h_da

HOCHSCHULE DARMSTADT
UNIVERSITY OF APPLIED SCIENCES

Computational Intelligence

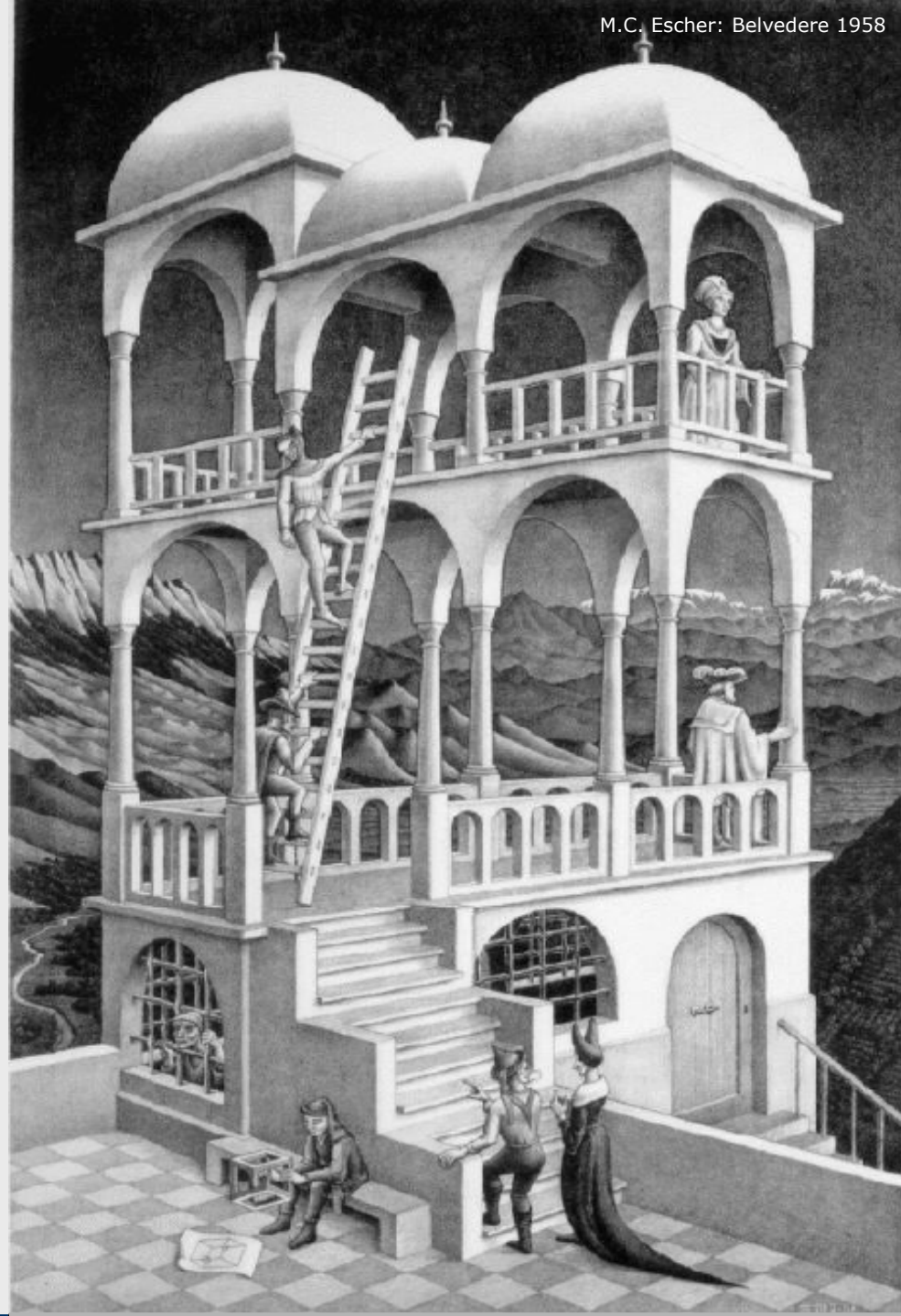
Kapitel 2: Fuzzy Logik

Teil D: Fuzzy Control

Dr. Norbert Waleschkowski

h_da Fachbereich Informatik

Sommersemester 2011
Master-Studiengang


M.C. Escher: Belvedere 1958

Kap. 2: Fuzzy Logik

Grundlagen der Fuzzy Control

- Nun die Theorie …
- Die größte Beachtung findet die Fuzzy Logik heute in der Steuer- und Regeltechnik, daher auch die Bezeichnung Fuzzy Control.
- Fuzzy Regler finden sich bereits in vielen Geräten des täglichen Gebrauchs wie etwa Wasch- und Spülmaschinen, Digitalcameras, etc.
- Der wesentliche Unterschied gegenüber klassischen Reglern liegt darin, dass es keine explizite mathematische Beschreibung des funktionalen Zusammenhangs von Ein- und Ausgabedaten gibt.
- Die Fuzzy Control wurde in ihren Grundzügen bereits 1972 von L. Zadeh umrissen.
- Mamdani und Assilian haben 1975 die Fuzzy Control erstmals zur Steuerung von Dampfmaschinen eingesetzt und die Architektur solcher Systeme detaillierter ausgearbeitet. Daher werden solche Systeme heute auch Mamdani-Systeme genannt.
- Wir werden uns im folgenden mit solchen Systemen beschäftigen.

