen *DataSpecies*-Elementen innerhalb des XML-Dokumentes ein Gewicht zu setzen. Dann wird das arithmetische Mittel auf das gewichtete, arithmetische Mittel erweitert:

$$x_{wam} = \frac{\sum_{i=1}^{n} w_i * x_i}{\sum_{i=1}^{n} w_i} \ . \tag{6.1}$$

Als Standardgewicht wird immer die 1 angenommen. Mit einem anderen Gewicht kann man eine Analyse dann entweder auf- oder abgewichten.

Die Implikation des Fuzzy Systems befindet sich in der *Evaluate(...)*-Methode der Klasse *CRuleGroup*. Dort wird, nach 5.3, das Maximum der Zugehörigkeiten des $\mathcal{V}$-Sets der einzelnen Minterme ermittelt und dieses nach Formel 5.9 mit Hilfe der Klasse *CFuzzyfierDefuzzyfier* defuzzyfiziert.

In Abbildung 6.10 sind die wesentlichen Klassen des Fuzzy Controllers und deren Abhängigkeiten anschaulich dargestellt.

6.8 Unit- und Lasttests

Als Testumgebung wurde zu Beginn des Projekts *CppUnit* (siehe [10]) ausgewählt. *CppUnit* erlaubt das Testen einzelner Klassen und Methoden nach dem Prinzip der Unit-Tests. Für sämtliche Klassen des PnPatFS-Moduls existieren entsprechende Testklassen, in welchen die Funktionalitäten der einzelnen Methoden einzeln und in ihrer Zusammenarbeit getestet werden.

Der Durchlauf der Tests ist fester Bestandteil der Debug-Konfigurationen des Buildprozesses und wird bei jeder Kompilierung durchgeführt. Dadurch hat man als Entwickler stets einen Überblick, ob die soeben kompilierte Software den Testanforderungen genügt. Da das PnPatFS-Modul innerhalb des Erfasungsmoduls von PROGNOST®-NT

zum Einsatz kommt, ist es Bestandteil einer zeitkritischen Anwendung, und eine geringe Ausführungsgeschwindigkeit war von Beginn an ein wichtiges Kriterium.

Aufgrund dieser wichtigen Anforderung wurde die Software ausgiebig auf einem realitätsnahen Testrechner überprüft. Als Testrechner diente ein Computer der Baureihe MaProg 9, der normalerweise in Produktivsystemen von PROGNOST®-NT zum Einsatz kommt. Anhand der Simulationsdaten einer realen Maschine mit vier Zylindern wurden sämtliche Tests durchgeführt.

Erste Tests haben gezeigt, dass das Laufzeitverhalten mit über 800ms pro *SetPattern(...)*-Aufruf nicht akzeptabel war. Ein erstes Review des Codes nach einem Profiling zeigte, dass an vielen Stellen komplette Listen und HashMaps als Argumente an Methoden übergeben wurden. Nachdem diese Fehler beseitigt wurden, konnte die Laufzeit eines Auswertungszyklus auf einen akzeptablen Wert von ca. 30ms pro Zylinder reduziert werden.

6.9 Optimierungen

Bei der Auswertung der Regeln gehen viele Prämissen mehrfach in eine Regelgruppe ein. Wenn viele dieser Analysen nach der selben Warnschwelle fuzzyfiziert werden und diese gleichzeitig in mehreren Mintermen auftaucht, wird diese Fuzzyfizierung für jede Prämisse erneut durchgeführt. Dieser Aufwand ließe sich einsparen, indem das Ergebnis der Fuzzyfizierung, also der Fuzzy Alpha Vektor $\vec{\alpha}$, zusammen mit der Information, welche Warnschwelle bereits fuzzyfiziert wurde, in den Analyseninformationen abgelegt wird. Bevor die Fuzzyfizierung einer Analyse durchgeführt wird, muss überprüft werden, ob für diese Warnschwelle bereits ein passender Fuzzy Alpha Vektor abgespeichert wurde. Das spart Zeit bei der Auswertung, besonders bei Regeln mit vielen oder-Verknüpfungen. Nach der Überführung solcher Regeln in die disjunktive Normalform entstehen viele Minterme mit einer ähnlichen Kombination von Prämissen. Bei dieser Optimierung ist darauf zu achten, dass der Geschwindigkeitszugewinn möglicherweise durch einen höheren Speicherverbrauch kompensiert wird.

