Quantoren sind wie folgt zu behandeln :

$$[\![\forall x : P(x)]\!] = \inf\{[\![P(x)]\!] \mid x\} \tag{4.31}$$

$$[\![\exists x : P(x)]\!] = \sup\{[\![P(x)]\!] \mid x\} \tag{4.32}$$

Für die Gleichheit von zwei Fuzzy-Mengen kann man die Abbildung

$$E^{(X)}_{\text{Luka}} : F(X) \times F(X) \rightarrow [0,1] \tag{4.33}$$

$$(u,v) \mapsto \inf\{1 - |\mu(x) - \nu(x)| \mid x \in X\} \tag{4.34}$$

verwenden, die bezüglich der t-Norm $\top_{\text{Luka}}$ (s. Gleichung (4.13)) eine Gleichheitsrelation darstellt.

4.3 Fuzzy-Regler

Ein Fuzzy-Regler (*fuzzy logic controller*, FLC) ist ein dynamisches System, welches mittels Expertenwissen, das in Regeln zusammengefaßt ist, Entscheidungen durchführt, i.a. die Steuerung von Prozessen übernimmt. In Abbildung 16 ist ein Überblick des Ablaufs der Fuzzy Regelung zu sehen.

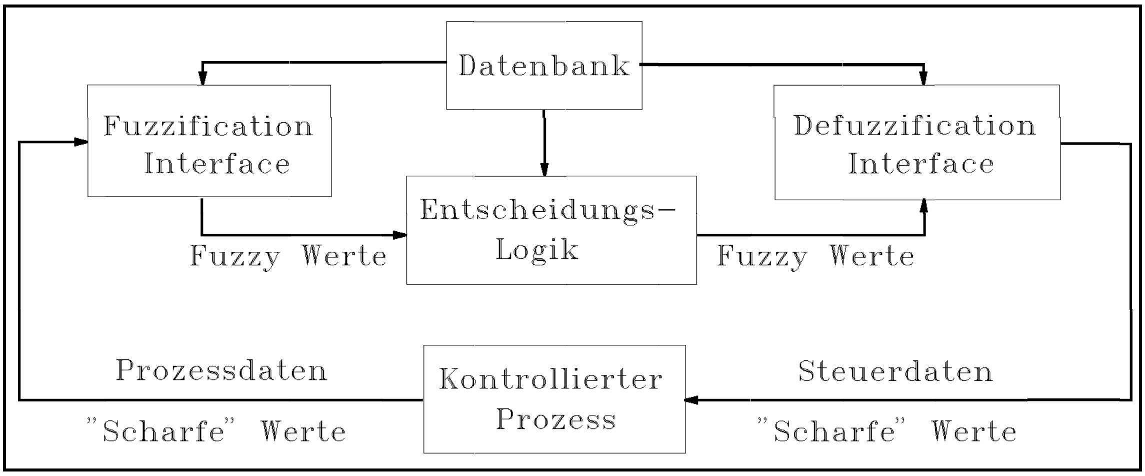

Abbildung 16: Grundaufbau eines Fuzzy Logic Controllers (FLC)

Bei einer umgangssprachlichen Beschreibung eines Prozesses oder eines Handlungsablaufes (z.B. Beschreibung von Krankheitsbildern, Qualitätseinschätzung von Schweißnähten) werden in der Regel keine genauen Werte wichtiger Prozeßgrößen verwendet, sondern unpräzise Angaben. In Fuzzy Systemen dienen linguistische Variable dazu, solche sprachlichen Konzepte auszudrücken.

Eine linguistische Variable ist ein 4-Tupel (N, U, G, M). Dabei ist N der Name der linguistischen Variable, U die Grundmenge der verwendeten Werte, G eine formale Grammatik mit den Wörtern der Sprache $L(G)$ als mögliche Werte der linguistischen Variable und M die Menge der semantischen Regeln, die jedem Wert $x \in L(G)$ die Bedeutung $M(x)$ zuordnen.

Als Beispiel sei die linguistische Variable $N = Alter$ gewählt. Die Grundmenge der Werte ist $U = \mathbb{N}$ und die Sprache $L(G) = \{$`etwa`(10)`, jung, etwa`(11)` oder jung,...`$\}$ wird durch die Grammatik G festgelegt :

<u>`Grundterm`</u> $\longrightarrow$ `etwa`(i) `| älter als` (i) `| ...` $i \in \mathbb{N}$

<u>`Grundterm`</u> $\longrightarrow$ `jung | alt | ...`

<u>`Term`</u> $\longrightarrow$ <u>`Grundterm`</u> `| (`<u>`Term`</u>` und `<u>`Term`</u>`) | sehr `<u>`Term`</u>` | (`<u>`Term`</u>` oder `<u>`Term`</u>`) | ...`

Jedem Grundterm $x \in L(G)$ wird eine Zugehörigkeitsfunktion $M(x)$ zugeordnet (*linguistisches Modellieren*). Die Terme werden mit Fuzzy Operatoren ausgewertet. Dabei ist die Wahl der verwendeten Operatoren ein Problem und die notwendigen Berechnungen sind bei realen Anwendung sehr komplex. Man begnügt sich daher mit einer geeigneten Approximation (*linguistisches Approximieren*).

Die aus dem zu regelnden Prozeß stammenden Daten müssen in Fuzzy Werte gewandelt werden (Fuzzifizierung, *fuzzification*). Dieses Verfahren ist vom Charakter der erfaßten Meßwerte abhängig. Präzise Werte können durch Fuzzy-Singletons repräsentiert werden, verrauschte Werte müssen durch andere Funktionen, z.B. Dreiecksfunktionen oder Trapezfunktionen dargestellt werden.

In der Datenbank und der Entscheidungslogik (s. Abbildung 16) sind die Kontrollregeln für den zu regelnden Prozeß und die Regeln zur Erzeugung bzw. Manipulation von Fuzzy Daten festgehalten. Dort findet die eigentliche Entscheidungsfindung statt. Die Regeln der Form

`if` *(eine Menge von Bedingungen ist erfüllt)*

`then` *(eine Menge von Entscheidungen wird gefällt)*

können mehrere Eingabe bzw. Ausgaben (*multi-input-single-output* – MISO, *multi-input-multi-output* – MIMO) haben. Ein Beispiel einer Regelbasis ist in Tabelle 2 zu sehen.

$$
\begin{array}{ll}
R_1: & \texttt{if } (x \texttt{ is } A_1) \texttt{ and } (y \texttt{ is } B_1) \texttt{ then } (z \texttt{ is } C_1)\ , \\
R_2: & \texttt{if } (x \texttt{ is } A_2) \texttt{ and } (y \texttt{ is } B_2) \texttt{ then } (z \texttt{ is } C_2)\ , \\
 & \vdots \\
R_n: & \texttt{if } (x \texttt{ is } A_n) \texttt{ and } (y \texttt{ is } B_n) \texttt{ then } (z \texttt{ is } C_n)\ .
\end{array}
$$

Tabelle 2: Beispiel einer Regelbasis eines FLC

Die „richtige“ Anzahl der notwendigen Regeln kann nicht im voraus bestimmt werden, zu viele Einflüsse spielen eine Rolle, z.B. Verhalten des Controllers, Effizienz der Berechnungen, Wahl der linguistischen Variablen. Die Regeln sollten auf Konsistenz überprüft werden, um die Wahrscheinlichkeit eines Widerspruchs möglichst klein zu halten.

Mit einer sinnvollen Fuzzy Partitionierung, also der Aufteilung des Wertebereichs eines Meßwertes in linguistische Werte respektive Zugehörigkeitsfunktionen, wird die Genauigkeit der Regelung bestimmt. Eine Aufteilung in beispielsweise drei linguistische Terme ist dabei natürlich „gröber“ als eine Aufteilung mit fünf Termen (s. Abbildung 17). Dabei darf man nicht die Vollständigkeit der Partitionierung vergessen. Das gesamte Intervall sollte mit Zugehörigkeitsfunktionen „abgedeckt“ sein. Abbildungen nehmen eine Diskre-

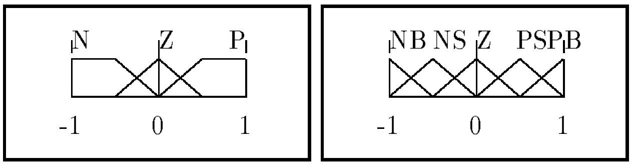

Abbildung 17: Unterschiedliche Fuzzy Partitionierungen eines Intervalls

tisierung (Quantisierung) und/oder Normalisierung der Wertebereiche in geeignete und besser zu handhabende Intervalle vor. Die Bezeichnungen in Abbildung 17 entsprechen denen in der Literatur in ähnlichem Zusammenhang üblicherweise verwendeten. Dabei bedeutet `N` - *negativ*, `Z` - *zero*, `P` - *positiv*, `NB` - *negativ big*, `NS` - *negativ small*, `PS` - *positiv small*, `PB` - *positiv big*.

Bei der Implementierung der Entscheidungslogik (s. Abbildung 16) des FLC, bzw. der Inferenzbildung, gibt es verschiedenen Möglichkeiten der Realisierung. Einige bekannte Verfahren sind in den Kapiteln 4.3.1, 4.3.2, 4.3.3 und 4.3.4 vorgestellt.

Die berechneten Fuzzy Werte müssen, bevor sie als Kontrollsignale an den zu steuernden Prozeß gesendet werden, in scharfe Werte umgewandelt werden (Defuzzifizierung, *defuzzification*). Übliche Verfahren zur Defuzzifizierung sind das

- Maximum-Kriterium (MAX)
  Auswahl des Wertes der Basisvariablen an dem das Maximum der Zugehörigkeitsfunktion liegt, z.B. der erste maximale Wert.

- Mittelwert der Maxima (*mean of maxima* - MOM)
  Auswahl des Mittelwertes aller Maxima.

- Flächenschwerpunkt (*center of area* - COA)
  Auswahl des Flächenschwerpunktes der Zugehörigkeitsfunktionen

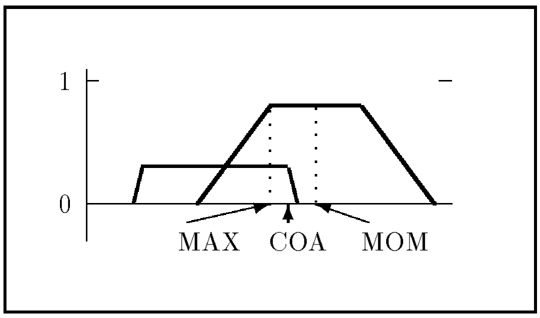

Abbildung 18: Defuzzifizierungs-Strategien

Bei Simulationen schnitt MAX am schlechtesten ab, COA lieferte bessere Ergebnisse als MOM. Darüber hinaus liegen die Stärken von MOM im dynamischen, von COA im statischen Regelverhalten (s. [Lee90a] und [Lee90b]).

4.3.1 Sugeno/Takagi-Regler

Jede Regel der Form

$$R_i \;:\; \texttt{if}\ (x\ \texttt{is}\ A_i)\ \texttt{and}\ (y\ \texttt{is}\ B_i)\ \texttt{then}\ (z = C_{0,i} + C_{1,i}\cdot x + C_{2,i}\cdot y) \tag{4.35}$$

hat als Ergebnis des Bedingungsteils (Erfülltheitsgrad, „Wahrheitswert“)

$$\alpha_i = \min\{\mu_{A_i}(x), \mu_{B_i}(y)\} \tag{4.36}$$

Wir erhalten das Resultat z_i der Regel i zu

$$z_i = C_{0,i} + C_{1,i}\cdot x + C_{2,i}\cdot y \tag{4.37}$$

und damit das Gesamtergebnis,

$$z = \frac{\sum_i \alpha_i \cdot z_i}{\sum_i \alpha_i} \tag{4.38}$$

welches ein „scharfer“ Wert ist und nicht mehr defuzzifiziert werden muß.

In Abbildung 19 ist eine Entscheidung für zwei Regeln mit jeweils zwei Eingängen $\mu_{A_i}(u_0)$ und $\mu_{B_i}(v_0)$ und einem Ausgang μ_{C_i} zu sehen.

