Die approbierte Originalversion dieser Diplom-/Masterarbeit ist an der Hauptbibliothek der Technischen Universität Wien aufgestellt (http://www.ub.tuwien.ac.at).

The approved original version of this diploma or master thesis is available at the main library of the Vienna University of Technology (http://www.ub.tuwien.ac.at/englweb/).

FAKULTÄT FÜR !NFORMATIK

Ausgewählte formale und programmtechnische Aspekte des medizinischen Expertensystems FuzzyKBWean

DIPLOMARBEIT

zur Erlangung des akademischen Grades

Diplom-Ingenieur

im Rahmen des Studiums

Informatik (E881)

eingereicht von

Wolfgang Koller
Matrikelnummer 8926813

an der
Fakultät für Informatik der Technischen Universität Wien

Betreuung:
Betreuer: Univ.-Prof. DI Dr. Klaus-Peter Adlassnig
Mitwirkung: DI Dr. Christian Schuh

Wien, 27.10.2008

(Unterschrift Verfasser) (Unterschrift Betreuer)

Kurzfassung

Das Ziel des Projektes FuzzyKBWean war es, auf Intensivstationen die Entwöhnung, das so genannte „Weaning“, beatmeter Patienten vom Beatmungsgerät (Respirator) durch ein medizinisches Expertensystem zu steuern. In der ersten Implementierungsstufe soll es als „open-loop“-System den behandelnden Ärzten eine möglichst effiziente, das heißt rasche und schonende, Entwöhnungsstrategie vorschlagen, während die Handlungsentscheidung dem Arzt überlassen bleibt. Aufbauend auf den in dieser Phase gewonnenen Erfahrungen, nach ausgedehnten Tests und nach Optimierung der Wissensbasis, ist als endgültiges Ziel ein „closed-loop“-System vorgesehen. Bei diesem wird das Expertensystem die Entwöhnung autonom steuern und der Arzt muss nur in Ausnahmefällen, wie zum Beispiel in Notsituationen, eingreifen.

In dieser Arbeit werden drei Komponenten beschrieben, die aufbauend auf einer bereits existierenden Implementierung von FuzzyKBWean realisiert wurden. Die *Wissenserwerbskomponente* ermöglicht eine interaktive Modellierung und Anpassung von Wissensbasen. Im *Datenbankkonzept* werden all jene Mechanismen beschrieben, die für die Speicherung der relevanten Daten sorgen. Die *Inferenzkomponente* schließlich interpretiert die vom Patientendatenmanagementsystem gelieferten physiologischen Daten und berechnet auf der Grundlage der in der Wissensbasis enthaltenen Regeln neue Stellwerte für das Beatmungsgerät.

Die aktuelle Version von FuzzyKBWean ist ein regelbasiertes Expertensystem mit Echtzeitverhalten. Seine hervorstechendste Eigenschaft ist, dass es zur Steuerung der Entwöhnung zwei Konzepte, nämlich Fuzzy- und Nicht-Fuzzy-Regelungsmodelle, einsetzt und dadurch einen direkten Vergleich der beiden Ansätze ermöglicht. Man geht in der Arbeitsgruppe FuzzyKBWean davon aus, dass ein unscharfes („fuzzy“) Regelungskonzept der Denk- und Ausdrucksweise von medizinischen Experten näher kommt als ein Modell auf der Basis von scharfer („crisp“) Logik. Dadurch erhofft man sich ein Regelungsverhalten des Systems, das einen kontinuierlicheren, für den Patienten schonenderen und insgesamt effizienteren Entwöhnungsvorgang zur Folge hat.

Abstract

The goal of the FuzzyKBWean project was to control the process of weaning ICU patients from mechanical ventilation by a medical expert system. In the first stage of development, its task as an open-loop system is to compute proposals for an efficient—i. e., quick and gentle—weaning strategy, whereas the final decisions are left to the physician. Based on the experience gained from this phase, after extensive testing and optimization of the knowledge base, the ultimate goal is a closed-loop system. This system will be able to wean patients autonomously, and the physician only has to intervene in cases of emergency.

This work describes three key components, that were implemented on the basis of an already existing version of FuzzyKBWean: The *knowledge acquisition component* allows an interactive modelling and adaption of knowledge bases. The *database concept* describes the techniques that are used for handling relevant data. The *inference engine* interprets the physiological data provided by the patient data management system and calculates new settings for the ventilator based on the rule set.

The current version of FuzzyKBWean is a rule-based expert system with real-time behaviour. Its key characteristic is the application of two different concepts, namely fuzzy and non-fuzzy models, in order to control the weaning process, which allows a thorough comparison of the two concepts. The FuzzyKBWean workgroup assumes that a fuzzy approach is more closely related to medical experts' way of thinking than a model based on crisp logic. We thus expect a more smooth control behaviour of the system that may result in a more efficient weaning process and prove to be less stressful for the patient.

Danksagung

Mein besonderer Dank gilt der Arbeitsgruppe FuzzyKBWean, DI Dr. techn. Christian Schuh und Dr. med. Christian Zelenka, Univ.-Prof. DI Dr. techn. Klaus-Peter Adlassnig und Ao. Univ.-Prof. Dr. med. Michael Hiesmayr sowie DI Andrea Rappelsberger.

Für ihre Unterstützung in allen Phasen dieser Arbeit bedanke ich mich bei meiner Familie und ganz besonders bei Ines Bürg.

Inhaltsverzeichnis

1. Motivation und Einleitung

"*We found that nurses and respiratory therapists, using protocol guidance, weaned patients from mechanical ventilation safely and more quickly than the team following the traditional practice of physician-directed weaning.*"

[Koll1997]

Dieses Zitat stammt aus der Studie „A randomized, controlled trial of protocol-directed versus physician-directed weaning from mechanical ventilation" von [Koll1997], die von 1995 bis 1997 an der Washington University School of Medicine, St. Louis, USA, an vier Intensivstationen im Barnes Hospital und Jewish Hospital durchgeführt wurde. Die Studie ging von der Hypothese aus, dass auf Intensivstationen das nichtärztliche Personal beatmete Patienten sicher vom Beatmungsgerät entwöhnen kann, indem es nach Richtlinien und Algorithmen vorgeht, die von leitenden Ärzten vorher in einem Protokoll festgelegt worden sind.

Die Entwöhnung vom Respirator (siehe auch Kap. 4. Entwöhnung vom Respirator („Weaning"), S. 20) ist eine alles andere als triviale Aufgabe. Man weiß zwar, was man darunter zu verstehen hat, nämlich die schrittweise Reduktion von mechanischer Beatmung bis hin zur selbständigen Spontanatmung des Patienten. Allein, über die richtige Strategie, um dieses Ziel möglichst rasch und effizient zu erreichen, herrscht keine Einigkeit.

1.1 Aufbau der Studie

Insgesamt 357 Patienten wurden in die Studie aufgenommen, wovon 179 (50,1%) nach einem vorher von Chefärzten erarbeiteten Protokoll durch nichtärztliches Personal und 178 (49,9%) durch Intensivmediziner nach deren klinischer Erfahrung entwöhnt wurden. Die Zuteilung der Patienten zu den beiden Vergleichsgruppen erfolgte nach dem Zufallsprinzip, es gab keine signifikanten Unterschiede in Alter, Geschlecht, ethnischer Herkunft, Vorerkrankungen, zugeteilter Intensivstation oder Beatmungsindikation.

Als primäre Ergebnisvariable wurde die Dauer der mechanischen Beatmung betrachtet, weiteres Augenmerk lag auf notwendiger Reintubation, Aufenthaltsdauer und Mortalitätsrate im Krankenhaus und den finanziellen Kosten.

1.2 Ergebnisse

Tab. 1: Ergebnisse der Studie [Koll1997]

Ergebnis	Protokollgeleitete Entwöhnung (n = 179)	von Ärzten geleitete Entwöhnung (n = 178)	Unterschiede zwischen den Gruppen (95% KI)	p-Wert
Beatmungsdauer vor Entwöhnungsbeginn (h)	**39,6 ± 81,7[a] [17][b]**	**58,3 ± 101,1 [22]**	**-18,7 (-40,2 bis 2,8)**	**0,016**
Beatmungsdauer (h)	**69,4 ± 123,7 [28]**	**102,0 ± 169,1 [35]**	**-32,6 (-63,4 bis –1,8)**	**0,029**
Beatmungsdauer > 7 d, n (%)	**21 (11,7)**	**31 (17,4)**	**-5,7 (-13,0 bis 1,6)**	**0,128**
Erforderliche Reintubation, n (%)	23 (12,8)	18 (10,1)	2,7 (-4,0 bis 9,4)	0,417
Mortalität im Krankenhaus, n (%)	40 (22,3)	42 (23,6)	-1,3 (-10,1 bis 7,5)	0,779
Aufenthaltsdauer im Krankenhaus, (Tage)	12,7 ± 9,4 [11]	14,2 ± 11,7 [11]	-1,5 (-3,7 bis 0,7)	0,517

KI ... Konfidenzintervall
[a] ... arithmetisches Mittel ± Standardabweichung
[b] ... Zahlen in eckigen Klammern repräsentieren Medianwerte

Detaillierte statistische Analysen sind in [Koll1997] zu finden. Die eindrucksvollsten Ergebnisse (in der Tabelle fett gedruckt) liegen bei der Entwöhnungsdauer vor. Hier haben sich statistisch signifikante Unterschiede zugunsten der protokollgeleiteten Gruppe ergeben, d. h. die Beatmungsdauer ist hier deutlich kürzer. Dagegen sind bei Mortalität und Aufenthaltsdauer im Krankenhaus kaum Unterschiede festzustellen.

1.3 Diskussion

Die Studie hat gezeigt, dass nichtärztliches Personal Patienten auf Intensivstationen, einem festgelegten Protokoll folgend, sicher und rascher vom Beatmungsgerät entwöhnen kann als Ärzte, die nach der traditionellen Methode ihrer persönlichen Erfahrung vorgehen. Obwohl in der nach Protokoll entwöhnten Gruppe die Reintubationsrate vergleichsweise höher und Mortalität, Aufenthaltsdauer sowie die Gesamtkosten niedriger sind, hat keiner dieser Parameter statistische Signifikanz erreicht. Dennoch konnte eine Einsparung von 42.960 US$ in der nach Protokoll entwöhnten Gruppe erzielt werden. Das bedeutet gerade in Zeiten, in denen sich das Gesundheitssystem immer höherem Kostendruck ausgesetzt sieht, einen deutlichen ökonomischen Vorteil.

Die Gründe für den insgesamt größeren Erfolg der Entwöhnung nach Protokoll liegen nach Meinung der Autoren darin, dass der Entwöhnungsprozess früher initiiert wurde und seine Dauer insgesamt kürzer war.

1.4 Protokoll vs. Expertensystem ?

Nun wäre es aber wohl nicht korrekt, anzunehmen, dass Ärzte „schlechter“ entwöhnen als Schwestern und Pfleger. Die Ursache liegt wahrscheinlich eher in der unterschiedlichen Art und Weise, wie ärztliches und nichtärztliches Personal den Entwöhnungsprozess kontrolliert. Einen dieser Unterschiede stellt die Engmaschigkeit der Überwachung dar. Je kontinuierlicher das Patientenmonitoring, umso rascher kann auf Änderungen des Zustandes reagiert werden, und umso effektiver und schonender gestaltet sich der Entwöhnungsvorgang. Diese Forderung wird von Schwestern und Pflegern besser erfüllt, da sie sich den Patienten in kürzeren Abständen widmen können als das Ärzten in der Regel möglich ist.

Auch in anderen Krankenhäusern lassen sich ähnliche Strategien wie in [Koll1997] finden. Beispielsweise wird im Allgemeinen Krankenhaus (AKH) Wien auf der Intensivstation für Herz-Thoraxchirurgie ein Protokoll mit Regeln verwendet, nach dem das nichtärztliche Personal die Entwöhnung von Patienten vom Beatmungsgerät steuert. Ärzte überwachen zwar den Gesamtablauf des Entwöhnungsvorganges, werden aber sonst nur in nicht beherrschbaren Situationen und Notfällen hinzugezogen. Die bisher damit gemachten Erfahrungen weisen in eine ähnliche Richtung wie in [Koll1997].

Analysiert man die Vorgangsweise von nichtärztlichem Personal, lassen sich folgende Charakteristika feststellen:

- Die Ausführenden besitzen im Vergleich zu den eigentlichen Experten, den Ärzten, nur begrenztes Wissen über die Problemdomäne.
- Sie gehen nach einem vorher festgelegten Protokoll, einem Satz von Regeln, vor, in denen das Expertenwissen enthalten ist.
- Die Überwachung der Patienten ist um vieles engmaschiger als jene durch die Experten.

Diese Eigenschaften erinnern deutlich an die Art und Weise, wie Expertensysteme arbeiten [Gott1990]. Es liegt daher nahe, anzunehmen, dass ein solches System den Entwöhnungsvorgang auf ähnliche Art und möglicherweise noch effektiver steuern könnte als Menschen. Für das Expertensystem spricht vor allem, dass der Patientenzustand in sehr kurzen Abständen interpretiert werden kann. Dadurch ist das System in der Lage, rascher auf Zustandsänderungen zu reagieren und der Entwöhnungsvorgang bekommt ein kontinuierliches Profil.

In dieser Arbeit sollen drei Schlüsselkomponenten des medizinischen Expertensystems **FuzzyKBWean** ([Schu1996], [Schu1998]) beschrieben werden, das zur Entwöhnung von Patienten vom Beatmungsgerät entworfen wurde. Diese Komponenten sind

- Wissenserwerbskomponente
- Inferenzkomponente, welche mit Fuzzy-Logik arbeitet
- Datenbankkonzept

Vor der Beschreibung des Expertensystems FuzzyKBWean sollen einige Grundlagen erläutert werden, die mit dem Thema im Zusammenhang stehen.

2. Die Atmung

Die Aufgabe der Atmung ist der Gasaustausch. Man unterscheidet zwischen der **äußeren Atmung** in den Lungen, wo Kohlendioxid (CO_2) abgegeben und das Blut mit Sauerstoff (O_2) angereichert wird, und der **inneren Atmung** in den Körpergeweben, mit der Abgabe von Sauerstoff aus dem Blut und der Aufnahme des Stoffwechselendproduktes Kohlendioxid.

Ein Atemzyklus besteht aus einer Einatmung (**Inspiration**) und einer Ausatmung (**Exspiration**).

2.1 Aufbau und Funktion der Atemwege

Anatomisch wird zwischen den oberen und den unteren Atemwegen unterschieden.

- obere Atemwege: Nasenhöhle, Rachen (Pharynx), Kehlkopf (Larynx)
- untere Atemwege: Luftröhre (Trachea), Bronchialsystem

Die Atmungsorgane von der Nasenhöhle bis zu den Bronchioli terminales gehören zum **gasleitenden System**. In ihnen wird die Atemluft transportiert, es findet dort aber kein Gasaustausch statt. Das Volumen im gasleitenden System wird als **Totraum** bezeichnet.

Im Gegensatz dazu steht das **gasaustauschende System**, bestehend aus den Bronchioli respiratorii und den Lungenbläschen (Alveolen).

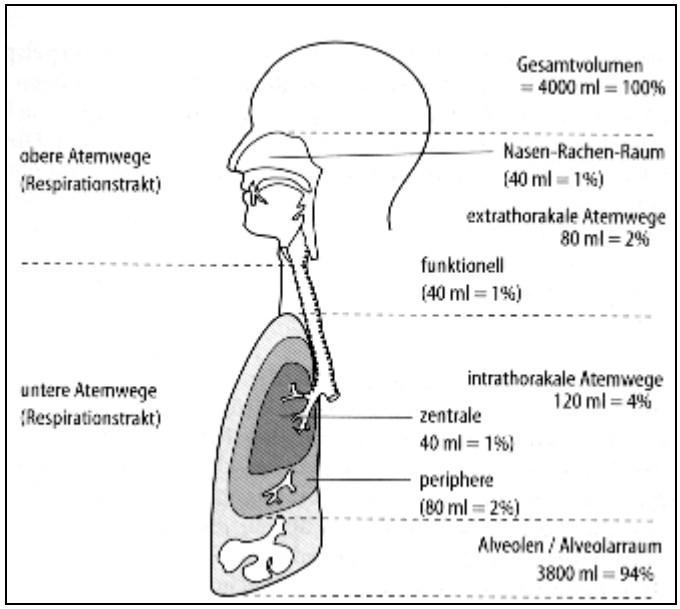


Abb. 1: Atemwege des Menschen (nach [Schm1995])

Die **Luftröhre** (Trachea) teilt sich nach 10–12 cm in den rechten und den linken Hauptbronchus, die am jeweiligen **Lungenhilus** in den rechten und den linken **Lungenflügel** eintreten. Dort verzweigen sich die Bronchien weiter bis in die **terminalen Bronchiolen** (Bronchioli terminales), die Endäste des gasleitenden Systems.

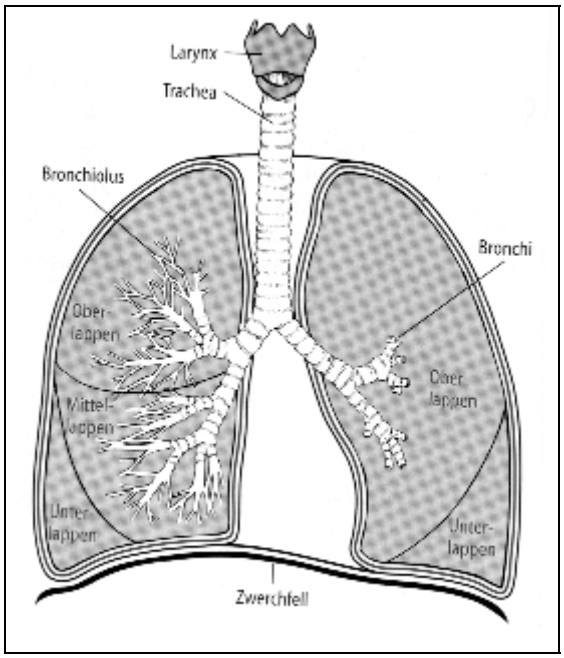


Abb. 2: Luftröhre (Trachea), Lungen und Bronchialbaum (nach [Schm1995])

Aus den Bronchioli terminales gehen die Bronchioli respiratorii hervor, die bereits zum gasaustauschenden System gehören. Sie münden in die **Lungenbläschen** (Alveolen), sechskantige bis kugelförmige Lufträume mit einem mittleren Durchmesser von 250–300 µm. Die gesamte von den Alveolen gebildete gasaustauschende Oberfläche beträgt je nach Geschlecht, Alter, Körpergröße und Trainingszustand 70–140 m^2.

Damit die Alveolen nicht kollabieren, wird von bestimmten Alveolarzellen ein Film aus Protein-Phospholipiden produziert. Dieser sogenannte **Surfactant** setzt die Oberflächenspannung der Lunge herab und stabilisiert dadurch die Alveolen.

Die Alveolen sind von feinen Blutgefäßen, den **Lungenkapillaren**, umkleidet. An dieser Grenze zwischen Blut und Luft, die aus den Basalmembranen des Alveolarepithels und den Kapillarwänden besteht, findet der Gasaustausch der äußeren Atmung durch **Diffusion** statt. Die Atemgase wandern durch die Membranen vom Ort höherer zum Ort niedrigerer Konzentration: Sauerstoff gelangt aus den Alveolen in das Kapillarblut, Kohlendioxid aus den Kapillaren in die Alveolen und wird von dort aus abgeatmet.

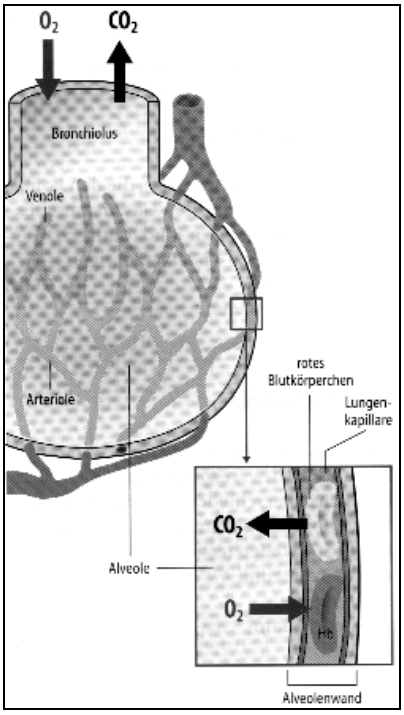


Abb. 3: Gasaustausch in den Alveolen (nach [Schm1995])

2.2 Lungenvolumina und -kapazitäten

Das in der Lunge befindliche Volumen kann durch folgende Größen beschrieben werden. Sie sind vom Alter und vom Geschlecht abhängig.

Totalkapazität (total lung capacity, TLC): Das nach einer maximalen Inspiration in der Lunge befindliche Volumen. Sie setzt sich aus Vitalkapazität und Residualvolumen zusammen.

Vitalkapazität (VC): Volumendifferenz zwischen maximaler Ein- und Ausatmung, ca. 74% der Totalkapazität. Sie kann weiter unterteilt werden in:

> **Inspiratorische Vitalkapazität** (VC_I): Volumen, das nach einer maximalen Ausatmung maximal eingeatmet werden kann.
>
> **Exspiratorische Vitalkapazität** (VC_E): Volumen, das nach einer maximalen Einatmung maximal ausgeatmet werden kann.
>
> **Atemzugvolumen** (AZV): Volumen, das bei jedem Atemzug ein- und ausgeatmet wird. Es beträgt beim Erwachsenen ca. 7ml/kg Körpergewicht (KG).
>
> **Inspiratorisches Reservevolumen** (IRV): Volumen, das nach einer normalen Inspiration noch zusätzlich eingeatmet werden kann.
>
> **Exspiratorisches Reservevolumen** (ERV): Volumen, das nach einer normalen Exspiration noch zusätzlich ausgeatmet werden kann.

Residualvolumen (RV): Jene Luftmenge, die nach einer maximalen Exspiration in der Lunge verbleibt, ca. 26% der Totalkapazität.

Funktionelle Residualkapazität (FRC): Summe aus RV und ERV.

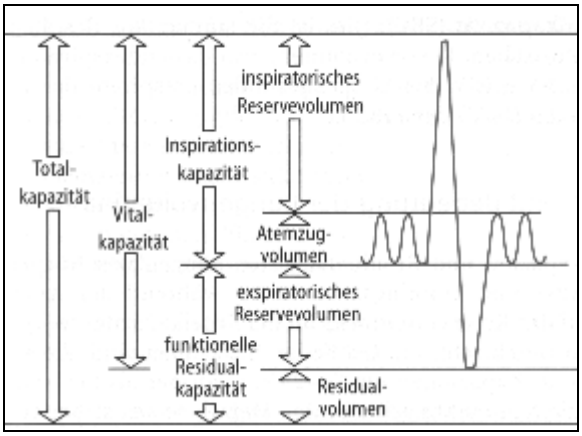


Abb. 4: Lungenvolumina und -kapazitäten (nach [Schm1995])

2.3 Weitere Atemgrößen

Atemfrequenz (AF): Anzahl der Atemzüge pro Minute, stark altersabhängig.

Atemminutenvolumen (AMV): Luftmenge, die pro Minute eingeatmet wird. AMV = AZV x AF

Totraumvolumen: Anteil am Atemzugvolumen, der nicht am Gasaustausch teilnimmt; beträgt beim Erwachsenen 2ml/kg KG oder ca. 30% des Atemzugvolumens.

I:E-Verhältnis: Verhältnis von Inspirations- zu Exspirationsdauer.

3. Beatmung

Die maschinelle Beatmung von Patienten ist ein Behandlungsverfahren der Notfall- und Intensivmedizin und wird bei einer ausgefallenen oder gestörten Atemfunktion eingesetzt. Sie ersetzt dabei die Eigenatmung des Patienten kurz- oder langfristig.

Tab. 2: Indikationen für maschinelle Beatmung [Baum1993]

Pumpversagen, -schwäche	Atemzentrum Nervenleitung Neuromuskuläre Übertragung Atemmuskulatur Kinematik der Thoraxwand Atemwege	
Lungenparenchymversagen	Alveolen	Gasphase gasaustauschende Oberfläche Wand (Surfactant, Epithel, Interstitium)
	Lungenkapillaren	Endothel
Pulmonale Perfusion		
Andere (extrapulmonal)	Hirnödem	Schädel-Hirn-Trauma Hirnoperationen
	Lungenödem	akutes Herzversagen
	nach Operationen oder Traumata	

Besonders in der postoperativen Phase müssen Patienten oft einige Zeit auf der Intensivstation verbringen und dort nachbeatmet werden, da es in dieser Situation oft zu einer Einschränkung der Lungenfunktion kommt [Ocze1996].

Die technische Umsetzung sieht so aus, dass von einem automatischen Beatmungsgerät (Respirator, engl. ventilator) Luft, der meist ein erhöhter Sauerstoffanteil zugemischt ist, mit Überdruck über einen Endotrachealtubus in die Atemwege des Patienten eingeblasen wird. Je nach Indikationsstellung werden unterschiedliche Beatmungsmuster eingesetzt. Das Spektrum der Beatmungsformen reicht von der **kontrollierten** (mandatorischen) Beatmung, bei der der Respirator die Atemtätigkeit zu 100% ersetzt, über verschiedene Arten der **augmentierenden** Beatmung, wo der Patient bereits einen Teil der Atemarbeit übernimmt, bis hin zur reinen **Spontanatmung**, die fallweise noch durch Sauerstoffanreicherung der Einatemluft unterstützt wird.

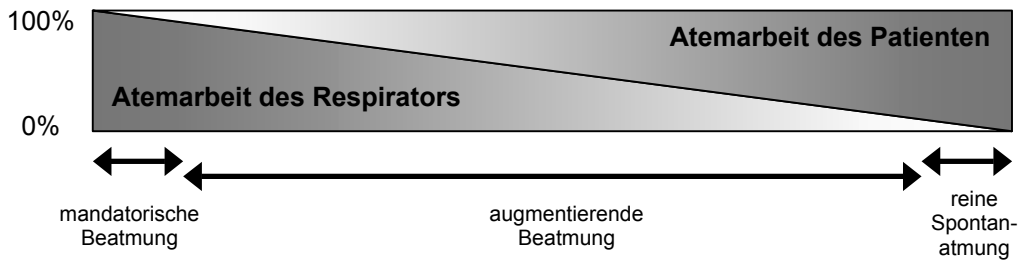


Abb. 5: Klassifikation der Beatmung nach Atemarbeit

PEEP (positive endexpiratory pressure): Positiver Druck, der am Ende der Ausatemphase in den Atemwegen aufrecht erhalten wird.

3.1 BIPAP-Beatmung (BIphasic Positive Airway Pressure)

Diese Beatmungsmethode kommt zusammen mit FuzzyKBWean zur Anwendung. Es handelt sich um eine Mischung aus Spontanatmung und zeitgesteuerter, druckkontrollierter Beatmung, die mandatorisch oder augmentierend erfolgen kann. In einem frei wählbaren Zeitraster wird zwischen zwei einstellbaren Druckniveaus umgeschaltet.

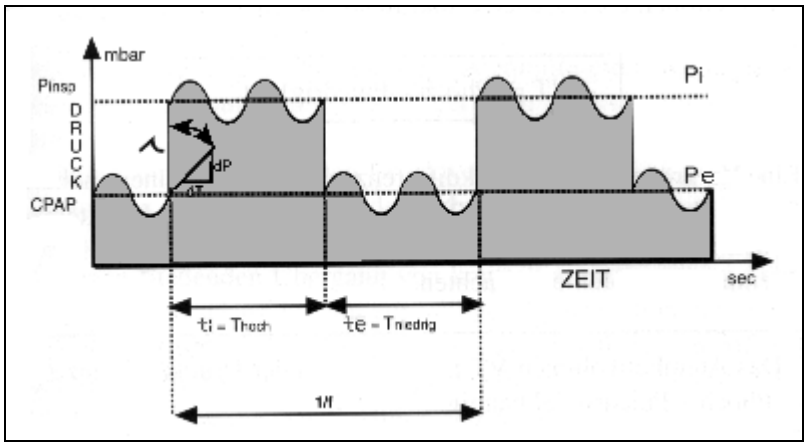


Abb. 6: BIPAP-Beatmung [Lars1997]

Am Beatmungsgerät können vier Parameter eingestellt werden:

- oberes (inspiratorisches) Druckniveau (p_i), PIP
- unteres (exspiratorisches) Druckniveau (p_e), PEEP
- Zeitdauer des oberen Druckniveaus (t_i)
- Zeitdauer des unteren Druckniveaus (t_e)

Der Vorteil des BIPAP ist seine große Flexibilität. Der Intensivmediziner hat die Möglichkeit, durch geeignete Wahl der vier Parameter eine Vielzahl von Beatmungsmustern zu realisieren. Der Patient wiederum kann in jedem Modus sowohl auf dem oberen als auch auf dem unteren Druckniveau spontan atmen. Das ermöglicht in der Entwöhnungsphase einen fließenden Übergang von der kontrollierten Beatmung zur Spontanatmung, ohne die Beatmungsform wechseln zu müssen. Die Entwöhnung wird dadurch zu einem kontinuierlichen Prozess.

Je nach Beteiligung des Patienten an der Atemarbeit lassen sich mehrere Formen der BIPAP-Beatmung unterscheiden. Die wichtigsten (siehe Abb. 7) sind:

- CMV-BIPAP: keine Spontanatmung, Beatmung erfolgt vollständig kontrolliert
- (S)IMV-BIPAP: Spontanatmung auf dem unteren Druckniveau
- Genuiner BIPAP: Spontanatmung auf beiden Druckniveaus
- CPAP: Spontanatmung, vollständige Angleichung der beiden Druckniveaus

CMV = *controlled mandatory ventilation*, kontrollierte Beatmung. Die gesamte Atemarbeit wird vom Respirator geleistet.

IMV = *intermittent mandatory ventilation*, Mischform zwischen Spontanatmung und kontrollierter Beatmung. Der Patient atmet spontan, aber mit unzureichendem Atemminutenvolumen. Das fehlende Volumen wird vom Respirator ergänzt, indem zwischendurch ein Beatmungshub mit fest vorgegebenem Volumen verabreicht wird.

Eine Variante davon stellt **SIMV** (s*ynchronized intermittent mandatory ventilation*, synchronisierte IMV-Beatmung) dar. Um zu verhindern, dass der Beatmungshub wie bei IMV möglicherweise in eine Ausatemphase des Patienten fällt, sorgt ein Trigger dafür, dass der Patient den Beatmungshub durch einen Einatemversuch selbst auslöst. Die Zeit, die der Respirator maximal auf den Einatemversuch wartet, kann eingestellt werden.

Genuiner BIPAP = Der Patient atmet spontan auf beiden Druckniveaus.

CPAP = *continuous positive airway pressure*, Spontanatmung mit kontinuierlichem positiven Atemwegsdruck. Der Patient atmet spontan auf einem erhöhten Druckniveau in den Atemwegen.