7 Toleranz von Fuzzy Regeln

Das in Kapitel 5 vorgestellte Fuzzy System ist in dieser Form auf viele unterschiedliche Maschinen anwendbar, da die Auswertung der Schadensfälle durch die Warnschwellensetzung unabhängig von den tatsächlichen Betriebsparametern einer solchen Maschine ist. Das Fuzzy System ist jedoch darauf angewiesen, dass sämtliche Informationen für alle Prämissen einer Fuzzy Regel zur Verfügung gestellt werden, damit diese ausgewertet werden kann.

In diesem Kapitel soll nun überprüft werden, ob es möglich ist, die Fuzzy Regeln auszuwerten obwohl die Informationen über die Prämissen unvollständig sind.

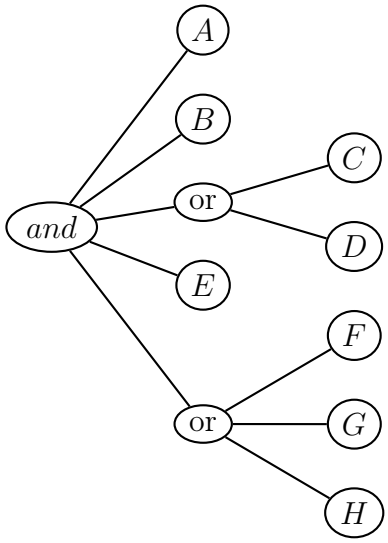

Abbildung 7.1: Zu untersuchende Referenzregel.

Dies könnte der Fall sein, wenn zum Beispiel ein Sensor defekt oder nicht vorhanden ist, da die Maschine nicht mit diesem ausgestattet wurde. In dem Fall des defekten Sensors ist es wichtig, dass die Überwachungssoftware der Maschine dem Fuzzy System mitteilt, dass die Informationen dieses Sensors nicht gültig sind. Ansonsten würde das Fuzzy System mit falschen Werten rechnen und unsinnge Ergebnisse liefern.

Die Fuzzy Regel, die in Abbildung 7.1 zu sehen ist, soll als Referenzregel dienen. Dieser Fuzzy Regel wird jeweils eine von drei unterschiedlichen Prämissen entfernt, die aufgrund ihrer Position in der Regel einen unterschiedlichen Einfluss auf das Gesamtergebnis der Regel hat. Dies führt zu vier unterschiedlichen Regeln, welche durch die folgenden disjunktiven Normalformen beschrieben sind:

$$\begin{aligned}\text{Referenz} : \; & ABCEF \oplus ABCEG \oplus ABCEH \oplus \\ & ABDEF \oplus ABDEG \oplus ABDEH \end{aligned} \tag{7.1}$$

$$\begin{aligned}\text{Ohne-A} : \; & BCEF \oplus BCEG \oplus BCEH \oplus \\ & BDEF \oplus BDEG \oplus BDEH \end{aligned} \tag{7.2}$$

$$\begin{aligned}\text{Ohne-C} : \; & ABEF \oplus ABEG \oplus ABEH \oplus \\ & ABDEF \oplus ABDEG \oplus ABDEH \end{aligned} \tag{7.3}$$

$$\begin{aligned}\text{Ohne-F} : \; & ABCE \oplus ABCEG \oplus ABCEH \oplus \\ & ABDE \oplus ABDEG \oplus ABDEH \end{aligned} \tag{7.4}$$

Die Referenz-Regel entspricht der Regel aus Abbildung 7.1. Der zweiten Regel fehlt die Prämisse A. Dort ist zu erwarten, dass sich diese Regel sehr ähnlich zur Referenz-Regel verhält, außer in dem Fall, indem die Prämisse A das Ergebnis der Referenz-Regel dominiert.

Bei der Regel "Ohne-C" ist eine Prämisse entfernt worden, die in genau der Hälfte der sechs Minterme vorhanden war. Für die letzte Regel "Ohne-F" ist die Prämisse F entfernt worden, die nur in einem Drittel der Minterme vorhanden war. Je seltener eine Prämisse in den Mintermen vorkommt, desto einfacher sollte es sein ihre Abwesenheit zu kompensieren

7.1 Verwendete Operatoren

Da das arithmetische Mittel als Operator zur Verknüpfung der Prämissen in den Mintermen wenig Spielraum lässt, werden vier verschiedene OWA-Operatoren untersucht. Das ist zum einen ein OWA-Operator mit Potenzverteilung, einer mit Polynomialverteilung (siehe 3.2) und zwei OWA-Operatoren, die als Quantifizierer (siehe 3.3) verwendet werden, einmal mit einer Transitionsbreite von 0.01 und einer von 0.1.