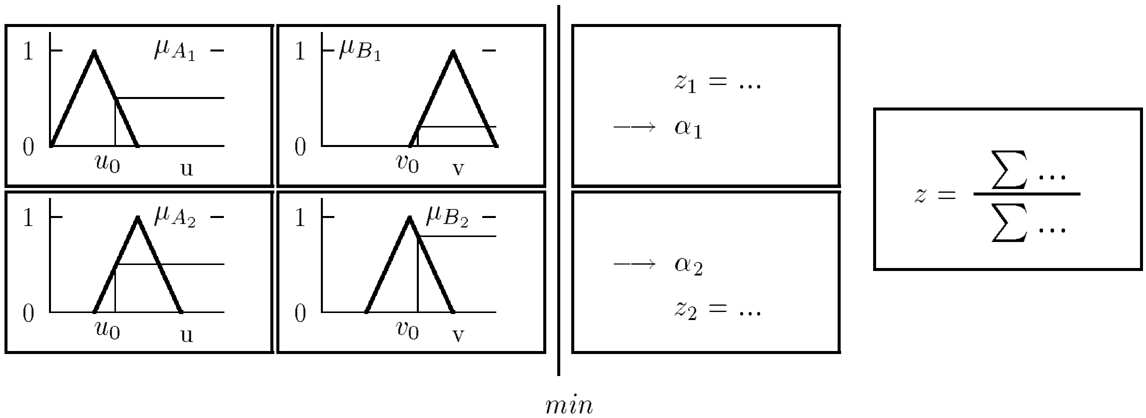

Abbildung 19: Sugeno Fuzzy Reasoning

Es wird eine (nicht-lineare) Input-Output-Relation definiert. Diese Art von Controllern sind besonders gut für die Anwendung von Lernverfahren geeignet. Es wurde gezeigt, daß sich jede stetige (reellwertige) Funktion auf einer kompakten Menge beliebig genau durch einen Controller aus der Klasse der (leicht modifizierten) Sugeno-Regler approximieren läßt.

4.3.2 Mamdami-Regler

Jede Regel der Form

$$R_i \;:\; \texttt{if}\ (x\ \texttt{is}\ A_i)\ \texttt{and}\ (y\ \texttt{is}\ B_i)\ \texttt{then}\ (z\ \texttt{is}\ C_i) \tag{4.39}$$

hat als Ergebnis des Bedingungsteils (Erfülltheitsgrad, „Wahrheitswert")

$$\alpha_{C_i} = \min\{\mu_{A_i}(u_0), \mu_{B_i}(v_0)\}\ . \tag{4.40}$$

Die Lösung der Regel i ergibt sich zu

$$\mu'_{C_i}(w) = \min\{\mu_{C_i}(w), \alpha_{C_i}\}\ . \tag{4.41}$$

Das Gesamtergebnis der Regelbasis berechnet sich dann

$$\mu_C = \max\{\mu'_{C_i}\} \quad \text{, für alle } i. \tag{4.42}$$

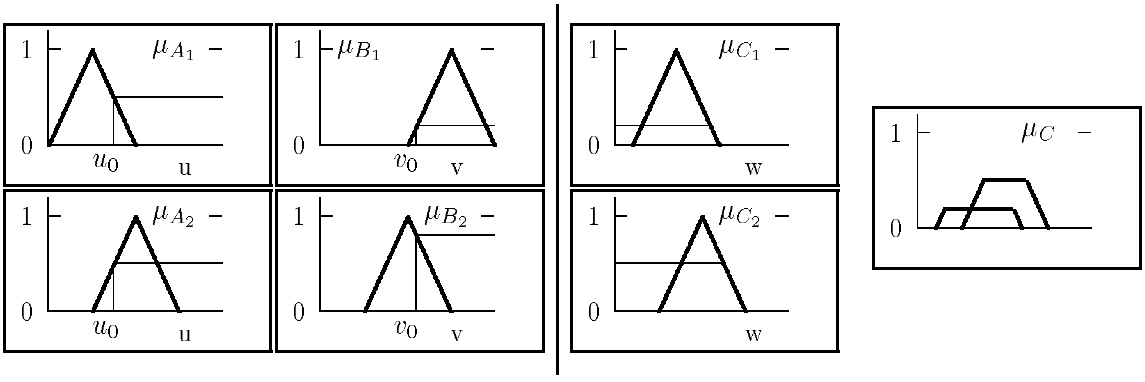

Abbildung 20: Mamdami's Fuzzy Reasoning

In Abbildung 20 ist eine Entscheidung für zwei Regeln mit jeweils zwei Eingängen $\mu_{A_i}(u_0)$ und $\mu_{B_i}(v_0)$ und einem Ausgang μ_{C_i} zu sehen. Die Abbildungen für μ_{A_i} und μ_{B_i} sind in diesem Beispiel die gleichen wie in Abbildung 19.
Das Ergebnis muß noch defuzzifiziert werden (s. Seite 18).

4.3.3 Larsen-Regler

Jede Regel der Form

$$R_i\ :\ \texttt{if}\ (x\ \texttt{is}\ A_i)\ \texttt{and}\ (y\ \texttt{is}\ B_i)\ \texttt{then}\ (z\ \texttt{is}\ C_i) \tag{4.43}$$

hat als Ergebnis des Bedingungsteils (Erfülltheitsgrad, „Wahrheitswert")

$$\alpha_{C_i} = \min\{\mu_{A_i}(u_0), \mu_{B_i}(v_0)\}\ . \tag{4.44}$$

Die Lösung der Regel i ergibt sich zu

$$\mu'_{C_i}(w) = \mu_{C_i}(w) \cdot \alpha_{C_i}\ . \tag{4.45}$$

Das Gesamtergebnis der Regelbasis berechnet sich

$$\mu_C(w) = \max\{\mu'_{C_i}(w)\} \quad \text{, für alle } i. \tag{4.46}$$

In Abbildung 21 ist eine Entscheidung für zwei Regeln mit jeweils zwei Eingängen $\mu_{A_i}(u_0)$ und $\mu_{B_i}(v_0)$ und einem Ausgang μ_{C_i} zu sehen. Die Abbildungen für μ_{A_i} und μ_{B_i} sind in diesem Beispiel die gleichen wie in Abbildung 19.
Auch hier muß das Ergebnis noch defuzzifiziert werden (s. Seite 18).

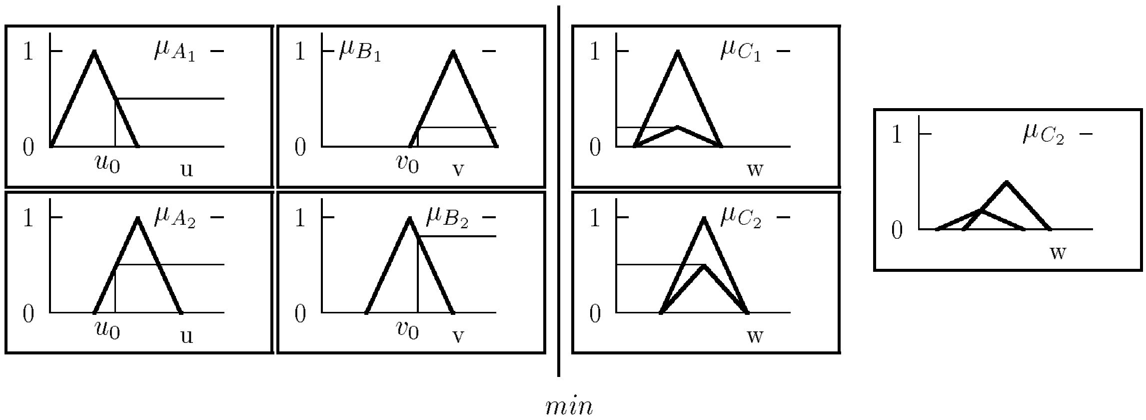

Abbildung 21: Larsen's Fuzzy Reasoning

4.3.4 Tsukamoto-Regler

Jede Regel der Form

$$R_i \ : \ \texttt{if} \ (x \ \texttt{is} \ A_i) \ \texttt{and} \ (y \ \texttt{is} \ B_i) \ \texttt{then} \ (z \ \texttt{is} \ C_i) \tag{4.47}$$

hat als Ergebnis des Bedingungsteils (Erfüllheitsgrad, „Wahrheitswert“)

$$\alpha_{C_i} = \min\{\mu_{A_i}(u_0), \mu_{B_i}(v_0)\} \, . \tag{4.48}$$

Wir erhalten das Resultat z_i der Regel i zu

$$z_i = \mu_{C_i}^{-1}(\alpha_{C_i}) \tag{4.49}$$

und damit das Gesamtergebnis,

$$z = \frac{\sum_i \alpha_{C_i} \cdot z_i}{\sum_i \alpha_{C_i}} \tag{4.50}$$

welches ein „scharfer“ Wert ist und nicht mehr defuzzifiziert werden muß.
Voraussetzung für dieses Verfahren ist die Verwendung von monotonen Zugehörigkeitsfunktionen (s. Abbildung 22).

In Abbildung 22 ist eine Entscheidung für zwei Regeln mit jeweils zwei Eingängen $\mu_{A_i}(u_0)$ und $\mu_{B_i}(v_0)$ und einem Ausgang μ_{C_i} zu sehen.

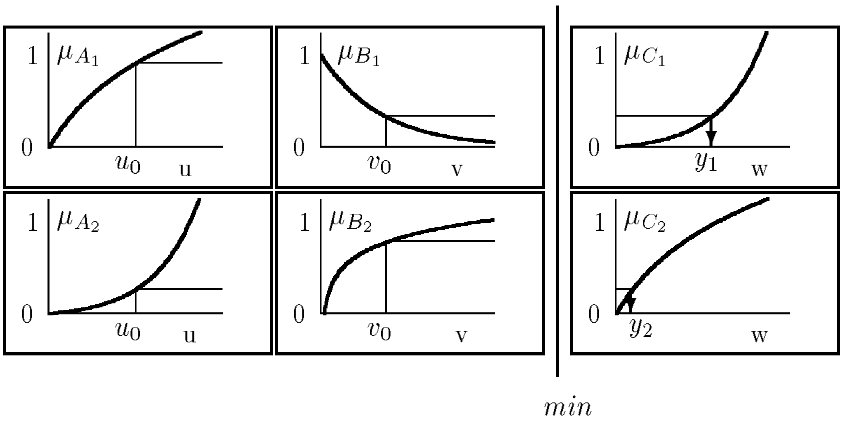

Abbildung 22: Tsukamoto's Fuzzy Reasoning

5 Kooperative Modelle von Fuzzy Systemen und Neuronalen Netzen

Durch die Verwendung von Fuzzy-Logik in Kontrollaufgaben wurde eine neue, mächtige Möglichkeit zur Regelung geschaffen (s. Kapitel 4). Ein Experte kann ohne genaues Wissen über den zu regelnden Prozeß mittels linguistischer Variablen Regeln aufstellen, die er bisher selbst intuitiv zur Regelung benutzte. Ein formales Modell des zu kontrollierenden Systems ist nicht mehr nötig, das Wissen über den Prozeß ist in den Kontrollregeln versteckt (s. Kapitel 4.3). Bei der Verwendung von Fuzzy-Control tritt jedoch ein Problem auf. Ein Experte kann nicht immer genau sagen warum er in einem bestimmten Augenblick eine Aktion startet und von welchen Vorbedingungen er diese Handlung abhängig macht. Außerdem werden die Zugehörigkeitsfunktionen nur subjektiv angegeben und müssen angepaßt werden. Diese Optimierungen sind sehr zeitaufwendig und nur mit heuristischen Methoden oder „trial-and-error“ durchzuführen.

Durch zusätzliche Verwendung von Neuronalen Netzen versucht man die genannten Nachteile zu vermeiden. Diese neue Art von Controllern nennt man *Fuzzy-Neuro-Controller* oder *Neuro-Fuzzy-Controller*. Man versucht die Vorteile Neuronaler Netze für die Regler zu nutzen. Das ist zum Ersten die massive Parallelität der Einheiten, was schnellere Berechnungen verspricht als bei sequentiellen Berechnungsweisen. Zweitens, auf Grund der hohen Parallelität ist ein Ausfall einzelner Neuronen zu verkraften, ein Neuronales Netz ist fehler-toleranter als herkömmliche Systeme. Drittens, wahrscheinlich der größte Vorteil, sind Neuronale Netze lernfähig und können Korrekturen vornehmen (s. Kapitel 3).