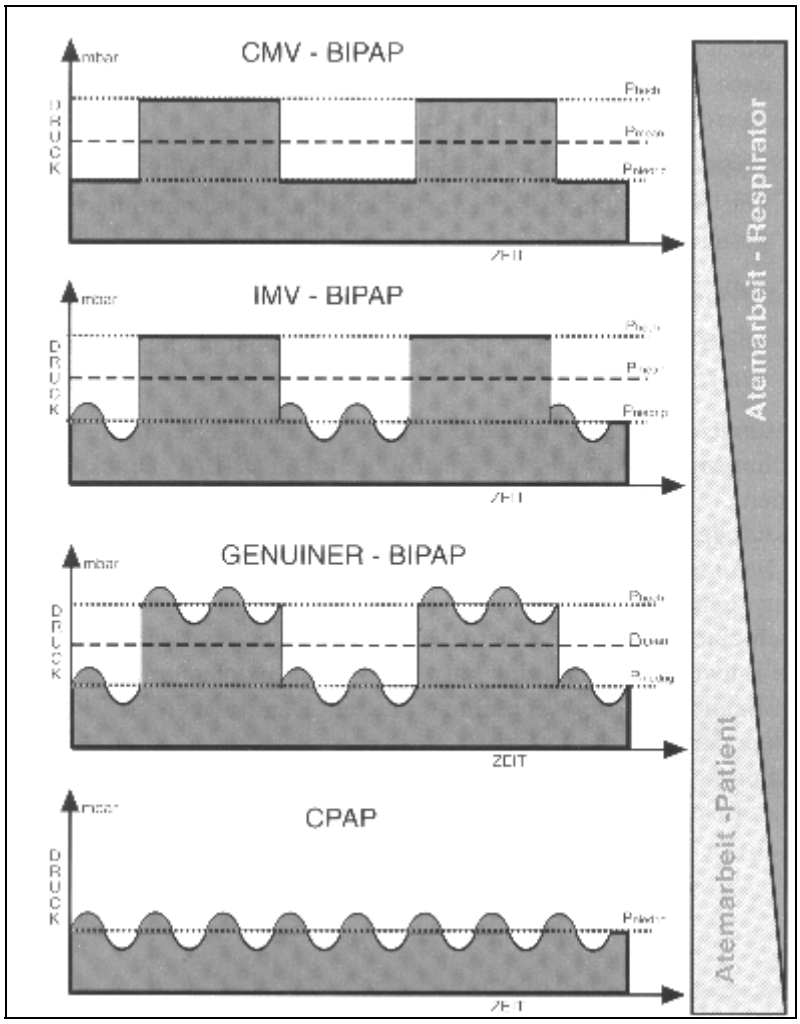


Abb. 7: Formen der BIPAP-Beatmung [Lars1997]

Weitere Varianten der BIPAP-Beatmung sind:

- IRV-BIPAP (inversed ratio ventilation BIPAP): BIPAP mit umgekehrtem Inspirations-/Exspirationszeit-Verhältnis (siehe Abb. 8).
- BIPAP-APRV (BIPAP airway pressure release ventilation): BIPAP, bei dem das obere Druckniveau kurzzeitig periodisch abgesenkt wird (siehe Abb. 9).

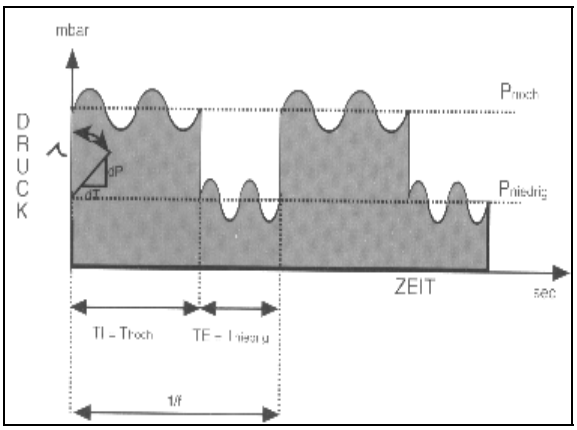


Abb. 8: IRV-BIPAP [Lars1997]

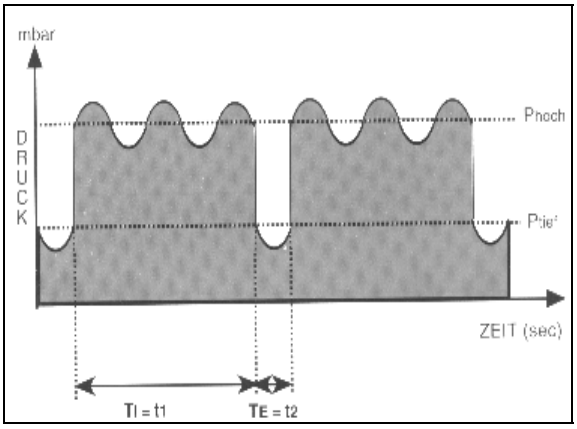


Abb. 9: BIPAP-APRV [Lars1997]

3.2 Negative Auswirkungen der Beatmung

Die maschinelle Beatmung soll aber nur eine vorübergehende Maßnahme sein. Sie wird nur so lange eingesetzt, bis der Patient wieder ausreichend spontan atmen kann. Der Grund dafür ist, dass eine Beatmung mit Überdruck letztlich eine unphysiologische und invasive Maßnahme ist, die mancherlei negative Auswirkungen auf den Organismus hat, beispielsweise auf das Herz-Kreislauf-System oder die Nierenfunktion. Darüber hinaus können auch ernste bis lebensbedrohliche Komplikationen auftreten.

Auswirkungen auf Organsysteme
Die maschinelle Beatmung kann das Herz-Kreislauf-System, die Nierenfunktion und die Durchblutung von Leber, Splanchnikus (Eingeweidenerv) und Gehirn negativ beeinflussen (zu Einzelheiten siehe [Lars1997]).

Sauerstofftoxizität
Bei Konzentrationen über 60% wirkt Sauerstoff (O_2) giftig und schädigt das Lungengewebe. Reiner Sauerstoff (100%) führt zu einer völligen Auswaschung von Stickstoff (N_2) und Edelgasen aus den Alveolen und es kommt zum Kollaps der Alveolen (Resorptionsatelektasen). Daher ist bei der Beatmung der Sauerstoffanteil prinzipiell nur so hoch wie nötig einzustellen.

Barotrauma
Hohe Beatmungsdrücke können die Lunge mechanisch schädigen. Es kommt dabei zur Zerreissung von bronchovaskulären und alveolären Membranen bis hin zum Luftaustritt. Weiters sind Schädigungen des Surfactant-Systems nachgewiesen. Das führt in mehreren Schritten zu einer völligen Abkoppelung von Lungenbezirken vom Gasaustausch (Atelektasen).

Respirator-assoziierte Pneumonie (Ventilator associated pneumonia, VAP)
Diese Form der Lungenentzündung ist eine nosokomiale Infektion, deren Erkrankungsrisiko bei intubierten Patienten um ein Mehrfaches höher liegt als bei nicht intubierten und mit der Dauer der Beatmung ansteigt. VAP besitzt eine sehr hohe Mortalität.

Lungenembolie
Bei der Lungenembolie kommt es zur Verschleppung von Blutgerinnseln (Emboli), meist aus den tiefen Beinvenen, in die Lungenstrombahn, wo sie Blutgefäße verlegen.

Gegenatmen des Patienten
Wenn der Patient gegen den Rhythmus des Respirators spontan atmet, kommt es zu vermehrter Atemarbeit mit erhöhtem O_2-Verbrauch und CO_2-Produktion. Ein solcher Zustand verursacht außerdem Stress und Angst beim Patienten.

Schwächung der Atemmuskulatur
Eine lange Sedierung und Relaxierung kann zu einer erheblichen und prolongierten Schwächung der Atemmuskulatur führen, was ein Pumpversagen (respiratorische Insuffizienz) verursachen und eine erfolgreiche Entwöhnung vom Respirator verhindern kann.

4. Entwöhnung vom Respirator („Weaning“)

Um die genannten Nebenwirkungen zu vermeiden, muss mit der Entwöhnung so früh als möglich begonnen werden. Man könnte sogar sagen, der Entwöhnungsvorgang beginnt bereits mit der Intubation. Durch eine geeignete Wahl von Beatmungsmustern und -parametern muss der Patient von der völligen Abhängigkeit vom Respirator zur völlig unabhängigen Spontanatmung übergeführt werden.

Dafür die geeignete Strategie zu finden ist aber nicht einfach. Es sind zahlreiche Faktoren zu berücksichtigen, und man muss etliche Forderungen erfüllen, die einander zum Teil widersprechen. Ein Entwöhnungsvorgang muss daher individuell auf den Patienten abgestimmt sein. Dies alles verlangt den Intensivmedizinern viel Erfahrung und auch Fingerspitzengefühl ab.

Leider zeigt sich in der Praxis, dass die Entwöhnung nicht selten nur suboptimal durchgeführt wird. Oft entsteht der Eindruck, dass mangels klarer Richtlinien zu viel herumprobiert wird, anstatt zielstrebig vorzugehen. Eine Vielzahl von Beatmungsmöglichkeiten, die meisten davon bereits im Respirator integriert, erschweren den Überblick und führen dazu, dass bei „schwierigen“ Entwöhnungen, besonders bei Langzeitbeatmeten, von einer Methode zur anderen gewechselt wird. Ein solches Vorgehen verwirrt und belastet Arzt und Patient.

4.1 Voraussetzungen für die Entwöhnung

Als Voraussetzungen für eine erfolgreiche Entwöhnung sind unter anderem folgende Kriterien ausschlaggebend [Baum1993]:

- psychologische Motivation, Hilfe durch das Pflegepersonal
- adäquate Mobilisierung
- entsprechender Ernährungszustand
- ausgeglichene Flüssigkeitsbilanz
- stabiler metabolischer Zustand
- gute Darmmotilität (Cave: Zwerchfellhochstand)
- keine Diskoordination der Atembewegungen
- stabile Herz-Kreislauf-Funktion
- adäquate respiratorische Funktion

So sehr diese Kriterien schon bei der reinen Beatmung anzustreben sind und eine Entwöhnung zweifellos auch effektiver verläuft, wenn sie erfüllt sind – unbedingte Voraussetzungen für den Entwöhnungsbeginn sind sie jedoch nicht [Baum1993]. Wartet man beispielsweise auf eine adäquate respiratorische Funktion, so kann sich die Entwöhnungsphase schwieriger und länger gestalten, da der optimale Zeitpunkt für den Entwöhnungsbeginn dann möglicherweise schon versäumt ist.

Andererseits hat ein verfrühter Entwöhnungsbeginn, der eine erneute Intubation und Anwendung von maschineller Beatmung nötig machen kann, ebenso negative Auswirkungen auf den Patienten.

Als geeignete Voraussetzungen für den Beginn der Entwöhnung lassen sich vielmehr folgende Atmungs- und Beatmungsparameter angeben:

Tab. 3: Voraussetzungen für den Entwöhnungsbeginn [Baum1993]

Parameter	Entwöhnungsbeginn bei
Vitalkapazität	$\geq$ 5 ml/kg KG
Inspiratorische Kraft	$\geq$ 10 cm H_2O
Höhe von PEEP / CPAP	$\leq$ 15 cm H_2O
pO_2 bei FiO_2 = 0,4	$\geq$ 60 mm Hg
pH-Wert	$\geq$ 7,3
Atemfrequenz	< 45/min
Atemminutenvolumen	< 18 l/min

4.2 Allgemeine Strategien zur Entwöhnung

Die Entwöhnung vom Beatmungsgerät ist keine statischer, sondern ein dynamischer Prozess, der individuell auf den Patienten abgestimmt sein muss. Daher ist es unumgänglich, bereits bei der Indikationsstellung und dem Beginn der Beatmung den Entwöhnungsvorgang ins Auge zu fassen. Die Entwöhnung selbst beginnt dann, wenn die Invasivität der Beatmung reduziert wird und verläuft schrittweise über immer weniger augmentierende Atemhilfen bis hin zur freien Spontanatmung und schließlich zur Extubation, bei der der Endotrachealtubus aus der Luftröhre entfernt wird. Der eigentliche Entwöhnungsprozess kann jedoch schon früher abgeschlossen sein, da der Patient durchaus auch über den Tubus spontan atmen kann. Für die schrittweise Reduktion der Invasivität ist die BIPAP-Beatmung besonders geeignet, da man hierbei einen fließenden Übergang von der kontrollierten zur spontanen Atmung erreichen kann, ohne die eigentliche Beatmungsform zu wechseln.

Als Beispiel sei eine Entwöhnungsstrategie angeführt, wie sie unter BIPAP-Beatmung angewendet wird:

Tab. 4: Entwöhnungsstrategie mit BIPAP [Baum1993]

Vorgehensschritt
Reduktion der FiO_2 auf < 0,5 anstreben
Reduktion des I:E-Verhältnisses auf $\leq$ 1:1
Reduktion des PEEP auf 7–9 mbar
Reduktion des oberen Druckniveaus in 2-bar-Schritten bis Δp von 8–12 mbar zwischen den beiden Druckniveaus erreicht ist
Ausdehnung des I:E-Verhältnisses auf 3 Sekunden (I:E = 3:3 sec)
Schrittweise Verlängerung der Phasenzeit E bis auf 12 Sekunden bei gleichzeitigem I von 3 Sekunden (entspricht einer maschinellen Atemfrequenz von 4/min)
Umstellung auf CPAP von ca. 6–8 mbar

Ein „schwieriger“ Entwöhnungsprozess ist gekennzeichnet durch einander abwechselnde Phasen von Entwöhnung, Beatmung und Stagnation bis schließlich der Durchbruch zur endgültigen Entwöhnung gelingt. Eine solche Situation belastet aber den Patienten und ist außerdem durch ihre Langwierigkeit sehr unökonomisch.

5. Expertensysteme

Ein **Expertensystem**, im folgenden **XPS** genannt, ist ein Computerprogramm, das seinen Anwendern Wissen und Fertigkeiten zur Verfügung stellt, über die normalerweise nur speziell ausgebildete und erfahrene Personen, die **Experten**, verfügen. Experten sind im wesentlichen durch folgende Eigenschaften gekennzeichnet [Gott1990]:

- Sie besitzen überdurchschnittliche Fähigkeiten, Probleme in einer bestimmten Domäne zufriedenstellend zu lösen, auch wenn diese Probleme keine eindeutige Lösung besitzen oder neu auftreten.
- Sie verwenden heuristisches Wissen und Erfahrungswerte, um spezielle Probleme zu lösen.
- Sie verfügen über Allgemeinwissen.
- Sie handeln oft intuitiv richtig, ohne ihren Entscheidungsvorgang begründen zu können.
- Sie sind in der Lage, Probleme unter Verwendung unvollständigen oder unsicheren Wissens zu lösen.
- Sie sind vergleichsweise selten und teuer.
- Ihre Leistungsfähigkeit schwankt je nach Tagesverfassung.
- Ein Experte allein ist oft nicht ausreichend (z. B. medizinisches Konsilium).
- Expertenwissen kann verlorengehen.

Ein XPS besitzt gegenüber menschlichen Experten mehrere wesentliche Vorteile:

- Es ist universell verfügbar, da für seinen Einsatz lediglich eine geeignete Hardware-Infrastruktur erforderlich ist.
- Es gibt keine Schwankungen der Leistungsfähigkeit.
- Für ein XPS sind alle Problemstellungen seiner Domäne gleich wahrscheinlich, während ein menschlicher Experte selten auftretende Konstellationen mitunter weniger gut erkennt als häufig vorkommende.

Nachteile sind dagegen:

- Die Einsatzmöglichkeiten eines XPS sind meist auf eng begrenzte Problemdomänen beschränkt.
- Es ist schwierig, dem XPS Allgemeinwissen („common sense knowledge“) zu vermitteln, welches für menschliche Experten dagegen selbstverständlich ist.
- Die Leistung eines XPS kann nur so gut sein wie die Qualität seiner Wissensbasis, die wiederum das Wissen menschlicher Experten enthält.

Mit der Konstruktion eines XPS sind drei Personen(gruppen) befasst:

Anwender sind jene Personen, die das fertige XPS zur Lösung von Problemen einsetzen. Sie besitzen üblicherweise im Vergleich zu Experten nur begrenztes Wissen über die Problemdomäne.

Experten stellen das zur Problemlösung erforderliche Wissen zur Verfügung.

Zwischen beiden Gruppen stehen die **Wissensingenieure** („knowledge engineers“). Ihre Aufgabe ist es, das Expertenwissen in eine für den Computer geeignete Repräsentationsform zu übertragen. Das ist kein einfacher Vorgang, da das Expertenwissen hoch komplex sein kann. Noch dazu lässt es sich oft nur schwer formalisieren, denn Experten handeln meist intuitiv und heuristisch.

5.1 Aufbau von Expertensystemen

Ein XPS besitzt schematisch folgenden Aufbau [Gott1990]:

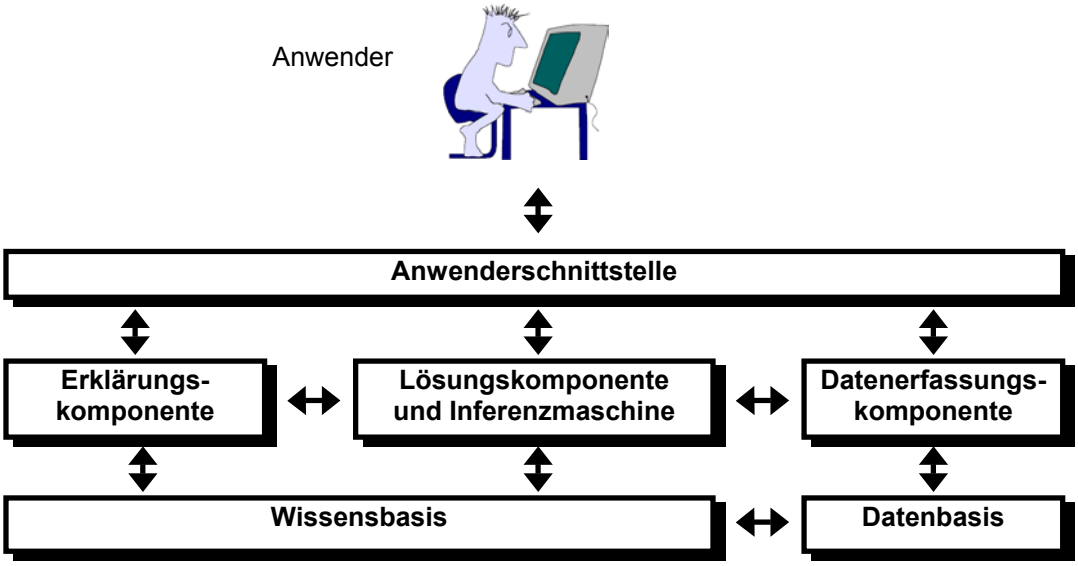


Abb. 10: Schematischer Aufbau eines Expertensystems

In der **Daten- und Wissensbasis** sind Informationen über die Problemdomäne gespeichert. Neues Wissen wird über die **Datenerfassungskomponente** akquiriert. Die **Lösungskomponente** und die **Inferenzmaschine** sind die zentralen Elemente eines XPS. Sie erarbeiten Lösungen für die gestellten Probleme, indem sie das Wissen der Wissensbasis systematisch auf die spezifischen Daten des Problems anwenden. Die **Erklärungskomponente** liefert Informationen über den Ablauf der Problemlösung. Durch sie wird der Lösungsvorgang des XPS dem Anwender nachvollziehbar dargestellt. Die **Anwenderschnittstelle** ermöglicht zwei Arten des Dialogs mit dem XPS. Einerseits mit dem Anwender zur Lösung eines Problems, andererseits mit dem Wissensingenieur für die Wartung und Erweiterung von Daten- und Wissensbasis.

5.2 Regelbasierte Expertensysteme

Wenn der Inhalt der Wissensbasis aus Regeln besteht, handelt es sich um ein **regelbasiertes XPS**. Diese Regeln liegen meistens in Form von WENN-DANN (IF-THEN) Konstrukten vor, d. h.

WENN <Prämisse> DANN <Konklusion>

Die Prämisse ist ein logischer Ausdruck, der entweder erfüllt (WAHR) oder nicht erfüllt (FALSCH) sein kann. Wenn die Prämisse erfüllt ist, kommt die Konklusion zur Anwendung. Auch menschliches Verhalten wird in vielen Situationen von solchen Regeln bestimmt.

Wenn die Konklusion eine Anweisung zur Handlung („action“) beinhaltet, werden derartige Regeln auch als „condition-action-rules“ bezeichnet.

Bsp.: WENN <EKG-Monitor Kammerflimmern zeigt> UND <kein Puls tastbar> DANN <Defibrillation>

5.3 Echtzeitverhalten

In manchen Problemdomänen kommt es nicht nur darauf an, dass das XPS korrekte Resultate liefert, sondern auch auf den Zeitpunkt, zu dem sie vorliegen. Beispielsweise kann ein richtiger Stellbefehl für das Beatmungsgerät, der zu spät eintrifft, genauso wertlos oder gar schädlich sein wie ein falscher.

Wenn der Zeitpunkt des Resultates relevant ist, spricht man von einem Echtzeitsystem (real time system). Der Zeitpunkt, zu dem das korrekte Resultat spätestens vorliegen muss, wird als „deadline“ bezeichnet. Wenn das Resultat auch nach der deadline noch Gültigkeit hat, heißt jene „soft“, andernfalls „firm“. Kann das Versäumen einer deadline zu katastrophalen Folgen führen, spricht man von einer „hard deadline“. Ein Echtzeitsystem, in dem mindestens eine hard deadline existiert, ist ein „hard real time system“ [Kope1997].

5.4 Medizinische Expertensysteme

Ein medizinisches XPS soll Ärzten bei der Lösung von medizinischen Problemen unterstützen. Meist handelt es sich dabei um diagnostische Aufgaben, für die bereits leistungsfähige Systeme im klinischen Einsatz sind (CADIAG, Hepaxpert, MYCIN, ...) [Adla2003]. Aber auch im therapeutischen Bereich kommen XPS zur Anwendung. Eines davon, FuzzyKBWean, ist Gegenstand der vorliegenden Arbeit.

Beim Einsatz von XPS ist zu beachten, dass die präsentierten Lösungen letztendlich auf dem Wissen von menschlichen Experten beruhen. Die Lösungsvorschläge sind in ihrer Güte entscheidend von der Qualität der Wissensbasis abhängig und dürfen daher nicht kritiklos akzeptiert werden, zumal das Wissen des XPS in den wenigsten Fällen vollständig ist. Das gilt vor allem für das Allgemeinwissen („common sense knowledge“), dessen sich menschliche Experten implizit bedienen, das aber in einem XPS nur schwer erfasst werden kann. XPS kommt daher oft die Rolle einer Assistentenfunktion zu, die menschliche Anwender bei schwierigen Fragestellungen unterstützen.

5.5 Das Expertensystem als Lehrer?

Wenn das XPS seine Entscheidungsfindungen logisch und leicht nachvollziehbar darlegen kann, mag das dem Anwender bisher verborgene oder nicht beachtete Aspekte aufzeigen und dadurch seinen Wissensstand verbessern. Dazu muss das XPS über eine hochwertige Erklärungskomponente verfügen.

6. Unscharfe Mengen und Fuzzy-Logik

Im täglichen Leben begegnen wir auf Schritt und Tritt Situationen, die nicht absolut präzise beschrieben werden können. Das kann an der zu großen Komplexität der Situation liegen oder auch daran, dass keine ausreichenden Informationen vorliegen, beispielsweise weil es sich um zukünftige Ereignisse handelt. Unsere Sprache spiegelt diese „Unschärfen" wider, sie ist voll von Aussagen wie „ungefähr", „ein bisschen", „noch etwas weiter" und ähnlichem [Zimm1993].

Gerade in der Medizin kommen derartige Formulierungen besonders häufig vor. Man betrachte zum Beispiel den Satz „Die Körpertemperatur des Patienten ist leicht erhöht.". Was versteht man unter dem Ausdruck „leicht erhöht"?

Wir Menschen interpretieren solche Phrasen meist auf intuitive Weise. Ein Computer dagegen kann damit nicht viel anfangen, da er auf konkrete, „scharfe", Werte angewiesen ist.

Computer arbeiten zudem fast immer auf der Basis einer zweiwertigen, „scharfen" Logik, die nur Ja/Nein- bzw. Wahr/Falsch-Entscheidungen zulässt und weitere Möglichkeiten ausschließt (lat. „tertium non datur"). Viele alltägliche und besonders medizinische Sachverhalte sind jedoch durch die bereits beschriebene inhärente „Un-Präzision" oder „Unschärfe" kennzeichnet. Eine scharfe Logik ist daher oft wenig geeignet, um für solche Probleme ein computergerechtes numerisches Modell zu erstellen.

Man lege zum Beispiel für die Sauerstoffsättigung des Blutes als Normalbereich das Intervall [96%,100%] fest. Nach der scharfen Logik wäre dann ein Wert von 95% bereits als „nicht normal" einzustufen. Das entspricht aber nicht unserer intuitiven Vorstellung. Umgangssprachlich würden wir einen solchen Wert wohl eher als „nicht mehr ganz normal" bezeichnen. Zur Modellierung derartiger Aussagen bedarf es besonderer mathematischer Werkzeuge.

6.1 Die Fuzzy-Set-Theorie

Das Konzept der unscharfen oder fuzzy[1] Mengen stammt von Lotfi A. Zadeh [Zade1965]. Es stellt eine Verallgemeinerung sowohl der klassischen Mengenlehre als auch der klassischen zweiwertigen Logik dar. Während in der klassischen Mengenlehre ein Element einer Menge entweder angehören kann oder nicht, wird bei unscharfen Mengen („fuzzy sets") für jedes Element der Grad der Zugehörigkeit angegeben. Er ist durch die Zugehörigkeitsfunktion oder charakteristische Funktion μ festgelegt, für deren Wertebereich üblicherweise das Intervall [0,1] gewählt wird. Dieses Konzept beinhaltet „volle Zugehörigkeit" (1) und „keine Zugehörigkeit" (0) als Grenzfälle, dazwischen gibt es jedoch einen kontinuierlichen Übergang.

Fuzzy-Logik repräsentiert jedoch weder „Wahrscheinlichkeit" (probability) noch „Unsicherheit" (uncertainty). Das zentrale Konzept ist der Begriff der „Vagheit" (vagueness) im Sinne von unklaren Grenzen zwischen Begriffen [Russ1995].

Bsp.: Ist eine Körpertemperatur von 38°C noch als „erhöht" oder bereits als „hohes Fieber" zu bezeichnen? Dies ist keine Frage von Wahrscheinlichkeit oder Unsicherheit, da der betreffende Temperaturwert ja exakt bekannt ist, sondern der Fall eines unscharfen Begriffes.

[1] fuzzy (engl.): unscharf, verschwommen

6.1.1 Zugehörigkeitsfunktionen

Definition: Sei U eine klassiche (scharfe) Menge. Eine unscharfe Menge (fuzzy set) A ist definiert als ein 2-Tupel (u, $\mu_A(u)$), mit u $\in$ U und $\mu_A(u)$:U$\rightarrow$[0,1]. Dann wird $\mu_A(u)$ als Zugehörigkeitsfunktion (membership function) bezeichnet [Schi1998].

Bei klassischen Mengen kann die Zugehörigkeitsfunktion nur die Werte 0 und 1 annehmen:

$$\mu_A(x) := \begin{cases} 1 & x \in A \\ 0 & x \notin A \end{cases}$$

Grundsätzlich kann jede beliebige analytische Funktion als Zugehörigkeitsfunktion verwendet werden. Im Hinblick auf effiziente Berechenbarkeit werden in der Praxis oft parametrisierte Funktionen verwendet, die durch ein Tupel von wenigen Werten dargestellt werden können. FuzzyKBWean beispielsweise bedient sich trapez- dreiecks-, rechtecks- und rampenförmiger Zugehörigkeitsfunktionen. Sie bestehen, grafisch betrachtet, lediglich aus geraden Streckenzügen, woraus die Zugehörigkeitsfunktion mittels einfacher geometrischer Methoden berechnet werden kann.

Definition: Die Trapezfunktion $\Pi_{\alpha,\beta,\gamma,\delta}(x)$:U$\rightarrow$[0,1] ist festgelegt durch

$$\Pi_{\alpha,\beta,\gamma,\delta}(x) := \begin{cases} 0 & x \le \alpha \\ \frac{x-\alpha}{\beta-\alpha} & \alpha < x \le \beta \\ 1 & \beta < x \le \gamma \\ \frac{\delta-x}{\delta-\gamma} & \gamma < x \le \delta \\ 0 & x > \delta \end{cases}$$

mit α, β, γ, δ, x $\in \Re$.

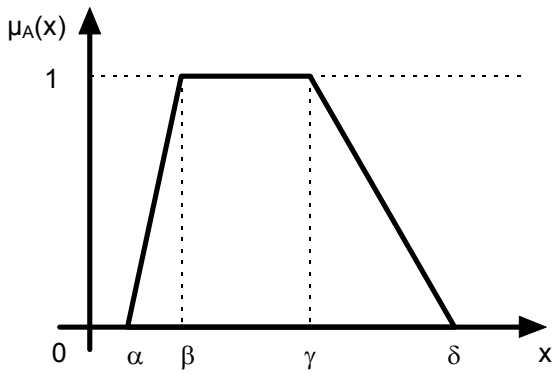


Abb. 11: Trapezfunktion

Definition: Die Dreiecksfunktion $\Delta_{\alpha,\beta,\gamma}(x)$:U$\rightarrow$[0,1] ist festgelegt durch

$$\Delta_{\alpha,\beta,\gamma}(x) := \begin{cases} 0 & x \le \alpha \\ \frac{x-\alpha}{\beta-\alpha} & \alpha < x \le \beta \\ \frac{\gamma-x}{\gamma-\beta} & \beta < x \le \gamma \\ 0 & x > \gamma \end{cases}$$

mit α, β, γ, x $\in \Re$.

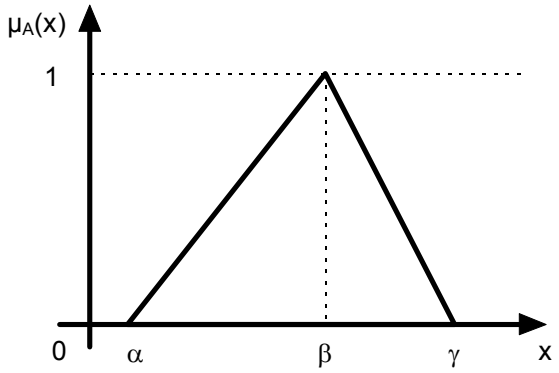


Abb. 12: Dreiecksfunktion

Die Dreiecksfunktion kann als Sonderfall der Trapezfunktion interpretiert werden, bei dem die beiden Parameter β und γ der Trapezfunktion zusammenfallen.

Eine weitere spezielle Form der Trapezfunktion ist die Rechtecksfunktion, bei der die Flanken einen unendlich steilen Anstieg haben.

$$\mu_A(x) := \begin{cases} 1 & \alpha \le x \le \beta \\ 0 & x < \alpha, x > \beta \end{cases}$$

mit α, β, $x \in \Re$.

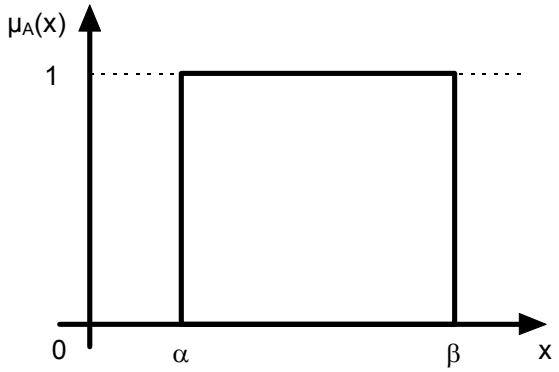


Abb. 13: Rechtecksfunktion

Diese Funktion entspricht der Zugehörigkeitsfunktion in der klassischen Mengenlehre, die nur die Fälle „keine Zugehörigkeit“ (0) und „volle Zugehörigkeit“ (1) unterscheidet.

Darüber hinaus sind durch geeignete Variation der Parameter verschiedene Mischformen aus den bisher genannten Funktionen darstellbar.

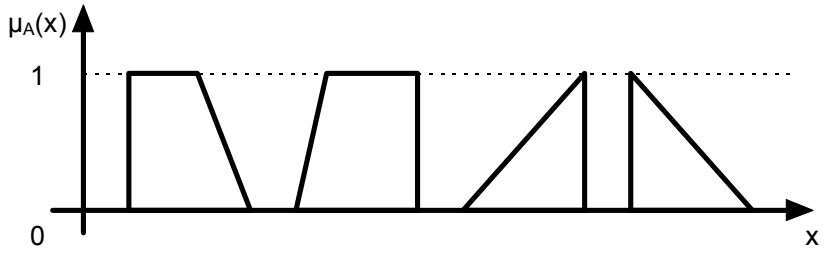


Abb. 14: weitere Zugehörigkeitsfunktionen

Definition: Die Rampenfunktion $\Gamma_{\alpha,\beta}(x) :U\rightarrow[0,1]$ ist festgelegt durch

$$\Gamma_{\alpha,\beta}(x) := \begin{cases} 0 & x \leq \alpha \\ \frac{x-\alpha}{\beta-\alpha} & \alpha < x \leq \beta \\ 1 & x > \beta \end{cases}$$

mit $\alpha, \beta, x \in \Re$.