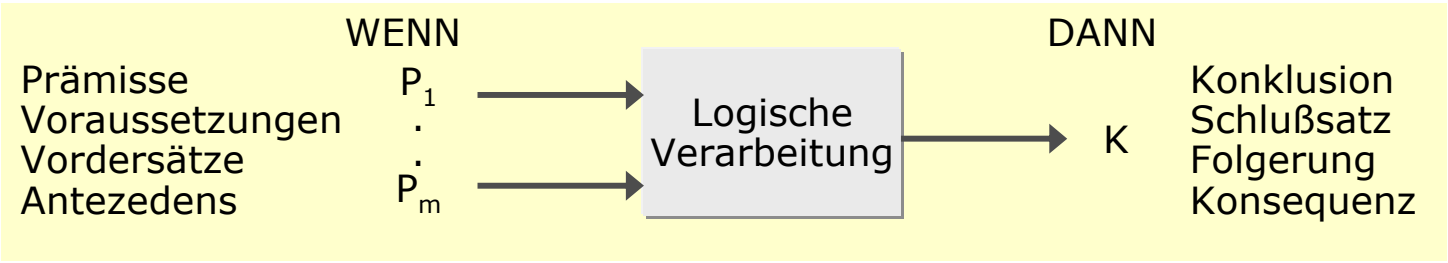
- Bevor wir dies tun, erfolgt aber zunächst ein kleiner Exkurs in die Arbeitsweise regelbasierter Systeme.

Exkurs: Arbeitsweise regelbasierter Systeme

Exkurs

Logisches Schließen (1)

- $P_1(x_1, \ldots, x_n)$, $P_2(x_1, \ldots, x_n)$, …, $P_m(x_1, \ldots, x_n)$ seien Aussageformen. Wenn für alle Belegungen der x_i mit einem Wert aus {w, f} (w= wahr, f= falsch), für welche die Aussageformen P_1, …,P_m =w (wahr) sind, auch die Aussageform $K(x_1,\ldots, x_n)$ wahr ist, so heißt K eine Konklusion aus den Aussageformen P_1, …, P_m.
- Wir schreiben: $P_1 \wedge P_2 \wedge \ldots \wedge P_m \Rightarrow K$
  und nennen diesen Ausdruck Implikation.



Exkurs

Logisches Schließen (2)

- **Wichtige Schlußregeln**:

- Modus ponens (Abtrennungsregel)

P	P ist wahr.
P → K	Wenn P wahr ist, dann ist K wahr.
K	

- Modus tollens (Widerlegungsregel)

P → K	Wenn P wahr ist, dann ist K wahr.
¬K	K ist nicht wahr.
¬P	Also ist P nicht wahr.

- Kettenschluß

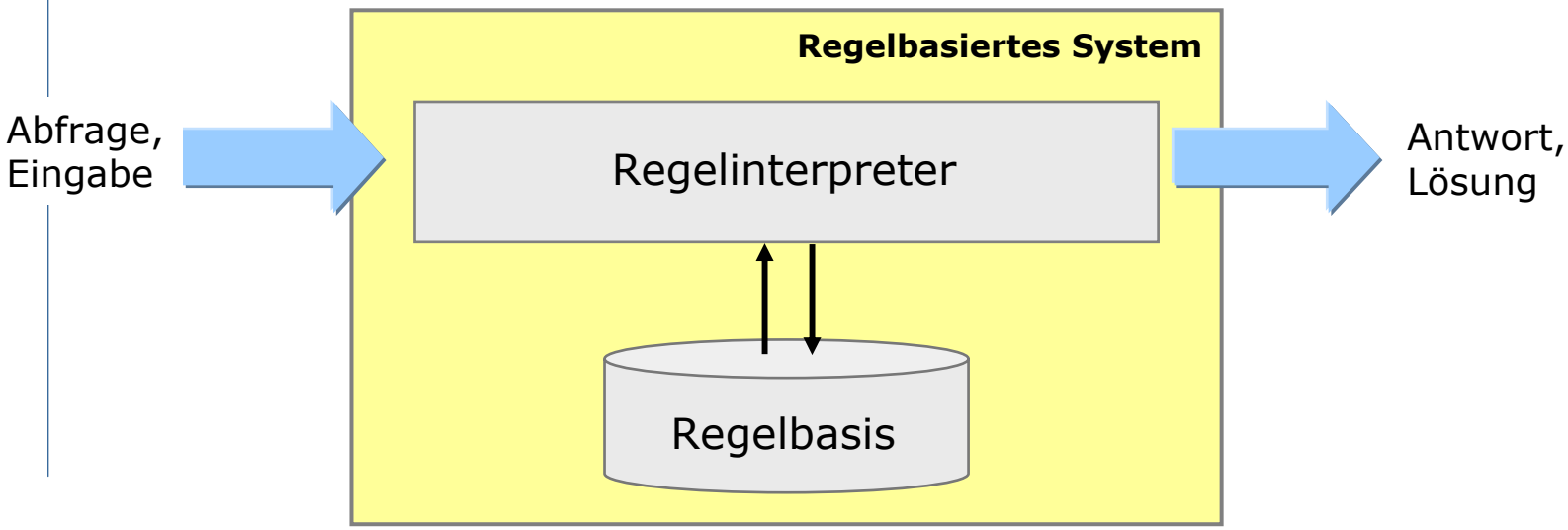
$$\frac{(P \rightarrow K_1) \wedge (K_1 \rightarrow K_2) \wedge \ldots \wedge (K_n \rightarrow K)}{P \rightarrow K}$$

Exkurs

Regelbasierte Systeme – Das Prinzip

- In einem klassischen Programm, also einer algorithmischen Problemlösung, ist das Wissen direkt im Programmcode hinterlegt.

- In einem regelbasierten System wird das Wissen in Form von Regeln repräsentiert und von der Verarbeitungslogik isoliert.