Neuronale Netze	Fuzzy Systeme
Vorteile : • Lernen aus Beispielen • Wettbewerbslernen • kein mathematisches Modell notwendig • kein Regelwissen notwendig	Vorteile : • kein Mathematisches Modell notwendig • a-priori (Regel-)Wissen nutzbar • einfach zu interpretieren • einfach zu implementieren
Nachteile : • Black Box • i.a. kein (Regel-)Wissen extrahierbar • Korrektheit nicht überprüfbar • Anpassung an veränderte Parameter eventuell schwierig (Lernvorgang wiederholen) • kein a-priori Wissen verwendbar	Nachteile : • Regelwissen muß verfügbar sein • nicht lernfähig • keine formalen Methoden für „Tuning“ • semantische Probleme bei der Interpretation „getunter“ Controller • Anpassung an veränderte Parameter eventuell schwierig

Tabelle 3: Vergleich von Neuronalen Netzen und Fuzzy Systemen

Grundsätzliche kann man in zwei Ansätze unterscheiden :

- *Kooperative Systeme*
  Das Fuzzy System und das Neuronale Netz sind streng voneinander getrennt und

übernehmen unabhängig voneinander Teilaufgaben des Kontrollablaufs.

- *Hybride Systeme*
  Es ist auf den ersten Blick keine klare Grenze zwischen Fuzzy-Contoller und Neuronalem Netz zu bestimmen. Die beiden Blöcke sind ineinander „verschmolzen“ (s. Kapitel 6).

Wie schon erwähnt, kann man bei Kooperativen Systemem streng zwischen Neuronalem Netz und Fuzzy-Controller trennen. Da Neuronale Netze relativ einfach lernen können, Muster richtig zu verarbeiten (s. Kapitel 3), bietet es sich an, damit beispielsweise die Regelbasis und Zugehörigkeitsfunktionen zu optimieren. Ein Experte gibt seine subjektive Einschätzung der Zugehörigkeitsfunktionen an, und ein Neuronales Netz optimiert, während der Regelung, diese Funktionen. Eine Veränderung der Regelbasis ist mit einem Faktor möglich, der den Einfluß jeder Regel auf das Gesamtergebnis steuert. Der Faktor 0 löscht gewissermaßen eine einzelne Regel. Als einführende Artikel zu diesem Thema sind [ERV91], [COB92], [HTHW92], [MWK+90], [NTUG91], [OWK+92] geeignet. Der Controller aus [CLH92] wird in Kapitel 5.1 ausführlicher erläutert.

Neuronale Netze können auch dazu eingesetzt werden aus gegebenen Ein-Ausgabedaten Regeln und Zugehörigkeitsfunktionen zu extrahieren und somit die Grundlage für Fuzzy-Control zu legen. Dazu kann ein Experte bei einer Steuerungsaufgabe beobachtet werden und seine Aktionen auf gegebene Signale aufgezeichnet werden. Diese Daten werden dann zum Lernen des Neuro-Fuzzy-Controllers verwendet. Näheres dazu in [YO92] respektive Kapitel 5.2.

Weitere interessante, einführende Informationen sind in [GG91], [YF92], [CH92], [Gup92], [Hal91], [Hay92], [HCB92], [IOT92], [Kel90], [KT92b], [KT92a], [KYT92], [LL92], [Ped91], [Tak90], [Wer92a], [Wer92b] und [Yag92] zu finden.

5.1 Chen-Regler

Ziel dieses Controllers von [CLH92] ist es, die Zugehörigkeitsfunktionen der Konsequenzteile der Regelbasis und die Stärke der Regeln zu optimieren. Das System besteht aus den Blöcken (s. Abbildung 23) Fuzzy-Regler (FLC) und Neuronalem Netz zum Anpassen der Regelbasis bzw. der Zugehörigkeitsfunktionen (*neuron adaptive elements* - NAE).

5.1.1 Der Fuzzy-Regler

Der Ablauf der Berechnung des Fuzzy Stellwertes erfolgt entsprechend dem in Kapitel 4.3.3 vorgestellten Verfahren. Die einzelnen Regeln haben nur eine Ausgabevariable (MISO)

$$\texttt{if}\ \textit{(Bedingungen)}\ \texttt{then}\ (U\ \texttt{is}\ U_i)$$

mit U als zu berechnende Kontrollaktion. U_i ist die Zugehörigkeitsfunktion des Konsequenzteils der Regel i und hat den Mittelpunkt u_i, es handelt sich um Dreiecksfunktionen mit einer Basislänge von 1. Die Basislänge hat dabei auf Grund der vereinfachten Defuzzifizierung keinen Einfluß auf die Ergebnisse des Reglers.

Der berechnete Fuzzy Wert wird dann mit der *center of area*-Methode (COA) (s. Seite

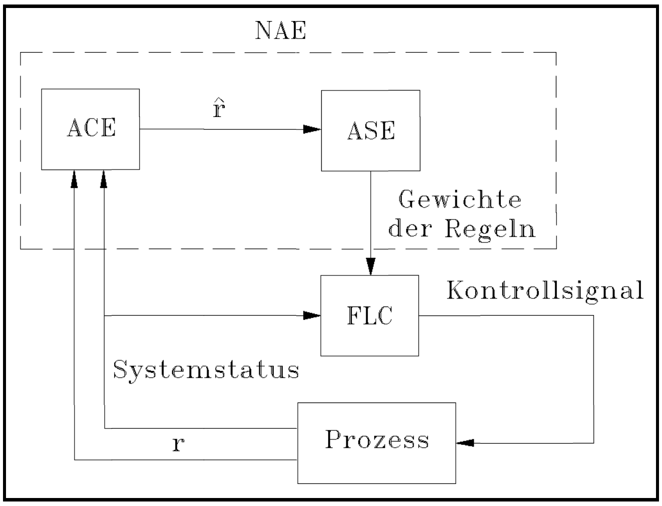

Abbildung 23: Aufbau des kooperativen Systems von Chen

18) in einen scharfen Ausgabewert u' defuzzifiziert

$$u' = \frac{\int C'(u)u\,du}{\int C'(u)\,du}\ , \tag{5.1}$$

zur Vereinfachung wird jedoch

$$u' = \frac{\sum u_i \mu_i}{\sum \mu_i} \tag{5.2}$$

verwendet, wobei $C'(u)$ das Gesamtergebnis aller Regeln und μ_i der „Wahrheitswert" der betreffenden Regel i ist.

5.1.2 Das Neuronale Netz

Das NAE besteht aus zwei Einheiten, dem *adaptive critic element* (ACE) und dem *associative search element* (ASE).

Adaptive Critic Element

Das ACE bekommt vom Prozeß das externe Reinforcement-Signal r, das das Verhalten des Prozesses beschreibt. Im ACE wird daraufhin eine Vorhersage des späteren Systemstatus gemacht und ein internes Reinforcement-Signal $\hat{r}$ erzeugt. Mittels diesem Wert lernen beide Einheiten Vorhersagen zu treffen und Regeln des FLC zu „bestrafen" bzw. „belohnen".

Das externe Reinforcement ist folgendermaßen definiert

$$r = \begin{cases} 1 & \text{, bei Verlassen eines Grenzbereiches } [a,b] \\ 0 & \text{, im Grenzbereich} \\ -\frac{1}{|a-b|}\left(\frac{1}{N}\sum_{k=1}^{N} |E(kT)|\right) & \text{, } T \text{ ist ein Zeitschritt, } NT \text{ die Lernperiode} \end{cases} \tag{5.3}$$

$E(\cdot)$ ist dabei der Fehler der Systemausgabe zu einem bestimmten Zeitpunkt.
Das ACE erhält außerdem die aktuellen Systemstatusvariablen als Eingabe (s. Abbildung 23). Das interne Reinforcement-Signal $\hat{r}$ für das ASE wird mit der Vorhersage $p(t)$ berechnet

$$\hat{r} = \gamma\, p(t) - p(t-1) \qquad , \; 0 \leq \gamma < 1 \; . \tag{5.4}$$

$p(t)$ selbst errechnet sich zu

$$p(t) = \sum_{i=1}^{n} v_i(t)\mu_i(t) \; . \tag{5.5}$$

Dabei ist $\mu_i(t)$ der „Wahrheitswert" der Regel i.
Die Gewichte v_i des ACE werden so verändert, daß $p(t)$ zu einer genauen Vorhersage konvergiert :

$$v_i(t+1) = v_i(t) + \beta\, \hat{r}\, \bar{\mu}_i(t) \tag{5.6}$$

$$\bar{\mu}_i(t) = \lambda\, \bar{\mu}_i(t-1) + (1-\lambda)\mu_i(t) \tag{5.7}$$

$0 \leq \beta < 1$ und $0 \leq \gamma < 1$ sind Lernraten und $\bar{\mu}_i(t)$ ein Maß der alten Werte von μ_i.

Associative Search Elements

Das ASE soll mit dem internen Reinforcement-Signal $\hat{r}$ die „Stärke" der Regeln der Regelbasis kontrollieren.

Die Kontrollaktionen des FLC sind durch die u_i bei der Defuzzifizierung festgelegt und mit den Gewichten des ASE in Beziehung durch

$$u_i = FL \cdot \tanh(kw_i) \, , \tag{5.8}$$

mit $k \in I\!R$ und FL als Grenzwert der Ausgabefunktion.
Die Gewichte des ASE werden mit

$$w_i(t+1) = w_1(t) + \alpha\; sign(E)|e_i(t)\,\hat{r}| \tag{5.9}$$

aktualisiert, wobei α die Lernrate und E der Fehler der Systemausgabe ist. Der „Eignungskoeffizient" e_i der Regel i wird durch

$$e_i(t) = \delta\, e_i(t-1) + (1-\delta)\, \mu_i(t) u_i(t) \tag{5.10}$$

verändert, mit $0 \leq \delta < 1$ als Abklangfaktor und $u_i(t)$ als der Mittelpunkt der Zugehörigkeitsfunktion des Konsequenzteils von Regel i.

5.1.3 Zusammenfassung

Zur Realisierung des Fuzzy-Reglers wird ein Standardverfahren verwendet, jedoch sind als Zugehörigkeitsfunktionen nur Dreiecksfunktionen mit fester Basislänge zugelassen. Diese Restriktion soll keinen Einfluß auf die Ergebnisse des Reglers haben. Bei der Defuzzifizierung wird auf das COA-Verfahren (s. Seite 33) zurückgegriffen. Die verwendeten Regeln sind vom Typ MISO, berechnen somit nur eine Ausgabevariable.

Das „normale“ Verhalten des FLC wird durch das NAE kontrolliert. Ein Reinforcement-Signal dient zur Vorhersage des Systemzustandes. Mit diesem Wert wird ein internes Reinforcement berechnet, welches zur Veränderung der Gewichte der Regeln, Eignungskoeffizienten genannt, und zur Verschiebung der Basispunkte der Zugehörigkeitsfunktionen dient.

Das Einsatzgebiet dieses Controllers ist für die Regelung dynamischer Systeme oder nichtlinearer Probleme angegeben. Für die in [CLH92] aufgeführten Beispiele, drei Simulationen dynamischer Systeme, ist zwar die Größe der Regelbasis und die Anzahl der linguistischen Variablen des FLC angegeben, eine Aussage über die benötigte Komplexität der Neuronalen Netze ist jedoch nicht zu finden. Weiterhin sind keine Ausführungen über die Wahl der Paramater, speziell die verschiedenen Lernraten und die Initialwerte der Neuronalen Netze, gemacht worden, eine Implementierung dieses Verfahrens würde also eine mehr oder weniger lange Testphase nach sich ziehen.

5.2 Yi-Regler

Das in [YO92] vorgestellte System (s. Abbildung 24) soll zur automatischen Erzeugung einer Regelbasis eines Fuzzy Controllers dienen. Das System setzt sich aus vier Komponenten zusammen (s. Abbildung 24):

- Regel-Erzeugung (s. Kapitel 5.2.1)
- Fuzzy Regler (s. Kapitel 5.2.2)
- Benutzer-Schnittstelle
- Wissensbasis

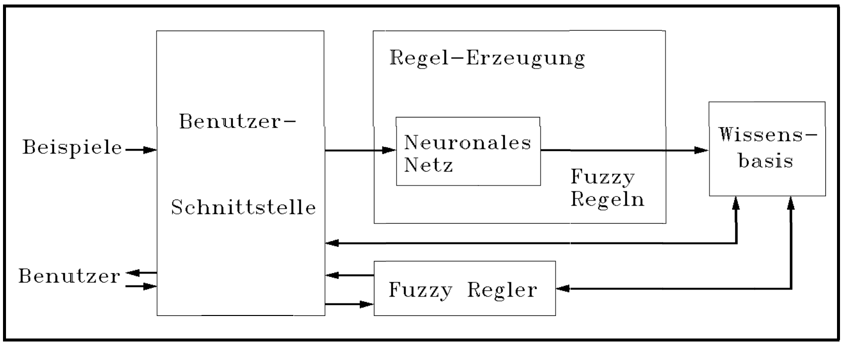

Abbildung 24: Aufbau des Systems von Yi

Durch die Präsentation von Ein-Ausgabe-Beispielen lernt das Neuronale Netz Beziehungen und erzeugt Fuzzy Regeln die mit einer *relativen Wichtigkeit* jedes Attributes des Beziehungsteils der Regel versehen sind. Mit diesen Regeln wird eine Wissensbasis aufgebaut. Über die linguistischen Variablen wird mit dem System gearbeitet. Im folgenden werden die Hauptkomponenten des Modells erläutert.