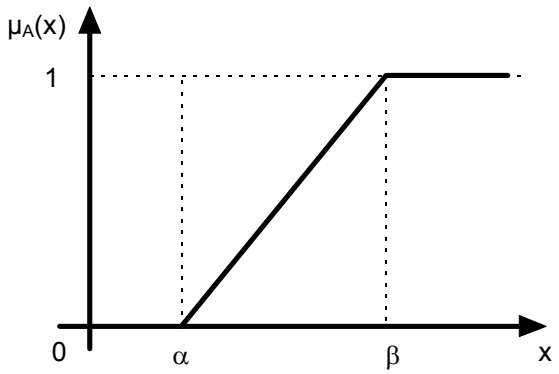


Abb. 15: Rampenfunktion Γ

Definition: Die Rampenfunktion $L_{\alpha,\beta}(x) :U\rightarrow[0,1]$ ist festgelegt durch

$$L_{\alpha,\beta}(x) := \begin{cases} 1 & x \leq \alpha \\ \frac{\beta-x}{\beta-\alpha} & \alpha < x \leq \beta \\ 0 & x > \beta \end{cases}$$

mit $\alpha, \beta, x \in \Re$.

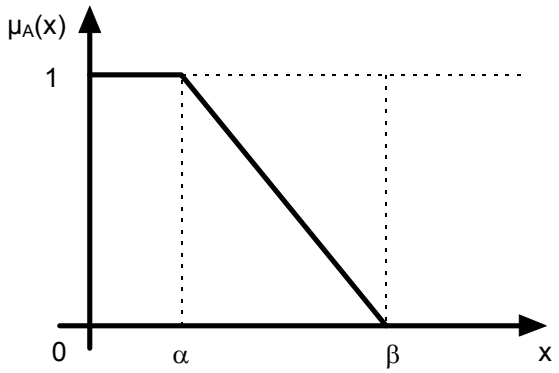


Abb. 16: Rampenfunktion L

6.1.2 Linguistische Variablen

Ein wesentliches Element der Fuzzy-Logik sind die linguistischen Variablen [Schi1998, Band1993]. Ihre Ausprägungen sind Elemente der natürlichen Sprache. Solche natürlichsprachlichen Ausdrücke sind oft „unscharf“, das heißt nicht exakt determiniert, sondern mit einer gewissen „lexikalischen Unsicherheit“ behaftet. Wie soll man beispielsweise den Ausdruck „Körpertemperatur *leicht erhöht*“ exakt definieren?

Aus diesem Grund werden die Ausprägungen von linguistischen Variablen in der Fuzzy-Logik nicht durch scharfe Zahlenwerte sondern durch Fuzzy-Mengen repräsentiert.

Man betrachte beispielsweise die linguistische Variable „Körpertemperatur“. Sie kann die Ausprägungen „niedrig“, „normal“, „erhöht“ und „hohes Fieber“ annehmen. Diese Ausprägungen werden als linguistische Terme bezeichnet, denen wiederum je eine Fuzzy-Menge assoziiert ist.

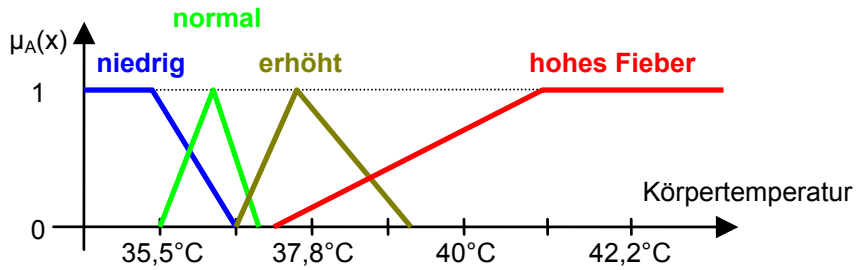


Abb. 17: Körpertemperatur in °C

Die Zuordnung von Fuzzy-Mengen zu den Termen einer linguistischen Variable erfolgt über eine Zuordnungsvorschrift M(T).

Bsp.: M(*hohes Fieber*) = $\{(x,\mu_{hohes\ Fieber}(x)) \mid 0 \le x \le 43\}$

mit $$\mu_{hohesFieber}(x) = \begin{cases} 0 & 0 < x \le 37{,}2 \\ \frac{x-37{,}2}{41{,}7-37{,}2} & 37{,}2 < x \le 41{,}7 \\ 1 & 41{,}7 < x \le 43 \end{cases}$$

und $T \in U = [0,43]$ U ... Grundmenge, zulässiger Wertebereich der Variable

6.2 Operatoren auf Fuzzy-Mengen

Wie in der klassischen Mengenlehre gibt es auch in der Fuzzy-Set-Theorie die Mengenoperationen Vereinigung, Durchschnitt und Komplementbildung, wobei gilt [Schi1998], [Band1993]:

Sei M eine Grundmenge und A, B $\subseteq$ M, dann gilt für x $\in$ M:

Vereinigung:	$x \in A \cup B$	$\Leftrightarrow$	$(x \in A) \vee (x \in B)$
Durchschnitt:	$x \in A \cap B$	$\Leftrightarrow$	$(x \in A) \wedge (x \in B)$
Komplementbildung:	$x \in C_M(A)$	$\Leftrightarrow$	$\neg(x \in A)$

Die hier vorkommenden logischen Operatoren $\vee$ (ODER), $\wedge$ (UND) und $\neg$ (NEGATION) sind in ihrer klassischen Form für die Anwendung auf Fuzzy-Mengen aber nicht brauchbar, da sie im Sinne einer konventionellen (zweiwertigen) Logik nur Aussagen mit den Werten WAHR (1) und FALSCH (0) verknüpfen können. Um unscharfe Aussagen mit ihren reellen Zugehörigkeitswerten im Intervall [0,1] verarbeiten zu können, muss die konventionelle Logik und damit ihre Operatoren zu einer für Fuzzy-Mengen geeigneten „Fuzzy-Logik“ erweitert werden.

Die Semantik der Fuzzy-Operatoren kann auf unterschiedliche Weise realisiert sein:
p, q $\in$ [0,1]

Tab. 5: Logische Operationen in der Fuzzy-Logik

Logischer Operator	Berechnungsmethode		Beispiel
UND ($\wedge$)	Minimum	min(p,q)	$0{,}8 \wedge 0{,}6 = \min(0{,}8 , 0{,}6) = 0{,}6$
	Algebraisches Produkt	p*q	$0{,}2 \wedge 0{,}5 = 0{,}2 * 0{,}5 = 0{,}1$
ODER ($\vee$)	Maximum	max(p,q)	$0{,}3 \vee 0{,}7 = \max(0{,}3 , 0{,}7) = 0{,}7$
NEGATION ($\neg$)	Komplement	1-p	$\neg 0{,}3 = 1 - 0{,}3 = 0{,}7$

Das logische UND kann beispielsweise als Minimumbildung oder als Multiplikation der zu verknüpfenden Zugehörigkeitswerte ausgeführt werden, während für das logische ODER der Maximum-Operator zur Anwendung kommt. Die logische NEGATION wird durch Komplementbildung des Zugehörigkeitswertes mit der Zahl 1 gebildet.

Da die Fuzzy-Logik eine Verallgemeinerung der klassischen zweiwertigen Logik darstellt, zeigen die Fuzzy-Operatoren UND, ODER und NEGATION ein „klassisches“ Verhalten, wenn die Zugehörigkeitswerte p und q den Grenzwerten 0 und 1 entsprechen.

Im Allgemeinen ist Fuzzy-Logik jedoch nicht mit Aussagen- und Prädikatenlogik konsistent, da Äquivalenzen wie

$$A \vee \neg A \Leftrightarrow T$$

in Fuzzy-Logik nicht mehr halten [Russ1995].

6.3 Fuzzy-Regelung (fuzzy control, FC)

Von einer Regelung (control) spricht man, wenn eine bestimmte Zustandsgröße eines technischen Prozesses dadurch innerhalb vorgegebener Grenzen gehalten wird, dass der Wert dieser Zustandsgröße ständig so kontrolliert wird, dass bei auftretenden Abweichungen dieser wieder auf den gewünschten Sollwert gebracht wird. Dabei entsteht ein als Rückwirkung bezeichneter Wirkungsablauf, der sich in einem geschlossenen Kreis – dem Regelkreis – vollzieht (DIN 19226) [Schi1998].

Eine Regelung ist somit überall dort erforderlich, wo in einem technischen Prozess eine Zustandsgröße durch den Einfluss von Störgrößen vom gewünschten Sollwert abweicht.

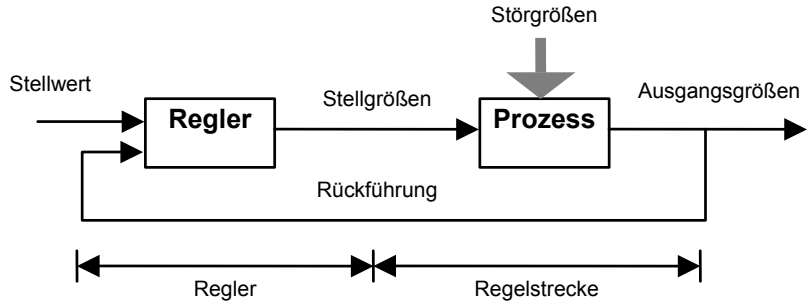


Abb. 18: Einfacher Regelkreis

Erfolgt die Regelung ohne dass ein menschlicher Operator im Regelkreis integriert ist, spricht man von „geschlossener Kopplung“ (closed-loop), andernfalls von „offener Kopplung“ (open-loop) [Kope1997].

Üblicherweise wird das zeitliche Verhalten von Regelstrecken nach Anregung durch eine eingangsseitige Funktion mit Differentialgleichungen n-ter Ordnung mit konstanten Koeffizienten beschrieben. Dieser exakte Ansatz wird aber schon extrem aufwändig, wenn die Voraussetzung der konstanten Koeffizienten nicht gilt – was bei realen Prozessen oft der Fall ist.

Es hat sich bei vielen Systemen als vorteilhafter erwiesen, das Verhalten eines menschlichen Dispatchers nachzuahmen. Dieser Ansatz bindet „Erfahrungswerte“, „Heuristiken“, „intuitives Verhalten“ und „Daumenregeln“ als Expertenwissen in den Entscheidungsprozess ein und benötigt im Gegensatz zur klassischen Methode kein exaktes Prozessmodell, was einen wesentlichen Vorteil darstellt. Die Umsetzung erfolgt mittels eines Fuzzy-Reglers.

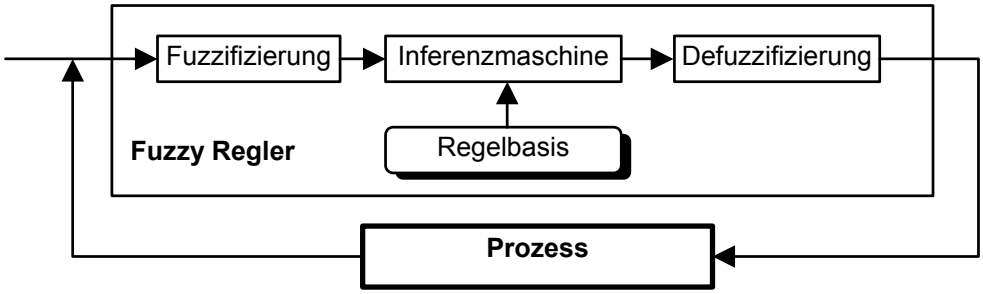


Abb. 19: Fuzzy-Regler

Die Arbeitsweise des Fuzzy-Reglers läuft in drei Schritten ab:

- Fuzzifizierung
- Regelauswertung (Inferenz)
- Defuzzifizierung

6.3.1 Fuzzifizierung

Bei der Fuzzifizierung werden mittels der Zugehörigkeitsfunktion die Zugehörigkeitswerte der (scharfen) Eingangsgrößen zu den Fuzzy-Mengen der jeweiligen linguistischen Variable bestimmt. Das Ergebnis ist eine Menge von Werten im Intervall [0,1].

6.3.2 Regelauswertung (Inferenz)

6.3.2.1 Regelbasis

In der Regelbasis ist das Expertenwissen zur Regelung des technischen Prozesses, meist in Form von sogenannten Produktionsregeln R_1, ..., R_n, gespeichert [Schi1998]. Eine solche Regel kann allgemein beschrieben werden als

R_k : IF $p_k(e)$ THEN $c_k(u)$

p_k ... *Prämisse* als Funktion der Eingangsgrößen e des Fuzzy-Reglers
c_k ... *Konklusion* als Aussage, die sich auf die Ausgangsgrößen u bezieht

Bsp.: IF $FiO_2 > 65$ THEN FiO_2change = -10

Die Regelbasis soll nach Möglichkeit vollständig und widerspruchsfrei sein. Ein solcher Idealzustand kann aber nicht immer erreicht werden, vor allem wenn der Umfang der Regelbasis ein bestimmtes Maß übersteigt. Die Komplexität der Basis wächst nämlich nicht-linear mit ihrer Größe.

6.3.2.2 Inferenz

Unter Inferenz versteht man die Auswertung der Regeln in der Regelbasis und die anschließende Zusammenfassung der daraus abzuleitenden Handlungsanweisungen (Konklusionen).

Die Auswertung einer Regel R_k läuft folgendermaßen ab:

1. Ermittlung des Wahrheitswertes der Prämisse. Er gibt an, in welchem Ausmaß die Regel feuert.

2. Bestimmung für jeden scharfen Ausgangswert, in welchem Maß er eine Handlungsanweisung der Regel darstellt.

3. Meist feuern mehrere Regeln der Regelbasis gleichzeitig. Daher muss als Gesamtresultat eine unscharfe Ergebnismenge durch Vereinigung der Ergebnismengen aller gefeuerten Regeln gebildet werden.

Je nach Implementierung der Fuzzy-Operatoren UND ($\wedge$) und ODER ($\vee$) unterscheidet man mehrere Inferenzmethoden. Die am häufigsten verwendeten sind MAX-MIN- und MAX-PROD-Inferenz.

Tab. 6: Inferenzmethoden

Inferenzmethode	Fuzzy UND	Fuzzy ODER
MAX-MIN	Minimumbildung	Maximumbildung
MAX-PROD	Algebraisches Produkt	Maximumbildung

6.3.3 Defuzzifizierung

Das Ergebnis der Inferenz ist wiederum eine Fuzzy-Menge und muss daher vor der Weiterverarbeitung in einen einzelnen numerischen Wert transformiert werden. Für die Defuzzifizierung gibt es verschiedene Methoden ([Schi1998], [Band1993]), wobei η im folgenden für das Resultat der Defuzzifizierung steht.

6.3.3.1 Maximum-Height

Diese Methode liefert als Resultat jenen Wert η, für den die Zugehörigkeitsfunktion der Ausgangsmenge ihr Maximum erreicht. Der Vorteil dieser Methode liegt in der einfachen Berechenbarkeit von η. Probleme treten nur dann auf, wenn die Zugehörigkeitsfunktion mehrere Maxima aufweist.

6.3.3.2 Mean-of-Maximum (MoM)

MoM liefert als Ergebnis η das arithmetische Mittel aller Werte, für die die Zugehörigkeitsfunktion ein Maximum erreicht. Diese Methode ist dann ungünstig, wenn im Verlauf der Zugehörigkeitsfunktion Strecken mit der Steigung 0, also plateauartige Stücke, enthalten sind.

6.3.3.3 Center-of-Gravity (CoG)

Die „Schwerpunktmethode“ CoG liefert die x-Komponente (Abszissenwert) des Schwerpunktes der Fläche, die der Graph der Zugehörigkeitsfunktion mit der Abszisse einschließt.

$$\eta = \frac{\sum_{r=1}^{k} c_i \cdot \mu_{u_r} \cdot u_r}{\sum_{r=1}^{k} c_i \cdot \mu_{u_r}} \qquad \text{für } k \geq 2,\ k \in \mathbb{N}$$

c_i ... Gewichtungsfaktor

μ_{u_r} ... Wert der Zugehörigkeitsfunktion

u_r ... Abszissenwert

Die Bedingung $k \geq 2$ ist notwendig, um eine Unzulänglichkeit der Schwerpunktmethode zu umgehen. Besteht die Summe in jenem degenerierten Fall nur aus einem Summanden ($k = 1$), so kürzen sich die Faktoren c_i und μ_{u_r} aus dem Bruch heraus. Übrig bleibt der unveränderte Wert u_r. Damit geht jedoch die fuzzy Eigenschaft der Berechnungsmethode verloren, sie wird „crisp“.

$$\eta = \frac{c_i \cdot \mu_{u_1} \cdot u_1}{c_i \cdot \mu_{u_1}} = u_1$$

Um das zu vermeiden, wird in der Inferenzkomponente von FuzzyKBWean eine modifizierte Version der CoG-Methode verwendet: Die abschließende Division wird nur dann ausgeführt, wenn $k \geq 2$ ist, andernfalls wird lediglich der Zähler des Bruches verwendet. Die Berechnungsmethode lautet daher:

$$\eta = \begin{cases} \dfrac{\sum_{r=1}^{k} c_i \cdot \mu_{u_r} \cdot u_r}{\sum_{r=1}^{k} c_i \cdot \mu_{u_r}} & k \geq 2 \\ c_i \cdot \mu_{u_1} \cdot u_1 & k = 1 \end{cases} \qquad k \in \mathbb{N}$$

6.4 Anwendung von Fuzzy-Logik und Fuzzy-Regelung in der Medizin

In der Medizin werden Fuzzy-Systeme vor allem in folgenden Gebieten eingesetzt ([Link2000], [Link2001]):

- Konservative Fächer (Interne Medizin): CADIAG [Adla2003]
- Invasive Medizin (Chirurgie, Anästhesie, ...)
- Bild- und Signalverarbeitung
- Labormedizin: TOXO [Adl2004]
- Grundlagenfächer

Die Systeme mit Fuzzy-Control sind zumeist regelbasiert (rule-based) und verwenden sowohl

- **offene Kopplung (open-loop)** im Sinne einer Unterstützung des medizinischen Anwenders bei der Entscheidungsfindung und
- **geschlossene Kopplung (closed-loop)** bei der das System den medizinischen Prozess unmittelbar regelt. In manchen Anwendungsbereichen haben sich closed-loop-Systeme als effektiver im Vergleich zu manueller Regelung erwiesen [Link2001].

6.5 Fuzzy-Logik und FuzzyKBWean

Der Ansatz von FuzzyKBWean ist, dass eine unscharfes („fuzzy“) Regelungskonzept der Denk- und Ausdrucksweise von medizinischen Experten näher kommt als ein Modell auf der Basis von scharfer („crisp“) Logik. Dadurch erhofft man sich ein Regelungsverhalten des Systems, das einen kontinuierlicheren, für den Patienten schonenderen und insgesamt effizienteren Entwöhnungsvorgang zur Folge hat.

Fuzzy-Logik und Fuzzy-Regelung haben sich vor allem in kommerziellen Anwendungen wie Steuerung von Zügen und Unterhaltungselektronik als sehr erfolgreich erwiesen [Russ1995]. Nach [Elka1993] liegt dies hauptsächlich an dem Umstand, dass diese Einsatzbereiche relativ kleine Regelbasen mit kurzen Kausalketten besitzen, und diese Systeme relativ einfach an die speziellen Erfordernisse angepasst werden können. Die Anpassung kann manuell oder durch selbstlernende Systeme erfolgen.

Bei komplexeren Aufgabenstellungen treten bei Fuzzy-Anwendungen aber ähnliche Probleme auf wie bei Systemen, die Ansätze wie „certainty factors“ oder die Dempster-Shafer-Theorie verwenden [Elka1993]: Bei langen Kausalketten können die Ergebnisse gegen 1 konvergieren oder unintuitive Resultate liefern.

Diese Probleme sollten bei FuzzyKBWean allerdings nicht auftreten, da dessen Wissensbasis relativ klein ist und die Kausalketten kurz sind.

7. Beschreibung von FuzzyKBWean

FuzzyKBWean ist ein medizinisches Expertensystem, das beatmungspflichtige Patienten auf Intensivstationen möglichst rasch und schonend vom Beatmungsgerät entwöhnen soll. Es arbeitet regelbasiert, verwendet Crisp- und Fuzzy-Logik und besitzt ein Quasi-Echtzeitverhalten.

7.1 Implementierung

FuzzyKBWean wurde für die Betriebssysteme Microsoft Windows 98/NT/2000/XP® erstellt. Als Entwicklungssystem diente Borland Delphi® 6.0 Client/Server Suite, als Datenbank wird eine InterBase®-Datenbank verwendet.

7.1.1 Hardwareumgebung

Für die Intensivstation steht ein zentraler Server zur Verfügung. Pro Intensivbett ist als Client je ein PC installiert. Die Bedienung der PC's erfolgt über Digiboard und Light pen (Lichtgriffel). Alle Computer sind über ein Subnetz des im AKH installierten Token-Ring-Netzwerkes verbunden.

7.1.2 Softwareumgebung

Als Patientendatenmanagementsystem (PDMS) wird das System PICIS® verwendet.

7.2 Arbeitsweise von FuzzyKBWean

Das XPS bekommt in Minutenabständen einen Satz Daten über den Patienten präsentiert. Es sind dies bestimmte physiologische Parameter, wie Sauerstoffsättigung, Gaskonzentrationen, Beatmungsdrücke und -frequenzen, etc. (siehe Tab. 7). Auf diese Daten werden die Regeln aus der Wissensbasis angewendet und aus den Ergebnissen neue Stellwerte für den Respirator errechnet. Diese Ausgabedaten werden am Bildschirm präsentiert. Der Arzt kann die Stellwerte akzeptieren oder eigene Werte verwenden. Da es vorläufig noch keine direkte Kopplung zwischen FuzzyKBWean und dem Beatmungsgerät gibt, sondern immer der Arzt zwischengeschaltet ist, wird diese Art von Betrieb als „offene Kopplung" oder „open-loop-System" bezeichnet. In weiterer Folge soll jedoch ein „closed-loop-Betrieb" realisiert werden, bei dem FuzzyKBWean den Respirator direkt ansteuert und der Arzt nur bei Notfällen eingreifen muss.

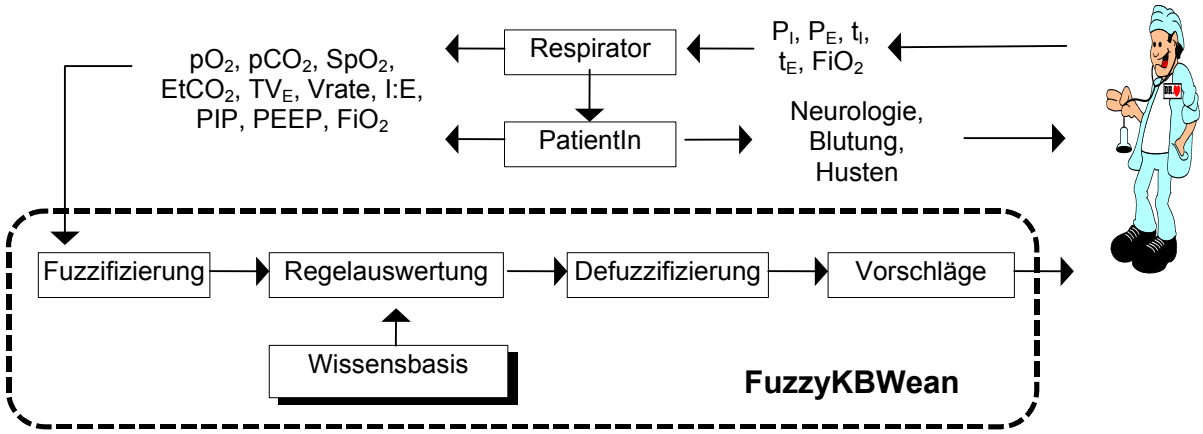


Abb. 20: Arbeitsweise von FuzzyKBWean

Tab. 7: FuzzyKBWean Eingabedaten

Eingabedaten für FuzzyKBWean	
pO_2	Sauerstoff-Partialdruck
pCO_2	Kohlendioxid-Partialdruck
SpO_2	Sauerstoffsättigung
$EtCO_2$	Endtidales Kohlendioxid
TV_E	Atemzugvolumen
Vrate	Atemfrequenz
I:E	Vehältnis Inspirations-/Exspirationszeit
PIP	peak inspiratory pressure
PEEP	positive endexpiratory pressure
FiO_2	Sauerstoffanteil in Einatemluft

Tab. 8: FuzzyKBWean Ausgabedaten

Ausgabedaten von FuzzyKBWean	
PIP	peak inspiratory pressure
PEEP	positive endexpiratory pressure
FiO_2	Sauerstoffanteil in Einatemluft

pO_2, pCO_2, SpO_2, $EtCO_2$ sind physiologische Parameter (siehe auch Anhang A – Physiologische Parameter, S. 97), die über verschiedene Verfahren am Patienten gemessen werden. SpO_2 und $EtCO_2$ stehen über Pulsoximetrie und Kapnometrie in Echtzeit zur Verfügung, pO_2 und pCO_2 müssen über eine Blutgasanalyse bestimmt werden.

$EtCO_2$ ist oft niedriger als pCO_2. Aus diesem Grund verwendet FuzzyKBWean die **Differenzkalibrierung**, bei der nach jeder Blutgasanalyse die Differenz

$$delta = pCO_2 - EtCO_2$$

gebildet wird. Die Variable *delta* wird dann in der Zeit bis zur nächsten Blutgasanalyse zum permanent messbaren $EtCO_2$ addiert und pCO_2 daraus rechnerisch ermittelt.

Bei TV_E, Vrate, I:E, PIP, PEEP und FiO_2 handelt es sich um Parameter, die am Beatmungsgerät eingestellt werden können. Sie fließen ebenfalls als Eingabewerte in FuzzyKBWean ein.

7.3 Arbeitsmodus

FuzzyKBWean verwendet zwei Arbeitsmodi, zwischen denen mittels geeigneter Regeln umgeschaltet werden kann:

Im Modus **Beatmung** („ventilation“) soll der Patient in einen stabilen Zustand gebracht werden. Das eigentliche Weaning erfolgt im Modus **Entwöhnung** („weaning“). Falls sich der Zustand des Patienten während dieses Prozesses verschlechtert, wird wieder in den Beatmungsmodus gewechselt usw.

Beide Modi verfügen über einen eigenen Satz von Regeln (siehe S. 48)

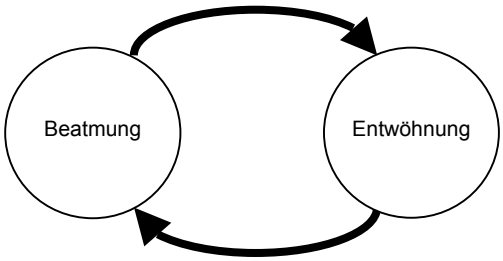


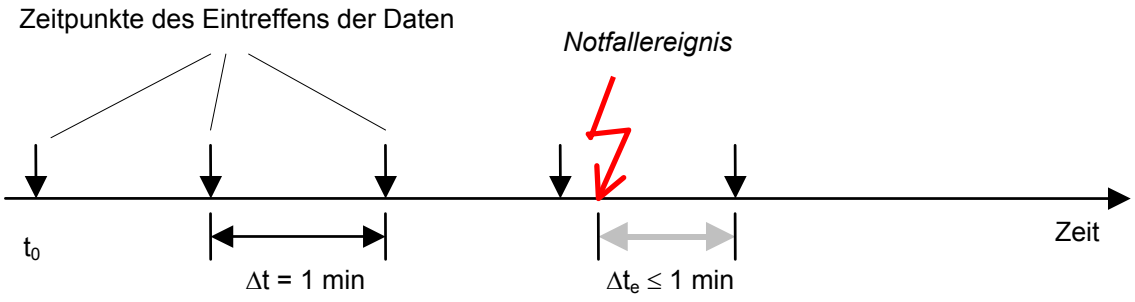
Abb. 21: Arbeitsmodi von FuzzyKBWean

7.4 Echtzeitverhalten

Das Echtzeitverhalten von FuzzyKBWean kann als zeitgesteuert („time-triggered“) beschrieben werden [Kope1997]. In Minutenabständen wird ein neuer Vektor mit Patientendaten geliefert, auf deren Grundlage die gesamte Wissenbasis durchlaufen und neue Resultate berechnet werden. Das Intervall von einer Minute erscheint für ein „Echtzeitsystem“ recht lang; vergegenwärtigt man sich jedoch, dass der Prozess der Entwöhnung an sich ein verhältnismäßig langfristiger ist, wird deutlich, dass dieses Minutenintervall einen ausgewogenen Kompromiss zwischen erforderlicher Antwortzeit und zu bewältigender Datenmenge darstellt.

In einer Minute kann die Wissensbasis, auch bei einer großen Datenbasis, auf einem geeigneten Rechner ohne Probleme komplett abgearbeitet werden. Dadurch lässt sich ein deterministisches Antwortverhalten garantieren. Für eingehendere Leistungsanalysen siehe auch Kapitel 9.6 Leistung der Datenbank (Performance), S. 67, und 11.3 Zeitverhalten, S. 86.

Während die einminütige Reaktionszeit für die Steuerung des Entwöhnungsvorganges ohne weiteres ausreicht, liegt der Fall bei Notsituationen dagegen anders.



Δt_e ... Zeitraum vom Eintritt des Notfallereignisses („emergency“) bis zum Erkanntwerden durch FuzzyKBWean

Tritt beispielsweise, unmittelbar nachdem ein Datensatz eingetroffen ist, ein Notfallereignis ein, kann es bis zu einer Minute dauern, bis die Situation von FuzzyKBWean erkannt wird. Das ist in manchen Fällen, wie z. B. bei malignen Herzrhythmusstörungen, schon zu lang. Allerdings fällt dieser Nachteil kaum ins Gewicht, da die Überwachungsgeräte (EKG, Pulsoximeter, Kapnometer, ...) ohnehin über integrierte Alarmfunktionen verfügen. Diese lösen in gefährlichen Situationen sofort einen Alarm aus, und das medizinische Personal kann augenblicklich einschreiten.

Aufgrund der bis zu einminütigen Reaktionszeit kann man bei FuzzyKBWean von einem zeitgesteuerten „Quasi-Echtzeitverhalten“ sprechen.

7.5 Wissensbasis von FuzzyKBWean

Die Wissensbasis von FuzzyKBWean besteht aus

- Variablen
- Werten
- Regeln
- Regelgruppen

Als **Variablen** werden alle veränderlichen Parameter bezeichnet, die zwischen FuzzyKBWean und dem PDMS bzw. dem Beatmungsgerät ausgetauscht werden. Variablen lassen sich nach Typ und Flussrichtung klassifizieren.

Der Typ einer Variable kann „numerisch“ oder „linguistisch“ sein. Während numerische Variablen mit Zahlen belegt werden, tragen linguistische Variablen eine verbale Bedeutung.

Bei der Flussrichtung unterscheidet man „Input“- und „Output“-Variablen. Erstere fließen vom Patienten bzw. Beatmungsgerät zu FuzzyKBWean, letztere als Stellgrößen von FuzzyKBWean zum Respirator.

Tab. 9: Beispiele für Variablen

	numerisch	linguistisch
input	SpO_2	Augenöffnen
output	PEEP	- - -

Linguistische Output-Variablen werden in FuzzyKBWean nicht explizit verwendet. Man könnte jedoch die Texte in den Regel-Konklusionen als solche Variablen interpretieren, da sie meist eine verbale Handlungsanweisung beinhalten.

Werte sind die möglichen Ausprägungen der Variablen. Ihr Typ richtet sich nach dem Typ der Variable, der sie zugeordnet sind.

Tab. 10: Beispiele für Werte

Variablentyp	Art der zugeordneten Werte	Beispiel
numerisch	Zahlen, Fuzzy-Mengen	95, PIP\normal
linguistisch	linguistische Terme	„Augenöffnen spontan“

Eine **Regel** hat immer die Form

IF <Prämisse> THEN <Konklusion>

Die Prämisse ist ein beliebig verschachtelter logischer Ausdruck, der entweder WAHR oder FALSCH sein kann. Ist die Prämisse WAHR, „feuert“ die Regel und ihre Konklusion kommt zur Anwendung. Die Konklusion beinhaltet entweder eine Handlungsanweisung für den Anwender, nach der neue Stellwerte für den Respirator gesetzt werden müssen, oder eine Informationsmeldung, beispielsweise wenn eine kritische Situation eintritt (Notfall).