Alle vier Operatoren sind über das komplette Spektrum der andness (siehe Formel 3.8) angegeben. Die Gewichte der OWA-Operatoren werden dazu von einer speziellen Klasse des JEFIS-Projektes kalibriert, um für jede andness aus dem Intervall $[0, 1]$ die Operatoren mit den passenden Gewichten zu versehen. An den Quantifizierern geschieht das über den Quantifizierungswert, der verändert wird. Die beiden anderen Operatoren stellen einen speziellen Parameter zur Einstellung der andness zur Verfügung.

7.2 Verwendete Eingangswerte

Für die Eingangswerte werden zwei unterschiedliche Szenarien verwendet. Die Werte die in den folgenden Listings zu sehen sind, enthalten die Eingangsdaten in einem zweidimensionalen Double-Array. Pro Durchgang wird jede Zeile jeder Fuzzy Regel einmal zugeführt. Die Elemente des inneren Arrays sind dabei nach der Buchstabierung der Prämissen von A bis H angeordnet.

```
double[][] scenario1_1 = {
    {0.18, 0.17, 0.16, 0.15, 0.14, 0.13, 0.12, 0.11},
    {0.17, 0.16, 0.15, 0.14, 0.13, 0.12, 0.11, 0.90},
    {0.16, 0.15, 0.14, 0.13, 0.12, 0.11, 0.90, 0.17},
    {0.15, 0.14, 0.13, 0.12, 0.11, 0.90, 0.17, 0.16},
    {0.14, 0.13, 0.12, 0.11, 0.90, 0.17, 0.16, 0.15},
    {0.13, 0.12, 0.11, 0.90, 0.17, 0.16, 0.15, 0.14},
    {0.12, 0.11, 0.90, 0.17, 0.16, 0.15, 0.14, 0.13},
    {0.11, 0.90, 0.17, 0.16, 0.15, 0.14, 0.13, 0.12},
    {0.90, 0.17, 0.16, 0.15, 0.14, 0.13, 0.12, 0.11}
};
```

Listing 7.1: Testscenario 1.1

Im ersten Szenario (siehe Listing 7.1) sind alle Eingangswerte ungefähr bei 0,15. Es ist also eigentlich mit keinem Schaden zu rechnen, auch wenn alle Analysen den "Gutbe-

reich" schon verlassen haben. Ab dem zweiten Eingangsdatensatz ist von H beginnend rückläufig bis A eine Fastverletzung der Warnschwelle von 0,9 eingetragen.

Das zweite Szenario ist das genau Komplement des ersten. Dafür wurde das erste Szenario herangezogen und jeder Eintrag wurde von 1 abgezogen um das genau gegenteilige Spektrum abzudecken.

```
double[][] scenario2 = {
    {1-0.18, 1-0.17, 1-0.16, 1-0.15, 1-0.14, 1-0.13, 1-0.12, 1-0.11},
    {1-0.17, 1-0.16, 1-0.15, 1-0.14, 1-0.13, 1-0.12, 1-0.11, 1-0.90},
    {1-0.16, 1-0.15, 1-0.14, 1-0.13, 1-0.12, 1-0.11, 1-0.90, 1-0.17},
    {1-0.15, 1-0.14, 1-0.13, 1-0.12, 1-0.11, 1-0.90, 1-0.17, 1-0.16},
    {1-0.14, 1-0.13, 1-0.12, 1-0.11, 1-0.90, 1-0.17, 1-0.16, 1-0.15},
    {1-0.13, 1-0.12, 1-0.11, 1-0.90, 1-0.17, 1-0.16, 1-0.15, 1-0.14},
    {1-0.12, 1-0.11, 1-0.90, 1-0.17, 1-0.16, 1-0.15, 1-0.14, 1-0.13},
    {1-0.11, 1-0.90, 1-0.17, 1-0.16, 1-0.15, 1-0.14, 1-0.13, 1-0.12},
    {1-0.90, 1-0.17, 1-0.16, 1-0.15, 1-0.14, 1-0.13, 1-0.12, 1-0.11}
};
```

Listing 7.2: Testscenario 2.1

7.3 Auswertung

Bei der Auswertung der Kombinationen erzeugte eine spezielle Testklasse des JEFIS-Projektes die Datafiles, aus denen mit gnuplot (siehe [13]) die folgenden Grafiken erstellt wurden. Bei zwei unterschiedlichen Szenarien, vier unterschiedlichen Regeln und neun unterschiedlichen Eingangsvektoren haben sich insgesamt 72 verschiedene Grafiken ergeben. Da diese Anzahl den Rahmen der Arbeit deutlich sprengen würde beschränkt sich diese These auf eine kleine Auswahl um bestimme Erkenntnisse hervor zuarbeiten.