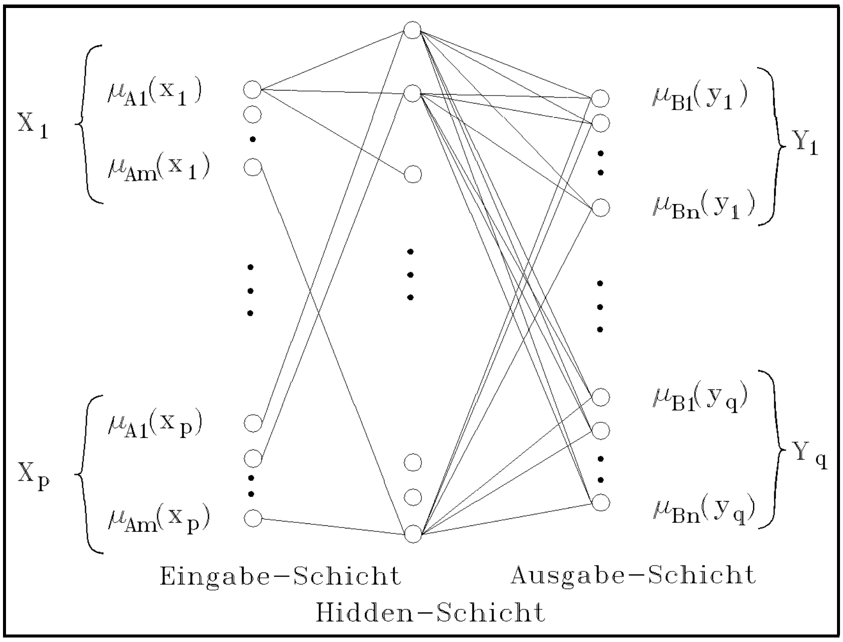

Abbildung 25: Neuronales Netz mit drei Schichten

5.2.1 Regel-Erzeugung

Die vom System erzeugten Regeln sind vom Typ *multi-input-multi-output* (MIMO). Für die Eingabevariablen X_i, $i = 1, ..., p$, werden m Zugehörigkeitsfunktionen μ_{A_j} oder einfacher A_j, $j = 1, ..., m$, respektive n Zugehörigkeitsfunktionen μ_{B_k} oder einfacher B_k, $k = 1, ..., n$, für die Ausgabevariablen Y_l, $l = 1, ..., q$, angegeben. Die Anzahl der möglichen Regeln ist damit $m^p * n * q$. Die Fuzzy Regeln werden durch das dreischichtige Neuronale Netz (s. Abbildung 25) mittles Backpropagation erzeugt.

Für jede Variable des Bedingungsteils einer Regel werden die linguistischen Werte festgelegt. Dadurch entsteht die Eingabe-Schicht des Netzes mit $m * p$ Neuronen. Die versteckte Schicht ist mit jedem Eingabe-Neuron verbunden, er besteht somit aus m^p Neuronen. Die Verbindungen stellen die Bedingungsteile der Regeln dar, m^p Regeln sind möglich. Die Bedingungsteile der Regeln sehen folgendermaßen aus (vgl. Abbildung 25) :

1.	if (X_1 is A_1) and (X_2 is A_1) ... (X_p is A_1)
2.	if (X_1 is A_1) and (X_2 is A_1) ... (X_p is A_2)
	$\vdots$
$m^p - 1$	if (X_1 is A_1) and (X_2 is A_m) ... (X_p is A_m)
m^p.	if (X_1 is A_m) and (X_2 is A_m) ... (X_p is A_m)

Die Ausgabe-Schicht hat $n * q$ Neuronen, für jede der q Ausgabevariablen gibt es n linguistische Werte. Alle Neuronen der Hidden-Schicht sind vollständig mit denen der Ausgabe-Schicht verbunden. Die Gewichte kontrollieren die erzeugten Regeln. Die Ausgabe des Netzes ist der Zugehörigkeitsgrad jedes Attributes im Konsequenzteil der Regeln. Jede Regel wird in der Wissensbasis wie folgt abgelegt :

`Regel`	*nummer :*		
`if`	$input\_var_1$	`is`	$(fuzzy\_variable\_1) \cdot \alpha_1$
`and`	$input\_var_2$	`is`	$(fuzzy\_variable\_2) \cdot \alpha_2$
	:		
`and`	$input\_var_p$	`is`	$(fuzzy\_variable\_m) \cdot \alpha_k$
`then`	$output\_var_1$	`is`	$(fuzzy\_variable\_1)$
	:		
`and`	$output\_var_q$	`is`	$(fuzzy\_variable\_n)$

Die Variabel α_i, $1 \le i \le k$, kennzeichnet die relative Wichtigkeit eines Attributes des Bedingungsteils. Der Algorithmus zur Erzeugung der Regeln ist in Tabelle 4 angegeben. Die Wichtigkeit einer Regel wird aus den Gewichten der Verbindungen der Hidden-Schicht mit der Ausgabe-Schicht bestimmt

$$\text{Wichtigkeit} = \frac{w_{ij}}{\max(w_{ij})}\,, \tag{5.11}$$

wobei w_{ij} das Gewicht der Verbindung des Hidden-Neuron i mit dem Ausgabe-Neuron j ist.

Die relative Wichtigkeit ($\in [0,1]$) kennzeichnet den Einfluß eines Attributes des Bedingungsteils einer Regel auf das gesamte Ergebnis der Regel.

Schritt 1:	an Hand der gegebenen Ein-Ausgabe-Beispiele werden die Zugehörigkeitsfunktionen $\mu_{A_l}(X_i)$ und $\mu_{B_k}(X_j)$ festgelegt und dementsprechend ein Neuronales Netz erzeugt
Schritt 2:	mittels Backpropagation wird gelernt
Schritt 3:	die Gewichte zwischen Hidden- und Ausgabe-Schicht kennzeichnen die *Wichtigkeit* einer Regel, die an Hand dieser *Wichtigkeit* ausgewählt werden
Schritt 4:	in den ausgewählten Regeln werden die Gewichte zwischen Eingabe- und Hidden-Schicht normalisiert und als *relative Wichtigkeit* zu jedem Attribut des Bedingungsteils hinzugefügt
Schritt 5:	Fuzzy Regeln mit relativer Wichtigkeit werden in der Wissensbasis gespeichert

Tabelle 4: Algorithmus zur Regelerzeugung

5.2.2 Fuzzy Entscheidungsfindung

Die bestimmte relative Wichtigkeit eines Attributes des Bedingungsteils beeinflußt die Zugehörigkeitsfunktion des Attributes. Die Fuzzy Menge $F(x)$ verändert sich mit der relativen Wichtigkeit α zu

$$\forall (x \in U) : F^{\alpha}(x) = \max\{(1-\alpha), F(x)\}\,. \tag{5.12}$$

Schritt 1:	die veränderte Fuzzy Menge A_i^α errechnet sich mit der relativen Wichtigkeit α
Schritt 2:	definiere den Schwellwert τ für SM
Schritt 3:	berechne jedes DM^i und SM^i; ist die Regel mit dem `and`-Operator verbunden, so errechnet sich SM zu SM$=\min(\mathrm{SM}^i)$
Schritt 4:	ist SM$>\tau$ wird die Regel ausgewählt
Schritt 5:	berechne MF auf Grund von SM und erzeuge das Ergebnis B' durch Anwendung von MF auf B, B'=MF(B)
Schritt 6:	hat mehr als eine Regel bei Schritt 4 gefeuert, werden ihre Ergebnisse B' mit dem max-Operator verknüpft
Schritt 7:	defuzzifiziere das Fuzzy Ergebnis B'

Tabelle 5: Algorithmus zur Entscheidungsfindung

Zur Bestimmung der Entscheidung wird ein mit AARS (*approximate analogical reasoning schema*) bezeichnetes Verfahren verwendet, das mit einem Übereinstimmungsfaktor SM (*similarity measure*) und einer Modifizierungsfunktion MF (*modification function*) arbeitet. Überschreitet SM einen Schwellwert, also stimmt der Bedingungsteil der Regel mit dem vorliegenden Fall hinreichend überein, feuert die Regel. In diesem Fall wird MF basierend auf SM berechnet und der Konsequenzteil der Regel verändert.
Bei einem vorliegenden Fall

Regel	`if`	X `is` A	`then`	Y `is` B
Fakt		X `is` A'		
Schluß				Y `is` B'

läuft die Berechnung des AARS in drei Schritten ab.

a. $\mathrm{DM}(A, A') = 1 - \sup_{x\in A}\{\mu_{A'\cap A(x)}\}$
dabei ist $\mu_{A'\cap A(x)} = \min\{\mu_{A'(x)}, \mu_{A(x)}\}$

b. SM$= \frac{1}{1+\mathrm{DM}}$, SM$\in [0, 1]$

c. Zur Berechnung des Ergebnisses gibt es zwei Möglichkeiten :
$$\mathrm{MF}_1 : B' = B * \mathrm{SM}$$
$$\mathrm{MF}_2 : B' = \min\{1, \frac{B}{\mathrm{SM}}\}$$

Der Algorithmus zur Entscheidungsfindung mit relativer Wichtigkeit ist mit sieben Schritten in Tabelle 5 angegeben. Die *richtige* Wahl von SM, MF und des Schwellwertes τ ist stark vom verwendeten Wertebereich abhängig. In Anwendungen ist das Verhalten bei unterschiedlichen Werten auszuprobieren. Der Schwellwert τ kontrolliert die Anzahl der feuernden Regeln. Bei einem Wert von $\tau = 1$ verhält sich das System wie ein konventionelles Expertensystem, bei $\tau = 0$ erhält man einen herkömmlichen Fuzzy Controller.

5.2.3 Zusammenfassung

Für gesammelte Ein-Ausgabe-Daten eines Kontrollproblems werden Regeln für einen FLC erzeugt. Die Partitionierungen der Wertebereiche der Eingabevariablen, also die einzelnen Zugehörigkeitsfunktionen, müssen jedoch angegeben werden. Die erzeugten Regeln sind mit einem Parameter, der relativen Wichtigkeit, versehen und vom Typ MIMO.

Für p Eingabevariable müssen jeweils m Zugehörigkeitsfunktionen modelliert werden, die Eingabe-Schicht besteht somit aus $m * p$ Neuronen. Die Hidden-Schicht ist vollständig mit der Eingabe-Schicht verbunden und modelliert damit m^p Bedingungsteile von Regeln. Die Ausgabe-Schicht besteht aus $n * q$ Neuronen, für q Ausgabevariable werden n linguistische Werte vom Benutzer festgelegt. Dieses dreischichtige Neuronale Netz erzeugt also die in den Ein-Ausgabe-Daten versteckten Regeln und legt sie in in einer Wissensbasis ab, auf die dann zugegriffen werden kann.

Eine Einschränkung des Systems ist, daß für verschiedene Variable eine gleiche Partitionierung des Wertebereichs vorgenommen werden muß. Durch die Angabe der Zugehörigkeitsfunktionen der Variablen wird weiterhin dem Benutzer Vorwissen über das zu regelnde Problem, bzw. dessen beschreibende Variablen, abverlangt, welches nicht immer in diesem Maße vorhanden ist.

Ist die Regelbasis des Fuzzy Controllers einmal erzeugt, werden keine Veränderungen mehr vorgenommen. Eine Feinabstimmung des Systems auf das Kontrollproblem ist nicht vorgesehen. Dies setzt voraus, daß die gewählten Beispiel-Daten das Kontrollsystem „ausreichend gut“ repräsentieren und damit auch eine gut abgestimmte Regelbasis erzeugt werden kann. Dies ist in vielen Fällen im voraus nicht mit Sicherheit zu sagen.