Regeln können nach Bedarf auch zu **Regelgruppen** zusammengefasst werden. Auf diese Weise lassen sich mehrere Regeln auf einmal in Prämissen und Konklusionen ansprechen.

Weiters kann über die Konklusion der **Arbeitsmodus** von FuzzyKBWean eingestellt sowie Regeln und Regelgruppen blockiert bzw. deblockiert werden (siehe S. 37)

7.6 Andere Systeme und Ansätze

In den letzten Jahrzehnten, beginnend in den frühen 1980er Jahren, wurde eine Reihen von Systemen zur Entscheidungsunterstützung im Rahmen der maschinellen Beatmung entwickelt ([Tehr2008], [Brun2002], [Hend2006], [Jouv2007], [Lell2006], [Rose2007]).

Die bisher entwickelten Ansätze lassen sich anhand von verschiedenen Kriterien beschreiben:

Tab. 11: Einteilung der Systeme nach verschiedenen Kriterien [Tehr2008]

Ansatz	regelbasiert	modellbasiert	regel- und modellbasiert
Beatmungsformen	pressure support (PS)	SIMV oder IMV + PS	mehrere
Patienten	Erwachsene	Neonaten	Patienten mit ARDS
optimierte Parameter	Blutgase	Entwöhnungszeit	Blutgase und Entwöhnungszeit
Technologie	open-loop	closed-loop	open-loop und closed-loop

Die meisten Systeme arbeiten **regelbasiert**, auf der Grundlage von klinischen Guidelines und Protokollen, während **modellbasierte** Ansätze auf einem physiologischen Modell des Patienten beruhen. Es werden auch Mischformen dieser beiden Ansätze verwendet.

So unterschiedlich wie die gewählten Ansätze waren auch die Ergebnisse. Einige Systeme konnten ihr Patientenklientel erfolgreich entwöhnen, bei anderen zeigten sich indirekte positive Effekte, wie zum Beispiel eine geringere Anzahl von erforderlichen Blutgasanalysen.

Die meisten derartigen Systeme konnten sich jedoch nicht im klinischen Alltag durchsetzen, was vor allem folgende Gründe hatte:

- Mangelnde Verfügbarkeit: Die Systeme wurden von verschiedenen Forschungsgruppen entwickelt und zum Teil in multizentrischen Studien evaluiert, waren aber dennoch nicht auf breiter Basis zugänglich.
- Kein ausreichendes Training für die potentiellen Anwender.
- Keine ausreichende Integration in die kommerziell verfügbaren Respiratoren.

Dennoch können solche Systeme die Strategie gerade bei „schwierigen“ Entwöhnungen nachhaltig unterstützen und somit Qualität und Kosteneffizienz verbessern [Tehr2008].

Bis zu einem routinemäßigen Einsatz, vor allem von closed-loop-Systemen, sind trotz vielversprechender Ansätze, jedoch noch weitere Forschungen und Evaluierungen nötig. Dies betrifft besonders die Qualität der Eingabedaten (zuverlässige Sensoren, Artefakterkennung, Vermeidung von Fehlerfortpflanzungen, ...) [Tehr2008], verbesserte Benutzerschnittstellen [Schu2004], etc.

8. Wissenserwerbskomponente

8.1 Bisheriger Ansatz

In der ursprünglichen Version von FuzzyKBWean liegt die gesamte Wissensbasis vorerst als Klartext vor. Dieser „Quelltext“ ist für den Anwender lesbar und kann mit einem konventionellen Textverarbeitungsprogramm erstellt und bearbeitet werden.

```
VARIABLES
   HR, SpO2, FiO2, EtCO2, PEEP, PIP;

CONSTANTS
   SpO2_normal = (96, 100];
   SpO2_low = (92, 96];
   SpO2_very_low = (85, 92];
   SpO2_dramatic = (00, 85];
   .
   .
   .

RULES

Rule OXY_1
if     [SpO2 in SpO2_normal]
   and [FiO2 in FiO2_normal]
   and [PEEP in PEEP_normal or PEEP in PEEP_low]
then
   "[OXY_1]  FiO2 -5%"
   .
   .
   .
```

Abb. 22: Ausschnitt aus einer FuzzyKBWean-Wissensbasis (alte Version)

Die drei Komponenten der Wissensbasis, Variablen, Werte und Regeln, sind in drei Sektionen des Quelltextes deklariert, die von den Schlüsselworten VARIABLES, CONSTANTS und RULES eingeleitet werden.

Die ursprüngliche Version von FuzzyKBWean arbeitet nur mit scharfer Logik, daher sind die Werte (CONSTANTS) noch keine Fuzzy-Mengen im eigentlichen Sinn, sondern lediglich Intervalle, mit denen ein bestimmter Wertebereich für die assoziierte Variable deklariert wird. Man könnte ein solches Intervall aber auch als Grenzfall einer trapezförmigen Fuzzy-Menge interpretieren, bei der die beiden Flanken einen unendlich steilen Anstieg bzw. Abfall haben.

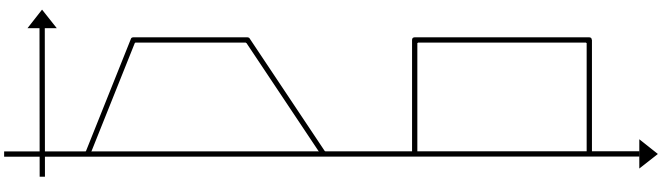


Abb. 23: Trapezförmige und rechteckige Fuzzy-Mengen

Weiters ist es möglich, wie in der mathematischen Schreibweise die Art der Intervallgrenzen festzulegen. Eine eckige Klammer „[“ oder „]“ bedeutet eine geschlossene Intervallgrenze, d.h. der betrachtete Grenzwert gehört noch zum Intervall. Von runden Klammern „(“ oder „)“ eingefasste Grenzwerte (offene Grenzen) gehören nicht zum Intervall.

Bsp.: Im Normalbereich der Sauerstoffsättigung des Blutes

```
SpO2_normal = (96, 100];
```

ist der obere Grenzwert 100 per definitionem noch Teil des Intervalls, der untere Grenzwert 96 jedoch nicht.

Jede Regel der Wissensbasis beginnt mit dem Schlüsselwort RULE, gefolgt vom eindeutigen Namen der Regel. Die Konklusion besteht aus einer Textmeldung, die meist eine Anweisung zum Einstellen von Beatmungsparametern oder eine Alarmmeldung beinhaltet.

Mit dem Quelltext der Wissensbasis kann FuzzyKBWean jedoch nicht unmittelbar arbeiten, er muss erst in eine geeignete, maschinenlesbare Form übergeführt werden. Bei jedem Start von FuzzyKBWean wird daher der Quelltext geladen und in zwei Schritten übersetzt:

Im ersten Schritt wird der Quelltext einer lexikalischen Analyse durch einen **Scanner**[2] unterzogen, der einen vorformatierten Zwischencode generiert. Dieser Zwischencode dient als Eingabe für den **Parser**, der die maschinenlesbare Version der Wissensbasis erzeugt.

Der ursprüngliche Ansatz weist jedoch einige Nachteile auf:

8.1.1 Fehlendes Entwicklungswerkzeug

Da man sich zum Erstellen der Wissensbasis eines konventionellen Texteditors bedienen muss, gestaltet sich der Entwicklungsprozess oft recht langwierig und unkomfortabel. Viele Standardelemente, beispielsweise die Schlüsselwörter zur Einleitung der Sektionen VARIABLES, CONSTANTS und RULES oder der Strichpunkt am Zeilenende müssen explizit angeschrieben werden und sorgen bei fehlerhafter Schreibweise, oder wenn sie überhaupt vergessen werden, für schwer auffindbare Fehler.

8.1.2 Komplizierte Fehlersuche

Je umfangreicher eine Wissensbasis ist, umso mehr Möglichkeiten gibt es, dass Fehler sich einschleichen. Ihr Spektrum reicht von syntaktischen Fehlern, wie falsch geschriebenen Schlüsselworten, über mehrfache Variablendeklarationen bis hin zu semantischen Fehlern, Inkonsistenzen im Regelaufbau der Wissensbasis selbst. Alle diese Fehler werden jedoch erst beim Laden der Wissensbasis manifest, also zur Laufzeit. Der Prozess der Fehlersuche und -korrektur, das sogenannte **Debugging**, sieht demnach folgendermaßen aus:

- Erstellen der Wissensbasis mit einem Texteditor
- Speichern der Wissensbasis und Beenden des Texteditors
- Starten von FuzzyKBWean
- Laden der Wissensbasis und Übersetzungsversuch
- Auftreten des Fehlers
- Beenden von FuzzyKBWean
- Starten des Texteditors
- Korrektur des Fehlers im Quelltext der Wissensbasis
- ... und der ganze Prozess beginnt von vorne, solange bis alle Fehler beseitigt sind.

8.1.3 Mangelnde Effizienz

Die Tatsache, dass die Wissensbasis jedes mal beim Start von FuzzyKBWean erneut übersetzt werden muss, ist recht unbefriedigend, da das eine Vergeudung von Rechenzeit darstellt. An sich

[2] Scanner (engl.) : „Abtaster“, Programmteil, der eine lexikalische Analyse des Quelltextes vornimmt

müsste es genügen, eine syntaktisch korrekte Wissensbasis nur einmal zu übersetzen, die ausführbare Version als eigene Datei abzuspeichern und beim Start von FuzzyKBWean zu laden.

Um alle diese Nachteile zu vermeiden, war also eine neue Wissenserwerbskomponente gesucht, die mehrere Forderungen erfüllen musste:

- Komfortable Unterstützung bei der Entwicklung von Wissensbasen und Erzeugung eines bereits vorformatierten und besser lesbaren Quelltextes.
- Fehlererkennungssystem, das bereits während der Entwicklungszeit eine präzise Identifikation aller Fehler und deren umgehende Korrektur ermöglicht.
- Erzeugung einer direkt ladbaren Version der Wissensbasis.
- Nicht nur Ausgabe von Textmeldungen durch die Regeln der Wissensbasis, sondern auch einer Liste von numerischen Stellwerten zur weiteren Verarbeitung.

8.2 Die Wissenserwerbskomponente: KBEdit

KBEdit, die neue Wissenserwerbskomponente, besteht aus zwei Subkomponenten, dem **Editor** und dem **Compiler** (siehe auch [Pomb1993]).

Der Editor dient der Erstellung des Quelltextes (source code), der vom Menschen lesbaren Form der Wissensbasis. Dagegen erzeugt der Compiler aus dem Quelltext eine ausführbare Version der Wissensbasis, die als eigene Datei gespeichert ist und von FuzzyKBWean unmittelbar, ohne weitere Übersetzungsschritte, geladen und verwendet werden kann (siehe auch [Aho1999]). Es bestehen Ähnlichkeiten zur Programmiersprache Java®, welche ebenfalls einen kompakten (plattformunabhängigen) Zwischencode, den so genannten „Bytecode“ erzeugt, welcher von einer „virtuellen Maschine“ (virtual machine, runtime environment „RTE“) ausgeführt wird [Abts2007].

Abb. 24: Erzeugung einer Wissensbasis mit KBEdit

Der Vorteil eines solchen Konzeptes liegt im wesentlichen darin, dass die Erstellung von Wissensbasen einfacher und gleichzeitig rascher bewältigt werden kann. Eine komfortable Benutzeroberfläche unterstützt die interaktive Modellierung aller Komponenten der Wissensbasis, Fehlersuche-Mechanismen ermöglichen ein gezieltes „Debugging“, noch bevor die ausführbare Fassung der Wissensbasis erzeugt wird. Dadurch wird garantiert, dass eine von FuzzyKBWean geladene und ausgeführte Wissensbasis bereits syntaktisch korrekt ist.

Der „Lebenszyklus“ einer Wissensbasis zerfällt somit in zwei voneinander abgekoppelte Phasen: die Entwicklungszeit und die Laufzeit.

8.2.1 Struktur der Quelltextes

Der Quelltext wird in konventionellen Textdateien gespeichert, die die Dateiextension „.kbs“ (für „knowledge base sourcecode“) tragen. Mit einem gewöhnlichen Texteditor betrachtet ist er jedoch vom Menschen nur schwer zu entziffern, erst wenn er von KBEdit geladen und dargestellt wird erhält er seine übersichtliche Form. Der Grund dafür ist, dass das Dateiformat in Hinblick auf effiziente Schreib- und Lesbarkeit für den Computer entworfen wurde.

Die Quelltextdatei ist in mehrere Abschnitte gegliedert. Jeder Abschnitt wird durch einen eindeutigen Namen, in eckigen Klammern stehend, eingeleitet und beinhaltet eine Liste mit Einstellungen. Für eine detaillierte Beschreibung des Formates der Quelltextdatei siehe Anhang B, S. 100.

8.2.2 Struktur der ausführbaren Wissensbasis

Die ausführbare Fassung der Wissensbasis ist für die rechnerinterne Verarbeitung in FuzzyKBWean bestimmt. Sie wird in Dateien mit der Extension „.kbe“ für „knowledge base executable“ gespeichert.

8.3 Erstellen von Wissensbasen mit KBEdit

8.3.1 Anlegen einer neuen Wissensbasis

Nach dem Start präsentiert sich KBEdit folgendermaßen:

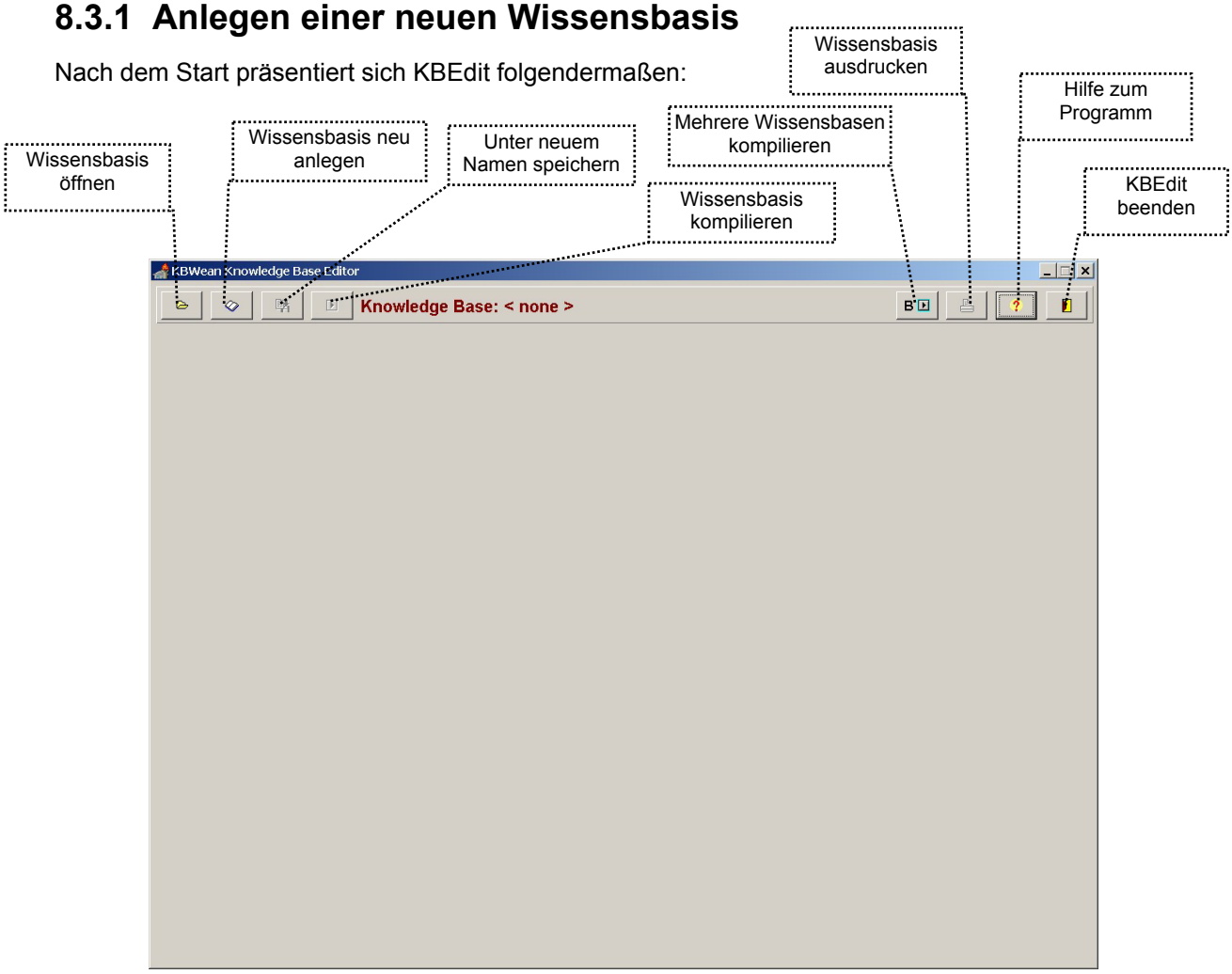


Abb. 25: KBEdit nach dem Programmstart

In der oberen Leiste befinden sich die zentralen Bedienungselemente. Hier können Wissensbasen geöffnet, neu erstellt, unter einem anderen Namen gespeichert, kompiliert und ausgedruckt werden. Durch Klicken auf die Schaltfläche „Wissensbasis neu anlegen“, wird eine neue Quelltextdatei erstellt. In einem Dialogfenster kann der Dateiname angegeben werden. Die Dateiextension „.kbs“ wird automatisch angefügt.

Sobald eine Wissensbasis neu angelegt oder eine bestehende Wissensbasis geöffnet wurde, erscheint das Editiermenü am unteren Rand des Programmfensters von KBEdit. Für jede Komponente der Wissensbasis steht dort eine Registermarke, über die die Seite mit dem gewünschten Inhalt aufgeschlagen werden kann.

8.3.2 Allgemeine Einstellungen (General settings)

Unter „General settings“ können allgemeine Eigenschaften wie Name und Beschreibung der Wissensbasis eingegeben und der Arbeitsmodus zu Beginn der Entwöhnung (siehe S. 37) festgelegt werden. Weiters ist es möglich, eine bestimmte Farbe anzugeben, mit der während des Betriebes von FuzzyKBWean Variablen gekennzeichnet werden, deren Werte außerhalb ihres Gültigkeitsintervalles liegen. Das Gültigkeitsintervall wird bei den Variablen eingestellt.

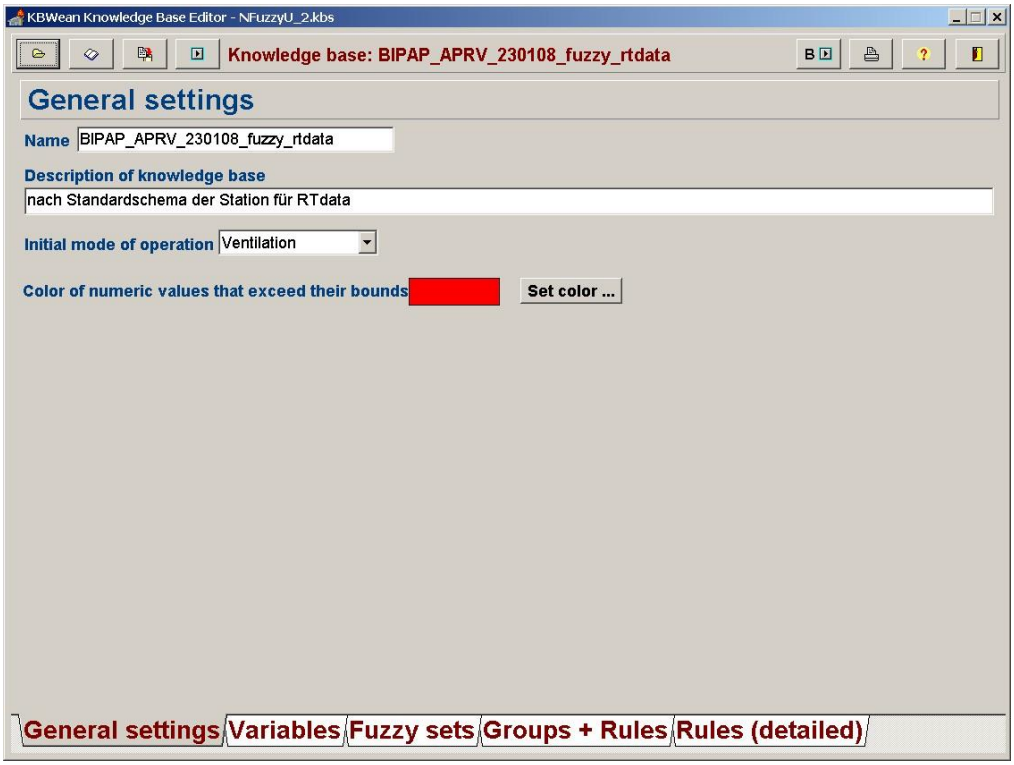


Abb. 26: KBEdit – Allgemeine Einstellungen

8.3.3 Variablen (Variables)

Auf dieser Seite werden Variablen definiert und ihre Eigenschaften festgelegt. Zusätzlich besteht die Möglichkeit, Variablennamen aus einer Textliste oder den Felddefinitionen einer Datenbank zu importieren.

An Eigenschaften können eingestellt werden:

Beschreibung (*Description of variable*): Beschreibung der Variable
Maßeinheit (*Unit of measurement*): Maßeinheit der Variable
Schrittweite für grafische Darstellung (*Increment (graphical representation)):* Gibt an, um wieviele Einheiten die Fuzzy-Sets in der grafischen Darstellung bei Betätigung der Verschiebe-Buttons versetzt werden.
Variablentyp (*Variable type*): Eine Variable kann von numerischem oder linguistischem Typ sein. Je nachdem wird sie mit reellen Zahlen oder Zeichenketten belegt.

Datenflussrichtung (*Data direction*): Input-Variablen sind z. B. Messwerte von Überwachungsgeräten, Output-Variablen z. B. Stellwerte für den Respirator. Nur letztere scheinen in den Konklusionen der Regeln (siehe S. 49) auf.
Gültigkeitsgrenzen (*Bounds*): Für jede Variable können absolute untere und obere Grenzwerte festgelegt werden, z. B. für die Sauerstoffsättigung das Intervall [0,100]. Falls solche Grenzen nicht existieren, kann auch „*unlimited*" eingestellt werden.
Gültigkeitsschwelle (*Validity threshold*): Zusätzlich kann ein Schwellwert angegeben werden, ab dem der Variablenwert als gültig interpretiert wird.

Für Variablentyp und Datenflussrichtung gilt eine Einschränkung: Output-Variablen können nicht vom Typ „linguistisch" sein, da es solche Variablen per definitionem im FuzzyKBWean-Konzept nicht gibt. Diese Kombination kann in KBEdit nicht eingestellt werden.

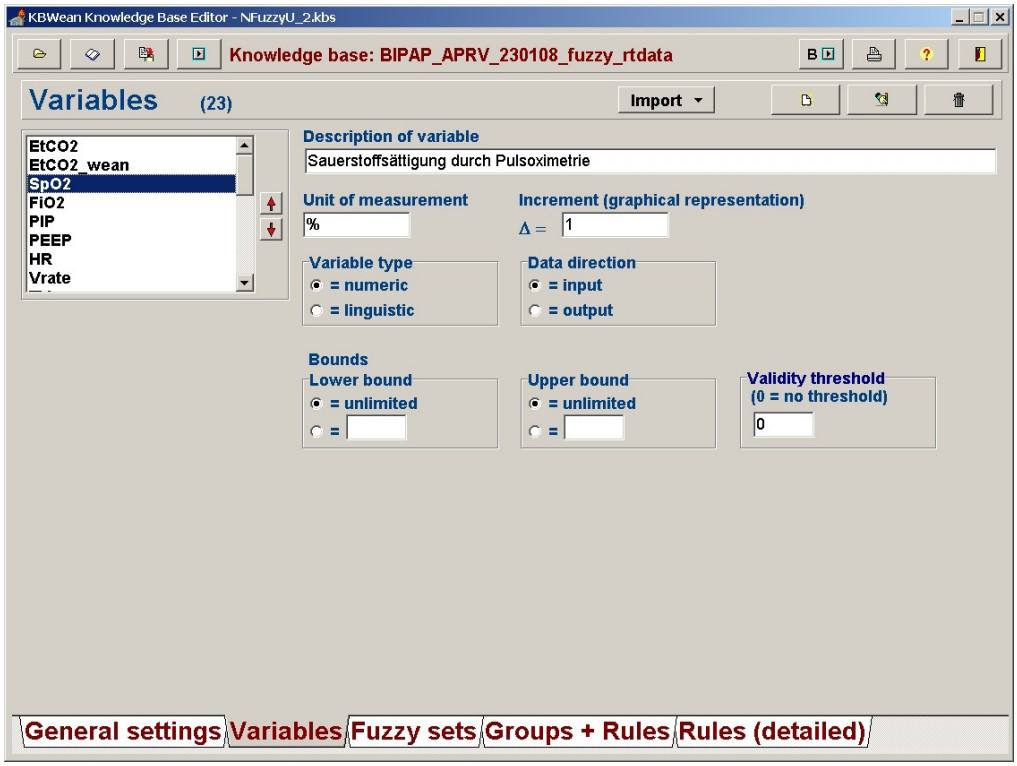


Abb. 27: KBEdit – Variablen

8.3.4 Fuzzy-Mengen / Linguistische Werte (Fuzzy sets / Values)

Hier werden die möglichen Werte(bereiche) der vorher angelegten Variablen definiert. Der Typ dieser Werte richtet sich nach dem Typ der momentan gewählten Variable:

- Fuzzy-Mengen (fuzzy sets) bei numerischen Variablen
- linguistische Werte bei linguistischen Variablen

Ein Fuzzy-Set in FuzzyKBWean ist immer trapezförmig, rechteckig oder dreieckig und wird definiert durch Angabe seiner vier Eckpunkte. Die Eckpunkte können direkt in Textfeldern eingegeben oder durch Betätigung der Pfeil-Schaltflächen verstellt werden. Der Versetzungsfaktor ist auf der Seite „Variablen" angegeben. Mittels der Klammern-Schaltflächen neben den Textfeldern werden die Intervallgrenzen festgelegt: „[" bzw. „]" für geschlossenes und „(" bzw. „)" für offenes Intervall.

Rampenförmige Fuzzy-Sets erhält man, indem man auf eine der Schaltflächen klickt, die die Eckpunktdaten des Sets flankieren.

Um die Modellierung der Fuzzy-Sets zu erleichtern gibt es die Möglichkeit einer grafischen Darstellung und verschiedene Hilfsmittel, wie Gitternetz, Ein- und Ausblenden einzelner Fuzzy-Sets und Skalierung auf den Nullpunkt des Koordinatensystems.

Für Fuzzy-Sets kann eine Beschreibung angegeben werden. Zusätzlich ist es möglich, eine Farbe einzustellen, mit der Messwerte in FuzzyKBWean eingefärbt werden, wenn sie im Wertebereich des betreffenden Fuzzy-Sets liegen.

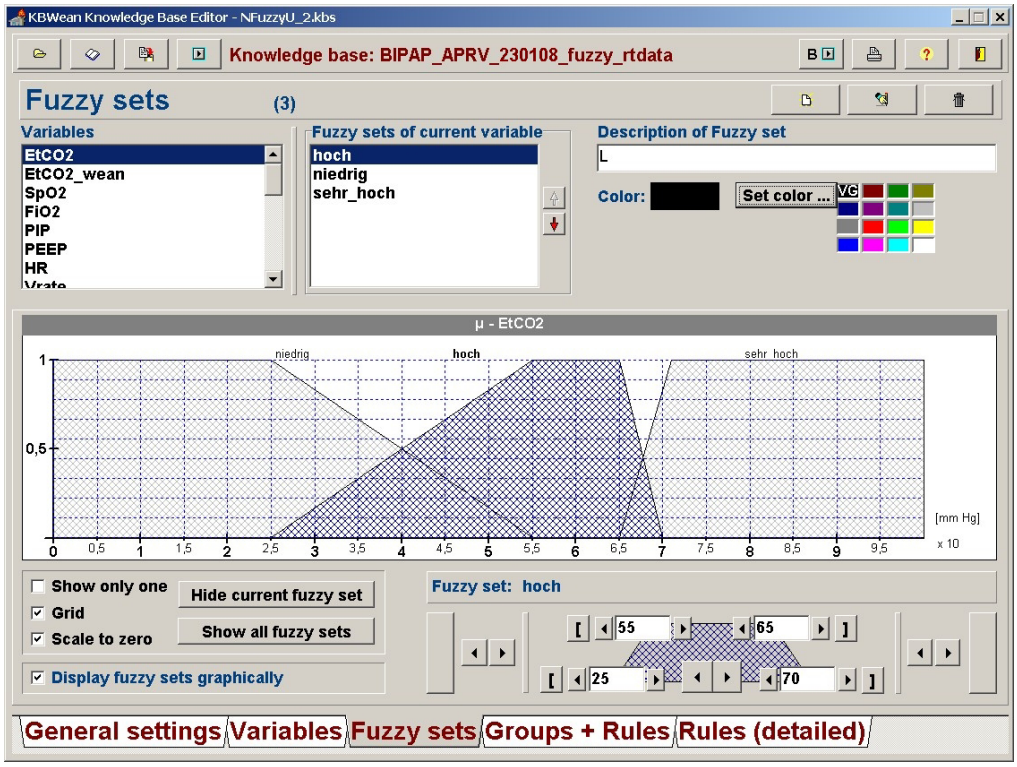


Abb. 28: KBEdit – Fuzzy sets

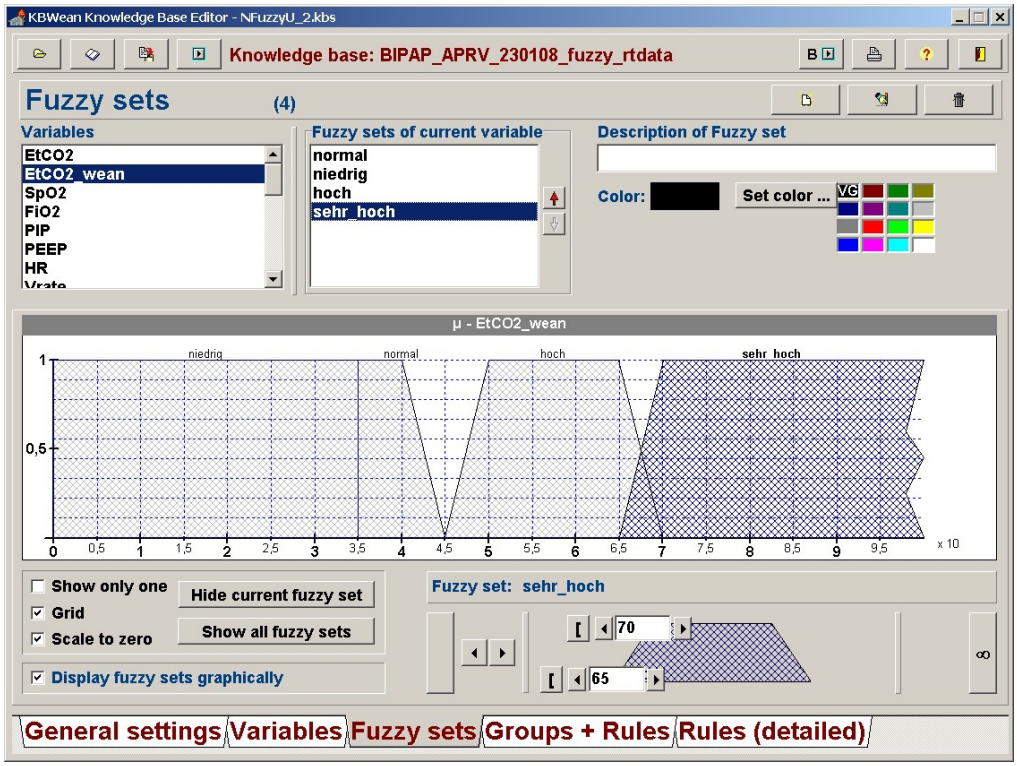


Abb. 29: Rampenförmiges Fuzzy-Set („sehr hoch“)

8.3.5 Regelgruppen (Groups)

Regeln können auch zu Gruppen zusammengefasst werden. Auf diese Weise lassen sich mehrere Regeln in einer Prämisse oder einer Konklusion ansprechen.