Exkurs

Arbeitsweise regelbasierter Systeme

- Grundstruktur von Regeln: Eine einfache WENN-DANN-Regel hat den folgenden Aufbau:
  - WENN alle Voraussetzungen erfüllt sind, DANN ziehe Schlussfolgerungen (WENN Prämisse DANN Konklusion)

    bzw.
  - **IF** P_1, P_2, ..., P_n **THEN** K_1, K_2, ..., K_m
- Wenn alle Bedingungen P_i erfüllt sind, dann können alle Schluß-folgerungen K_j gezogen werden. Die P_i und K_j sind Wahrheitswerte, die nur WAHR (TRUE) oder FALSCH (FALSE) sein können.
- Neben Regeln gibt es auch noch Fakten. Das sind Aussagen, die immer erfüllt sind. Fakten können als Regeln ohne Prämisse betrachtet werden.
- Die Konklusion, also der THEN-Teil, kann auch Aktionen A enthal-ten, die bei Erfüllung aller Prämissen ebenfalls auszuführen sind. Es kann sogar sein, dass nur Aktionen auszuführen sind, also
  - **IF** P_1, P_2, ..., P_n **THEN** K_1, K_2, ..., K_m **DO** A_1, A_2, ..., A_l

    bzw.
  - **IF** P_1, P_2, ..., P_n **THEN DO** A_1, A_2, ..., A_l

Aktionen sind eine Art Nebeneffekt. Wir nehmen im folgenden an, dass in den Aktionen, also im **DO**-Teil, keine Prämissen gesetzt werden können, damit das Schlussfolgerungsschema explizit sichtbar bleibt.

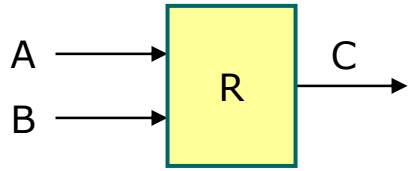
Exkurs

Wir beschränken uns hier auf Regeln mit nur einer Konsequenz. Dabei handelt es sich nicht um eine wirkliche Einschränkung, da sich jede Regel mit n Konsequenzen in Form von n Regeln mit denselben Bedingungen darstellen lässt.

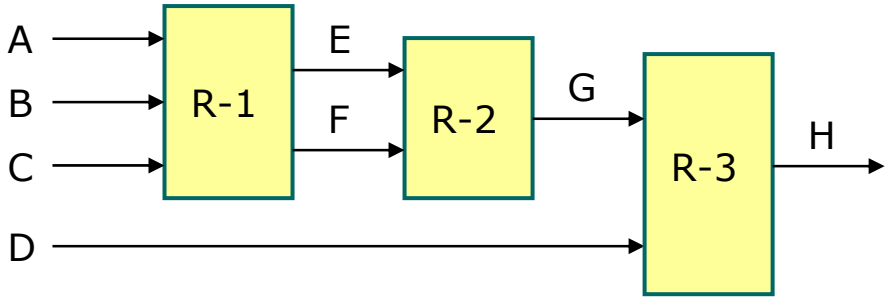
Vorwärtsverkettung von Regeln

- Die Konklusionen einer Regel können Prämissen einer anderen Regel sein. Dadurch ergeben sich viele potentielle Verkettungsmöglichkeiten von Regeln.
- Zur Verdeutlichung verwenden wir die folgende Notation:

  Regel R: **IF** A, B **THEN** C



- Wir betrachten nun das folgende Regelsystem, bestehend aus den Regeln R-1, R-2 und R-3: Wir nehmen an, es gelten die Fakten A, B, C und D. Da A, B und C gelten, kann mittels R-1 auf E und F geschlossen werden. Mittels R-2 kann dann direkt auf G weitergeschlossen werden. Wegen D und R-3 gilt dann auch H.

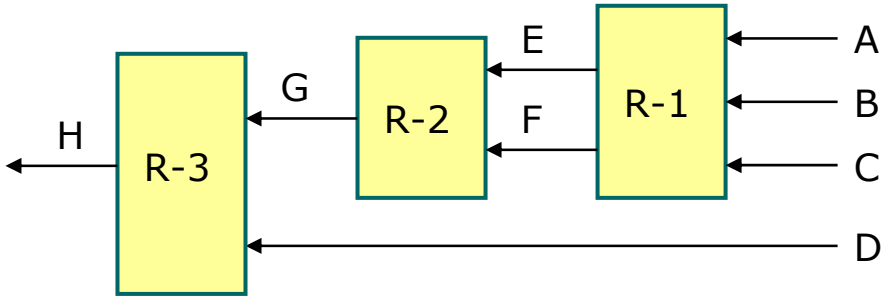


- Diese Art der Regelverkettung heißt Vorwärtsverkettung (Forward Chaining).

Exkurs

Rückwärtsverkettung von Regeln

- Wir betrachten dasselbe Regelsystem, um nun umgekehrt – rückwärts – zu schließen.



- Dabei besteht das Ziel darin, die Aussage H zu beweisen. H gilt gemäß Regel R-3, wenn G und D gelten. D gilt gemäß Voraussetzung. G lässt sich aus Regel R-2 ableiten, wenn E und F gelten. E und F lassen sich aus R-1 ableiten, da A, B und C gelten. H kann also bewiesen werden, weil A, B, C und D gelten.
- Diese Form der Regelverkettung heißt Rückwärtsverkettung (Backward Chaining).

Exkurs

Vergleich Vorwärts-/Rückwärtsverkettung

- Regelsysteme können sehr groß werden. Z.B. hatte das System R1/XCON zur Konfiguration von DEC-Mainframe-Computern, das erste kommerziell erfolgreiche wissensbasierte System, eine Regelbasis aus ca. 14.000 Regeln.
- Bei der Vorwärtsverkettung kann es sehr leicht passieren, dass Unmengen von Schlussfolgerungen gezogen werden, die gar nicht benötigt werden. Es kommt zur kombinatorischen Explosion von Regelausführungen. Es werden also Strategien benötigt, um das unkontrollierte Feuern der Regeln zu steuern.
- Bei der Rückwärtsverkettung geht man von dem zu beweisenden Satz aus und arbeitet sich schrittweise bis zu den gegebenen Voraussetzungen durch. Die Programmiersprache PROLOG basiert z.B. auf dem Prinzip der Rückwärtsverkettung. Die Rückwärts-verkettung arbeitet wesentlich zielgerichteter.