6 Hybride Modelle von Fuzzy Systemen und Neuronalen Netzen

Wie in Kapitel 5 erwähnt, ist die exakte Trennung zwischen Neuronalem Netz und Fuzzy-Regler bei hybriden System in vielen Fällen nicht zu vollziehen. Vielmehr modelliert in der Regel ein Neuronales Netz einen FLC. Dabei repräsentieren die Gewichte der Verbindungen und die Schwellwerte der Neuronen die Parameter des Controllers, wie beispielsweise Zugehörigkeitsfunktionen, Gewichte der Regeln u.s.w.

In [KWOA92] wird ein Algorithmus vorgestellt, der dazu dient ein Fuzzy-Modell eines Reglers in ein Neuronales Netz abzubilden. Dieser Hybride Controller lernt bei der Regelung des Prozesses und kann dann wiederum als ein (verbessertes) Fuzzy-Modell interpretiert werden. Näheres in Kapitel 6.1.

In Kapitel 6.2 wird ein aus [HNW89] stammender Neuro-Fuzzy-Controller beschrieben. Dieser Regeler lernt an Hand von Ein-Ausgabe-Tupeln Zugehörigkeitsfunktionen und Regeln.

In [Ber92] und Kapitel 6.3 wird eine Reglersystem vorgestellt, das mit Reinforcement Learning (s. Kapitel 3.3) arbeitet. Der Controller wurde in Kapitel 7 mit einer Anwendung, der Simulation eines invertierten Pendels, untersucht.

Als letzter ausführlich beschriebener Regler ist in Kapitel 6.4 ein Neuro-Fuzzy-Controller aus [NK92a] aufgeführt.

Weitere interessante, einführende Informationen sind in [CH92], [GR92], [GPK89], [GQ91], [GG91], [Gup92], [Hal91], [Hay92], [HCB92], [HFU92a], [IOT92], [Jan91], [Kel90], [KT92b], [KT92a], [KYT92], [LL92], [MY92], [MSF91], [NHW92], [PC92], [Ped91], [Tak90], [Wer92a], [Wer92b], [Yag92] und [YF92] zu finden.

Weiterführende Bemerkungen stehen in [BTP92]. Es wird ein Kohonen Netzwerk vorgestellt, das intern mit Fuzzy-Werten arbeitet und Nachteile des konventionellen Modells vermeidet.

In [KL92a] und [KL92b] wird ein modifiziertes Verfahren zur Fuzzy-Entscheidungsfindung vorgestellt und die Umsetzung in ein Neuronales Netz erläutert.

Genetische Algorithmen dienen in [Md92] dazu, veränderte Umgebungsparameter in den Neuro-Fuzzy-Controller einzubeziehen.

Der aus [HFOU90] und [HFU92b] stammende Regler findet und optimiert selbstständig Regeln und Zugehörigkeitsfunktionen.

6.1 Kawamura-Regler

In [KWOA92] wird ein Neuro-Fuzzy System vorgestellt, dessen Arbeitsweise in Abbildung 26 skizziert ist und in fünf Schritten beschrieben werden kann :

1. Zugehörigkeitsfunktionen, Fuzzy-Regeln und das „Verhalten“ der Fuzzy-Operation, beispielsweise „`und`“, werden spezifiziert. Ein Defuzzifizierungsverfahren, in dem in [KWOA92] vorgestellten Beispiel wird das COA-Verfahren (*center of area*, s. auch Kapitel 4.3) verwendet, wird festgelegt.

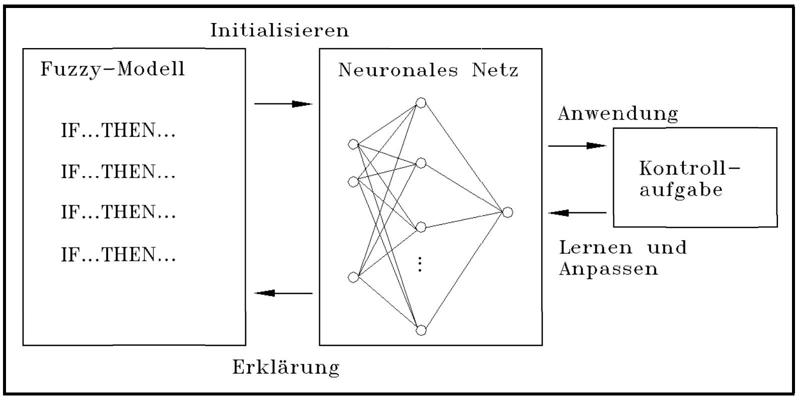

Abbildung 26: Kawamuras Neuro-Fuzzy System

2. Aus den Parametern der Operationen und Regeln wird ein Neuronales Netz initialisiert.

3. Das Neuronale Netz wird zur Regelung des Kontrollsystems verwendet.

4. Das Neuronale Netz lernt aus Kontrolldaten des Systems und optimiert seine Gewichte.

5. Nach dem Lernen können die Einheiten und Gewichte des Neuronalen Netzes wieder als Zugehörigkeitsfunktionen und Fuzzy-Regeln zum besseren Verständnis der internen Arbeitsweise interpretiert werden.

In Abbildung 27 ist ein strukturiertes Neuronales Netz zu sehen, welches Fuzzy-Regeln modelliert (s. Anwendung in [KWOA92]). Die unterschiedlichen Ausgabefunktionen der Neuronen entsprechen denen in den Gleichungen (3.12) und (3.15)

$$S(s) = \begin{cases} \frac{1}{1+\exp(s)} & \text{sigmoides Neuron} \\ s & \text{lineares Neuron} \end{cases} \qquad s = \sum_i x_i w_i \tag{6.1}$$

Das Fuzzy-Modell wird folgendermaßen auf das Neuronale Netz abgebildet :

- Die Neuronen der dritten Schicht entsprechen den vorgeschlagenen Bedingungen der Regeln
- Die Neuronen der vierten Schicht entsprechen den Fuzzy-Regeln
- Die Neuronen der fünften Schicht entsprechen den Bedingungen der Konsequenzteile der Regeln

Die Konvertierung der Parameter des aufgestellten Fuzzy-Modells in Parameter des Neuronalen Netzes geschieht wie folgt (mit Bezug auf das Beispiel aus [KWOA92] und damit auf Abbildung 27) :

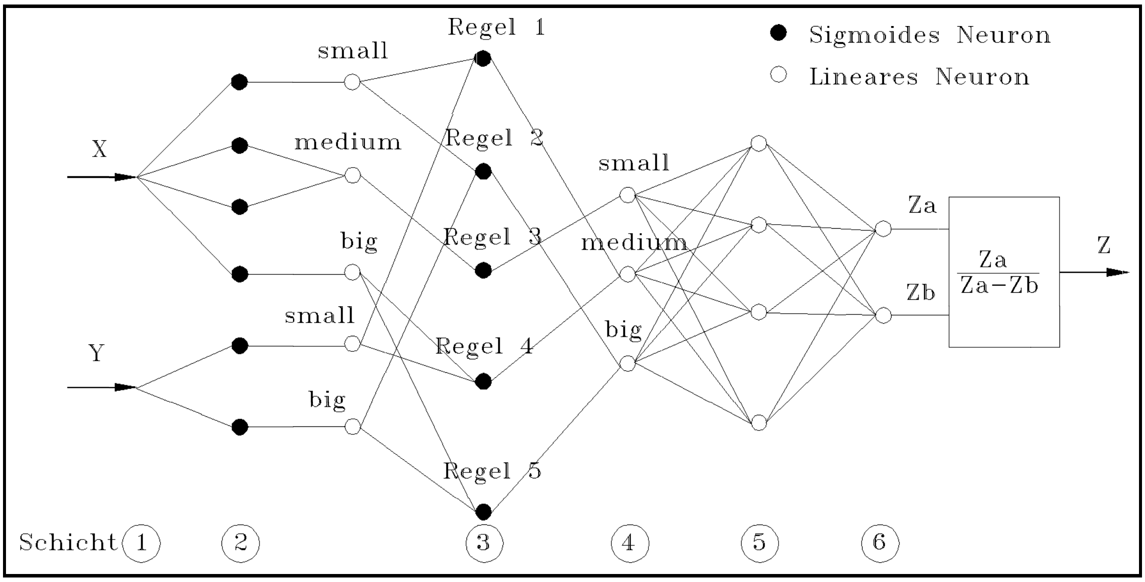

Abbildung 27: Modellierung von Fuzzy-Regeln durch Neuronales Netz

1. Fuzzy-Regeln
   Die Regel 1 „*if (X is small) and (Y is small) then (Z is medium)*“ wird durch Verbindungen des Neurons der Regel 1 in Schicht 4 mit den Neuronen „*X is small*“ und „*Y is small*“ der dritten Schicht realisiert. Außerdem wird das Neuron der Regel 1 mit dem „*Z is medium*“-Neuron auf Schicht 5 mit einem Gewicht von 1.0 verbunden, die Schwellwerte der Neuronen in der fünften Schicht sollten auf 0.0 gesetzt werden.

2. Zugehörigkeitsfunktionen des Bedingungsteils
   In Abbildung 28 sind die Realisierungen der linguistischen Werte „*small*“, „*medium*“ und „*big*“ mit Zugehörigkeitsfunktionen zu sehen. Die leeren Kreise stellen dabei lineare Neuronen und die dunklen sigmoide Neuronen dar. Als Zahlenwerte sind die Gewichte und Schwellwerte angegeben. Monoton steigende bzw. fallende Zugehörigkeitsfunktionen werden durch Verändern des Gewichtes und des Schwellwertes mit einem Neuron modelliert. Zugehörigkeitsfunktionen in Form einer Glockenkurve werden als Differenz von zwei sigmoiden Neuronen dargestellt.

3. Fuzzy-Operationen
   Die Neuronen der dritten Schicht werden mit dem „Operations-Neuron“ der vierten Schicht verbunden. Die Gewichte der Verbindungen und die Schwellwerte der Neuronen werden von Kawamura offensichtlich heuristisch gewählt.

4. Zugehörigkeitsfunktionen des Konsequenzteils
   Die Neuronen der fünften Schicht entsprechen den Zugehörigkeitsfunktionen des Konsequenzteils und die Neuronen der sechsten Schicht repräsentieren Punkte auf der Z-Achse, also der Ausgabevariable (s. Abbildung 29). Die Gewichte zwischen

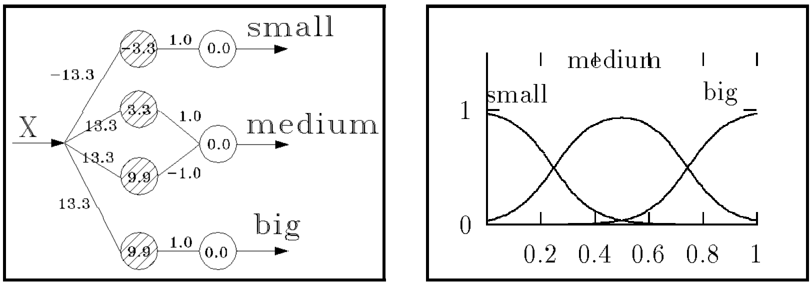

Abbildung 28: Zugehörigkeitsfunktionen des Bedingungsteils

zwei Neuronen der Schichten werden mit den Werten der entsprechenden Zugehörigkeitsfunktion an der Stelle der Z-Achse, die das Neuron der sechsten Schicht darstellt, initialisiert. Es liegt also eine Diskretisierung der Ausgabevariable vor.
Beispielsweise hat die Funktion „*small*" die Werte 1.0 bei 0.0, 0.0 bei 0.33, 0.0 bei 0.67 und 0.0 bei 1.0. Die Gewichte vom Neuron „*Z is small*" der fünften Schicht mit denen der sechsten Schicht sind somit 1.0, 0.0, 0.0, 0.0. Die Schwellwerte sollten auf 0.0 gesetzt werden.