7.3.1 Szenario 1.1

In Abbildung 7.2 ist der normale Verlauf der Operatoren zu sehen. Da alle Eingangswerte sehr ähnlich sind, gibt nicht viel zu erkenne. Die Abbildung soll zum Vergleich dienen. Bereits beim ersten Vektor mit einem Ausreißer lässt sich in Abbildung 7.3erkennen, dass bei sehr kleiner und sehr großer andness die Operatoren stark schwanken. Einzig der Quantifizierer mit der schmalen Transitionsbreite zeigt sich für eine andness zwischen ca. 0.25 und 0.8 von dem Ausreißer unbeeindruckt. Ein ähnliches Bild zeigen auch die

Abbildungen 7.4 und 7.5, wobei die erstere den Bereich der stabilen andness auf 0.5 nach oben beschränkt.

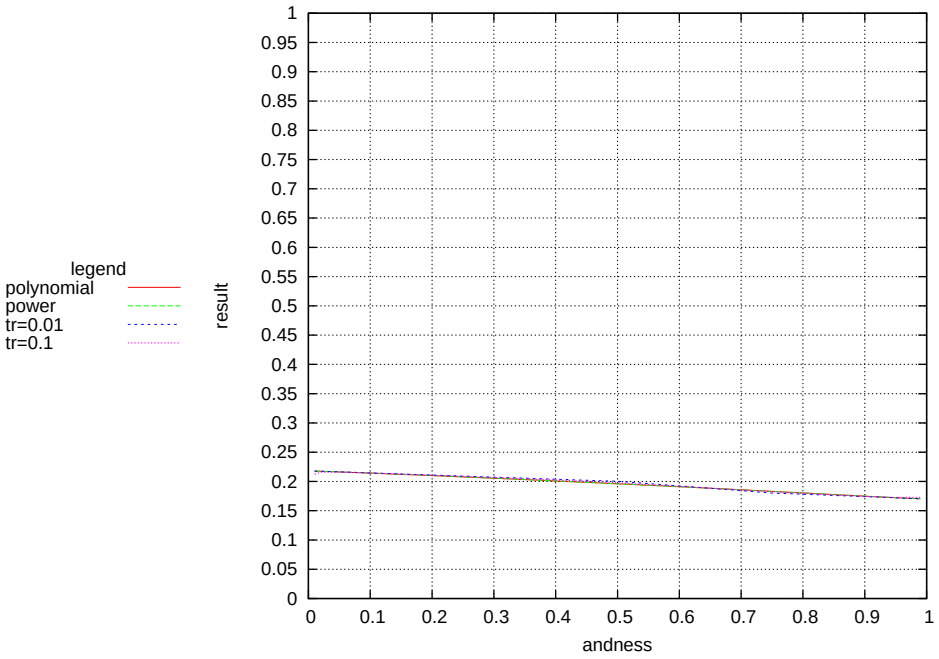

Abbildung 7.2: Scenario 1.1, Vektor 0, Referenzregel.

7.3.2 Szenario 2.1

Bei diesem Szenario war zu beobachten, dass sich von Vektor 0 bis Vektor 3 einschließlich, das Bild der Verteilung der Operatoren nur minimal von Abbildung 7.6 unterscheidet. Im ersten Vektor ist kein Ausreißer vorhanden und bis Vektor 3 befanden sich die Ausreißer alle im großen Oder-Zweig der Referenzregel und fielen wegen ihrer geringen Marginalität nicht ins Gewicht.

Ebenso fiel bei Vektor 4 und Vektor 7 auf, dass alle Grafiken ähnlich waren, bedingt dadurch dass keine der Regeln einen Minterm ohne die Prämissen B und E besitzt. Auch bei diesen beiden Vektoren ist zu erkennen, dass der Quantifizierer mit der schmalen