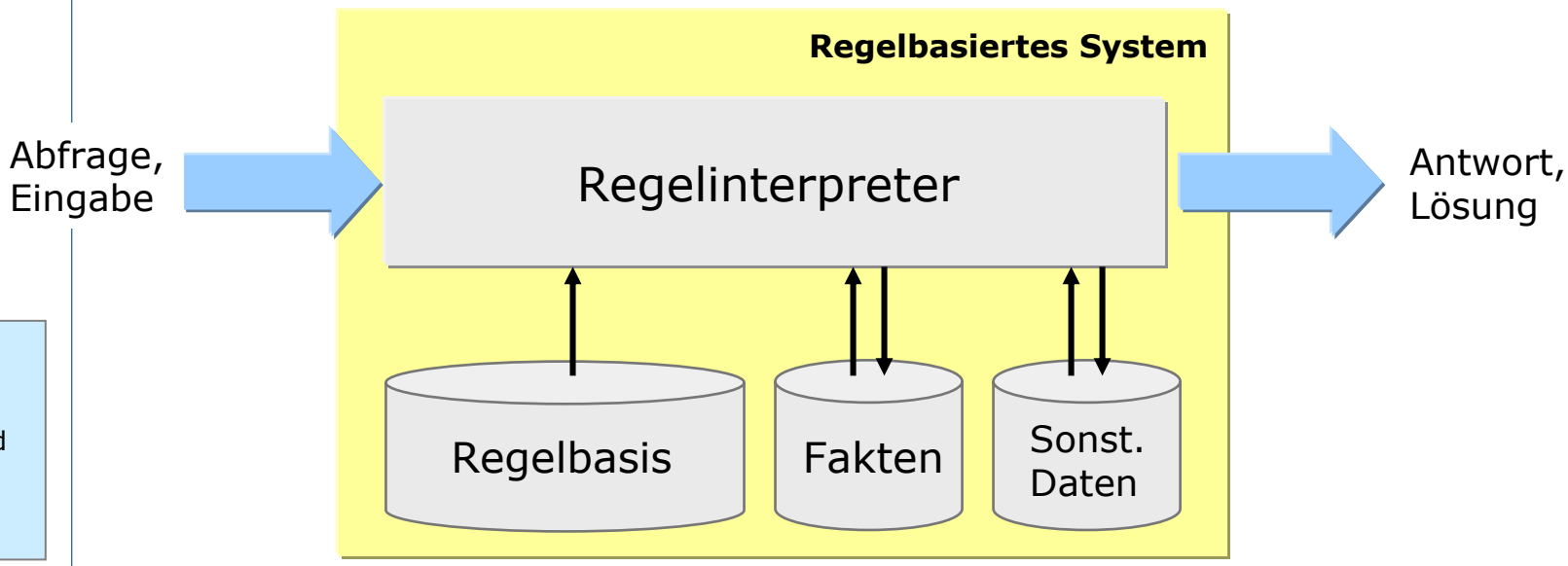
- Vorwärtsverkettung:
  - datengetrieben (data driven)
  - synthetisch, Ableiten aller Folgerungen
  - geeignet bei kleinen Regelmengen
  - Regeln sind anwendbar, wenn Prämissen erfüllt sind

Rückwärtsverkettung:
- zielgetrieben (goal driven)
- analytisch, Beweis der Hypothese
- nötig bei größeren Regelmengen
- Regeln sind anwendbar, wenn Folgerung zu beweisen ist.

Exkurs

Regelinterpreter (1)

- Zur Verarbeitung der Regeln wird noch eine Problemlösungs-komponente benötigt, der Regelinterpreter.



Aktionsroutinen manipulieren die sonstigen Daten des Systems und sorgen für die Interaktion mit dem Benutzer.

- Nach dem Start wählt der Regelinterpreter die anzuwendenden Regeln aus und führt einen ersten Schlussfolgerungszyklus durch.
- Dabei ausgeführte Aktionsroutinen manipulieren sonstige Daten und erfragen ggf. neue Fakten. Außerdem können Regelausführungen zu einer neuen Faktenlage führen. Ein neuer Zyklus beginnt.