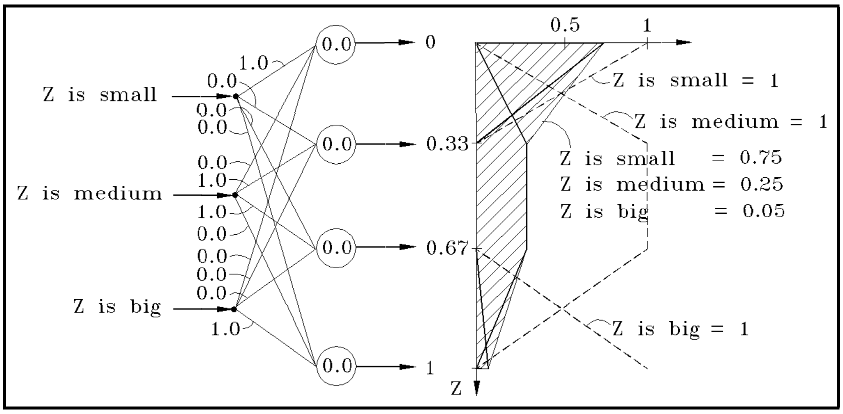

Abbildung 29: Zugehörigkeitsfunktionen des Konsequenzteils

Intern läuft die Berechnung nun folgendermaßen ab :
In Abbildung 29 bedeutet „*Z is small*"=1.0, daß die Ausgabe des Neurons 1.0 ist (gestrichelte Linien). Trifft diese Bedingung jedoch nur teilweise zu, also z.B. mit dem Grad 0.75, so wird die ursprüngliche Zugehörigkeitsfunktion mit diesem Grad multipliziert, also hier um 25% vermindert (durchgezogene Linien). Da alle Neuronen der sechsten Schicht eine lineare Ausgabefunktion haben werden alle Ausgaben der Neuronen bzw. alle modifizierten Funktionen addiert. Der schraffierte Bereich in Abbildung 29 ist also das Ergebnis „*Z is small*"=0.75, „*Z is medium*"=0.25 und „*Z is big*"=0.05.

5. Berechnung des COA
   In Abbildung 30 kann man die Vernetzung zur Berechnung des COA sehen. Die

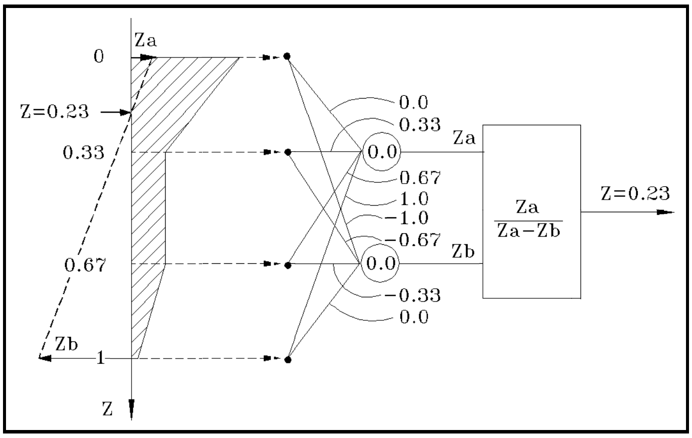

Abbildung 30: Berechnung des COA

Gewichte zwischen den Neuronen der sechsten Schicht mit Neuron *Za* entsprechen den jeweiligen Z-Werten der entsprechenden Neuronen. Die Gewichte zu Neuron *Zb* ergeben sich genauso, es muß nur 1.0 subtrahiert werden.
Die Ausgabe von *Za* ist gleich der Summe der Produkte der Ausgaben der sechsten Schicht und den zugehörigen Gewichten, entspricht also dem Rotationsmoment um den Punkt „Z=0.0“. Der Wert von *Zb* ist demnach äquivalent zum Rotationsmoment um den Punkt „Z=1.0“. Der COA ergibt sich somit zu $\frac{Za}{Za-Zb}$.

Das so entstandene Neuronale Netz kann nun zur Regelung eingesetzt werden und lernt mittels Backpropagation (s. Seite 15). Eine Veränderung der Gewichte zwischen Neuronen der vierten und fünften Schicht kann als Modifikation der Gewichtung der entsprechenden Regel interpretiert werden. Alle anderen Veränderungen entsprechen Optimierungen der Zugehörigkeitsfunktionen. Eine Interpretation des Neuronalen Netzes nach der Lernphase läuft entsprechend dem oben angegebenen Verfahren umgekehrt ab.

6.1.1 Zusammenfassung

Der vorgestellte Algorithmus zur Umsetzung eines Fuzzy-Modells in ein Neuronales Netz, respektive Neuro-Fuzzy-Controller, gibt gute Einblicke in die internen Verarbeitungsweisen solcher Modelle. Es wird ein sechs-schichtiges Neuronales Netz erzeugt. Die verwendete Neuronen haben je nach Aufgabe lineare bzw. sigmoide Ausgabefunktionen (s. auch Kapitel 3.2).

Im anzugebenden Fuzzy-Modell sind die Regeln, Zugehörigkeitsfunktionen und verwendeten Operationen zu spezifizieren. Das Modell dient also lediglich dazu, einen vorhandenen FLC zu verbessern und spätere Interpretationsmöglichkeiten des gelernten, modifizierten Neuro-Fuzzy-Controllers zu ermöglichen.

6.2 Hayashi-Regler

6.2.1 Neuro-Fuzzy-Regler

Der in [HNW89] eingeführte Controller wurde in [HNW90] und [HNYW92] erweitert und verbessert. An Hand des invertierten Pendels versuchen die Autoren die Leistungsfähigkeit ihres Systems zu erläutern. In Abbildung 31 ist ein Blockdiagramm des *neural network driven fuzzy reasoning* (NDF) zu sehen.

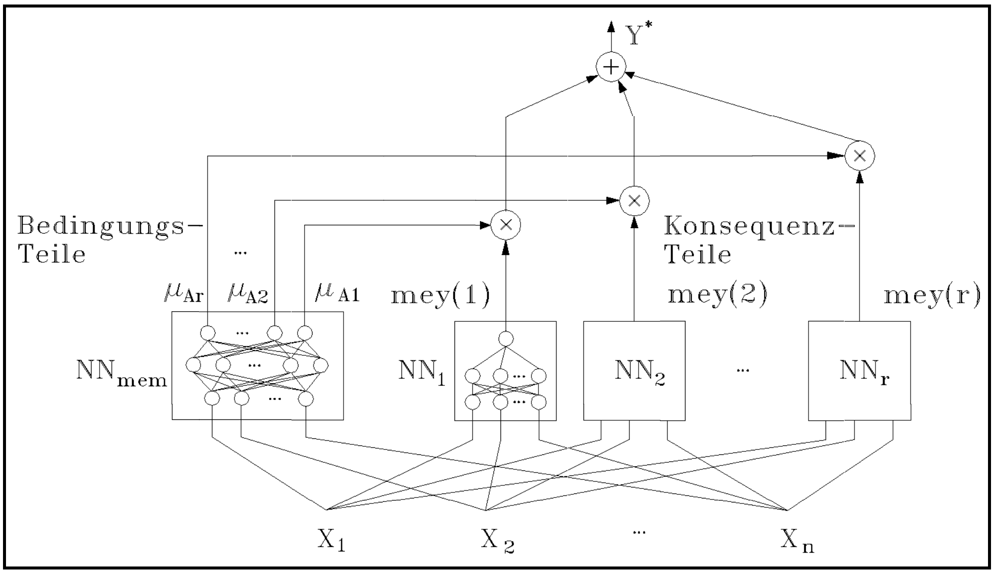

Abbildung 31: Hayashis NDF-System

Die verwendeten Regeln haben die Form

$$R_s \quad : \quad \texttt{if } \mathbf{x} = (x_1, x_2, ..., x_n) \texttt{ is } A_s, \texttt{ then } y_s = NN_s(x_1, x_2, ..., x_m),$$
$$s \quad = \quad 1, 2, ..., r\,;\;\; m \leq n$$

Dabei ist r die Anzahl der Regeln, A_s eine Fuzzy-Menge des Bedingungsteils einer Regel und $NN_s(x_1, x_2, ..., x_m)$ beschreibt eine Funktion, die durch ein M-schichtiges Neuronales Netz mit der Eingabe $\mathbf{x} = (x_1, x_2, ..., x_n)$ und der Ausgabe y modelliert wird. Den Zugehörigkeitsgrad einer Eingabe $\mathbf{x} = (x_1, x_2, ..., x_n)$ zu dem Bedingungsteil der s-ten Regel erhält man aus dem Zugehörigkeitswert der Eingabe $\mathbf{x}$ zur Fuzzy-Menge A_s.

Die Zugehörigkeitsfunktionen des Bedingungsteil werden mit dem Neuronalen Netz NN_{mem} festgelegt. Mit diesem Backpropagation-Netz (s. auch Seite 15) werden die Zugehörigkeitsgrade von Eingaben zu Regeln gelernt. Die Ausgabe des Netzes NN_{mem} ist somit ein Vektor $\mu_{\mathbf{A}} = (\mu_{A1}(\mathbf{x}), \mu_{A2}(\mathbf{x}), ..., \mu_{Ar}(\mathbf{x}))$ dessen Komponenten $\mu_{As}(\mathbf{x})$ den Zugehörigkeitsgrad der Eingabe $\mathbf{x}$ zur s-ten Regel angeben. Es folgt der Algorithmus zur Berechnung der Ausgabe y_i^* der i-ten Eingabe $\mathbf{x}_i$, $i = 1, 2, ..., N$, (s. dazu Abbildung 31).

1. Festlegen der Ein-Ausgabepaare $(\mathbf{x}_i, y_i)$ mit $\mathbf{x}_i = (x_{i1}, x_{i2}, ..., x_{in})$, $i = 1, 2, ..., N$.

2. Trenne die Ein-Ausgabedaten in r Klassen R_s, $s = 1, 2, ..., r$. Jede Partition wird dabei als Regel R_s aufgefaßt, die Ein-Ausgabedaten jeder Regel R_s werden als

$(\mathbf{x}_i^s, y_i^s)$, $i = 1, 2, ..., N_s$, ausgedrückt, vorausgesetzt N_s ist die Anzahl der Ein-Ausgabedaten für R_s.

3. Bestimme die Zugehörigkeitsfunktionen des Bedingungsteils mit Hilfe des M-schichtigen Backpropagation-Netzes NN_{mem} (s. Abbildung 31).

4. Bestimme die Modelle des Konsequenzteils mit Hilfe der M-schichtigen Backpropagation-Netze $NN_1, NN_2, ..., NN_r$ und wähle die beste Modellierung.
Dazu wird folgendes iterative Verfahren verwendet :

   (a) Für $k = n$ wird die Eingabe $\mathbf{x}_i = (x_{i1}, x_{i2}, ..., x_{ik})$, $i = 1, 2, ..., N$, an die Eingabeschichten jedes NN_s und die zugehörige Ausgabe y_i an die Ausgabeschichten der NN_s angelegt. Die Menge der Eingabevariablen wird mit

$$Q_s = \{x_1, x_2, ..., x_k\} \tag{6.2}$$

   bezeichnet, und die Menge der Eingabevariablen die den Ausgabeschichten jedes NN_s zugewiesen wird, ist mit

$$T_s = \{y\} \tag{6.3}$$

   beschrieben.

   (b) Für die Eingabedaten $x_{i1}, x_{i2}, ..., x_{ik}$ wird nach wiederholtem Lernen die Ausgabe ey_i berechnet und der mittlere quadratische Fehler J_k^s bestimmt :

$$J_k^s = \left[\sum_{i=1}^{N} (y_i - ey_i)^2 \right] / N, \qquad s = 1, 2, ..., r\,. \tag{6.4}$$

   (c) Um die Korrelation der Eingabe x_j mit der Ausgabe y zu bestimmen, wird x_j aus der Eingabemenge entfernt. Die neue Eingabe $x_{i1}, ..., x_{ij-1}, x_{ij+1}, ..., x_{ik}$, $i = 1, 2, ..., N$ wird an die Eingabeschicht des M-schichtigen Backpropagation-Netzes und die Ausgabe y_i an die Ausgabeschicht angelegt. Nach Lernen der Eingabe $x_{i1}, ..., x_{ij-1}, x_{ij+1}, ..., x_{ik}$ kann die Ausgabe ey_i' bestimmt werden. Mit diesen Werten wird erneut der mittlere quadratische Fehler berechnet

$$J_{k-1}^{sj} = \left[\sum_{i=1}^{N} (y_i - ey_i')^2 \right] / N, \qquad s = 1, 2, ..., r\,. \tag{6.5}$$

   Diese Berechnung wird für alle x_s, $s = 1, 2, ..., k$ durchgeführt und das Minimum

$$J_{k-1}^{sc} = \min_j J_{k-1}^{sj}, \qquad j = 1, 2, ..., k \tag{6.6}$$

   bestimmt.