Es gilt eine 1:n-Zuordnung, d. h. eine Regel kann nur genau einer Gruppe angehören, Gruppen dagegen können beliebig viele Regeln enthalten.

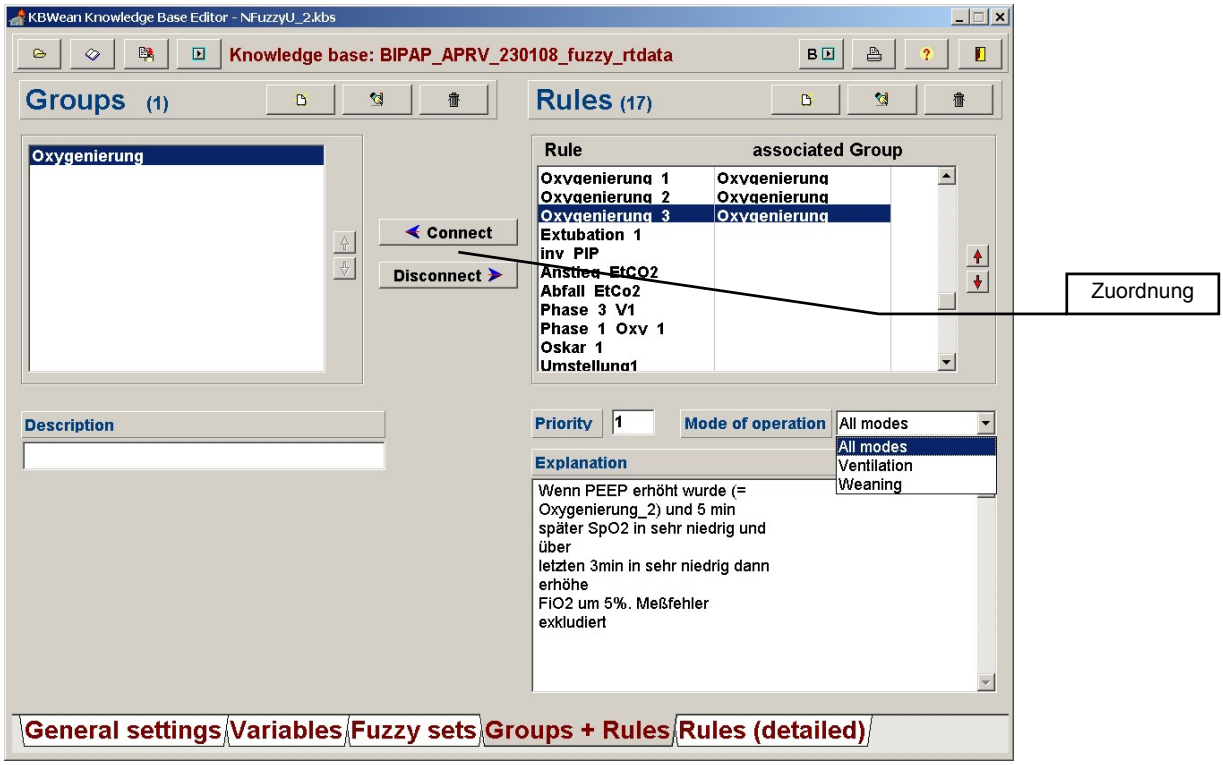


Abb. 30: KBEdit – Regelgruppen

An Eigenschaften können eingestellt werden:

Beschreibung (*Description*): Beschreibung der Regelgruppe
Priorität (*Priority*): Priorität der Regel
Arbeitsmodus (*Mode of operation*): Arbeitsmodus, dem die Regel zugeordnet ist
Erklärung (*Explanation*): Erklärung zur ausgewählten Regel

8.3.6 Regeln (Rules)

Nachdem eine neue Regel mit einem eindeutigen Namen generiert wurde, muss zunächst die Prämisse erstellt werden. Sie trägt eine baumartig verschachtelte Struktur.

8.3.6.1 Prämisse

⇒ Anklicken des Knotens, in dem der Eintrag erfolgen soll, mit der linken Maustaste. (Entfällt zu Beginn, wenn die Prämisse noch leer ist.)

⇒ Durch Klicken auf die rechte Maustaste während sich der Mauszeiger im Prämissenfeld befindet, öffnet sich das Editiermenü. Es enthält vordefinierte Einträge und Editierfunktionen.

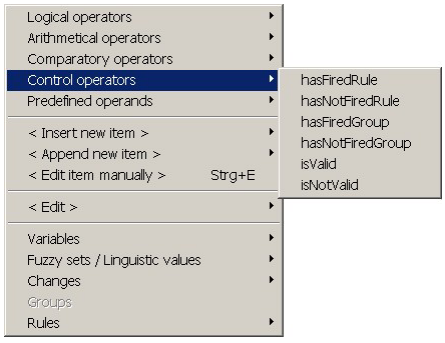


Abb. 31: Editiermenü – Beispiel 1

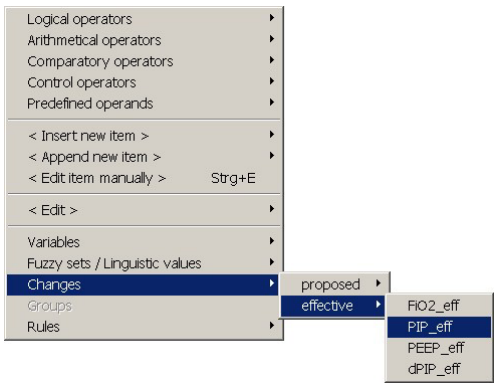


Abb. 32: Editiermenü – Beispiel 2

Tab. 12: Struktur des Editiermenüs

Ebene 1	Ebene 2	Ebene 3	Beschreibung
Logical operators	And		∧
	Or		∨
	Not		¬
Arithmetical operators	+		Addition
	-		Subtraktion
	*		Multiplikation
	/		Division
	DiffAbs		absolute Differenz
	DiffRel		relative Differenz
	Ago		vergangener Variablenwert
	Mean		arithmetisches Mittel
	StdDevAbsolute		absolute Standardabweichung
	StdDevRelative		relative Standardabweichung
Comparatory Operators	=		gleich
	<>		ungleich
	<		kleiner
	<=		kleiner oder gleich
	>		größer
	>=		größer oder gleich
	In		in Fuzzy-Set enthalten
	Is		Gleichheit für linguistische Variablen
Control Operators	HasFiredRule		Test ob Regel gefeuert hat
	HasNotFiredRule		Test ob Regel nicht gefeuert hat
	HasFiredGroup		Test ob Regelgruppe gefeuert hat
	HasNotFiredGroup		Test ob Regelgruppe nicht gefeuert hat
	IsValid		Test auf Gültigkeit von Variablen
	IsNotValid		Test auf Ungültigkeit von Variablen
Predefined Operands	All		alle
	Any		irgendein
	Ever		irgendwann
< Insert new item >	at the same level		Neuen Eintrag in der selben Ebene einfügen
	one level below		Neuen Eintrag eine Ebene darüber einfügen
	one level above		Neuen Eintrag eine Ebene unterhalb einfügen
< Append new item >	at the same level		Neuen Eintrag in der selben Ebene anhängen
	one level below		Neuen Eintrag eine Ebene unterhalb anhängen
< Edit item manually >			Eintrag direkt bearbeiten
< Edit >	Delete		Eintrag löschen
	Cut		Eintrag ausschneiden
	Copy		Eintrag kopieren
	Paste		Eintrag einfügen
Variables	{ Variablen }		Liste der Variablen
Fuzzy sets / Linguistic values	{ Variablen }	{ Werte}	Liste der Werte
Changes	Proposed	{ Ausgabevariablen }	vorgeschlagene Stellwertänderungen
	Effective	{ Ausgabevariablen }	tatsächliche Stellwertänderungen
Groups	{ Gruppen }		Liste der Regelgruppen
Rules	{ Regeln }		Liste der Regeln

⇒ Anwählen des gewünschten Elementes oder der gewünschten Funktion im Editiermenü

⇒ Wiederholung des Vorganges, solange bis die Prämisse fertig erstellt ist.

8.3.6.2 Konklusion

Im zweiten Schritt wird die Konklusion der Regel festgelegt. Sie besteht aus vier Teilen, nämlich

- Liste von Ausgabewerten (hauptsächlich Stellwertänderungen für den Respirator)
- Ausgabetext (verbale Konklusion, beliebiger Text)
- Blockierung und Deblockierung von Regeln oder Regelgruppen
- Einstellung des Arbeitsmodus: „Beatmung“ (ventilation) oder „Entwöhnung“ (weaning)

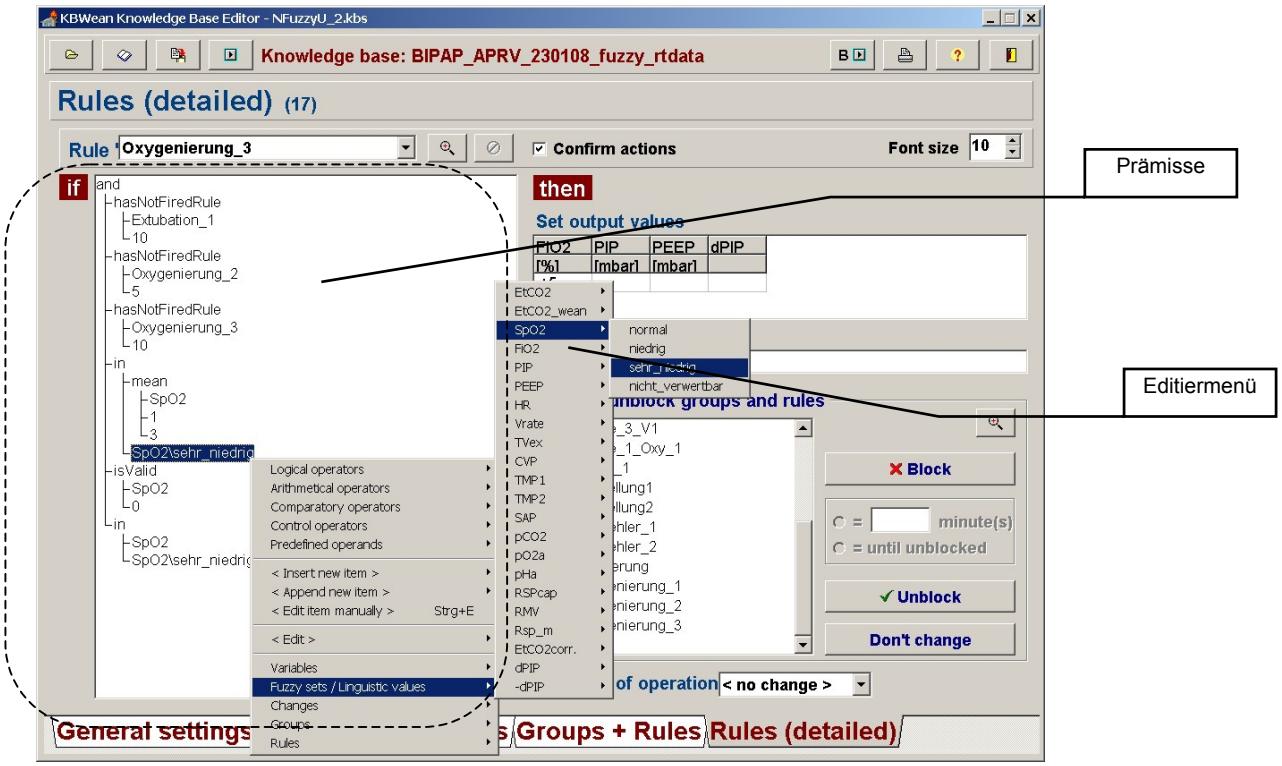


Abb. 33: KBEdit – Erstellung der Prämisse

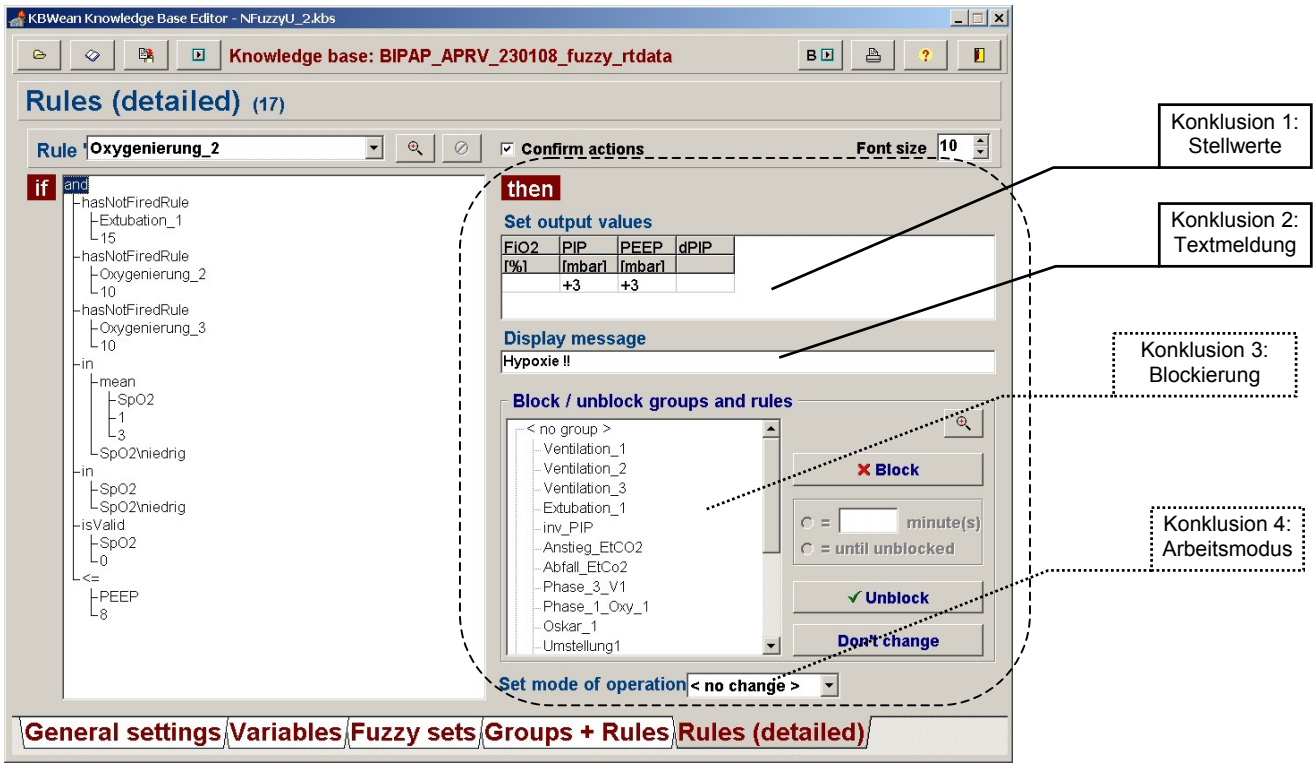


Abb. 34: KBEdit – Konklusionen

Alle vier Teile der Konklusion sind optional und können in beliebigen Kombinationen auftreten. In den meisten Fällen wird zumindest einer davon in der Regel verwendet werden. Es kann jedoch durchaus sinnvoll sein, auch Regeln ohne jegliche Konklusion zu definieren (siehe Kap. 8.6.3 Zeitrasterregeln, S. 58).

Die Stellwertänderungen für den Respirator können auf zwei Arten angegeben werden:

- absolut
- relativ (als Prozentangabe), durch Anfügen eines Prozentzeichens zum Wert

Eine *absolute Stellwertänderung* gibt an, wieviele Einheiten im Maßsystem der betrachteten Variable beim Feuern der Regel zum momentanen Wert addiert oder, bei negativem Vorzeichen, subtrahiert werden.

Im folgenden Beispiel werden, sollte die Regel feuern, die gerade aktuellen Werte der Variablen PIP und PEEP um jeweils 3 Millibar erhöht.

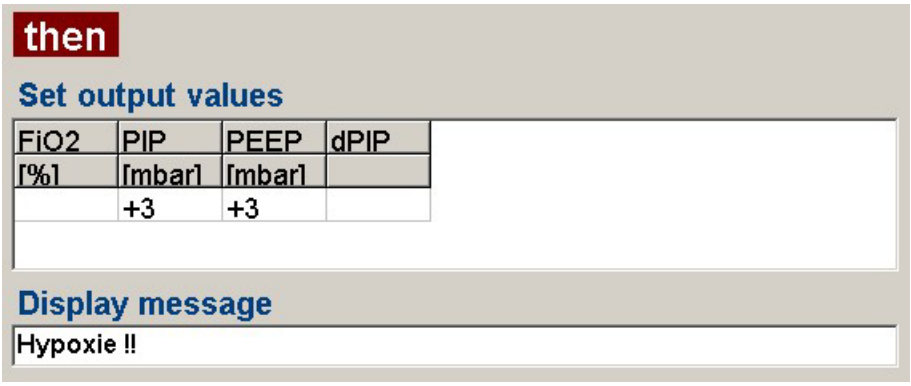


Abb. 35: absolute Stellwertänderung

Relative Stellwertänderungen modifizieren den Wert der betrachteten Variable dagegen um einen prozentuellen Betrag, ausgehend vom momentanen Wert.

Das folgende Beispiel zeigt, wie der Wert der Variable PEEP beim Feuern der Regel um 10% des aktuellen Wertes erniedrigt wird.

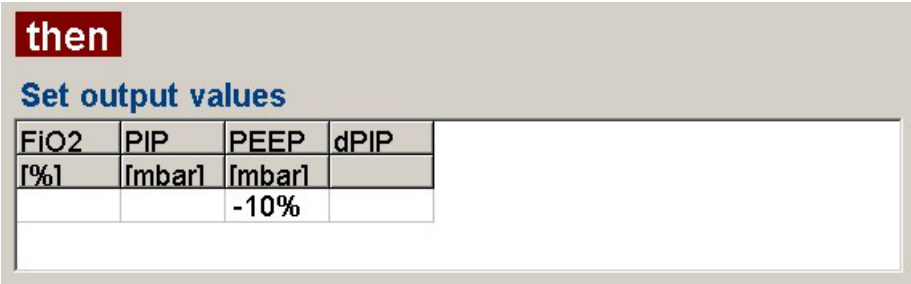


Abb. 36: relative Stellwertänderung

Die relative Stellwertänderung darf nicht mit der Maßeinheit „Prozent (%)“, wie sie bei manchen Parametern vorkommt, verwechselt werden. Beispielsweise wird FiO_2, der Sauerstoffanteil in der Beatmungsluft, in Prozent angegeben. Die folgenden Stellwertänderungen führen dann zu unterschiedlichen Ergebnissen.

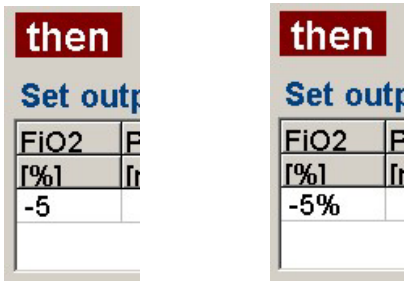


Abb. 37: Stellwertänderungen absolut (a) und relativ (b)

Die Konklusion (a) vermindert FiO2 immer um 5 Einheiten „Prozent“, unabhängig vom gerade aktuellen Wert. Konklusion (b) dagegen subtrahiert 5 Prozent des momentanen Wertes vom Parameter.

Bsp.: $FiO_2(t\text{-}1) = 40$ (a) $FiO_2(t) = FiO_2(t\text{-}1) - 5 = 40 - 5 = 35$
(b) $FiO_2(t) = FiO_2(t\text{-}1) - FiO_2(t\text{-}1) * 0{,}05 = 40 - 2 = 38$

Es sind natürlich noch weitere Methoden denkbar, eine Variablenänderung durchzuführen. Diese können linearer oder nichtlinearer Art sein. Sollte sich während der klinischen Erprobung von FuzzyKBWean die Notwendigkeit ergeben, zusätzliche Methoden zu verwenden, können diese ohne großen Aufwand implementiert werden. Beide Systeme, KBEdit und FuzzyKBWean, beinhalten die erforderlichen Mechanismen, um spätere Erweiterungen aufzunehmen.

Der **Ausgabetext** („verbale Konklusion“) soll die feuernde Regel und ihre Aktionen in kurzer und prägnanter Form erläutern.

Abb. 38: Ausgabetext

Die Anweisungen zur **Blockierung** oder **Deblockierung** können sowohl Regeln als auch Regelgruppen umfassen. Die Blockierung kann für die Dauer eines Zeitintervalls oder bis zur expliziten Deblockierung durch eine andere Regel erfolgen. Blockierte Regeln oder Gruppen können nicht feuern.

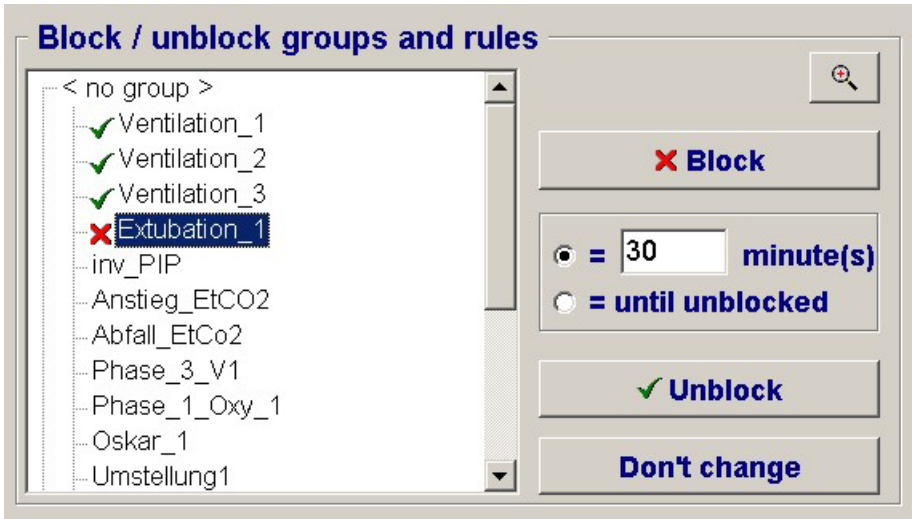


Abb. 39: Blockierung und Deblockierung von Regeln

In diesem Beispiel werden die Regeln „Ventilation_1“ bis „Ventilation_3“ deblockiert und die Regel „Extubation_1“ für 30 Minuten blockiert. Alle anderen Regeln werden nicht beeinflusst.

Die Einstellung des **Arbeitsmodus** kann „Beatmung“ (ventilation), „Entwöhnung“ (weaning) oder „keine Änderung“ (no change) umfassen. Für beide Modi kann ein geeigneter Satz von Regeln definiert werden.

Abb. 40: Einstellung des Arbeitsmodus

8.3.6.3 Erklärung

Eine (optionale) Erklärung zur Regel dient dem besseren Verständnis und macht den Inferenzvorgang dem Anwender transparenter. Das soll zu einer erhöhten Akzeptanz des Systems FuzzyKBWean führen.

Weiters kann für jede Regel eine **Priorität** eingestellt werden. In der zum jetzigen Zeitpunkt verwendeten Version von FuzzyKBWean findet die Priorität allerdings noch keine Verwendung. Sollten die Erfahrungen der klinischen Testphasen eine Priorisierung der Regeln erforderlich machen, sind nur mehr geringfüge Modifikationen an FuzzyKBWean notwendig.

Die Erstellung von Wissensbasen mit KBEdit bietet einige wesentliche Vorteile:

- KBEdit sorgt für eine eindeutige Namensgebung bei Variablen, Fuzzy-Sets und Regeln. Mehrfachvergabe von Namen werden nicht akzeptiert und daraus resultierende Inkonsistenzen vermieden.

- Die Eingabe von häufig benötigten Elementen wie Operatoren, Variablen und Fuzzy-Sets kann durch Klicken mit der Maus getätigt werden. Dadurch geht die Prämissenerstellung rascher vonstatten und Tippfehler werden minimiert.

- KBEdit sorgt für Konsistenz bei der Benennung. Wird beispielsweise eine Variable umbenannt, erfolgt automatisch eine entsprechende Korrektur in allen Regeln.

- KBEdit erzwingt eine gewisse Vorstrukturierung der Regeln. Dadurch können häufige Fehler, wie sie bei einer unbeschränkten, manuellen Eingabe unweigerlich auftreten würden, deutlich verringert werden. Dazu gehört beispielsweise eine Plausibilitätsprüfung. Der Anwender wird gegebenenfalls darauf aufmerksam gemacht, dass einige Elemente nicht an bestimmten Positionen der Prämisse eingetragen werden dürfen. So dürfte ein numerischer Wert nicht in einem Knoten stehen, der seinerseits Unterknoten besitzt.

8.3.7 Kompilieren von Wissensbasen

Unter Kompilieren versteht man die Generierung einer unmittelbar ausführbaren Fassung der Wissensbasis durch Übersetzung des Quelltextes in eine kompakte Form, die von FuzzyKBWean leicht und effizient verarbeitet werden kann (siehe auch S. 42).

Der Kompiliervorgang wird durch Betätigen der Schaltfläche „Kompilieren“ in der obersten Menüleiste von KBEdit gestartet.

Sollte die Wissensbasis syntaktisch noch nicht korrekt sein, erscheint ein Meldungsfenster, in dem die Art der Unkorrektheit und ihre Lokalisation angezeigt werden. Dabei wird unterschieden zwischen

- Warnungen und
- Fehlern

Warnungen deuten auf bestimmte Unvollständigkeiten in der Wissensbasis hin. Dadurch wird ihre Verwendung in FuzzyKBWean aber nicht oder nur wenig beeinträchtigt und der Kompiliervorgang schließt daher erfolgreich.

Bsp.: Regeln, die keine Konklusion besitzen (das kann in manchen Fällen aber erwünscht sein, siehe auch Kap. 8.6.3 Zeitrasterregeln, S. 58).

Fehler dagegen machen einen einwandfreien Programmablauf unmöglich, weswegen die Wissensbasis nicht erfolgreich kompiliert werden kann.

Bsp.: Syntaxfehler, unkorrekte Anzahl von Operanden, ...

Kann die Wissensbasis erfolgreich kompiliert werden – ob mit oder ohne Warnungen –, erscheint ebenfalls ein entsprechendes Meldungsfenster. In diesem Fall wird eine Datei erzeugt, die den selben Namen wie der Quelltext der Wissensbasis trägt, jedoch mit der Dateiextension „.kbe“, und die die ausführbare Wissensbasis beinhaltet.

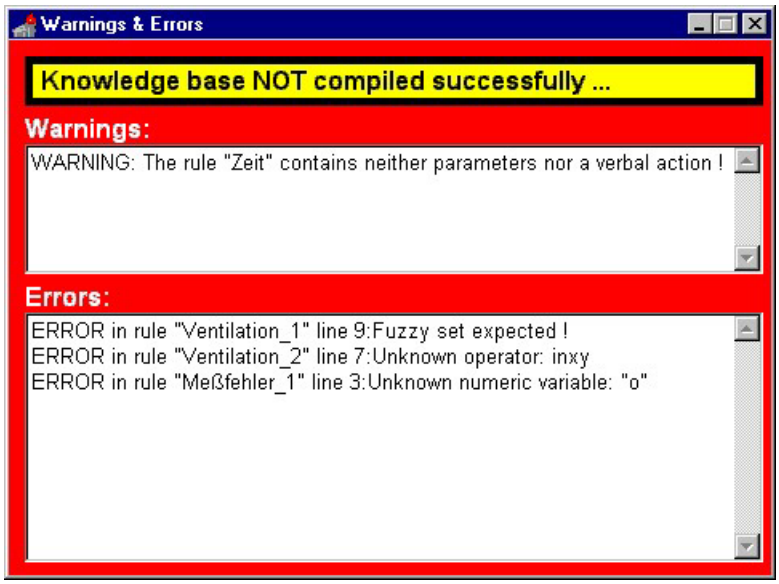


Abb. 41: Fehlgeschlagener Kompilierversuch

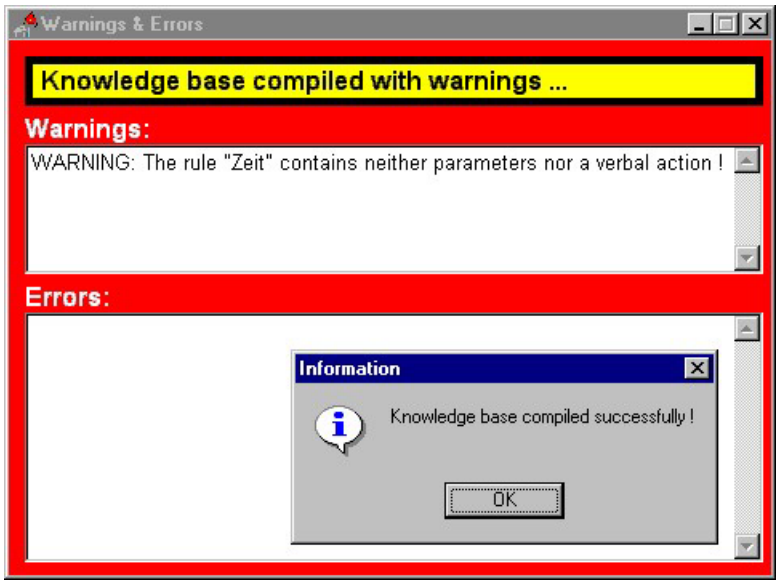


Abb. 42: Erfolgreicher Kompiliervorgang (mit Warnungen)

8.4 Der Befehlssatz von FuzzyKBWean

Die Operatoren von FuzzyKBWean lassen sich in vier Klassen unterteilen:

- Logische Operatoren
- Vergleichsoperatoren
- Arithmetische Operatoren
- Kontrolloperatoren

8.4.1 Logische Operatoren

Diese Operatoren führen Verknüpfungen auf logischen Ausdrücken durch, also auf beliebig verschachtelten Konstrukten, die in ihrer Gesamtheit ein „Boole'sches" Ergebnis (WAHR oder FALSCH) liefern.

8.4.2 Vergleichsoperatoren

Diese Operatoren führen Vergleiche zwischen numerischen oder linguistischen Ausdrücken durch. Als numerischen Ausdruck bezeichnet man einen beliebig verschachtelten Ausdruck, der nach seiner Auswertung eine reelle Zahl ergibt, z. B. 14/2, (x+y)*2, ... oder eine Zahl selbst, z. B. 3,14, 28, ...

Ein linguistischer Ausdruck ist ein Ausdruck, der nach seiner Auswertung eine Zeichenfolge („string") ergibt, z. B. „Hallo, Patient !", „Weaning" + „ forever !"

8.4.3 Arithmetische Operatoren

Arithmetische Operatoren dienen zur Ausführung von arithmetischen und statistischen Berechnungen. Sie liefern eine reelle Zahl als Ergebnis.

8.4.4 Kontrolloperatoren

Kontrolloperatoren liefern Informationen über die Gültigkeit von Variablen und über in der Vergangenheit gefeuerte Regeln. Sie ermöglichen dadurch eine Steuerung des Auswertungsflusses in der Wissensbasis.