Exkurs

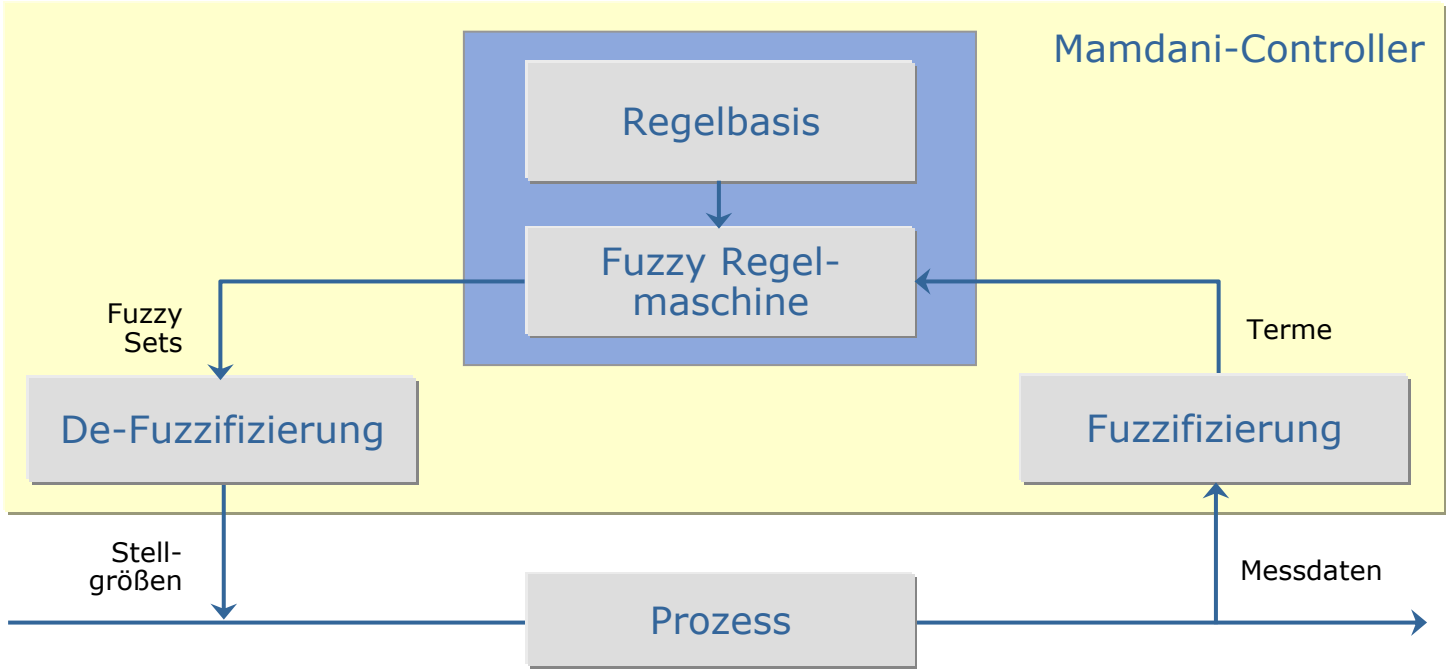
Regelinterpreter (2)

- Die „sonstigen Daten" stehen für alle anderen Daten, die mit dem eigentlichen Schlussfolgerungsmechanismus des Regelinterpreters nichts zu tun haben. Sie sind Nebeneffekt der Aktionsroutinen.
- Es gibt verschiedene Strategien, um die Auswahl der anzuwendenden Regeln zu steuern:
  - Einfachste Form:
    - Suche passende anwendbare Regel.
    - Führe Regel aus.
  - Alternative 1:
    - Suche alle anwendbaren Regeln.
    - Feuere alle Regeln nacheinander.
  - Alternative 2:
    - Suche alle anwendbaren Regeln.
    - Wähle bestimmte Regeln nach einer gewissen Heuristik aus.
    - Feuere die Regeln in der heuristisch bestimmten Reihenfolge.
  - Sonstige: ...
- Intelligentere Strategien verwenden Heuristiken, die durch Meta-Regeln gesteuert werden können. Meta-Regeln sind Regeln, die die Auswahl und Reihenfolge der anzuwendenden Regeln steuern.
  - Prioritäten von Regeln
  - Prioritäten von Fakten
  - Häufigkeit der früheren Verwendung
  - Spezialisierungsgrad der Regeln
  - Gruppenbildungen von Regeln etc.

Zurück zur Fuzzy Control

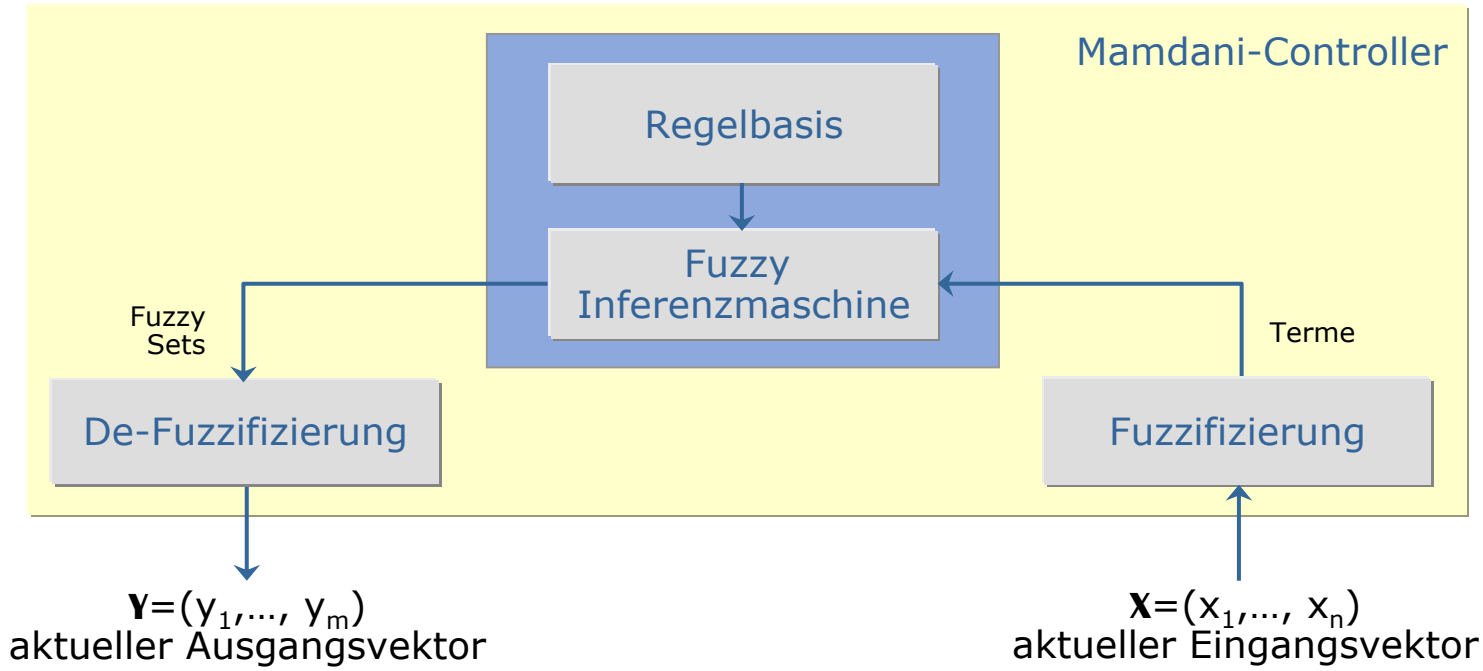
Grundstruktur eines Fuzzy Control-Systems