   (d) Nach Vergleich von J_{k-1}^{sc} mit J_k^s aus Gleichung (6.4) wird die Menge der Variablen Q_s verändert :

$$Q_s = \{x_1, x_2, ..., x_{c-1}, x_{c+1}, ..., x_k\}, \qquad \text{wenn } J_{k-1}^{sc} < J_k^s \tag{6.7}$$

$$Q_s = \{x_1, x_2, ..., x_k\}, \qquad \text{wenn } J_{k-1}^{sc} \geq J_k^s \tag{6.8}$$

Bei Anwendung von Gleichung (6.7) kann der mittlere quadratische Fehler durch Weglassen von x_c verringert werden, d.h. ey_i' ist eine bessere Darstellung von y_i als ey_i. Weil die Korrelation zwischen x_c und den Ausgaben y klein ist, kann die Eingabe aus der Eingabemenge Q_s entfernt werden, diese Menge hat somit nur noch $k-1$ Variablen.
Bei Anwendung von Gleichung (6.8) sind die Eingabevariablen x_c stark mit y korreliert, die Anzahl der Eingabevariablen in Q_s bleibt unverändert bei k.
Bei einer Verringerung der Eingabevariablen wird k auf $n-1, n-2, ..., 1$ gesetzt und Punkt 4. wird solange wiederhholt bis Gleichung (6.8) angewendet werden kann. Die Verringerung der Eingabevariablen für das Backpropagation-Netz NN_s ist bei Verwendung der Gleichung (6.8) abgeschlossen.
Mit der endgültigen Menge von Eingabevariablen $Q_s = \{x_1, x_2, ..., x_m\}$ ist das Neuronale Netz NN_s optimal zur Repräsentation des Konsequenzteils der Regel R_s. Punkt 4. wird für alle NN_s durchgeführt, um alle Konsequenzteile der Regeln zu bestimmen.

5. Die Ausgabe y^* wird aus der Gleichung

$$y_i^* = \frac{\sum_{s=1}^{r} \mu_{As}(x_{i1}, x_{i2}, ..., x_{in}) \times mey_i(s)}{\sum_{s=1}^{r} \mu_{As}(x_{i1}, x_{i2}, ..., x_{in})}, \quad i = 1, 2, ..., N \tag{6.9}$$

errechnet, wobei $mey_i(s)$ die Ausgabe des optimalen Backpropagation-Netzes NN_s aus Punkt 4. ist.
In Abbildung 31 ist zu sehen, daß die Ausgabe y_i^* durch Summation der Produkte der Zugehörigkeitswerte der Bedingungsteile, z.B. $\mu_{As}(x_{i1}, x_{i2}, ..., x_{in})$, und der Ausgaben der Konsequenzteile, z.B. $mey_i(s)$, zu berechnen ist.

Obwohl es auch möglich ist einen nichtlinearen Zusammenhang mit nur einem Backpropagation-Netz herzustellen, ist die Verwendung von jeweils einem Backpropagation-Netz für jedes Teilgebiet (bzw. jeden Teilraum) als vorteilhafter anzusehen.

6.2.2 Neuro-Fuzzy-Regler mit veränderbarer Regelbasis

Das NDF Verfahren ist nicht in der Lage seine Regelbasis anzupassen, wenn sich die Umgebung, speziell der zu kontrollierende Prozeß, dynamisch verändert. Das jetzt vorgestellte System *neural network driven fuzzy reasoning with learning function* (NDFL) soll dem Problem Abhilfe verschaffen. Mit der Muster-Suchmethode werden die optimalen Koeffizienten der Konsequenzteile der Regeln bestimmt. Die Muster-Suchmethode wird in zwei Teile getrennt : die Forschungs-Bewegung und die Muster-Bewegung. Die Such-Bewegung legt die Richtung der optimalen Lösung durch Berechnung von Werten der Entwicklungsfunktion f fest. Die Muster-Bewegung wird zur Verschiebung des Suchpunktes in die für die optimale Lösung benötigte Richtung verwendet.
Die Anzahl der Suchparameter ist m, die Koordinaten am Ende der $p-1$-ten Such-Bewegung des k-ten Musters ist mit $W_p^k = (w_0^k, w_1^k, ..., w_p^k, ..., w_m^k)$ bezeichnet. Ein Einheitsvektor mit einer 1 an der p-ten Stelle sei durch d_p dargestellt.

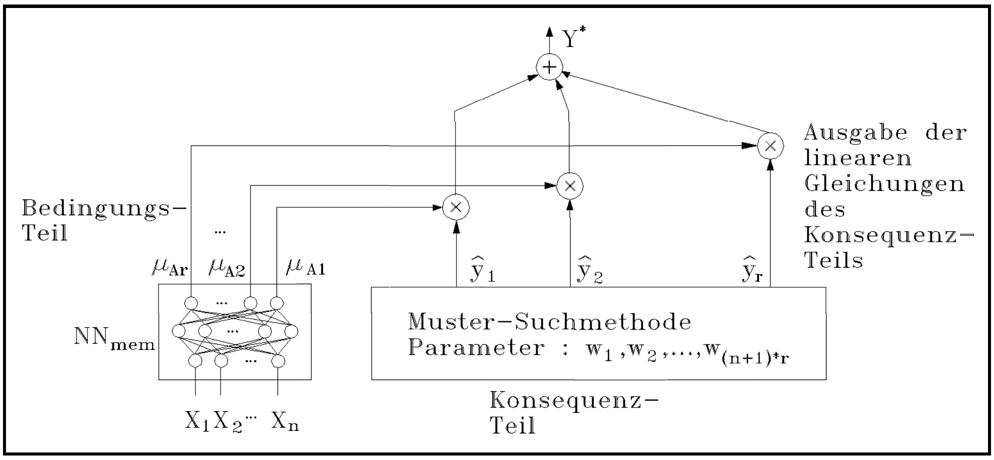

Abbildung 32: Hayashis NDFL-System

1. Such-Bewegung
   Iterationsbeginn sei $p = 1$. Wenn
   $$\int (W_p^k + \alpha_p^k d_p) \ < \ f(W_p^k), \quad \alpha_p^k > 0 \tag{6.10}$$
   erfüllt ist, ist die p-te Such-Bewegung erfolgreich gewesen und für die $p+1$-te Such-Bewegung wird $W_{p+1}^k = W_p^k + \alpha_p^k d_p$ gesetzt.
   Ist die Bedingung jedoch nicht erfüllt, wird $\alpha_p^k = -\alpha_p^k$ gesetzt und dieselbe Operation wiederholt. Klappt beides nicht, so wird $W_{p+1}^k = W_p^k$ gesetzt. α_p^k zeigt den Bereich der Suche an.

2. Muster-Bewegung
   Die Koordinaten die durch die m-te Such-Bewegung der k-ten Mustersuche werden als $W_B^k = (w_0^k, w_1^k, ..., w_p^k, ..., w_m^k)$ geschrieben. W_B^k wird als Basispunkt der k-ten Mustersuche bezeichnet. Mustersuche bedeutet die folgende Bewegung vom Basispunkt W_B^k nach W^k
   $$W^k = W_B^k + (W_B^k - W_B^{k-1}). \tag{6.11}$$

Eine Veränderung des Konsequenzteils einer Regel zur Minimierung der Entwicklungsfunktion f durch Anwendung der Muster-Suchmethode ist für Regel R_s dargestellt

$$\begin{aligned} R_s \quad &: \quad \texttt{if } \mathbf{x} = (x_1, x_2, ..., x_n) \texttt{ is } A_s \texttt{ then } y_s^k = w_{s0}^k + w_{s1}^k x_1 + ... + w_{sn}^k x_n \\ s \quad &= \quad 1, 2, ..., r. \end{aligned}$$

A_s ist dabei eine Fuzzy-Menge des Bedingungsteils, und $w_{s0}^k, ..., w_{sn}^k$ sind Koeffizienten die durch die Muster-Suchmethode angepaßt werden.
Mit der i-ten Eingabe $X_i = (x_{i1}, ..., x_{in})$ wird die Ausgabe y_i^* aus der Gleichung

$$y_i^* = \frac{\sum_{s=1}^r \mu_{As}(x_{i1}, ..., x_{in}) \ \times \ (w_{s0}^k + w_{s1}^k x_{i1} + ... + w_{sn}^k x_{in})}{\sum_{s=1}^r \mu_{As}(x_{i1}, ..., x_{in})}, \quad i = 1, 2, ..., N \tag{6.12}$$

bestimmt.
Ein Blockdiagramm des Ablaufs ist in Abbildung 32 zu sehen. Es ist zu erkennen, daß die Form der Fuzzy-Menge A_s des Bedingungsteils durch ein Backpropagation-Netz NN_{mem} bestimmt wird. Die Mustersuche wird dazu verwendet, für die Entwicklungsfunktion f einen optimalen Wert zu erreichen.

Der Algorithmus zur Berechnung der Bewegungen ist bei Bedarf in [HNYW92] zu finden.

6.2.3 Zusammenfassung

Der Neuro-Fuzzy-Regler
Der Aufbau dieses Controllers hebt sich deutlich von den bisher vorgestellten Modellen ab. Jeder Konsequenzteil einer Regel wird mit einem Neuronalen Netz berechnet. Parallel zur Berechnung des Konsequenzteils werden in einem anderen Neuronalen Netz die Ergebnisse der Bedingungsteile verarbeitet.

Das Training erfolgt mit Ein-Ausgabe-Tupeln für die Klassen angegeben werden müssen. Jede Klasse wird von einer Regel entschieden. Die Zugehörigkeitsfunktionen der Bedingungsteile werden vom Controller automatisch erfasst. Für die Konsequenzteile werden die Funktionen mit einem iterativen Suchverfahren bestimmt.

Das vorgestellte System scheint sehr aufwendig, gerade die Lernphase der verschiedenen Netze wird viel Zeit in Anspruch nehmen. Die Wahl der Ein-Ausgabe-Tupel ist nicht ohne Vorwissen über das zu kontrollierende System zu machen, dieses Vorwissen ist in vielen Fällen jedoch nicht verfügbar. Sind die Beispieldaten ausgewählt, steht nicht fest ob mit ihnen eine „ausreichend gute“ Repräsentation des Problems gefunden ist.

Der Neuro-Fuzzy-Regler mit veränderbarer Regelbasis
Dieses modifiziertes Verfahren des „normalen“ Neuro-Fuzzy-Reglers soll eine Anpassung der Regelbasis an sich dynamisch verändernde Kontrollprobleme ermöglichen.

Für die Eingabedaten werden die jeweils optimalen Konsequenzteile der Regeln mit einem Muster-Suchverfahren bestimmt.

6.3 Berenji-Regler

Das in [Ber92] vorgestellte Modell (weiteres auch in [LB89]) ist in Abbildung 33 dargestellt. Die *approximate reasoning based intelligent control*-Architektur (ARIC) besteht aus zwei Hauptkomponenten.

Das *action-state evaluation network* (AEN) ist dabei zur Bewertung der Kontrollaktionen vorgesehen, es gibt dem Hauptcontroller ein internes Reinforcement zur Verbesserung der Gewichte. Das *action selection network* (ASN) modelliert einen Fuzzy-Regler (FLC). Es handelt sich bei beiden Modulen um mehrschichtige Neuronale Netze.

6.3.1 Das Action-State-Evaluation-Netzwerk

Das AEN empfängt als Eingabe den Status des Systems und ein Fehlersignal. Die Struktur des Netzwerkes ist in Abbildung 34 zu sehen.