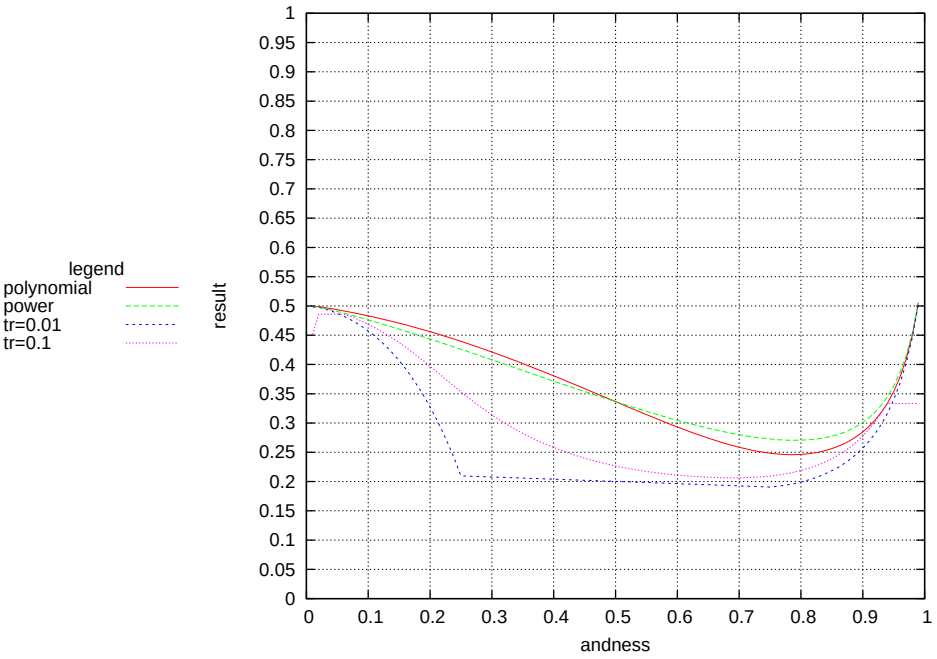

Abbildung 7.3: Scenario 1.1, Vektor 1, Referenzregel.

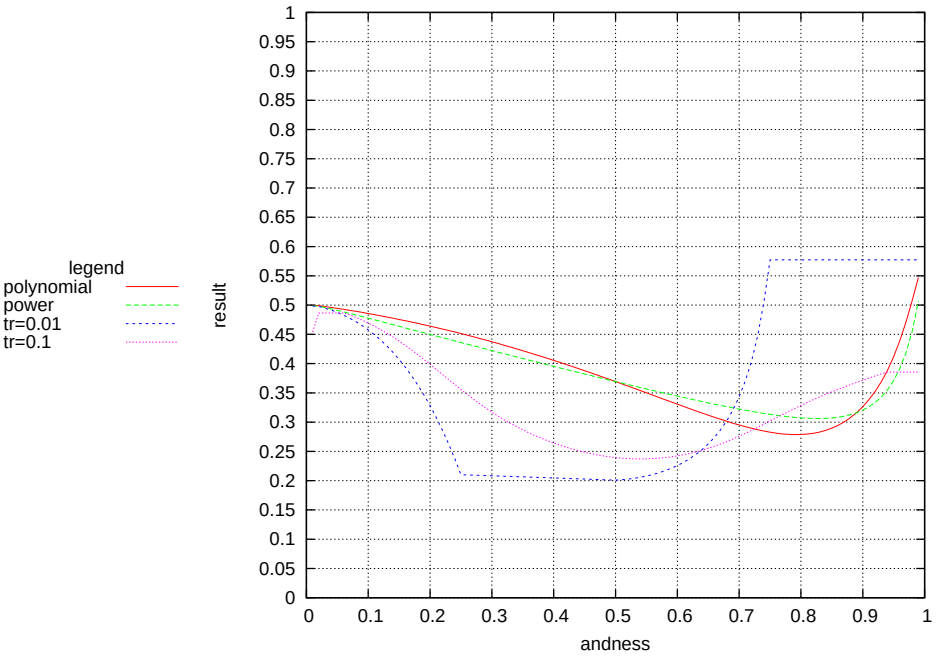

Abbildung 7.4: Scenario 1.1, Vektor 1, “Ohne C”-Regel.