- In der letzten LE wurde bereits die Grundstruktur eines Mamdani-Controllers vorgestellt. Er besteht aus
  - einem Fuzzifizierer, der die scharfen Daten in unscharfe Terme überführt
  - einer Regelbasis, bestehend aus unscharfen Regeln,
  - einem Regelinterpreter zur Verarbeitung der unscharfen Regeln sowie
  - einem Defuzzifizierer, der aus einer unscharfen Menge eine scharfe Aussage erzeugt.



Die Verarbeitungsschritte

- Die Verarbeitung vollzieht sich in 3 Schritten:
  - A: Fuzzifizierung
  - B: Inferenz
  - C: Defuzzifizierung…



Schritt A: Die Fuzzifizierung

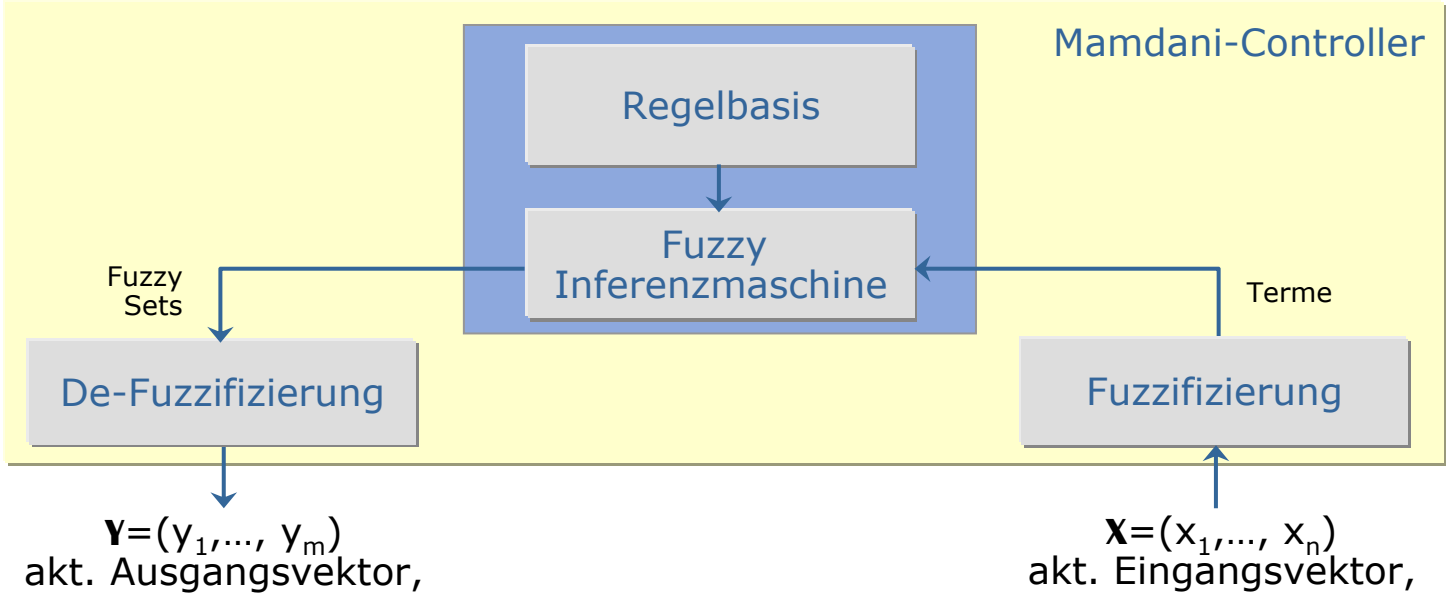
- Unter der Fuzzifizierung der Eingangsgrößen versteht man die Ermittlung des Erfüllungsgrades des Momentanwertes der Eingangsgröße x_i der Größe X_i mit den entsprechenden unscharfen Zuständen.
- Eine Größe X_i wird dabei als linguistische Variable angesehen. Das Skalenniveau von X_i wird dadurch auf m Ausprägungsbereiche (wie „hoch", „mittel", „niedrig", …) verteilt.
- D.h.: $X_i = \{A_{i1}, A_{i2}, \ldots, A_{im}\}$, wobei die unscharfen Mengen A_{ik} die Zugehörigkeitsfunktionen $\mu_{A_{ik}}$ besitzen. Daraus ergibt sich dann der jeweilige Zugehörigkeitswert:

$$a_{ik} = FUZZ(x_i) = \mu_{A_{ik}}(x_i),\ 1 \leq k \leq m$$

- Bsp.: Eine Brennkammertemperatur von 910°C zu einem Grade von 0,8 „sehr hoch" und zu einem Grade von 0,3 „hoch" (s. Bsp. aus „*VCI_04_FuzzyControl_Bsp.pdf*").