Tab. 13: Überblick über die Operatoren von FuzzyKBWean

Klasse	Operator	Beschreibung
Logisch	and	logisches UND (∧)
	or	logisches ODER (∨)
	not	NEGATION (¬)
Vergleich	<	kleiner als
	<=	kleiner oder gleich als
	=	gleich
	>=	größer oder gleich als
	>	größer als
	<>	ungleich
	in	Test ob Ausdruck in Intervall enthalten
	is	"=" für linguistische Ausdrücke
Arithmetisch	+	Addition
	-	Subtraktion
	*	Multiplikation
	/	Division
	ago	Vergangener Wert einer Variable
	mean	Mittelwert
	stdDevAbsolute	Absolute Standardabweichung
	stdDevRelative	Prozentuelle Standardabweichung
	diffAbs	Absolutwert der Differenz zweier Ausdrücke
	diffRel	Prozentuelle Differenz zweier Ausdrücke
Kontrolle	hasFiredRule	Test ob Regel gefeuert hat
	hasNotFiredRule	Test ob Regel nicht gefeuert hat
	hasFiredGroup	Test ob Gruppe gefeuert hat
	hasNotFiredGroup	Test ob Gruppe nicht gefeuert hat
	isValid	Test auf gültigen Wert
	isNotValid	Test auf ungültigen Wert

8.5 Notation der Operatoren

FuzzyKBWean verwendet eine modifizierte Präfix-Notation. Bei dieser Schreibweise steht der Operator an erster Stelle, gefolgt von seinen Operanden. Der Vorteil der Präfix-Notation von FuzzyKBWean besteht darin, dass kompliziertere, tief verschachtelte Ausdrücke leichter lesbar sind, da man ohne die Verwendung von Klammern auskommt. Auch entspricht diese Schreibweise eher der rechnerinternen Arbeitsweise der Inferenzkomponente von FuzzyKBWean, was zu einer besseren Leistung führt.

Man betrachte als Beispiel einen logischen Ausdruck in der uns vertrauten Infix-Notation, bei der der Operator zwischen seinen Operanden steht:

```
((x <= (y - 5)) and (z = 13)) or (((a - b) * 2) = 20)
```

In konventioneller Präfix-Notation würde dieser Ausdruck folgendermaßen aussehen:

```
or and <= x - y 5 = z 13 = * - a b 2 20
```

Eine solche Schreibweise ist nicht leicht überschaubar. In FuzzyKBWean dagegen würde der Ausdruck so lauten:

```
or
├─ and
│  ├─ <=
│  │  ├─ x
│  │  └─ -
│  │     ├─ y
│  │     └─ 5
│  └─ =
│     ├─ z
│     └─ 13
└─ =
   ├─ *
   │  ├─ -
   │  │  ├─ a
   │  │  └─ b
   │  └─ 2
   └─ 20
```

Die FuzzyKBWean-Präfix-Notation verwendet einen baumartigen Aufbau. Der Operator steht immer in einem „Knoten“, seine Operanden um eine Ebene eingerückt in „Unterknoten“. Diese Struktur kann beliebig verschachtelt sein.

8.6 Regeltypen

8.6.1 Beatmungsregeln

Bevor mit der Entwöhnung begonnen werden kann, müssen bestimmte Voraussetzungen erfüllt sein (siehe Kap. 4.1 Voraussetzungen für die Entwöhnung, S. 20). Beatmungsregeln haben die Aufgabe, den Patienten in einen stabilen Zustand überzuführen bzw. zu halten, von dem aus eine Entwöhnung eingeleitet werden kann.

Bsp.:

```
if and
     ├─ in
     │   ├─ pCO2
     │   └─ pCO2\normal
     └─ <
         ├─ VTE
         └─ 0.5
then <erhöhe PIP um 2>
```

8.6.2 Entwöhnungsregeln

Diese Regeln dagegen steuern den konkreten Entwöhnungsvorgang. Sie führen dabei den Patienten aus seinem stabilen Zustand wieder heraus, indem sie die Invasivität der Beatmung schrittweise reduzieren und zur Spontanatmung hinführen.

Bsp.:

```
if and
     ├─ in
     │   ├─ pO2
     │   └─ pO2\normal
     └─ in
         ├─ pCO2
         └─ pCO2\normal
then <erniedrige PIP um 2>
```

Beatmungs- und Entwöhnungsregeln werden bei ihrer Definition mittels der Wissenserwerbskomponente KBEdit dem jeweiligen **Arbeitsmodus** von FuzzyKBWean (Beatmung oder Entwöhnung, S. 37) zugeordnet. Der Arbeitsmodus kann über die Konklusionen geeigneter Regeln eingestellt werden. Es kommen dann nur die dem Modus zugeordneten Regeln zur Anwendung.

8.6.3 Zeitrasterregeln

Manche Regeln benötigen ein Bezugssystem, eine Folge von Punkten auf der Zeitachse in regelmäßigen Abständen. Ein solches Bezugssystem kann durch eine einfache Regel implementiert werden, die folgendermaßen aufgebaut ist:

```
StundeRaster: if hasNotFiredRule
                    ├─ StundeRaster
                    └─ 59
              then
```

Diese Regel ist ein Beispiel für eine fehlende Konklusion. Beim Kompilieren würde daher eine entsprechende Warnung erscheinen, die aber ignoriert werden kann.

Die Aufgabe solcher Regeln ist lediglich das Feuern. Auch wenn keine Konklusion vorhanden ist und daher auch keine Handlungsanweisungen auftreten, wird die Regel aktiviert und ihr Feuern registriert. Über den `hasNotFiredRule`-Operator fragt sie sich daraufhin ständig selbst ab und tritt jeweils nach 60 Minuten wieder in Aktion. Über die Zeit betrachtet entsteht dadurch ein Raster von Referenzpunkten, auf das sich andere Regeln beziehen können.

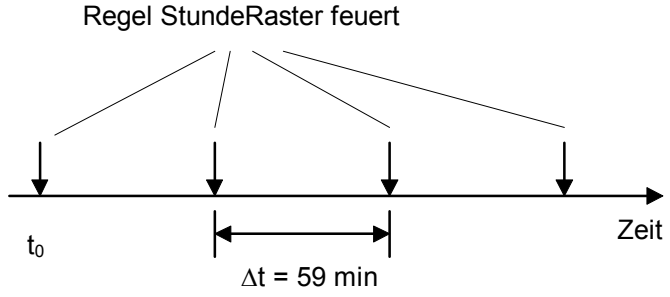


Abb. 43: Funktionsprinzip einer Zeitrasterregel

Durch geeignete Wahl des zweiten Operanden im `hasNotFiredRule`-Operator kann die Granularität des Rasters variiert werden. Dabei ist zu beachten, dass dieser Operand der Länge des gewünschten Rasters *minus eins* entsprechen muss. Andernfalls feuert die Rasterregel immer um eine Minute zu spät. So muss bei einem Stundenraster (60 Minuten) im `hasNotFiredRule`-Operator eine Intervall-Länge von 59 angegeben werden.

Es ist auch möglich, verschiedene Zeitrasterregeln parallel laufen zu lassen und dadurch mehrere Raster mit unterschiedlichen Intervalllängen übereinander zu legen.

Die folgende Regel Vent1 verwendet die Zeitrasterregel StundeRaster als Referenz:

```
Vent1:          if and
                     ├─ hasNotFiredRule
                     │    ├─ StundeRaster
                     │    └─ 58
                     └─ in
                          ├─ EtCO2
                          └─ EtCO2\normal
                then

StundeRaster:   if hasNotFiredRule
                       ├─ StundeRaster
                       └─ 59
                then
```

Besonders wichtig dabei ist, dass in der Regel, die auf eine Zeitrasterregel verweist, im betreffenden hasFired- bzw. hasNotFiredRule-Operator der Zeitoperand wiederum um mindestens 1 niedriger gewählt werden muss als in der Zeitrasterregel. Sonst würde die verweisende Regel entweder immer oder niemals feuern, je nachdem ob diese Regel mittels des `hasFiredRule`- oder des `hasNotFiredRule`-Operators die Zeitrasterregel referenziert.

9. Datenbankkonzept

9.1 Datenquelle: PDMS PICIS®

In einer modernen Intensivstation wird von den Patienten eine Vielzahl von physiologischen Messdaten erfasst. Um diese Datenflut zu bewältigen, werden zunehmend PDMS auf Computeranlagen eingesetzt, die sämtliche Überwachungsdaten speichern. Diese Systeme bieten im Vergleich zu einer manuellen Datenverwaltung mancherlei Vorteile:

- Viele Geräte zur Patientenüberwachung (Pulsoximeter, EKG, ...) verfügen bereits über EDV-Schnittstellen und können ihre Messdaten ohne Zeitverzögerung dem PDMS zur Verfügung stellen. Dadurch ergibt sich ein permanent aktuelles Bild des Patientenzustandes – jedenfalls soweit er über numerische Parameter erfassbar ist.

- Räumlich und zeitlich nicht zusammenhängende Daten können miteinander verknüpft werden.

- Nicht direkt messbare Parameter können berechnet werden – und das schneller und vor allem fehlerfrei, im Gegensatz zu manueller Berechnung. (Bsp.: Flüssigkeitsbilanzen, Kreatininclearance bei der Nierenfunktionsbestimmung, ...)

- Zeitersparnis bei aufwändigen Arbeitsvorgängen (Erstellung und Übertragung von Laborbefunden, ...)

- Ökonomischerer Betrieb.

- Lückenlose Dokumentation, auch aus forensischen Gründen wichtig.

- Qualitätskontrolle und Überwachung des Therapieerfolges werden vereinfacht. Aus der Analyse der genau dokumentierten Krankheits- und Therapieverläufe können möglicherweise neue therapeutische Ansätze abgeleitet werden.

FuzzyKBWean arbeitet auf der Basis von physiologischen Daten des Patienten, die vom PDMS „PICIS®“ (Patient Integrated Clinical Information System, Caresuite 97 Chart +, Paris, Barcelona, 1998) geliefert werden. Die hauptsächliche Datenquelle ist eine PICIS-Datei namens rtData-File. In dieser Datei liegen die benötigten Parameter allerdings in zehnminütigen Abständen vor. Diese Intervalle sind für eine kontinuierliche Steuerung des Entwöhnungsprozesses aber zu lang, da FuzzyKBWean auf plötzliche Änderungen des Patientenzustandes nicht rasch genug reagieren könnte. Um ein effizientes Arbeiten zu ermöglichen, bedarf es einer „sampling rate“ der Daten von einer Minute. In solchen Abständen stehen die Patientendaten nur im sogenannten „Hourbuffer“ von PICIS zur Verfügung. Dieser Speicher ist als Ringbuffer aufgebaut und beinhaltet die Daten der vergangenen 60 Minuten.

9.2 Vorteile eines Datenbankmodells

Die von PICIS gelieferten Rohdaten liegen in einer Form vor, die für eine weitere Verarbeitung wenig geeignet ist. Für eine effiziente Datenverarbeitung ist es vorteilhaft, wenn die Werte in den Tabellen einer Datenbank gespeichert sind.

- Die strukturierte Form der Daten erleichtert den Zugriff und es existieren für diesen Zweck bereits vordefinierte Datenbankfunktionen. Das gilt besonders für Zugriffe auf Werte, die in der Vergangenheit liegen (z. B. „ago“-Operator) oder auf ganze Serien von zeitlich aufeinanderfolgenden Werten, wie sie beispielsweise von den statistischen Funktionen „mean“, „stdDevAbsolute“ und „stdDevRelative“ benötigt werden.

- In der Datenbank sind nur jene Parameter enthalten, die für FuzzyKBWean von Interesse sind. PICIS liefert nämlich eine Fülle von Informationen, aber nur wenige davon werden zur Steuerung der Entwöhnung wirklich benötigt.

- Es erfolgt eine lückenlose Dokumentation des Entwöhnungsprozesses. Das ermöglicht und erleichtert statistische Auswertungen und retrospektive Analysen.

- Mit ein und derselben Datenmenge können mehrere Probeläufe gefahren werden, beispielsweise um das Verhalten von FuzzyKBWean unter Verwendung von verschiedenen Wissensbasen zu testen und zu vergleichen.

- Der Datenaustausch zwischen FuzzyKBWean und der Datenbank erfolgt über genau definierte Schnittstellen, ungeachtet der physikalischen Realisierung der Datenbank. Dadurch ist es nicht notwendig, die den Datenfluss steuernden Komponenten von FuzzyKBWean zu modifizieren, sollte sich die Struktur der Datenbank ändern.

- Das Datenbankmodell ermöglicht eine Kompatibilität zu eventuellen zukünftigen Versionen des PDMS. In einem solchen Fall müsste lediglich die Schnittstelle zwischen PDMS und Datenbank neu implementiert werden. Am Zugriff auf die Daten von FuzzyKBWean aus ändert sich nichts.

- Sollte es erforderlich werden, noch andere physiologische Parameter als die bisher verwendeten für die Steuerung der Entwöhnung heranzuziehen, können diese ohne großen Aufwand nachträglich eingebaut werden. Dazu müssen lediglich die Tabellenstruktur einer bestimmten Tabelle in der Datenbank und die relevanten Regeln in der Wissensbasis ergänzt werden.

Die Datenbank soll als Drehscheibe für den gesamten Datenfluss in FuzzyKBWean dienen.

9.3 Aufbau der Datenbank

Für jede Intensivstation existiert eine Datenbank, die auf einem zentralen Server gespeichert ist. Diese Datenbank enthält zu Beginn fünf Tabellen:

PATIENTS
DAT_TEMPLATE
RUL_TEMPLATE
CON_TEMPLATE
CHG_TEMPLATE

Die Tabellen mit dem Suffix „_TEMPLATE“ dienen als Vorlagen. Sie tragen bereits eine geeignete Struktur, beinhalten aber keine Daten.

9.3.1 Nomenklatur der Tabellen

Wenn ein neuer Patient zur Entwöhnung kommt, wird für ihn ein neuer Datensatz in der Tabelle PATIENTS (siehe Relationenmodell) eingetragen und ein neuer Satz von Tabellen angelegt. Dabei wird von jeder _TEMPLATE-Tabelle eine Kopie erstellt und nach folgendem Schema benannt:

<Präfix>_<BettNr>_<Beginndatum>_<Beginnzeit>

<Präfix> : entspricht den ersten drei Buchstaben des Namens der _TEMPLATE-Tabelle
<BettNr : Nummer des Patientbettes (immer zweistellig)
<Beginndatum> : Datum des Entwöhnungsbeginns (Format: YYYYMMDD)
<Beginnzeit> : Zeit des Entwöhnungsbeginns (Format: HHNN)

Dadurch ist eine eindeutige Benennung und Zuordnung der Tabellen zu den Patienten und Entwöhnungsvorgängen gewährleistet.

Bsp.: DAT_02_20080401_1200
CON_12_20080113_1347

9.3.2 Relationenmodell und Beschreibung der Tabellen

PATIENTS	FAMNAME	varchar(30)	Familienname des Patienten
	NAME	varchar(30)	Vorname
	BIRTHDATE	char(10)	Geburtsdatum (YYYY.MM.DD)
	GENDER	char(1)	Geschlecht (M/W)
	S_DATE	char(10)	Beginndatum der Entwöhnung (YYYY.MM.DD)
	S_TIME	char(5)	Beginnzeit der Entwöhnung (HH:NN), NN = Minute
	BEDNUMBER	short	Nummer des Patientenbettes
	KNOWBASE	varchar(255)	Pfad und Dateiname der zur Entwöhnung verwendeten Wissensbasis
	E_DATE	char(10)	Enddatum der Entwöhnung
	E_TIME	char(5)	Endzeit der Entwöhnung

Diese Tabelle enthält die Stammdaten der Patienten, pro Patient einen Datensatz. Sie existiert in der Datenbank nur einmal.

DAT_BB_YYYYMMDD_HHNN	D_DATE	char(10)	Datum
	D_TIME	char(5)	Zeit
	HR	char(6)	Herzfrequenz
	SPO2	char(6)	Sauerstoffsättigung (pulsoximetrisch)
	ETCO2	char(6)	Endtidales CO_2
	FIO2	char(6)	Inspiratorische O_2-Konzentration
	PEEP	char(6)	Endexspiratorischer Beatmungsdruck (gemess.)
	PIP	char(6)	Inspiratorischer Spitzendruck (gemess.)
	RSP_M	char(6)	Atemfrequenz
	VRATE	char(6)	Beatmungsfrequenz
	I_E	char(6)	Inspirations-Exspirationszeitverhältnis
	TVEX	char(6)	Ausatemvolumen
	PO2A	char(6)	O_2-Partialdruck
	PCO2	char(6)	CO_2-Partialdruck

Diese Tabellen sind die zentralen Elemente der Datenbank. Sie enthalten alle Eingabedaten für FuzzyKBWean (physiologische Messwerte vom Patienten und Respiratordaten), die wiederum in Minutenabständen von PICIS geliefert werden. FuzzyKBWean entnimmt daraus alle Daten, die es zur Steuerung des Entwöhnungsvorganges benötigt. Pro Entwöhnung existiert immer eine DAT-Tabelle in der Datenbank, die mit Bettnummer, Datum und Zeit des Entwöhnungsbeginns einen eindeutigen Namen trägt.

RUL_BB_YYYYMMDD_HHNN	R_DATE	char(10)	Datum
	R_TIME	char(5)	Zeit
	RULE	varchar(120)	Name der gefeuerten Regel

In diesen Tabellen sind die zu einem bestimmten Zeitpunkt gefeuerten Regeln enthalten.

CHG_BB_YYYYMMDD_HHNN	C_DATE	char(10)	Datum
	C_TIME	char(5)	Zeit
	PIP_prp	char(6)	vorgeschlagener PIP
	PIP_eff	char(6)	tatsächlich eingestellter PIP
	:		
	COMMENT	memo	Kommentar

In diesen Tabellen stehen die Ausgabedaten („Changes“) von FuzzyKBWean, die vorgeschlagenen und tatsächlich gesetzten Stellwerte zu einem bestimmten Zeitpunkt. Pro Stellwert existieren im Datensatz immer zwei Felder: Jene mit dem Suffix ‘_prp’ enthalten die von FuzzyKBWean vorgeschlagenen („proposed“) und jene mit dem Suffix ‘_eff’ die vom Anwender tatsächlich gesetzten („effective“) Stellwerte für das Beatmungsgerät. Zusätzlich kann jeder Datensatz mit einem Kommentar versehen werden, beispielsweise wenn der Arzt andere als die vorgeschlagenen Werte einstellt und das begründen möchte.

CON_BB_YYYYMMDD_HHNN	P_RULE	varchar(120)	Name der Regel
	LF_DATE	char(10)	Datum des letzten Feuerns
	LF_TIME	char(5)	Zeit des letzten Feuerns
	CONCLUSION	varchar(255)	Konklusion der Regel
	EXPLANATION	memo	Erklärung zur Regel

Hier sind die Konklusionen und Erklärungen der Regeln enthalten, sowie Datum und Zeit, an denen die Regel das letzte mal gefeuert hat. Letztere Informationen werden vor allem von den „hasFired“- und „hasNotFired“-Operatoren verwendet, die feststellen, ob eine bestimmte Regel oder Gruppe in der Vergangenheit gefeuert hat oder nicht.

9.3.3 EER-Diagramm

Das EER[3]-Diagramm veranschaulicht die Beziehungen der Tabellen einer Datenbank untereinander. Die unterstrichenen Feldnamen bezeichnen den Primärschlüssel der Tabelle. Darunter versteht man eine Untermenge der Felder eines Datensatzes, die den Satz eindeutig identifizieren. In den oben beschriebenen Tabellen ist das oft der Zeitstempel, also Datum und Zeit, zu denen der Datensatz angelegt wurde (beispielsweise in der DAT-Tabelle). Das ist deshalb zulässig, da die Eingabedaten immer in Minutenabständen eintreffen und daher in der selben Minute niemals mehrere Datensätze in der DAT-Tabelle angelegt werden. In diesem Fall besteht der Primärschlüssel aus zwei Feldern, dem Datum und der Zeit.

Ein anderes Beispiel ist der Regelname. Hier genügt dieses eine Feld als Primärschlüssel, da der Regelname per definitionem bereits eindeutig sein muss.

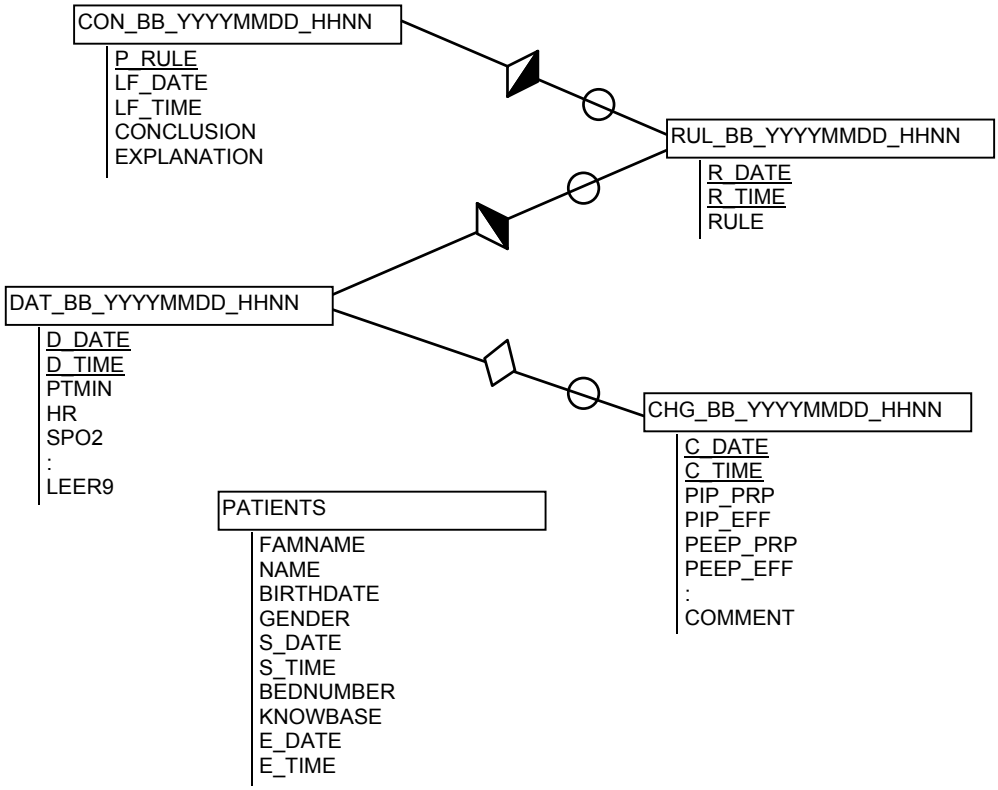


Abb. 44: EER-Diagramm des FuzzyKBWean-Datenbankmodells

In der DAT-Tabelle sind 1 bis n Datensätze enthalten, die durch Datum und Zeit, den sogenannten „Zeitstempel“, eindeutig gekennzeichnet sind. Für jeden dieser Datensätze existieren 0 bis n Sätze in der RUL-Tabelle, d. h. für jeden Eingabedatensatz können mehrere Regeln gefeuert werden. Die Zuordnung erfolgt über einen gleichen Zeitstempel. Ebenso kann mit jedem Eingabedatensatz über einen gleichen Zeitstempel ein (einziger) Ausgabedatensatz in der CHG-Tabelle verknüpft sein, muss aber nicht.

Jede Regel in der CON-Tabelle kann 0 bis n mal gefeuert werden (Eintrag in der RUL-Tabelle).

[3] EER: Abkürzung für „extended entity relationship“

9.4 Implementierung der Datenbank

Die Datenbank von FuzzyKBWean ist nach dem Client-Server-Prinzip organisiert. Sie liegt auf einem zentralen Rechner (Server) und die einzelnen FuzzyKBWean-Instanzen (Clients), die an den Patientenbetten laufen, greifen darauf zu.

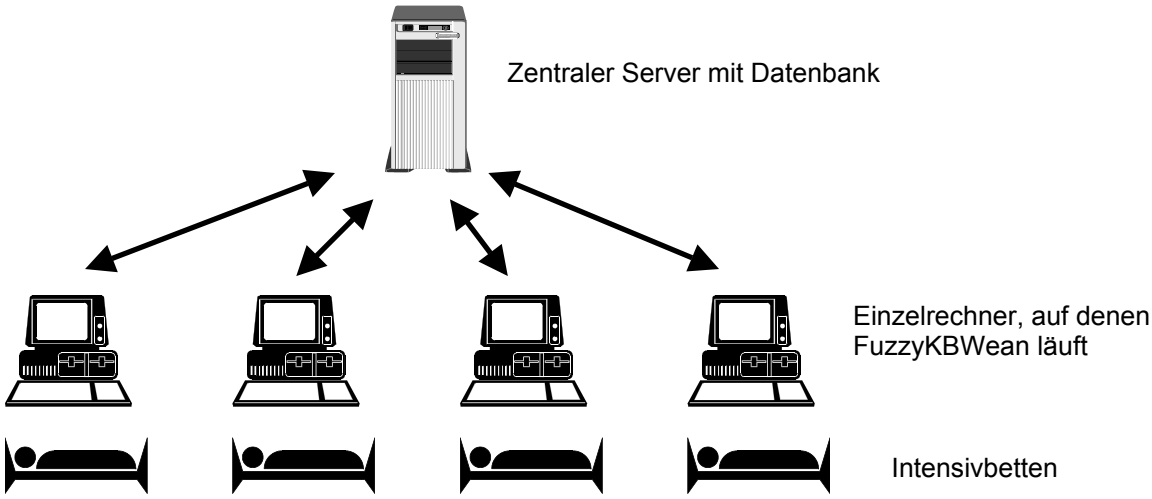


Abb. 45: Funktionsprinzip der Datenbank von FuzzyKBWean (schematisch)

Konkret handelt es sich um eine Datenbank vom Typ InterBase®.

9.5 Arbeitsweise der Datenbank

9.5.1 Beginn eines Entwöhnungsvorganges

Zuerst wird für den Patienten ein neuer Datensatz in der Tabelle PATIENTS angelegt und dort seine Stammdaten sowie Datum und Zeit des Entwöhnungsbeginns und die zur Entwöhnung verwendete Wissensbasis (bzw. ihr Dateiname) eingetragen.

Dann werden von den Mustertabellen DAT_TEMPLATE, RUL_TEMPLATE, CON_TEMPLATE und CHG_TEMPLATE Kopien angelegt und erhalten eindeutige Namen (siehe Kap. 9.3.1 Nomenklatur der Tabellen, S. 62). Wird beispielsweise für den Patienten im Bett Nr. 17 der Station am 28. Februar 2008 um 11:43 Uhr der Entwöhnungsvorgang begonnen, entstehen folgende vier Tabellen:

```
DAT_17_20080228_1143
RUL_17_20080228_1143
CON_17_20080228_1143
CHG_17_20080228_1143
```

Diese werden bis zur Beendigung der Entwöhnung von der entsprechenden FuzzyKBWean-Instanz, welche am Intensivbett Nr. 17 arbeitet, verwendet. Über die Tabelle PATIENTS und die Informationen Bettnummer, Beginndatum und Beginnzeit kann der zugeordnete Patient eindeutig identifiziert werden.

Die Tabelle CON wird mit den Regel-Konklusionen der zur Entwöhnung verwendeten Wissensbasis gefüllt.

9.5.2 Laufender Entwöhnungsvorgang

Während des Entwöhnungsvorganges arbeiten drei Komponenten über die Datenbank zusammen: das PICIS-System, das FuzzyKBWean-Hauptprogramm und die Inferenzkomponente (siehe Kap. 10. Inferenzkomponente, S. 70). Der Datenfluss zwischen diesen sieht folgendermaßen aus:

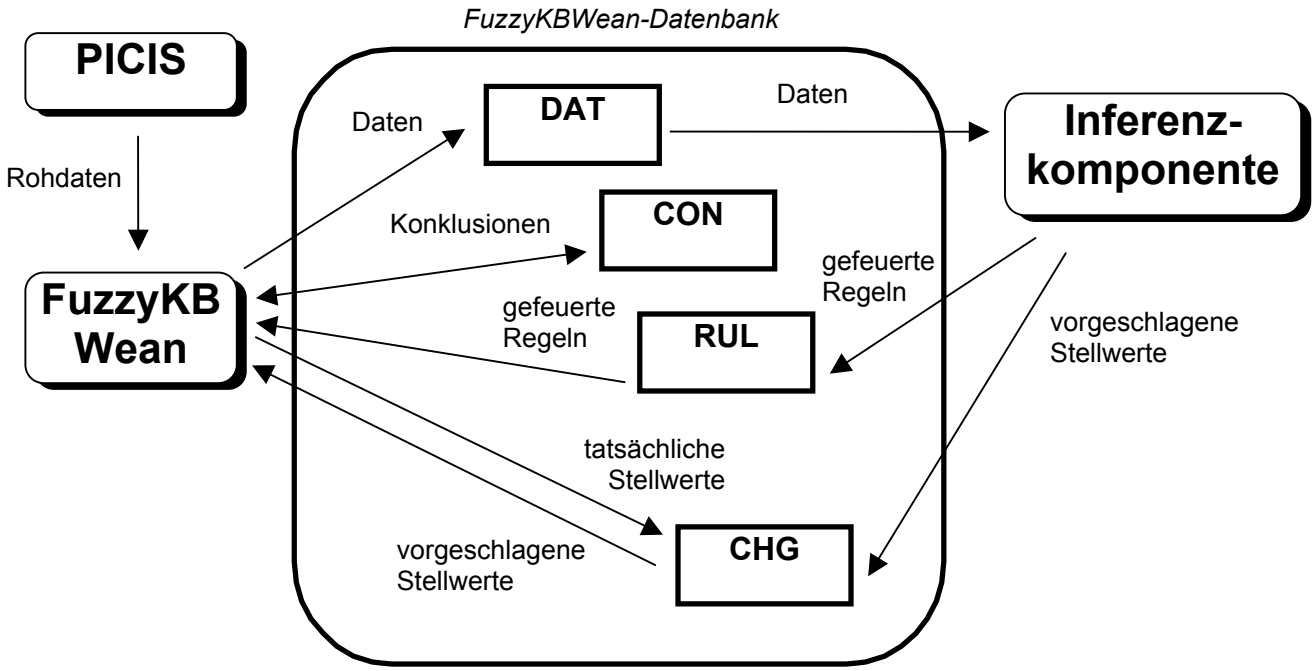


Abb. 46: Datenfluss (schematisiert)

PICIS liefert die Rohdaten. Darin ist eine Vielzahl von Informationen über den Patienten und das Beatmungsgerät enthalten, von denen aber nur relativ wenige zur Steuerung des Entwöhnungsvorganges benötigt werden. Daher extrahiert FuzzyKBWean in Minutenabständen die relevanten Parameter und speichert sie zusammen mit der aktuellen Datums- und Zeitinformation in einem neuen Datensatz in der DAT-Tabelle.

Die Inferenzkomponente von FuzzyKBWean greift auf diese Datensätze zu und wertet sie aus, den Regeln ihrer Wissensbasis folgend. Dabei erzeugt sie zwei Arten von Ausgabedaten: Die auf der Basis der Eingabedaten berechneten neuen Stellwerte für den Respirator und eine Liste jener Regeln aus der Wissensbasis, die für den aktuellen Eingabedatensatz gefeuert haben. Die neuen Stellwerte werden in der CHG-Tabelle gespeichert, die gefeuerten Regeln in der RUL-Tabelle. Für beide wird auch die Datums- und Zeitinformation mit eingetragen, was eine eindeutige Zuordnung zum assoziierten Eingabedatensatz ermöglicht.

FuzzyKBWean liest die vorgeschlagenen Stellwerte und die gefeuerten Regeln aus der CHG- und RUL-Tabelle und zeigt beide am Bildschirm an. Über die gefeuerten Regeln ermittelt es außerdem aus der CON-Tabelle die zugehörigen Konklusionen, welche ebenfalls am Bildschirm ausgegeben werden. Der Arzt kann die vorgeschlagenen Stellwerte akzeptieren oder andere Werte einstellen. Diese veränderten Werte werden zusammen mit den vorgeschlagenen in der CHG-Tabelle gespeichert und können mit einem erklärenden Kommentar versehen werden.

Eine Regel muss nicht notwendigerweise immer Stellwerte vorschlagen. Manche Regeln enthalten in ihren Konklusionen lediglich verbale Meldungen, die eine Information übermitteln oder einen alarmierenden Zustand anzeigen.

Bsp.:

```
if and
     ├─ >=
     │   ├─ -
     │   │   ├─ ago
     │   │   │   ├─ EtCO2
     │   │   │   └─ 30
     │   │   └─ EtCO2
     │   └─ 5
     └─ in
         ├─ EtCO2
         └─ EtCO2\high
then
  "Ventilation Tendenz gut, daher keine Änderung"

if in
     ├─ SpO2
     └─ SpO2\dramatic
then
  "Pulsoximeter neu positionieren !!  Oder doch schwere Hypoxie ?"
```

9.5.3 Beenden des Entwöhnungsvorganges

Wenn die Entwöhnung abgeschlossen ist, werden Datum und Zeit dieses Ereignisses im entsprechenden Datensatz der Tabelle PATIENTS gespeichert. Die dem Entwöhnungsvorgang zugeordneten DAT-, RUL-, CON- und CHG-Tabellen verbleiben in der Datenbank, bis sie explizit gelöscht oder exportiert und separat abgespeichert werden. Auf diese Weise stehen sie für retrospektive statistische Auswertungen und vergleichende Analysen zur Verfügung.