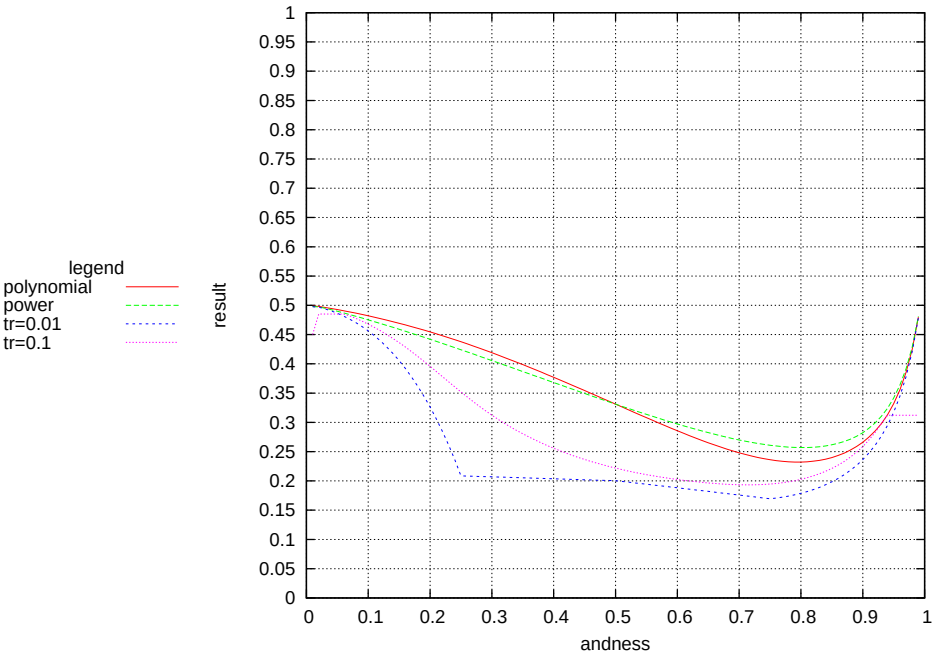

Abbildung 7.5: Scenario 1.1, Vektor 5, “Ohne C”-Regel.

Transitionsbreite von 0.01 für einen sehr breiten andness-Bereich zwischen ca. 0.25 und 0.8 sich von dem Ausreißer sehr unberührt zeigt, siehe Abbildungen 7.7 und 7.8.

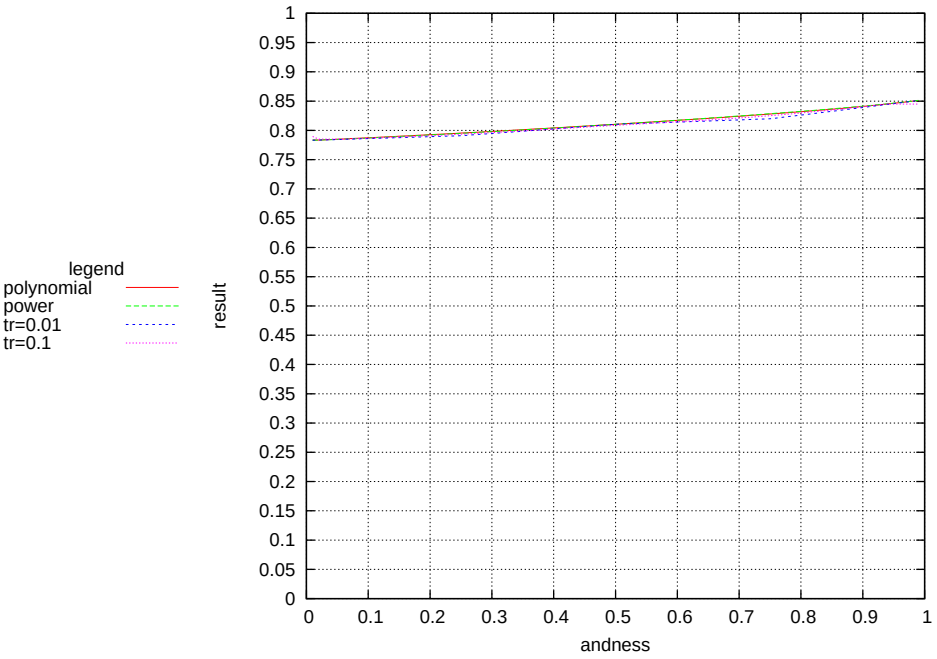

Abbildung 7.6: Scenario 2.1, Vektor 0, Referenzregel.