Schritt B: Die unscharfe Inferenz (1)

- Wir betrachten zunächst die Arbeitsweise regelbasierter Fuzzy-Systeme.
- Die Regelbasis enthält alle WENN-DANN-Regeln R_j, wobei $1 \leq j \leq k$.
- Die Prämisse, also der WENN-Teil der Regel R_j, besteht aus einem Ausdruck der Form $X_i = P_{ji}$ oder aus Verknüpfungen solcher Ausdrücke wie $(X_1 = P_{j1})$ *o* $(X_2 = P_{j2})$ *o* … $(X_n = P_{jn})$, wobei *o* die UND- bzw. ODER-Konjunktion sein kann.
- Die Regel R_j lautet dann also: WENN P_j DANN K bzw. $P_j \Rightarrow K$.
- Jeder Regel R_j läßt sich noch ein Vertrauensmaß cer_j (certainty) zwischen 0 (kein Vertrauen) und 1 (volles Vertrauen) zuordnen.



Schritt B: Die unscharfe Inferenz (2)

- Der Controller habe nun neue und bereits fuzzifizierte Messwerte erhalten. Nun muss untersucht werden, ob diese Werte für die Regelprämissen relevant sind. Dies muss für jede Regel geschehen.
- Für jedes Element der Prämisse wurde der Erfüllungsgrad bei der Fuzzifizierung festgelegt. Wie wird aber der Erfüllungsgrad der gesamten Prämisse festgelegt? Und welche Schlussfolgerung wird gezogen?
- Dies geschieht im Inferenzschritt, der in 3 Teilschritten abläuft:
  - Aggregation (AGG)
    Die Aggregation der Elemente der Prämisse ermittelt den Erfüllungsgrad der gesamten Prämisse.
  - Implikation (IMP)
    Die Implikation ermittelt den Erfüllungsgrad der zu ziehenden Schlussfolgerung K (wenn P → K). Die Wirkung der Regel hängt also vom Erfüllungsgrad der Prämisse ab.
  - Akkumulation (AKK)
    Die Akkumulation führt die Ergebnisse der Regelauswertung zusammen. Es ergibt sich eine unscharfe Ergebnismenge.
- Im Verarbeitungsschritt C, der Defuzzifizierung, muss diese unscharfe Ergebnismenge in einen konkreten, scharfen Wert überführt werden, um die darin enthaltene Information möglichst gut abzubilden.

Schritt B: Die unscharfe Inferenz (3)

Der Inferenzschritt: (1) Aggregation

- Die einzelnen Zugehörigkeitswerte a_{ij} werden zu einem Gesamtwert a_j für die Prämisse der Regel R_j mittels einer geeigneten Aggegationsfunktion AGG aggregiert:
$$a_j = AGG\ (a_{j1}, a_{j2}, \ldots, a_{jn})$$
- Dieser Aggregationsoperator AGG wird aufgabenspezifisch gewählt und muss die Verknüpfungen der Regelbedingungen geeignet abbilden können.
- Üblicherweise werden hier i.a. t-Normen, t-Conormen sowie auch kompensatorische Operatoren verwendet.
- Sofern Sicherheitsfaktoren $0 \leq cer \leq 1$ verwendet werden, ist anschließend noch ein modifizierter Erfüllungsgrad a_j^* zu ermitteln. Hierzu ist die Größe a_j noch mit dem Sicherheitsmaß cer_j zu verknüpfen:
$$a_j^* = CER(a_j, cer_j)$$
- Ein Sicherheitsfaktor kann auch als Zugehörigkeitsgrad einer unsicheren Regel zur Menge der sicheren Regeln aufgefasst werden.
- Die CER-Operation muss eine t-Norm sein, da die Größe a_j mit dem Sicherheitsmaß cer_j via UND zu verknüpfen ist. Oft verwendet man den Minimum- oder den PROD-Operator. Bei absolut sicheren Regeln erfolgt damit keine Veränderung der Größe a_j.

Schritt B: Die unscharfe Inferenz (4)

Der Inferenzschritt: (2) Implikation

- Die Schlussfolgerung einer Regel R_j sollte höchstens den Erfüllungsgrad a_j^* wie ihr WENN-Teil besitzen:

  $$K_j^* = IMP\ (a_j^*, K_j)$$

  Für den Operator IMP lassen sich alle nicht-kompensatorischen t-Normen verwenden (, denn der Glaube an die Gültigkeit einer Regel und der Erfülltheitsgrad ihrer Prämisse gleichen sich nicht aus).

Der Inferenzschritt: (3) Akkumulation

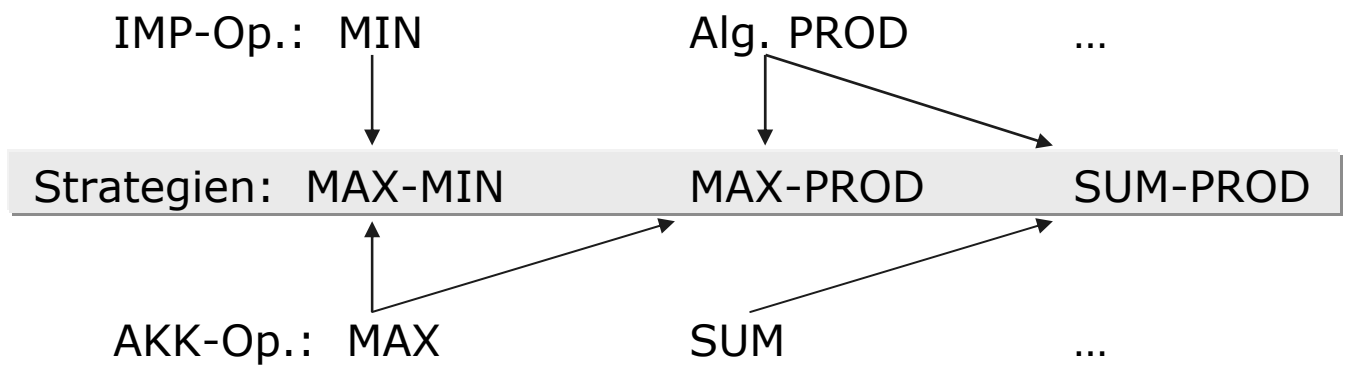
- Wenn die Implikationen aller Regeln vorliegen, führt die Akkumulation AKK die Einzelresultate zusammen:

  $$K^* = AKK(K_1^*, K_2^*, \ldots, K_k^*)$$

- Die AKK-Operation entspricht einer ODER-Verknüpfung, da die Regeln einen alternativen Charakter besitzen; das Gesamtergebnis entsteht durch Vereinigung der Einzelergebnisse. Daher werden für den AKK-Operator üblicherweise t-Conormen verwendet. Häufig verwendet werden z.B. der MAX- und der Additionsoperator.