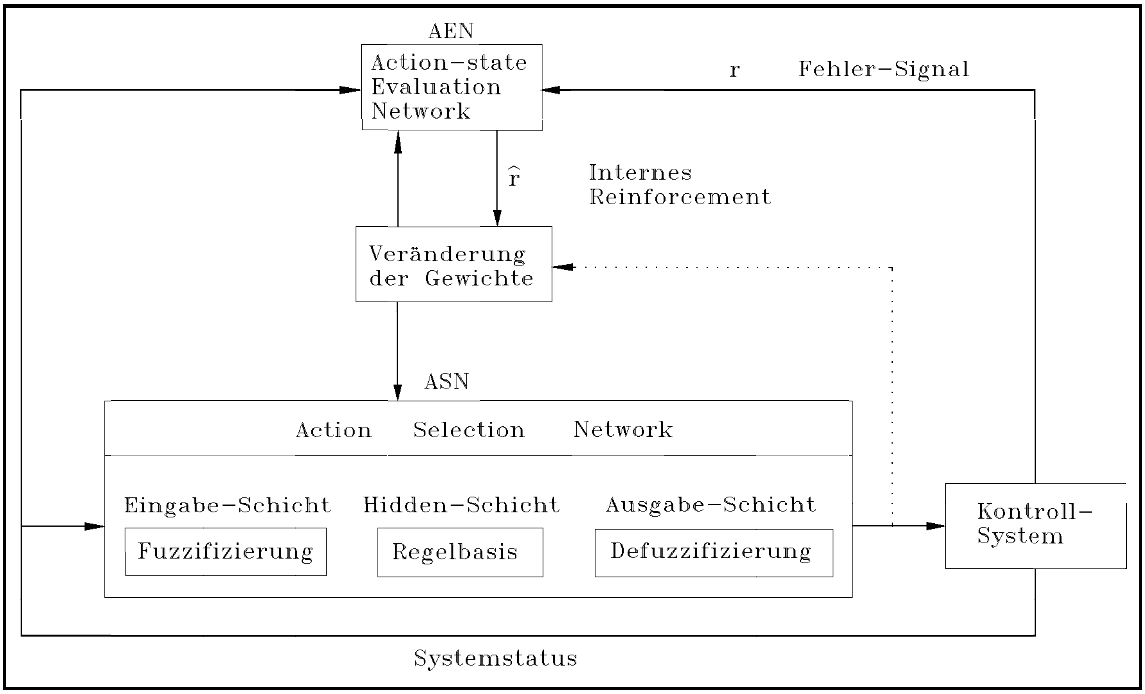

Abbildung 33: Berenjis ARIC-Architektur

Es besteht aus n Eingabeeinheiten, h versteckten Einheiten und einer Ausgabeeinheit. Das AEN erzeugt eine Vorhersage des zukünftigen Reinforcements, auch heuristisches oder inneres Reinforcement genannt, für einen bestimmten Systemstatus, die Veränderungen dieser Vorhersage werden zur Steuerung des ASN bei der Aktionswahl verwendet. Wechselt das System aus einem Status mit geringem Reinforcement zu einem mit größerem Reinforcement, so wird auf Grund dieser positiven Änderung die Auswahl der verantwortlichen Aktion verstärkt.

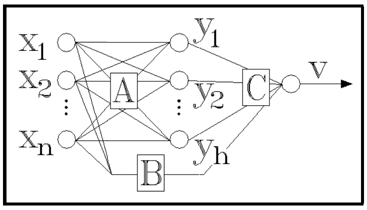

Abbildung 34: AEN Aufbau

Die Ausgabe der Einheiten der Hidden-Schicht berechnet sich zu

$$y_i[t, t+1] = g\left(\sum_{j=1}^{n} a_{ij}[t] x_j[t+1]\right) \tag{6.13}$$

wobei (vgl. Gleichung (3.15))

$$g(s) = \frac{1}{1 + \exp(-s)} \tag{6.14}$$

gilt und t bzw. $t+1$ Zeitschritte sind.
Die Ausgabeeinheit erhält Eingaben sowohl von den Einheiten der Hidden-Schicht (z.B. y_i) als auch direkt von den Eingabeeinheiten (z.B. x_i) :

$$v[t, t+1] = \sum_{i=1}^{n} b_i[t] x_i[t+1] + \sum_{i=1}^{h} c_i[t] y_i[t, t+1] \tag{6.15}$$

dabei ist $v[t, t+1]$ die Reinforcement-Vorhersage. Die Abhängigkeit von zwei Zeitschritten wird in den obigen und den folgenden Gleichungen zur Verbesserung der Stabilität bei der Änderung der Gewichte verwendet.
Das interne Reinforcement berechnet sich nun zu

$$\hat{r}[t+1] = \begin{cases} 0 & \text{, bei Start} \\ r[t+1] - v[t, t] & \text{, bei Fehler in der Steuerung} \\ r[t+1] + \gamma v[t, t+1] - v[t, t] & \text{, sonst} \end{cases} \tag{6.16}$$

$0 \leq \gamma \leq 1$ ist ein Koeffizient zur Verminderung des Einflusses von Werten von v die für nachfolgende Zeitschritte berechnet sind. Das heuristische oder interne Reinforcement ergibt sich also aus der Summe von externem Reinforcement und der Änderung der Werte von v.
Die Gewichte des Neuronalen Netzes werden nach einem „Belohnungs-/Bestrafungsprinzip" verändert. Bei positivem (negativem) internen Reinforcement werden die Gewichte „belohnt" („bestraft") und in die Richtung geändert, die ihren Anteil an der Gesamtsumme wachsen (sinken) läßt. Die Gewichte b_i, in Abbildung 34 mit $\boxed{\text{B}}$ gekennzeichnet, werden folgendermaßen verändert

$$b_i[t+1] = b_i[t] + \beta \hat{r}[t+1] x_i[t], \tag{6.17}$$

dabei ist $\beta > 0$ eine Konstante und $\hat{r}[t+1]$ das innere Reinforcement $t+1$. Ein positives $\hat{r}$ hat also ein Wachsen der Gewichte zu Folge, ein negatives $\hat{r}$ somit ein Sinken. Die Gewichte zwischen der Hidden-Schicht und der Ausgabe-Schicht, in Abbildung 34 mit $\boxed{\text{C}}$ gekennzeichnet, werden mit

$$c_i[t+1] = c_i[t] + \beta \hat{r}[t+1] y_i[t, t] \tag{6.18}$$

verändert. Die Anpassung der Gewichte der Hidden-Schicht, in Abbildung 34 mit $\boxed{\text{A}}$ gekennzeichnet, geschieht mit einem modifizierten Error-Backpropagation Algorithmus (s. Seite 15). $\hat{r}$ spielt dabei die Rolle des Fehlers. Die Veränderung läuft somit folgendermaßen ab

$$a_{ij}[t+1] = a_{ij}[t] + \beta_h \hat{r}[t+1] y_i[t, t](1 - y_i[t, t]) \mathrm{sgn}(c_i[t]) x_j[t], \tag{6.19}$$

mit $\beta, \beta_h > 0$.

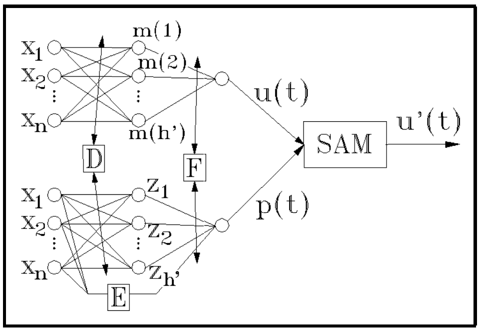

Abbildung 35: ASN Aufbau

6.3.2 Das Action-Selection-Netzwerk

Das ASN enthält zwei zweischichtige Neuronale Netze, wobei eines einen Fuzzy-Regler (FLC) modelliert. In Abbildung 35 ist das Schema des ASN zu sehen.

Das ASN verändert die vom FLC gewählten Aktionen zufällig. Diese Änderung ist bedeutsamer für Aktionen in denen kein großes internes Reinforcement empfangen wird (z.B. großes Fehlersignal). Ist das interne Reinforcement groß, wird das ASN die gewählte Aktion nur sehr wenig ändern. Kann also der FLC das System gut steuern (nach einer gewissen Lernphase), so werden keine oder nur sehr kleine Änderungen zu machen sein.

Der FLC des ASN
Der FLC ist im oberen Teil der Abbildung 35 zu sehen. Die Eingabe-Schicht spielt die Rolle des Fuzzifizierers, die Elemente der Hidden-Schicht stellen die Regeln dar. Das ASN hat dabei genau soviele Element in der versteckten Schicht, wie es Regeln in der Regelbasis gibt. Die Eingaben der versteckten Schicht sind dabei die „Wahrheitswerte" der Vorbedingungen der Regeln und die Ausgaben sind die Fuzzy-Ergebnisse der Konsequenzteile. In der Ausgabe-Schicht wird aus allen Ausgaben der Regeln ein scharfer Wert mit Tsukamoto's-Defuzzifiziermethode berechnet (s. Kapitel 4.3.4).

Die Regel haben mehrere Eingaben und eine Ausgabe (MISO), $w(i)$ ist der Wert mit dem Regel i auf die Eingabe $\mathbf{x}$ zutrifft

$$w(i) = \min\{d_{i1}\, \mu_{i1}(x_1), ..., d_{ij}\, \mu_{ij}(x_j), ..., d_{in}\, \mu_{in}(x_n)\}, \tag{6.20}$$

$\mu_{ij}(x_j)$ ist dabei der Zugehörigkeitsgrad der Eingabe x_j zu einer Fuzzy-Menge in der j-ten Bedingung von Regel i, d_{ij} ist das Gewicht der Verbindung von Einheit j zu Einheit i und n ist die Anzahl der Eingaben.
$m(i)$ ist das Ergebnis nach Anwendung von $w(i)$ auf den Konsequenzteil von Regel i

$$m(i) = \mu_{C_i}^{-1}(w(i)), \tag{6.21}$$

μ_{C_i} ist eine monotone Zugehörigkeitsfunktion des linguistischen Wertes aus dem Konsequenzteil von Regel i. Die Kontrollaktion $u(t)$ ergibt sich aus allen Regeln zu

$$u(t) = \frac{\sum_{i=1}^{h} f_i \times m(i) \times w(i)}{\sum_{i=1}^{h} w(i) \times f_i}, \tag{6.22}$$

mit h als Zahl der Einheiten der Hidden-Schicht bzw. der Regeln des Modells und f_i als Gewicht von Einheit i der versteckten Schicht zur Ausgabeschicht.

Das Neuronale Netz des ASN

Das Neuronale Netz ist im unteren Teil der Abbildung 35 zu sehen. Zur gleichen Zeit, in der die Ausgabe $u(t)$ des FLC berechnet wird, wird dazu eine Art Vertrauenskoeffizient p erzeugt. Dieser Wert wird dann zur Veränderung von $u(t)$ verwendet.

Die Ausgabe der versteckten Schicht ist

$$z_i[t, t+1] = g\left(\sum_{j=1}^{n} d_{ij}[t]x_j[t+1]\right) \tag{6.23}$$

wobei $g(s)$ eine sigmoide Ausgabefunktion ist (vgl. Gleichung (3.15) und (6.14)). Die Ausgabeeinheit erhält Eingaben von sowohl den Elementen der Hidden-Schicht, als auch denen der Eingabeschicht. Die zur Aktion $u(t)$ des FLC gehörende Ausgabe $p[t, t+1]$ berechnet sich zu

$$p[t, t+1] = \sum_{i=1}^{n} e_i[t]x_i[t+1] + \sum_{i=1}^{h'} f_i[t]z_i[t, t+1]. \tag{6.24}$$

Als Gesamtausgabe wird

$$u'[t] = o(u[t], p[t, t+1]) \tag{6.25}$$

mit Hilfe einer stochastischen Funktion o, die $u[t]$ auf Grund von p verändert, erzeugt. Als Meßwert dieser stochastischen Aktionsänderung (*stochastic action modification* - SAM, s. Abbildung 35) wird

$$s[t] = k(u[t], u'[t]) \tag{6.26}$$

verwendet. Die Auswahl der passenden Funktionen o und k ist abhängig von der Anzahl der möglichen Kontrollaktionen und von der Art der benötigten Entscheidung (diskreter oder kontinuierlicher Wert).

Die Änderung der Gewichte der Verbindungen zur Ausgabeschicht, in Abbildung 35 mit $\boxed{\text{E}}$ gekennzeichnet, ist proportional zu $\hat{r}, s$ und der zugehörigen Eingabe

$$e_i[t+1] = e_i[t] + \rho\hat{r}[t+1]s[t]x_i[t], \tag{6.27}$$

$\rho > 0$ ist die Lernrate. Die Gewichte der Verbindungen von der versteckten Schicht zur Ausgabeschicht, in Abbildung 35 mit $\boxed{\text{F}}$ gekennzeichnet, werden entsprechend angepasst

$$f_i[t+1] = f_i[t] + \rho\hat{r}[t+1]s[t]z_i[t, t]. \tag{6.28}$$

Ist $\hat{r}$ einmal eine gute Vorhersage der folgenden Aktion, kann sein Produkt mit dem Wert $s[t]$ einer gewählten Aktion als gemessener Fehler verwendet werden. Durch Error-Backpropagation können die Gewichte des Netzes, in Abbildung 35 mit $\boxed{\text{D}}$ gekennzeichnet, neu berechnet werden

$$d_{ij}[t+1] = d_{ij}[t] + \rho_h\hat{r}[t+1]z_i[t, t](1 - z_i[t, t])\text{sgn}(f_i[t])s[t]x_j[t], \tag{6.29}$$