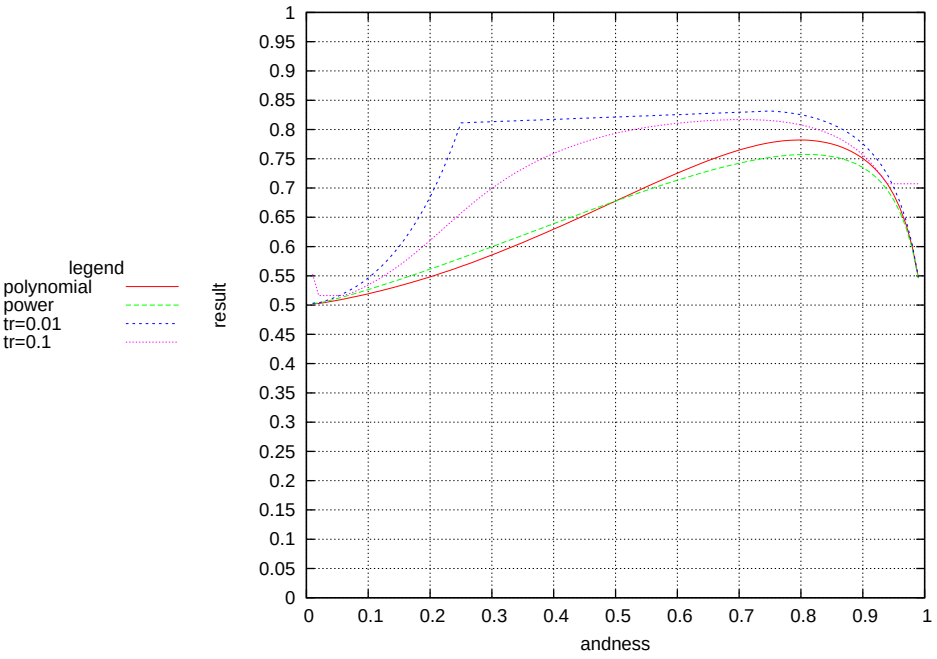

Abbildung 7.7: Scenario 2.1, Vektor 4, Referenzregel.

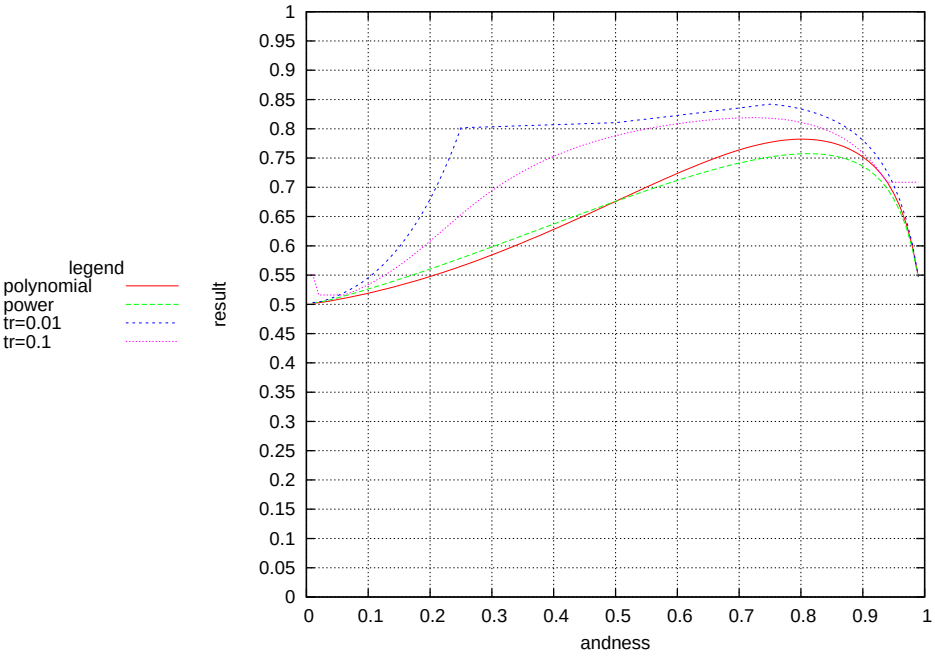

Abbildung 7.8: Scenario 2.1, Vektor 7, Referenzregel.

8 Zusammenfassung

Ziel der dieser Arbeit war die erfolgreiche Entwicklung eines regelbasierten Fuzzy Systems zur Auswertung gegebener Expertenregeln, welche bestimmte Schäden an Kolbenverdichtern beschreiben. In Kapitel 4 wurde gezeigt wie sich diese Regeln in eine disjunktive Normalform überführen lassen. Die Minterme dieser Regeln in disjunktiver Normalform können direkt in die bestehende Musterdatenbank eingegeben werden um deren Qualität zu verbessern.