Schritt B: Die unscharfe Inferenz (5)

- Häufig benutzte Inferenzstrategien sind z.B. die MaxMin- und die MaxProd-Strategie. Dabei repräsentiert der 1. Teil des Namens den verwendeten AKK-Operator und der 2. Teil den verwendeten IMP-Operator.
- Beispiele für Inferenzstrategien:



- Beispiel: IMP=PROD=μ_{Prod}: $\mu_{K_j*}(x) = a_j^* \cdot \mu_{K_j*}(x)$

  AKK=Summe: $\mu_{K*}(x) = \sum_j \mu_{K_j*}(x)$

  $\Rightarrow$ SUM-PROD -Strategie

Schritt C: Die Defuzzifizierung (1)

- Die unscharfe Ergebnismenge muss nun in einen konkreten, scharfen Wert überführt werden:

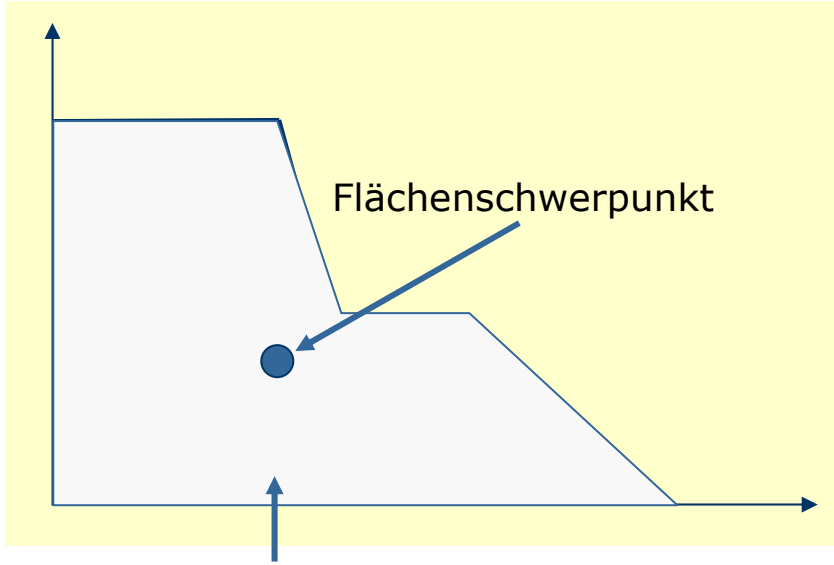
  $$k^* = \text{DEFUZZ}(K^*) = D(x_1, x_2, \ldots, x_n)$$

  Dabei muss die in K^* enthaltene Information möglichst gut durch einen Wert repräsentiert werden. Dies ist zwangsläufig mit einem Informationsverlust verbunden.

- Wir diskutieren im folgenden alternative Verfahren der Defuzzifizierung.

- I. a. erhält man als Ergebnismenge eine Vereinigung von mehreren unscharfen, nicht notwendig normalisierten Mengen.

- Der Max-Operator führt zur Zugehörigkeitsfunktion in Form der äußeren Umhüllung der beteiligten Mengen.



Die Defuzzifizierung (2)

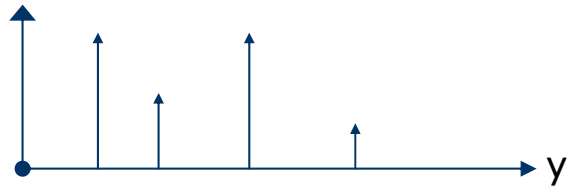
- Flächenschwerpunkt COA (center of area):

$$\bar{y} = COA(K^*) = \frac{1.\text{ Moment von } \mu_K}{0.\text{ Moment von } \mu_K} = \frac{m_1(K^*)}{m_0(K^*)} = \frac{\sum_i x_i \cdot \mu_K(x_i)}{\sum_i \mu_K(x_i)} \; bzw. \; \frac{\int y \cdot \mu_K(y)\,dy}{\int \mu_K(y)\,dy}$$

Wenn man sich die Ergebnismenge mit einer dünnen Masseschicht belegt denkt, ergibt sich so der Flächenschwerpunkt.

- Mittelwert der Maximalwerte (MOM bzw. mean of maximum)

$$\bar{y} = \frac{1}{n}\sum_i h(K_i), h(K_i) = \mu_{K_i}(y_{K_i})$$



- Median:

y teilt die Ergebnismenge in zwei Hälften gleicher Fläche.

- Parametrisierte Familie von Defuzzifizierungsmethoden:

$$\bar{y} = \frac{\sum_i \left(\int ..\right)\mu^{\gamma}(x_i) \cdot x_i}{\sum_i \left(\int ..\right)\mu^{\gamma}(x_i)}$$

$\gamma = 1$: COA
$\gamma \to 0$: $y=(x_1+....x_n)/n$
$\gamma \to \infty$: Mean-Max-Methode
$\gamma < 1$: Werte von μ(x) werden verstärkt.
$\gamma > 1$: Werte von μ(x) werden abgeschwächt.