9.6 Leistung der Datenbank (Performance)

Da es sich bei FuzzyKBWean um ein Echtzeitsystem handelt, wenn auch mit einem relativ groben Zeitraster, ist die Leistung der Datenbank entscheidend für die Qualität des gesamten Systems. Es müssen ja bestimmte Antwortzeiten a priori garantiert werden können. Von besonderem Interesse sind dabei die Leistungsmerkmale „Speicherplatzbedarf“ und „Zugriffszeit“ der Datenbank.

Ein Entwöhnungsvorgang kann unterschiedlich viel Zeit in Anspruch nehmen. Aus den Erfahrungen der klinischen Praxis lassen sich Zeiträume von drei Stunden bis etwa eineinhalb Tagen angeben, die bei den folgenden Berechnungen als Grenzwerte dienen sollen. Es ist natürlich nicht ausgeschlossen, dass auch längere oder kürzere Entwöhnungszeiten auftreten können. Für eine zuverlässige Leistungsabschätzung der Datenbank nehmen wir jedoch an, dass bei 90% der Entwöhnungsvorgänge der Zeitbedarf innerhalb der angegebenen Grenzen liegt.

Tab. 14: Zeitbedarf von Entwöhnungen

	Zeitbedarf
t_{min}	3 h
t_{max}	36 h

Sogenannte „schwierige“ Entwöhnungen, bei denen lange kein Durchbruch erzielt wird und öfters Rückschläge auftreten, können sich allerdings bis zu einer Woche (168 Stunden) hinziehen.

9.6.1 Erforderlicher Speicherplatz pro Entwöhnung

Wie in Kap. 9.5.1 Beginn eines Entwöhnungsvorganges beschrieben, werden für jeden Patienten vier Tabellen angelegt und im Lauf der Entwöhnung in unterschiedlichem Ausmaß mit Daten gefüllt.

9.6.1.1 Patientendaten (PATIENTS)

Für jeden Patienten wird in der Tabelle PATIENTS ein Satz mit patientenspezifischen Daten eingetragen. Diese Daten nehmen maximal 358 Bytes pro Patient in Anspruch und sind daher, im Verhältnis zur Größe anderer Tabellen, vernachlässigbar.

9.6.1.2 Konklusionen (CON)

Die Tabelle CON wird nur einmal, zu Beginn der Entwöhnung, gefüllt. Ihr Umfang hängt von der Anzahl der Regeln und der Länge der Konklusionen und Regelerklärungen ab. Da sie im Lauf der Zeit jedoch nicht wächst, trägt auch sie kaum entscheidend zur Größe der Datenbank bei.

9.6.1.3 Physiologische Daten (DAT)

An die DAT-Tabelle wird pro Minute ein Datensatz angefügt. Seine Größe ist von der Anzahl der verwendeten Parameter abhängig und beträgt 10 + 5 + 6 * *Anzahl_der_Parameter* Bytes. In ihrer jetzigen Form (mit 12 Parametern) belegt jeder Satz der DAT-Tabelle somit 87 Bytes. Pro Stunde fallen daher maximal **5220 Bytes** (5,1 KB) an Daten an. Je weniger Parameter für die Entwöhnung benötigt werden, umso kleiner ist auch die über die Zeit erzeugte Datenmenge.

9.6.1.4 Gefeuerte Regeln (RUL)

Ein Datensatz der RUL-Tabelle belegt 10 + 5 + *Länge_des_Regelnamens* Bytes. Wieviele Datensätze pro Minute anfallen, ist sehr von der Aktivität der Regeln abhängig. Als (stark unterschiedliche) Extremfälle sind „keine Regel feuert jemals“ und „in jeder Minute feuern alle Regeln“ theoretisch denkbar. In der Praxis hat sich in mehreren Testläufen über reale physiologische Daten ermitteln lassen, dass pro Minute im Mittel ca. 4 Regeln feuern. Geht man davon aus, dass ein Regelname meist zwischen vier und zwölf Zeichen lang ist, erhält man ein durchschnittliches Datenaufkommen von **5520 Bytes** (5,4 KB) pro Stunde.

9.6.1.5 Stellwerte (CHG)

Ähnliches gilt für die Tabelle CHG, in der die Stellwerte enthalten sind. Auch hier werden nur dann Daten eingetragen, wenn Regeln feuern. Allerdings wird pro Minute nur entweder kein oder ein einziger Datensatz produziert, nämlich dann, wenn mindestens eine Regel feuert, die mindestens einen Stellwert berechnet. Man kann, wiederum auf der Basis von Testläufen, davon ausgehen, dass in der Minute durchschnittlich 0,7 Datensätze eingetragen werden. Ein solcher Datensatz belegt 10 + 5 + 6 * *Anzahl_der_Stellwerte* Bytes. In der vorliegenden Version von FuzzyKBWean werden drei Stellgrößen – PIP, PEEP und FiO_2 – verwendet. Somit fallen pro Minute 33 * 0,7 = 23,1 Bytes an Daten in der CHG-Tabelle an, bzw. **1386 Bytes** (1,4 KB) pro Stunde. Zusätzlich kann jeder Datensatz dieser Tabelle vom Anwender mit einem Kommentar versehen werden.

Für das pro Entwöhnung anfallende Datenvolumen sind hauptsächlich die Tabellen DAT, RUL, CHG verantwortlich. Insgesamt vergrößert sich die Datenbank im Mittel um **12126 Bytes** (11,8 KB) pro Stunde. Für jeden Entwöhnungsvorgang muss daher mit folgendem Speicherplatzbedarf gerechnet werden:

Tab. 15: Speicherbedarf pro Entwöhnung

	Zeitbedarf	Speicherbedarf
t_{min}	3 h	36328 Bytes (35,5 KB)
t_{max}	36 h	436536 Bytes (426,3 KB)

9.6.2 Zugriffszeit

Die Zugriffszeiten auf die Datenbank sind im wesentlichen von der Netzwerklast abhängig. In der praktischen Erprobung unter realen Bedingungen lagen die Zeiten, bis zu denen die angeforderten

Daten zur Verfügung standen, immer um Größenordnungen unter der kritischen Grenze von einer Minute. Die Zugriffszeit hat daher keine signifikante Bedeutung.

9.6.3 Konsequenzen für die Praxis

Bei Verwendung von heute üblicher (oder in Zukunft noch leistungsfähigerer) Technologie dürften im realen Betrieb von FuzzyKBWean wohl kaum Performance-Probleme auftreten, sowohl was Speicherplatzbedarf auf den Festplatten, als auch Rechenleistung, Datenbank-Zugriffszeiten und Übertragungsraten über das Netzwerk betrifft. Dennoch kann die Gesamtleistung auch von seiten der Anwender noch verbessert werden.

Die Datenbank sollte nicht übermäßig groß werden. Das kann verhindert werden, indem man regelmäßig die den abgeschlossenen Entwöhnungsvorgängen assoziierten Tabellen aus der Datenbank exportiert und separat archiviert. Sie stehen auch dann noch für nachträgliche Analysen und Statistiken zur Verfügung.

Zur Größenbegrenzung der Datenbank trägt auch bei, nur die wirklich für die Entwöhnung benötigten physiologischen Parameter in die Wissensbasis, und damit in die DAT-Tabelle, aufzunehmen. Eine solche Untermenge aller möglichen Parameter muss von den Experten bestimmt werden.

Die Regeln sollten nach Möglichkeit kurze Namen tragen, andererseits ist es auch wünschenswert, dass die Namen aussagekräftig und auf den ersten Blick identifizierbar sind. In der Praxis haben sich Regelnamen mit fünf bis zehn, maximal fünfzehn Buchstaben als tauglicher Kompromiss erwiesen.

10. Inferenzkomponente

Die Inferenzkomponente ist das zentrale Element jedes Expertensystems. Ihre Aufgabe ist es, für das gegebene Problem Lösungsvorschläge zu erarbeiten, indem es die Regeln der Wissensbasis auf den durch die Eingabedaten repräsentierten Problemzustand anwendet.

Die Inferenzkomponente von FuzzyKBWean stützt sich auf die Datenbasis, welche den (messbaren) Patientenzustand der Vergangenheit und der Gegenwart beinhaltet.

10.1 Datenbasis

Die Datenbasis kann als $(m \times n)$-Matrix interpretiert werden, bei der die Zeilenvektoren die verschiedenen physiologischen Parameter beinhalten und die Spaltenvektoren die Ausprägungen der Parameter über die Zeit.

$$\begin{pmatrix} FiO_{2t_0} & PIP_{t_0} & PEEP_{t_0} & \cdots & EtCO_{2t_0} \\ FiO_{2t_1} & PIP_{t_1} & PEEP_{t_1} & \cdots & EtCO_{2t_1} \\ \vdots & \vdots & \vdots & & \vdots \\ FiO_{2t_{n-1}} & PIP_{t_{n-1}} & PEEP_{t_{n-1}} & \cdots & EtCO_{2t_{n-1}} \\ FiO_{2t_n} & PIP_{t_n} & PEEP_{t_n} & \cdots & EtCO_{2t_n} \end{pmatrix}$$

Abb. 47: Datenbasis von FuzzyKBWean in Matrixform

Die Datenbasis ist als Tabelle im Datenbanksystem von FuzzyKBWean realisiert. Sie wächst während des Entwöhnungsvorganges dynamisch, da in einminütigen Abständen neue Zeilenvektoren d_i angefügt werden:

$$d_i = (FiO_{2i}, PIP_i, PEEP_i, ..., EtCO_{2i})$$

Die Selektion der Daten aus der Datenbasis erfolgt über Operatoren aus dem Befehlssatz von FuzzyKBWean (siehe S. 55). Die Datenauswahl kann auf verschiedene Arten erfolgen.

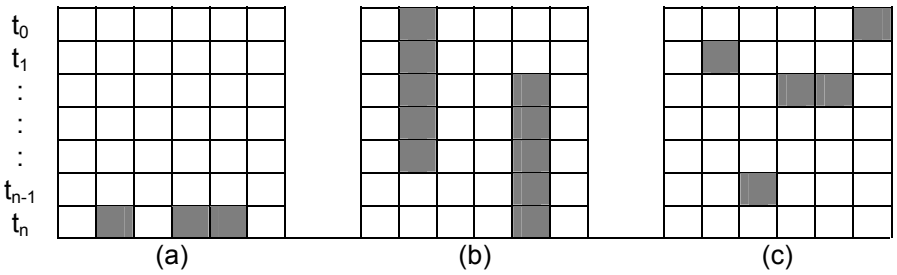


Abb. 48: Auswahl der Daten aus der Datenbasis

(a) Direkte Variablenabfrage nach dem Muster

```
if <
    ├─ SpO2
    └─ 96
```

Das Ergebnis ist der augenblicklich aktuelle Werte der Variable.

(b) Statistikfunktionen (Mittelwert, Standardabweichung, ...)
Diese Funktionen benötigen einen Vektor von Variablenwerten über die Zeit.

```
if >
    ├─ mean
    │    ├─ EtCO2
    │    ├─ 5
    │    └─ 5
    └─ 45
```

(c) Abfrage mittels des ago-Operators

```
if <
    ├─ ago
    │    ├─ SpO2
    │    └─ 5
    └─ 92
```

Die Auswertung der abgefragten Variablen erfolgt nach der Methode „call-by-value“ (siehe auch [Aho1999]), d. h. überall dort, wo im Text der Regel eine Variable, bzw. ein ago- oder statistischer Operator, vorkommt, wird sie bzw. er durch den konkreten numerischen Wert ersetzt, welcher aus der Datenbasis ermittelt wird.

10.2 Syntax von FuzzyKBWean

Die Syntax von FuzzyKBWean kann durch eine kontextfreie Grammatik G in Extended-Backus-Naur-Form (EBNF) beschrieben werden.

G = < V, T, P, S >

V = { RuleCollection, Rule, LExpr, LFaktor, CExpr, Expr, LingExpr, Conclusion, Variable, FuzzySet, RuleName }

T = { "if", "then", "and", "or", "not", "<", "<=", "=", ">=", ">", "<>", "in", "is", "hasFiredRule", "hasNotFiredRule", "hasFiredGroup", "hasNotFiredGroup", "isValid", "isNotValid", "+", "-", "*", "/", "ago", "mean", "stdDevAbsolute", "stdDevRelative", "diffAbs", "diffRel", <string>, <reelleZahl> }

P :

RuleCollection	→	Rule { Rule }
Rule	→	"if" LExpr "then" Conclusion
LExpr	→	("and" | "or") LExpr LExpr { LExpr } | LFaktor
LFaktor	→	"not" LExpr | CExpr
CExpr	→	("<" | "<=" | "=" | ">=" | ">" | "<>") Expr Expr | "in" Expr fuzzySet | "is" LingExpr LingExpr | ("hasFiredRule" | "hasNotFiredRule") RuleName | ("hasFiredGroup" | "hasNotFiredGroup") GroupName | ("isValid" | "isNotValid") Variable
Expr	→	("+" | "–" | "*" | "/") Expr Expr | "ago" Variable Expr | ("mean" | "stdDevAbsolute" | "stdDevRelative") Variable Expr Expr | ("diffAbs" | "diffRel") Expr Expr | Variable | <reelleZahl>
LingExpr	→	"ago" Variable Expr | Variable | <string>
Conclusion	→	<string>
Variable	→	<string>
FuzzySet	→	<string>
RuleName	→	<string>
GroupName	→	<string>

S = RuleCollection

<string> ... Folge von beliebigen alphanumerischen Zeichen
<reelleZahl> ... { x | x ∈ ℜ }

V ... Menge von metalinguistischen Variablen
T ... Menge von Terminalsymbolen
P ... Menge von Produktionsregeln
S ... Startsymbol

10.3 Implementierung der Inferenzkomponente

FuzzyKBWean wurde mittels des Systems Borland Delphi® 6.0 Client Server Suite erstellt und in der Programmiersprache ObjectPascal implementiert. Die Inferenzkomponente ist in zwei Pascal-Units enthalten:

`DKnowBasN.dcu` : enthält Konstanten-, Typ- und Objektdeklarationen
`DInterprN.dcu` : enthält die eigentliche Inferenzkomponente

10.3.1 Wissensbasis

Die maschinenlesbare Version der Wissensbasis ist im Objekt `TKnowBaseN` enthalten:

```
TKnowBaseN = Class(TObject)
    OpMode : TOperationMode;     { Arbeitsmodus (Beatmung, Entwöhnung) }
    OutOfBoundsColor : TColor;
    Vars: TList;                 { Variablen }
    Consts: TList;               { Fuzzy sets }
    Params: TParamList;          { Stellgrößen }
    Groups: TList;               { Regelgruppen }
    Rules: TRuleCollectionN;     { Regeln }
    constructor Create;
    destructor Done; virtual;
    procedure Load(fname : string; var ok : boolean);
    procedure SetOpMode(aOpMode : TOperationMode);
    procedure CommitBlock;
    procedure DecrementBlockTime;
    function IsVariable(var Id: SHORTString): boolean;
end;
```

`SetOpMode` setzt den Arbeitsmodus der Wissensbasis (siehe S. 37).
`CommitBlock` führt die Blockierung von Regeln aus (siehe S. 49ff). Dies erfolgt in einem Durchgang (commit) nach der Abarbeitung aller Regeln.
`DecrementBlockTime` verringert den Blockierungszähler von Regeln bei zeitabhängiger Blockierung.

10.3.1.1 Create und Done

Über die Methoden `Create` und `Done` wird eine Instanz des Objektes erzeugt bzw. vernichtet.

10.3.1.2 Load

Die Methode `Load` sorgt dafür, dass die Wissensbasis aus einer Datei geladen wird. Die Komponenten der Wissensbasis – Variablen, Werte und Regeln – werden dabei in den Listen `Vars`, `Consts`, `Params` und `Rules` gespeichert, und zwar in Form von Objekten, die von der Symbolklasse `TSymbolN` abgeleitet sind:

```
TSymbolN = Class                { Standardsymbol }
    Id       : shortstring;
    classe   : TClassN;
    ArgCount : integer;
    constructor Create(aId : shortstring; aClass: TClassN; aArgCount :
      integer);
end;
```

Diese Klasse und ihre Tochterklassen `TNumberSymbol`, `TVarSymbol` und `TConstSymbol` beinhalten Informationen über das zu speichernde Objekt, wie den Namen (`Id`), das Symbol des

Operators (`classe`) und die Anzahl der Operanden (`ArgCount`). In den von `TSymbolN` abgeleiteten Klassen sind darüber hinaus noch detailliertere Attribute enthalten.

Die Liste Rules besteht aus einer Folge von Symbolen, in denen die atomaren Elemente des Prämissentextes der Regel gespeichert sind. Diese Elemente entsprechen den Terminalen in der Syntaxdefinition und können Operatoren, Variablen oder Werte sein. Sie werden auch als *Token* bezeichnet.

10.3.1.3 IsVariable

Diese Funktion stellt fest, ob die als Argument angebene Variable in der Liste der Variablen enthalten ist oder nicht. Je nachdem liefert die Funktion true oder false.

10.3.2 Inferenzkomponente

Das Objekt TInterpreterN enthält die eigentliche Inferenzkomponente. In der folgenden Deklaration werden nur die wesentlichen Attribute und Methoden des Objekts angeführt. Alle weiteren Bestandteile haben nur unterstützende Funktion und werden daher an dieser Stelle nicht näher erläutert.

```
TInterpreterN = Class
    LogicMode    : TLogicMode;           { Crisp oder Fuzzy }
    InfMode      : TInferenceMode;       { Art der Inferenz }
    DefuzzMode   : TDefuzzificationMode; { Art der Defuzzifizierung }
     :
    KBN          : TKnowBaseN;           { enthält Wissensbasis           }
    DatTable     : TTable;               { Diese vier Attribute ver-      }
    RuleTable    : TTable;               { weisen auf die von FuzzyKBWean }
    ChangesTable : TTable;               { benötigten Datenbank-          }
    ConTable     : TTable;               { tabellen (siehe S. 62)         }
     :
    Data         : TPicisData;           { Daten von PICIS (siehe S. 60) }

    constructor Create(aKBN: TKnowBaseN; aData: TPicisData; aLogicMode : TLogicMode;
         aInfMode : TInferenceMode; aDefuzzMode : TDefuzzificationMode; aSolveMode :
         TSolveMode);
    destructor Done; virtual;
     :
    procedure Run(var aDatTable, aRuleTable, aChangesTable, aConTable : TTable);

    procedure LExpr(var Value: boolean; var Alpha : real);
    procedure LFaktor(var Value: boolean; var Alpha : real);
    procedure CExpr(var Value: boolean; var Alpha : real);
    procedure Expr(var Value: real; var validResult : boolean);
    procedure LingExpr(var Value : shortstring; var validResult : boolean);

    procedure EvaluateIf(aDD, aTT : shortstring; RuleName: shortstring; V1: boolean;
         Alpha : real);
    function EvaluateNot(V: boolean): boolean;
    function EvaluateAnd(V1, V2: boolean): boolean;
    function EvaluateOr(V1, V2: boolean): boolean;
    function FuzzyNot(V1: real): real;
    function FuzzyAnd(V1, V2: real): real;
    function FuzzyOr(V1, V2: real): real;
    function EvaluateLess(V1, V2: real): boolean;
     :
    function EvaluateIn(V1: real; V2: TSymbolN; var Alpha : real): boolean;
    function EvaluateIs(V1: shortstring; V2: TSymbolN): boolean;
    function EvaluateHasFiredGroup(gNr : integer; V1 : real) : boolean;
    function EvaluateHasNotFiredGroup(gNr : integer; V1 : real; opT : TClassN) :
         boolean;
    function EvaluateHasFiredRule(st : shortstring; V1 : real; opT : TClassN) : boolean;
    function EvaluateHasNotFiredRule(st : shortstring; V1 : real) : boolean;
    function EvaluateIsValid(V1 : TSymbolN; vt : TVarType) : boolean;
    function EvaluateIsNotValid(V1 : TSymbolN; vt : TVarType) : boolean;
    function EvaluateAgoNumeric(V1: TSymbolN; V2: real; var validResult: boolean): real;
    function EvaluateAgoLinguistic(V1: TSymbolN; V2: real; var validResult: boolean):
         shortstring;
    function EvaluateMean(V1 : TSymbolN; V2, V3 : real; var validResult: boolean) :
         real;
    function EvaluateStdDevAbsolute(V1 : TSymbolN; V2, V3 : real; var validResult:
         boolean) : real;
```

```
        function EvaluateStdDevRelative(V1 : TSymbolN; V2, V3 : real; var validResult:
             boolean) : real;
        function EvaluateTrend(V1: TVarSymbolN; V2,V3,V4: real): boolean;
        function EvaluatePlus(V1, V2: real): real;
         :
        function EvaluateDiffAbs(V1, V2 : real) : real;
        function EvaluateDiffRel(V1, V2 : real) : real;
        function EvaluatePicisVarNumeric(aSym: TSymbolN; aDeltaTime: real; var V: real):
             boolean;
        function EvaluatePicisVarLinguistic(aSym: TSymbolN; aDeltaTime: real; var V:
             shortstring): boolean;

    private
        StatQuery  : TQuery;
        MeanList   : TList;
        SDAList    : TList;
        SDRList    : TList;
        RulTabName : string;
        DD : string[10];
        TT : string[5];

        function SolveConflict(vList : TList) : real;
        function Min(V1, V2: real): real;
        function Max(V1, V2: real): real;
        function IndexBySQL(l : TList; sq : string) : integer;
        procedure AddTime(oldDate, oldTime : string; back : word; var newDate, newTime :
             string);
        procedure EmptyRead(n : integer);
    end;
```

10.3.3 Initialisierung der Inferenzkomponente

Die Inferenzkomponente wird mittels der Methode `create` erzeugt. Bei ihrem Aufruf müssen sechs Argumente übergeben werden:

- `aKBN`: Ein Objekt vom Typ `TKnowBaseN`. Es enthält die zu verwendende Wissensbasis, welche schon vorher erzeugt worden sein muss.
- `aData`: Ein Objekt vom Typ `TPicisData`, welches die Verbindung zum PDMS PICIS® herstellt. Über `aData` werden die Werte der Variablen aus der Datenbank ermittelt.
- `aLogicMode`: bestimmt die Art der Logik, nach der die Inferenz ablaufen soll. `aLogicMode` kann die Werte `lmFuzzy` für unscharfe und `lmCrisp` für scharfe (zweiwertige) Logik annehmen.
- `aInfMode`: legt die Inferenzmethode fest: `imMaxProd` für MAX-PROD-Inferenz und `imMaxMin` für MAX-MIN-Inferenz (siehe Kap. 6.3.2.2 Inferenz, S. 32)
- `aDefuzzMode`: gibt die Methode zur Defuzzifizierung an. Mögliche Werte sind `dmMaximumHeight`, `dmMeanOfMaximum` und `dmCenterOfGravity`.
- `aSolveMode`: Legt fest, auf welche Art Stellwertkonflikte bei scharfer Logik aufgelöst werden. In der vorliegenden Version von FuzzyKBWean ist nur die Mittelwertbildung als Lösungsmethode implementiert (`smMean`).

10.4 Ablauf des Inferenzprozesses

In Minutenabständen treffen neue Informationen über den Patienten ein, d. h. es ändert sich die Datenbasis. Aufgrund der geänderten Situation wird dann immer ein neuer Inferenzvorgang begonnen und die gesamte Wissensbasis über der Datenbasis abgearbeitet. Die Inferenz startet mittels der Methode `Run`, der als Aufrufparameter die vier mit dem Entwöhnungsvorgang assoziierten Datenbanktabellen übergeben werden. Nun wird jede Regelprämisse Token für Token gelesen.

10.4.1 Die Methode „Run“

```
procedure TInterpreterN.Run(var aDatTable, aRuleTable, aChangesTable, aConTable :
TTable);
  var i, j, c     : Integer;
      :

    procedure InterpretRule(aRule: TRuleN; var hasValues : boolean);
    begin
        :
      EvaluateIf(DD, TT, RuleName, aV, Alpha);
        :
      if aV then begin  { Regel feuert }
          :
        { neue Stellwerte berechnen }
          :
(1)     { ggf. Gruppen und Regeln blockieren bzw. deblockieren }
          :
        { ggf. neuen Arbeitsmodus setzen }
          :
      end;
    end;

begin
  :
  for i := 0 to KBN.Rules.Count - 1 do begin
    WRul := TRuleN(KBN.Rules.Items[i]);

    { Regel wird nur berücksichtigt, wenn sie dem aktuellen Arbeitsmodus zugeordnet }
    { oder „global“ ist                                                              }

    if (WRul.OpMode = CurOpMode) or (WRul.OpMode = omAll) then begin
      if WRul.BlockTime = 0 then begin  { Regel wird nur ausgewertet, wenn sie }
        InterpretRule(WRul, hV);        { nicht blockiert ist (BlockTime = 0)  }
        hasValues := hasValues or hV;
      end;
    end;
  end;
  KBN.CommitBlock;         { Blockierung ausführen }
  KBN.DecrementBlockTime;  { Blockierungszeit dekrementieren, falls > 0 }

  if hasValues then begin  { = es wurden neue Stellwerte berechnet }
    :
    for i := 0 to KBN.Params.Count - 1 do begin
      pSym := KBN.Params.Items[i];

      if (pSym.Denominator <> 0) or (pSym.ValueList.Count > 0) then begin
        if LogicMode = lmFuzzy then begin
          pSym.AddItem(pSym.Numerator / pSym.Denominator); { für Sugeno-Operator }
        end;

        change := SolveConflict(pSym.ValueList);    { nur in Crisp-Logik }

        if Abs(change) < pSym.Threshold then begin  { Wird nur angewendet, wenn sie }
          change := 0;                              { den Schwellwert überschreitet }
        end;

        resultVal := pSym.OldValue + change;
        rVal := FormatFloat(FLOATFORMAT, resultVal);
      end else begin
        rVal := INVALID_V;  { keine gültigen Änderungen }
      end;
      :
    end;
    :
  end;
end;
```

(1) Blockierungszeit
- n > 0 : Blockierung für n Minuten
- n = 0 : Blockierung aufgehoben bzw. nicht blockiert
- n = -1 : Blockierung bis explizit durch andere Regel aufgehoben

10.4.2 Auswertung der Regelprämissen

Die Abarbeitung der Wissensbasis erfolgt gemäß den syntaktischen Regeln, wie sie in der kontextfreien Grammatik von FuzzyKBWean (siehe Kap. 10.2 Syntax von FuzzyKBWean, S. 72) definiert wurden und entspricht, bildhaft betrachtet, einem Baum. Für jede der Produktionsregeln LExpr, LFaktor, CExpr, Expr und LingExpr existiert eine gleichnamige Methode des Objektes `TInterprN`.

Als Beispiel sei die Methode `LFaktor` nun näher beschrieben:

```
procedure TInterpreterN.LFaktor(var Value: boolean; var Alpha : real);
    var aV : boolean;
begin
    case Sym.Classe of
        opNot : begin
                    ReadSym;
                    LExpr(aV, Alpha);
                    Value := EvaluateNot(aV);
                    Alpha := FuzzyNot(Alpha);
                end;
    else
        begin
            CExpr(aV, Alpha);
            Value := aV;
        end;
end;
```

Der Programmtext in ObjectPascal entspricht der Produktionsregel aus der kontextfreien Grammatik - die Alternativen der Produktionsregel werden durch die „`case`“-Anweisung abgebildet [Matt2006]. Im Prozedurkopf sind die beiden Variablenparameter „`Value`“ und „`Alpha`“ definiert, welche `LFaktor` benötigt. Sie werden von oben nach unten und umgekehrt „durchgereicht“, d. h. sie können sowohl von `LFaktor` als auch von den im Ablaufbaum darunterliegenden Prozeduren `LExpr` und `CExpr` verändert werden und gelangen dann als Ergebnis wieder zur aufrufenden Prozedur `LExpr` zurück. `Value` entspricht dem Boole'schen Ergebnis der darunterliegenden Teile der Regelauswertung, entweder WAHR (TRUE) oder FALSCH (FALSE), das der darüberliegenden Prozedur übergeben wird. Die Variable `Alpha` enthält parallel dazu das „fuzzy“ Ergebnis der bisherigen Regelauswertung, also ihren Erfülltheitsgrad als reelle Zahl im Intervall [0,1].

Als Entscheidungsvariable für die case-Anweisung dient der Typ des aktuellen Tokens. Handelt es sich um den Operator „not“, wird der verbleibende Teil der Prämisse in der Prozedur `LExpr` ausgewertet, andernfalls in der Prozedur `CExpr`. Im ersten Fall werden die zurückgegebenen Variablenwerte `Value` und `Alpha`, entsprechend dem not-Operator, negiert, indem sie den Funktionen `EvaluateNot` und `FuzzyNot` übergeben werden.

```
function TInterpreterN.EvaluateNot(V: boolean): boolean;
begin
    EvaluateNot := Not(V);
end;

function TInterpreterN.FuzzyNot(V1: real): real;
begin
    result := (1 - V1);
end;
```

Funktionen, die mit `Evaluate..` beginnen, führen in FuzzyKBWean elementare Operationen, wie logische Verknüpfungen, arithmetische Operationen oder Variablenauswertungen durch.

10.5 Ausgewählte Methoden

Im folgenden sollen einige Methoden der Inferenzkomponente ausführlicher erläutert werden.

10.5.1 Fuzzifizierung

Die Methode `EvaluateIn` überprüft, ob ein gegebener Wert in einer Fuzzy-Menge liegt, liefert in diesem Fall TRUE zurück und berechnet außerdem die Zugehörigkeitsfunktion (`Alpha`). Als Eingabeparameter werden ihr der zu prüfende Wert `V1` und die Fuzzy-Menge `V2` übergeben. Die Funktion nimmt als Verallgemeinerung eine trapezförmige Fuzzy-Menge an, aber auch rechteck- und dreieckförmige Mengen werden korrekt verarbeitet.

```
function TInterpreterN.EvaluateIn(V1: real; V2: TSymbolN; var Alpha : real): boolean;
  var j   : integer;
      TCS : TConstSymbolN;
begin
  Alpha := 0;
  j := StrToInt(V2.Id);
  if j <> -1 then begin
    TCS := TConstSymbolN(KBN.Consts.Items[j]);
    with TCS do begin
(1)   EvaluateIn := ((LoLoClosed and (V1 >= LoLoBound)) or (V1 > LoLoBound) or LoInf) and
                    ((UpUpClosed and (V1 <= UpUpBound)) or (V1 < UpUpBound) or UpInf);
(2)   if ((LoClosed and (V1 >= LoBound)) or (V1 > LoBound) or LoInf) and
       ((UpClosed and (V1 <= UpBound)) or (V1 < UpBound) or UpInf) then begin
        Alpha := 1;
(3)   end else if ((LoLoClosed and (V1 >= LoLoBound)) or (V1 > LoLoBound)) and
       (V1 <= LoBound) then begin
        Alpha := (V1 - LoLoBound) / (LoBound - LoLoBound);
(4)   end else if ((UpUpClosed and (V1 <= UpUpBound)) or (V1 < UpUpBound)) and
       (V1 >= UpBound) then begin
        Alpha := (UpUpBound - V1) / (UpUpBound - UpBound);
      end;
    end;
  end else begin
    Error(RuleName, 'Fuzzy set not declared');
  end;
end;
```

(1): Hier wird der Rückgabewert ermittelt. Er ist TRUE, wenn V1 in der Fuzzy-Menge V2 liegt, sonst FALSE. Dabei werden auch die Intervallgrenzen berücksichtigt, die entweder offen oder geschlossen sein können, und ebenso rampenförmige Fuzzy-Sets.
(2) – (4): An diesen Stellen wird die Zugehörigkeitsfunktion berechnet, und zwar: bei (2) liegt volle Zugehörigkeit vor (Alpha = 1), (3) bezeichnet die ansteigende und (4) die fallende Flanke.