Das entwickelte Fuzzy System ist in Kapitel 5 beschrieben. Es benutzt die Expertenregeln in disjunktiver Normalform um aus diesen die internen Fuzzy Regeln zu generieren. Durch den Einsatz von Konzepten des Natural Logic Controler (siehe [4]) war es möglich die extrem dünn besetzte Regelbasis auszuwerten. Für die daraus resultierende minimale Fuzzy Partition im Ausgang lässt sich eine geschlossen Form für die Berechnung eines Center of Gravity angeben. Die Anwendung des Maximumprinzips zur Auswertung der Fuzzy Regeln, ließ sich durch die besondere Form der Defuzzyfizierung extrem effizient umsetzen.

Die Untersuchung der Fuzzy Regeln in Kapitel 7 ergab, dass sich der Quantifizierer -Operator als sehr robust gegenüber starken Ausreißern in den Eingangsdaten erwiesen hat. Diese Untersuchung deckt jedoch nur einen sehr kleinen Teil des möglichen Testszenarios ab. Eine tiefergehende Untersuchung wäre besonders im Hinblick auf die Entwicklung von Schadensfällen interessant, wenn der Operator einige chaotische Ausreißer während des Verlaufs der Schadenserkennung ausbügeln könnte.

8.1 Ausblick

Nach Abschluss dieses Projektes, im Rahmen der Masterthesis, ist sicherlich die Einführung in den produktiven Betrieb und die Verifizierung der angestrebten Funktionalität in der Realität ausserhalb kontrollierter Testumgebungen notwendig. Das entwickelte PnPatFS-Modul kann parallel zur existierenden Schadenserkennung auf Basis der neu-

ronalen Netze betrieben werden. Dadurch kann sicher gestellt werden, dass das neue System keinen Schadensfall übersieht, den das alte gefunden hätte.

Da das neue System schon früher Daten über einen mlöglichen Schaden liefern kann, könnte es Sinn machen die Daten auszuwerten, um dann zu versuchen eine Vorhersage über den weiteren Verlauf des Schadensfalls zu treffen.

Bis jetzt existiert noch keine direkte Verbindung zwischen der Schadensmusterdatenbank und dem PnPatFS-Modul. Die Schadensmuster dieser Datenbank ließen sich ähnlich den Mintermen der Fuzzy Regeln auswerten. So könnten diese bereits existierenden Muster ebenfalls unscharf ausgewertet werden.

Abbildungsverzeichnis

Listingverzeichnis

Literaturverzeichnis

[1] Dickerson, J.A.; Kosko, B. “Fuzzy function approximation with ellipsoidal rules”, IEEE Transactions on Systems, Man, and Cybernetics, Part B, Volume 26, Issue 4 (1996), 542 - 560

[2] Zadeh, L. A. “Fuzzy Sets”, Information and Control 8 (1965), 338 - 353

[3] Biewer, B. “Fuzzy-Methoden: Praxisrelevante Rechenmodelle und Fuzzy-Programmiersprachen”, Springer (1997), 85ff

[4] Aceves-Lopez, A.; Aguilar-Martin, J. “A simplified version of mamdanis fuzzy controller: the natural logic controller”, IEEE Transactions on Fuzzy Systems 14 (2006), 16 - 30

[5] Yager, R. R, “On ordered weighted averaging aggregation operators in multi-cirteria decision making”, IEEE Transactions on Systems, Man and Cybernetics 18 (1988), 183 - 190

[6] Mamdani, E.H. ; Assilian S. “An Experiment in Linguistic Synthesis with Fuzzy Logic Controller”, International Journal of Man-Machine Studies 7 (1975) 1 - 13

[7] Hammer, B.; Strickert, M.; Bojer, T. “Generalized relevance LVQ for time series”, Artificial Neural Networks - ICANN’2001, Springer ICANN’2001 (2001), 677 - 683

[8] Koers, C.; Bojer, T.; Hammer B. “Monitoring technical systems with prototype based clustering”, ESANN’2003 proceedings - European Symposium on Artificial Neural Networks (2003), 433 - 439

[9] Oberschelp, W.; Vossen G. “Rechneraufbau und Rechnerstrukturen”, Oldenbourg Wissenschaftsverlag (2006), 18 - 19

Sämtliche folgenden Internetadressen wurden am angegebenen Datum auf Existenz und Korrektheit überprüft!

[10] CppUnit - *http://sourceforge.net/projects/cppunit*
Stand vom 13. Oktober 2008

[11] JEFIS - *http://www.lab4inf.fh-muenster.de/cc/buildresults/Jefis*
Stand vom 13. Oktober 2008

[12] Prognost Systems GmbH - *http://www.prognost.com*
Stand vom 13. Oktober 2008

[13] gnuplot - *www.gnuplot.info/*
Stand vom 13. Oktober 2008