10.5.2 Defuzzifizierung

Zur Defuzzifizierung, also zur Überführung der resultierenden unscharfen Ergebnismenge in einen scharfen numerischen Wert, verwendet FuzzyKBWean eine modifizierte Sugeno-Methode (siehe Kap. 6.3.3.3 Center-of-Gravity (CoG), S. 33):

$$\eta = \frac{\sum_{r=1}^{k} c_i \cdot \mu_{u_r} \cdot u_r}{\sum_{r=1}^{k} c_i \cdot \mu_{u_r}} \quad \text{für } k \geq 2,\ k \in \mathbb{N}$$

10.5.2.1 Vorbereitung der Defuzzifizierung

Die Summen in Zähler und Nenner werden schon während der Auswertung der Regeln aufaddiert, sodass nach der Bearbeitung der Wissensbasis nur mehr die abschließende Division durchgeführt werden muss. Die Summenbildung erfolgt in der Prozedur `InterpretRule`. Dabei werden nach Abarbeiten der Regel alle in ihrer Konklusion enthaltenen Output-Variablen (z. B. Stellgrößen des Beatmungsgerätes, ...) durchlaufen. Das Objekt `pIt` enthält die aktuelle Output-Variable.

```
.
.
.
if pSym.OldValueValid then begin
    case pIt.ChangeMode of
        cmAbsolute : newValue := pIt.Value;
        cmPercent  : newValue := (pSym.OldValue / 100) * pIt.Value;
    end;
    case LogicMode of
        lmCrisp : pSym.AddItem(newValue);
        lmFuzzy : KBN.Params.AddTo(pIndex, newValue * Alpha, Alpha);
    end;
end;
.
.
.
```

Es werden nur gültige Werte (`OldValueValid = true`) in die Berechnung aufgenommen. Ungültige Werte können beispielsweise durch Messfehler verursacht worden sein und sind in der Datenbanktabelle durch den Eintrag „*“ gekennzeichnet.

In der ersten der beiden `case`-Anweisungen wird, abhängig vom Attribut `ChangeMode`, die Stellwertänderung `newValue` entweder absolut (`cmAbsolute`) oder prozentuell (`cmPercent`) ermittelt. Im ersten Fall wird der aus der Regelauswertung gewonnene Stellwert `pIt.Value` unverändert übernommen, andernfalls als Prozentanteil des alten, zum Zeitpunkt t_{i-1} vorliegenden Wertes `pSym.OldValue` berechnet.

Die zweite `case`-Anweisung bestimmt, auf welche Art die so ermittelten Werte zwischengespeichert werden. Arbeitet FuzzyKBWean mit unscharfer Logik (`LogicMode = lmFuzzy`), dann wird mittels der Methode `AddTo` des Objektes `TKnowBasN` für die betrachtete Output-Variable eine Aufsummierung gemäß der Center-of-Gravity-Formel durchgeführt. Bei scharfer Logik (`LogicMode = lmCrisp`) ist das nicht erforderlich, und daher wird der berechnete Stellwert über die Methode `AddItem` unverändert an die Ergebnisliste der Output-Variable angefügt.

10.5.2.2 Ausführung

Nach Durchlauf aller Regeln schließt die Defuzzifizierung mit

```
.
.
.
if LogicMode = lmFuzzy then begin
    if pSym.k > 1 then begin
        pSym.AddItem(pSym.Numerator / pSym.Denominator);
    end else begin
        pSym.AddItem(pSym.Numerator);
    end;
end;

change := SolveConflict(pSym.ValueList);
resultVal := pSym.OldValue + change;
.
.
.
```

Die abschließende Division wird nur dann durchgeführt, wenn die Anzahl der Summanden (`pSym.k`) größer als 1 ist, was seinen Grund in einer bestimmten Eigenschaft der CoG-Methode hat (siehe Kap. 6.3.3.3 Center-of-Gravity (CoG), S. 33).

Wenn mit Fuzzy-Logik gearbeitet wird, ist die Ergebnisliste `pSym.ValueList` zu diesem Zeitpunkt noch leer, da nur bei scharfer Logik schon vorher Elemente angefügt wurden. Daher kann nun das Ergebnis der Center-of-Gravity-Methode, der Quotient aus schon vorher aufsummiertem Zähler und Nenner, als nunmehr einziges Element in dieser Liste plaziert werden. Bei scharfer Logik bleibt die Ergebnisliste unverändert.

10.5.2.3 Stellwertkonflikte

Das endgültige Ergebnis, die Stellwertänderung `change`, ist das Resultat der Funktion `SolveConflict`, welche bei Verwendung von scharfer Logik Stellwertkonflikte zwischen Regeln auflösen soll. Beispielsweise könnten zwei Regeln `rule1` und `rule2` durchaus gleichzeitig feuern, aber unterschiedliche Stellwerte für dieselbe Output-Variable berechnen. Ein solcher Widerspruch könnte auf Inkonsistenzen in der Wissensbasis hindeuten, oder aber Ausdruck eines der Aufgabenstellung inhärenten Problems sein. Das gilt aber nur für die scharfe Logik, da unterschiedliche Berechnungsergebnisse bei Fuzzy-Logik durchaus erwartet werden und in die Defuzzifizierung einfließen. Wie auch immer, bei Verwendung von scharfer Logik müssen eventuell vorkommende Stellwertkonflikte aufgelöst werden, und das geschieht mittels der Funktion `SolveConflict`. Sie bildet aus den einander möglicherweise widersprechenden Elementen der Ergebnisliste `pSym.ValueList` das arithmetische Mittel und liefert es als Ergebnis zurück. Bei Fuzzy-Logik bleibt dieser Berechnungsschritt ohne Auswirkungen, da in diesem Fall, wie oben erwähnt, die Ergebnisliste ja nur aus einem einzigen Element besteht.

Es sind natürlich noch andere Arten denkbar, um Stellwertkonflikte aufzulösen. In der vorliegenden Version von FuzzyKBWean ist bislang nur die arithmetische Mittelwertbildung implementiert. Sollten sich andere Berechnungsmethoden als zielführender erweisen, können sie ohne großen Aufwand ergänzt werden.

10.5.3 Auswertung von Variablen

Die Auswertung von Variablen, also Ermittlung ihrer Werte aus der Datenbank, wird nicht unmittelbar von der Inferenzkomponente übernommen. Sie ruft vielmehr eine Methode des Objektes `Data` (vom Typ `TPicisData`) namens `GetValueBefore` auf (*).

```
function TInterpreterN.EvaluatePicisVarNumeric(aSym: TSymbolN;
           aDeltaTime: real; var V: real): boolean;
    var j   : integer;
        aId : shortstring;
begin
    j := StrToInt(aSym.Id);
    case aSym.classe of
        tyInputVar  : aId := TVarSymbolN(KBN.Vars.Items[j]).Id;
        tyOutputVar : aId := TParamSymbolN(KBN.Params.Items[j]).Id;
    end;
(*) result := Data.GetValueBefore(aId, aDeltaTime, V);
end;
```

Dieser Methode wird der Name der Variable und der „offset“ übergeben. Der offset gibt an, um wieviele Minuten der gesuchte Wert in der Vergangenheit liegt. Ein offset von „0“ liefert den gerade aktuellen Wert der Variable. Der Rückgabewert von `GetValueBefore` gibt an, ob die Variable erfolgreich ausgewertet werden konnte (TRUE) oder nicht (FALSE).

Bevor die Methode `GetValueBefore` aufgerufen werden kann, muss erst der Name der zu übergebenden Variable ermittelt werden. Im Token der Variable, wie es im Regeltext enthalten ist, steht nämlich nicht der Variablenname selbst, sondern lediglich eine Indexzahl (siehe Kap. 10.6.1 Indizes statt Variablennamen, S. 82)

Eine Methode namens `EvaluatePicisVarLinguistic` ist ähnlich wie `EvaluatePicisVarNumeric` aufgebaut, sie dient aber, im Gegensatz zu jener, nicht zur Auswertung von numerischen, sondern von linguistischen Variablen. Ihre „Partnermethode“ im Objekt `Data` heißt `GetStringBefore`.

Diese Art der Variablenauswertung wird auch „call-by-value“ genannt, d. h. jedes Vorkommnis eines Variablennamens im Regeltext wird unmittelbar durch ihre konkrete Belegung ersetzt [Aho1999].

Die Methoden `EvaluatePicisVarNumeric` und `EvaluatePicisVarLinguistic` werden indirekt auch noch für einen anderen Zweck eingesetzt, nämlich um festzustellen, ob die übergebene Variable überhaupt mit einem gültigen Wert belegt ist. Diese Information benötigen die Operatoren `isValid` und `isNotValid`.

```
function TInterpreterN.EvaluateIsValid(V1 : TSymbolN; vt : TVarType) :
  boolean;
    var d1 : real;
        d2 : shortstring;
begin
    case vt of
        vtNumeric    : EvaluateIsValid :=
                            EvaluatePicisVarNumeric(V1, 0, d1);
        vtLinguistic : EvaluateIsValid :=
                            EvaluatePicisVarLinguistic(V1, 0, d2);
    end;
end;
```

Abhängig vom Typ `vt` der übergebenen Variable ruft diese Methode ihrerseits die Methoden `EvaluatePicisVarNumeric` und `EvaluatePicisVarLinguistic` auf. Der ermittelte Wert der Variable wird nicht weiter verwendet, zurückgegeben wird lediglich die Information, ob die Auswertung erfolgreich war (TRUE) oder fehlgeschlagen ist (FALSE).

10.5.4 Berechnung von Stellwerten

```
procedure InterpretRule(aRule: TRuleN; var hasValues : boolean);
  var i, j, k, pIndex    : integer;
      :
  begin
    :
    EvaluateIf(DD, TT, RuleName, aV, Alpha);
    if (Sym.Classe = tyString) then begin
      if aV then begin                              { Regel feuert              }
        for i := 0 to aRule.Prs.Count - 1 do begin { Neue Stellwerte berechnen }
          :
          if pSym.OldValueValid then begin
(1)         case pIt.ChangeMode of
              cmAbsolute : newValue := pIt.Value;                          { absolut     }
              cmPercent  : newValue := (pSym.OldValue / 100) * pIt.Value; { prozentuell }
            end;
            case LogicMode of
              lmCrisp : pSym.AddItem(newValue);                            { Crisp Logik }
              lmFuzzy : KBN.Params.AddTo(pIndex, newValue * Alpha, Alpha); { Fuzzy Logik }
            end;
          end;
        end;
        :
      end;
      :
    end;
  end;
```

(1) In Abhängigkeit des Attributs `ChangeMode` des Parametersymbols `pIt` wird der neue Stellwert durch einfache Übernahme des berechneten Wertes entweder „absolut“ gesetzt, oder „prozentuell“ durch Berechnung eines Prozentanteiles aus dem vorhergehenden Wert.

10.6 Leistungsoptimierungen

Computer heutiger – und erst recht zukünftiger – Bauart verfügen über ausreichende Speicher- und Rechenleistung, um die bei der Entwöhnung anfallenden Datenmengen zu bewältigen. Ein zufriedenstellendes Laufzeitverhalten von FuzzyKBWean ist daher mit den auf der Intensivstation eingesetzten PC's grundsätzlich gewährleistet. Dennoch – FuzzyKBWean ist ein Echtzeit-Expertensystem und muss somit vorgegebene Zeitlimits per definitionem einhalten können. Auch programmiertechnisch bestehen einige Möglichkeiten, um im Inferenzalgorithmus bestehende Redundanzen abzubauen und die Leistung des Systems insgesamt zu verbessern. Einige wesentliche Maßnahmen sollen nun beschrieben werden.

10.6.1 Indizes statt Variablennamen

Variablennamen können in FuzzyKBWean bis zu 120 Zeichen lang sein. Wird eine Variable, die einen langen Namen trägt, im Regeltext oft verwendet, bläht das die Länge der ausführbaren (executable) Wissensbasis unnötig auf. Auch benötigt das Suchen von Variablen in der Variablenliste der Wissensbasis mehr Zeit, da längere Zeichenketten verglichen werden müssen. Aus diesen Gründen enthält die ausführbare Wissensbasis nicht die eigentlichen Variablennamen, sondern einen Listenindex, repräsentiert durch eine Ganzzahl (integer).

Während des Kompiliervorganges werden zunächst zwei Listen von Variablen angelegt, eine für Input- und eine für Output-Variablen. Danach werden die Regeln kompiliert und dabei jede Variable durch den Index, den sie in der jeweiligen Liste innehat, ersetzt.

Bsp.:

Variablenliste:

Index	Variable
0	SpO2
1	PEEP
2	FiO2
3	EtCO2

Vor Kompilierung:

```
and
 ├─ >
 │   ├─FiO2
 │   └─30
 └─ >
     ├─SpO2
     └─96
```

Nach Kompilierung:

```
and
 ├─ >
 │   ├─2
 │   └─30
 └─ >
     ├─0
     └─96
```

Während der Laufzeit von FuzzyKBWean ermitteln die Methoden, die Variablen auswerten, – `EvaluatePicisVarNumeric` und `EvaluatePicisVarLinguistic` – den Variablennamen über den Index aus der Liste, bevor sie ihn an die Methoden `GetValueBefore` und `GetStringBefore` übergeben:

```
function TInterpreterN.EvaluatePicisVarNumeric(aSym: TSymbolN;
          aDeltaTime: real; var V: real): boolean;
    var j   : integer;
        aId : shortstring;
begin
    j := StrToInt(aSym.Id);
    case aSym.classe of
        tyInputVar  : aId := TVarSymbolN(KBN.Vars.Items[j]).Id;
        tyOutputVar : aId := TParamSymbolN(KBN.Params.Items[j]).Id;
    end;
    result := Data.GetValueBefore(aId, aDeltaTime, V);
end;
```

10.6.2 Unvollständige Auswertung von logischen Ausdrücken

Man betrachte den folgenden logischen Ausdruck:

$$(a = 1) \wedge (b = 7) \wedge (c = 0) \wedge (d = 12) \wedge (e = 13)$$

Wenn a nicht gleich 1 ist, evaluiert der erste Term des Ausdrucks zu FALSCH. Damit ist, bei einer logischen UND-Verknüpfung aller Terme, aber auch bereits der gesamte Ausdruck FALSCH. Auf eine Auswertung der vier restlichen Terme kann verzichtet werden, ohne dass ein unkorrektes Gesamtergebnis entsteht.

Ähnlich verhält es sich bei ODER-verknüpften Termen. Hierbei ist der gesamte Ausdruck schon dann logisch WAHR, wenn mindestens ein Term WAHR ergibt.

$$(a = 1) \vee (b = 7) \vee (c = 0) \vee (d = 12) \vee (e = 13)$$

Wenn hier a gleich 1 ist, trifft auch die Gesamtaussage zu, wiederum brauchen die restlichen Terme nicht ausgewertet werden, wenn nur das Gesamtergebnis von Interesse ist.

FuzzyKBWean macht sich diese Eigenschaften von logischen Aussagen zunutze, indem es die Auswertung abbricht, sobald ein Term als fuzzy WAHR bzw. FALSCH erkannt wird, und nimmt das bisherige Ergebnis als Gesamtergebnis an.

Tab. 16: Auswertung logischer Verknüpfungen

		Terme verknüpft mit	
		$\wedge$ (UND)	$\vee$ (ODER)
Betrachteter einzelner Term	FALSCH	abbrechen	fortsetzen
	$0 < \alpha < 1$	fortsetzen	fortsetzen
	$\alpha = 1$	fortsetzen	abbrechen

Auf diese Weise kann die Verarbeitungszeit von umfangreichen logischen Ausdrücken signifikant verkürzt werden.

Diese Methodik funktioniert auch bei Fuzzy-Logik. Ersetzt man im ersten Beispielausdruck die Terme durch Zugehörigkeitswerte „α (alpha)“ und nimmt wieder an, dass bereits der erste Term FALSCH ist ($\alpha = 0$), ergibt sich folgendes Bild:

$$0 \wedge 0{,}2 \wedge 0{,}73 \wedge 1 \wedge 0{,}8$$

Sowohl mit Minimum- als auch Produktbildung bei der Auswertung des Fuzzy-UND kommt es zu einem Gesamtergebnis von 0, was FALSCH entspricht.

Für Fuzzy-ODER gilt aber eine Einschränkung. Hier ist es nicht ausreichend, wenn ein Term WAHR ist ($\alpha > 0$), da es ohne weiteres noch Folgeterme geben kann, die „WAHRER“ sind, also deren Zugehörigkeitswert α noch größer als der vorherige ist. Die Abbruchbedingung lautet für Fuzzy-ODER daher nicht bloß Term = WAHR, sondern $\alpha = 1$.

Die unvollständige Auswertung von UND- bzw. ODER-Verknüpfungen, ermöglicht eine Leistungsoptimierung: Durch geeignetes Anordnen der einzelnen Terme in der Prämisse müssen zeitaufwändige Operationen, wie z. B. Statistikfunktionen, oft gar nicht erst ausgewertet werden.

Bsp.:
```
if and
     ├─ <
     │   ├─ SpO2
     │   └─ 95
     └─ <
         ├─ mean
         │   ├─ SpO2
         │   ├─ 1
         │   └─ 5
         └─ 95
then
```

Wenn der aktuelle Wert von `SpO2` unter 95 liegt, wird die Auswertung der AND-Verknüpfung abgebrochen und der zweite Term nicht mehr berechnet.

Daher sollten Terme, deren Auswertung wenig Zeit beansprucht, möglichst weit oben in der Prämisse angeordnet werden.

10.6.3 Zwischenspeicherung von Datenbankabfragen

Die statistischen Operatoren `mean`, `stdDevAbsolute` und `stdDevRelative` benötigen eine ganze Folge von Einzeldaten, welche sie über sogenannte Abfragen an die Datenbank zur Verfügung gestellt bekommen. Diese Abfragen benötigen aber vergleichsweise viel Zeit, weswegen sie nur dann eingesetzt werden sollen, wenn das Ergebnis nicht auf anderem Weg zu erhalten ist.

In großen Wissensbasen kann es vorkommen, dass identische Statistikoperationen in unterschiedlichen Regeln mehrfach vorhanden sind. In solchen Fällen würde es genügen, das Ergebnis der ersten Abfrage evident zu halten – zu „cachen“ – und bei allen gleichlautenden Abfragen einzusetzen, anstatt es von neuem aus der Datenbank zu ermitteln und wiederholt zu berechnen. Daher werden alle Datenbankabfragen im Rahmen von Statistikoperatoren in Listen zwischengespeichert. Bei jeder neuen solchen Operation wird zuerst in der Liste gesucht, ob das Ergebnis dort bereits vorhanden ist. Erst wenn das nicht der Fall ist, wird eine neue Abfrage an die Datenbank initiiert. Diese Listen werden zu Beginn jedes Auswertungszyklus der Wissensbasis wieder gelöscht.

11. Bisherige Ergebnisse

Im ersten Schritt wurde FuzzyKBWean „in vitro“ getestet. Als Testumgebung dienten hierbei durch reale Entwöhnungsvorgänge gewonnene Datenmengen und speziell entworfene Datenbasen, um das Regelungsverhalten von FuzzyKBWean in bestimmten Situationen zu erforschen.

Als nächster Schritt erfolgten Testläufe an einer Intensivstation des Wiener Allgemeinen Krankenhauses. [Schu2004] beschreibt eine prospektive randomisierte Studie an 28 Patienten während ihrer postoperativen Behandlung. Dabei zeigte sich, dass das behandelnde medizinische Personal mit größerer zeitlicher Verzögerung auf Hyper- bzw. Hypoventilation reagierte als das Programm.

Tab. 17: Verzögerung der Reaktion auf Hyperventilation durch behandelndes Personal (korrigiert nach [Schu2004])

Patient	Episode	Vorschlag (Zeit)	Ausführung (Zeit)	Verzögerung (min)
X	1	16:30	19:42	192
	2	23:50	01:55	125
	3	04:27	07:43	196
B	1	21:17	03:58	401
D	1	22:57	23:16	19
E	1	22:30	01:15	165
	2	01:15	01:50	35
K	1	13:45	14:45	60
	2	20:20	20:48	28
C	1	17:03	18:27	84
	2	08:48	16:36	468
G	1	20:02	20:12	10
	2	20:12	20:30	18
	3	20:30	22:47	137
	4	22:47	23:22	35
L	1	21:05	21:40	35

Im Mittel betrug die Verzögerung der Reaktion im Fall von Hyperventilation **126 Minuten**, im Fall von Hypoventilation **50 Minuten**.

11.1 Wissenserwerbskomponente

Das Wissenserwerbssystem KBEdit hat sich bewährt. Die Erstellung und Adaptierung von Wissensbasen kann damit rascher und einfacher bewerkstelligt werden. Es wurden einige Erweiterungen vorgenommen wie zum Beispiel die Einführung von Regelgruppen (S. 47), der Blockierungsmechanismus von Regeln und die Einstellmöglichkeit des Arbeitsmodus (S. 37) über Regeln.

11.2 Datenbankmodell

Ähnliches gilt für das Datenbankmodell. Es ermöglicht auf einfache Weise Datenzugriffe in die Vergangenheit, statistische Operationen, graphische Auswertungen in Echtzeit, retrospektive Analysen und vieles mehr.

11.3 Zeitverhalten

Für einen erfolgreichen Einsatz von FuzzyKBWean ist ein Datenakquisitionsintervall, ein „sampling interval“, von längstens einer Minute unbedingt erforderlich. Nur so kann das System auf Änderungen des Patientenzustandes rasch genug reagieren. Mit einer geringeren Sampling-Rate, wie zum Beispiel alle zehn Minuten, entstehen im Datenverlauf zu große Sprünge, was zu einem oszillierenden Regelungsverhalten und zu überschießenden Sprungantworten führt. Solche abrupten Stellwertänderungen sind für den Entwöhnungsverlauf nicht günstig, da sie den Organismus des Patienten unnötig belasten. Vielmehr sollte ein „sanftes“ und damit schonenderes Verlaufsprofil angestrebt werden, was nur durch eine rasche Anpassung der Stellgrößen an den Patientenzustand und folglich durch eine relativ hohe Sampling-Rate erreicht werden kann.

11.4 Vergleich von Crisp- und Fuzzy-Logik

Die Stärke von Fuzzy-Logik und Fuzzy-Control liegt in ihrem adaptiven Regelverhalten. Im Vergleich zu Crisp-Logik treten bei Fuzzy-Logik weniger abrupte Sprungantworten auf, wodurch man sich dem angestrebten sanften Verlaufsprofil des Entwöhnungsvorganges eher annähert. Die von FuzzyKBWean unter Testbedingungen berechneten Stellgrößen waren bei Verwendung von Fuzzy-Logik auch tatsächlich besser an die Eingabedaten angepasst als unter Crisp-Logik. Ob dieses Regelungsverhalten auch unter klinischen Bedingungen signifikant positive Auswirkungen auf den Patientzustand haben wird, bleibt noch zu untersuchen.

11.5 Probleme

Einige Probleme stehen noch zur Lösung an, bevor die Realisierung eines closed-loop-Systems zur Entwöhnung von Patienten in Betracht gezogen werden kann. Teilweise entsprechen diese den Erfahrungen, die mit anderen Systemen gemacht wurden (siehe auch S. 39).

11.5.1 Qualität der Eingabedaten

Die vom PDMS PICIS® übernommenen Daten weisen zum Teil noch ungenügende Qualität auf. Es bedarf daher zusätzlicher Mechanismen zur Fehler- bzw. Artefakterkennung.

11.5.2 Unvollständige Eingabedaten

So ist es in der vorliegenden Version von PICIS® noch nicht möglich, Informationen über bestimmte pflegerische Tätigkeiten am Patienten abzurufen. Diese Tätigkeiten, wie zum Beispiel das Absaugen von Sekret aus den Atemwegen, verursachen aber Änderungen an Blutgaswerten, Sauerstoffsättigung, etc. des Patienten, was eine Reaktion von seiten des Systems FuzzyKBWean bewirken würde – obwohl das in solchen Fällen gar nicht erwünscht ist, da es sich bei diesen Blutgasveränderungen um unschädliche „Artefakte“ handelt!

In solchen und ähnlichen Situationen wäre es vorteilhaft, ein umfassenderes Bild vom Gesamtzustand des Patienten für FuzzyKBWean zugänglich zu machen. Ob dieses Bild allein über numerische Parameter, wie Gaskonzentrationen und Frequenzen, gezeichnet werden kann, ist fraglich. Möglicherweise schaffen hier zukünftige Versionen von PICIS® Abhilfe, die mehr Informationen, auch nicht-numerischer Art, zur Verfügung stellen können. FuzzyKBWean jedenfalls ist durch das Konzept der linguistischen Variablen (siehe S. 29 und 44) darauf vorbereitet. Seine Regelgrammatik beinhaltet bereits Mechanismen zur Verarbeitung von solchen linguistischen Informationen.

Allerdings verwenden Intensivmediziner bei der Entwöhnung auch Informationen, die nur schwierig auf automatischem Weg erfasst werden können, z. B. „Erscheinungsbild“, „Gesamtzustand“, „Augenöffnen“, etc. Es bleibt zu untersuchen, ob und inwieweit derartige Eingaben für ein closed-loop-System relevant sind und wie sie gegebenenfalls akquiriert werden können (siehe auch S. 88).

11.5.3 Benutzerschnittstelle

Um die Entscheidungsfindung zu erleichtern müssen Informationen auf einfache und intuitive Art präsentiert werden. Ein derartiger Ansatz wird in [Schu2004] beschrieben. Tests mit intuitiven Benutzerschnittstellen stehen allerdings noch aus.

12. Zukünftige Perspektiven

Das langfristige Ziel des FuzzyKBWean-Projektes ist es, ein closed-loop-System zu realisieren, welches den Patienten völlig selbständig und so effizient wie möglich, also so rasch und so schonend es geht, vom Beatmungsgerät entwöhnt. Das medizinische Personal soll nur in Notfällen oder in Situationen, die FuzzyKBWean nicht selbst beherrschen kann, eingreifen müssen.

Die hardwaretechnischen Voraussetzungen dafür sind im Prinzip gegeben. Man benötigt einen D/A-Wandler und einen entsprechend modifizierten Respirator, damit eine Verbindung zwischen Steuercomputer und Beatmungsgerät hergestellt werden kann.

Bevor allerdings das closed-loop-System ernsthaft ins Auge gefasst werden kann, müssen die in Kap. 11.5 genannten Probleme gelöst sein. Den nächsten Schritt könnte der Aufbau einer realitätsnahen Testumgebung darstellen, da man ein closed-loop-System verständlicherweise – aus ethischen und forensischen Gründen – nicht an realen Patienten austesten kann. Diese Testumgebung würde aus einem bereits existierenden Lungenmodell und einem Patientensimulator bestehen. Der Simulator wäre ein Programm, das wie ein realer Patient einen Datenstrom von physiologischen Parametern produziert, welche als Eingabedaten für FuzzyKBWean dienen. Diese Daten können entweder nach einem vorprogrammierten Muster erzeugt oder vom Bediener unmittelbar verändert werden. Gleichzeitig soll der Simulator auf die von FuzzyKBWean berechneten Stellgrößen reagieren, wodurch eine Interaktion zwischen FuzzyKBWean und dem Simulator entsteht.

Auf diese Weise kann das Regelungsverhalten unter Normalbedingungen und in kritischen Situationen und Grenzfällen getestet werden. Die daraus gewonnenen Erkenntnisse fließen dann unmittelbar in die Verbesserung der Wissensbasis ein, so lange bis eine optimale Entwöhnungsstrategie vorliegt.

Wie in Kap. 11.5.2 Unvollständige Eingabedaten (S. 87) beschrieben, lassen sich zum jetzigen Zeitpunkt noch nicht alle Informationen, die von Medizinern als Grundlage bei der Entwöhnung verwendet werden, auf automatischem Weg über ein PDMS erfassen. Es könnte sich daher eine Art von hybridem System als vorteilhaft erweisen, bei dem sowohl automatisch akquirierte Patientendaten als auch manuell eingegebene Informationen in den Regelungsvorgang einfließen. Dies bleibt noch zu untersuchen.

Die verwendeten Konzepte und bisher realisierten Programmsysteme sind effektiv, die bisher erzielten Ergebnisse positiv und ermutigend. Dennoch ist bis zur Erreichung des angestrebten Ziels noch einiges an Forschungs- und Entwicklungsarbeit zu leisten. Ein geeigneter Ausgangspunkt dafür ist in der jetzigen FuzzyKBWean-Implementierung zweifellos vorhanden.

Abbildungen

Tabellen

Literatur

[Abts2007] Abts D.: „Grundkurs Java“, Vieweg+Teubner Verlag, Wiesbaden, 2007

[Adl2004] Adl C.: „Entwicklung und praktischer Einsatz der wissensbasierten Systeme Crisp- und FuzzyTempToxopert zur serologischen Diagnostik von Toxoplasmose“, Magisterarbeit, Technische Universität Wien, Wien, 2004

[Adla2003] Adlassnig K.-P., Rappelsberger A., Seising R.: „Expertensysteme in der Medizin“, Skriptum, Institut für Medizinische Computerwissenschaften, Universität Wien, Wien, 2003

[Aho1999] Aho A. V., Sethi R., Ullman J. D.: „Compilerbau“, Oldenbourg Verlag, München, 1999

[Band1993] Bandemer H., Gottwald S.: „Einführung in Fuzzy-Methoden“, Akademie Verlag, Berlin, 1993

[Baum1993] Baum M., Benzer H.: „Einsatz von Atemhilfen“, in Benzer H., Burchardi H., Larsen R., Suter, P. M. (Hrsg.) „Intensivmedizin“, Springer-Verlag, Berlin, 1993

[Brun2002] Brunner J.: “History and principles of closed-loop control applied to mechanical ventilation”, *Netherlands Journal of Critical Care,* Vol. 6, 6–9, 2002

[Elka1993] Elkan C.: “The paradoxical success of fuzzy logic”, in *Proceedings of the Eleventh National Conference on Artificial Intelligence AAAI-93*, AAAI Press, Washington D.C., 608–703, 1993

[Gott1990] Gottlob G., Frühwirth T., Horn W. (Hrsg.): „Expertensysteme“, Springer-Verlag, Wien, 1990

[Hend2006] Hendrix H., Kaiser M., Yusen R., Merk J.: “A randomized trial of automated versus conventional protocol-driven weaning from mechanical ventilation following coronary artery bypass surgery”, *European Journal of Cardio-Thoracic Surgery,* Vol. 6, 957–963, 2006

[Jouv2007] Jouvet P., Farges C., Hatzakis G., Monir A., Lesage F., Dupic L., Brochard L., Hubert P.: “Weaning children from mechanical ventilation with a computer-driven system (closed-loop protocol): a pilot study.”, *Pediatric Critical Care Medicine,* Vol. 8, 425–432, 2007

[Koll1997] Kollef M. H., Shapiro S. D., Silver P., St. John R. E., Prentice D., Sauer S., Ahrens T. S., Shannon W., Baker-Clinkscale D.: “A randomized, controlled trial of protocol-directed versus physician-directed weaning from mechanical ventilation”, *Critical Care Medicine,* Vol. 25, 567–574, 1997

[Kope1997] Kopetz H.: “Real-Time Systems”, Kluwer Academic Publishers, Boston, 1997

[Lars1997] Larsen R., Ziegenfuß T.: „Beatmung – Grundlagen und Praxis“, Springer-Verlag, Berlin, 1997

[Lell2006] Lellouche F., Mancebo J., Jolliet P., Roeseler J., Schortgen F., Dojat M. et al.: “A multicenter randomized trial of computer-driven protocolized weaning from mechanical ventilation”, *American Journal of Respiratory and Critical Care Medicine,* Vol. 174, 894–900, 2006

[Link2000] Linkens D. A., Abbod M. F., Mahfouf M.: “Intelligent systems in biomedicine”, in *Proceedings of the European Symposium on Intelligent Techniques ESIT 2000*, ERUDIT, Aachen, 46–61, 2000

[Link2001] Linkens D. A., Abbod M. F., Mahfouf M.: “A survey of fuzzy logic monitoring and control utilisation in medicine”, *Artificial Intelligence in Medicine,* Vol. 21, 27–42, 2001

[Matt2006] Matthäus W.-G.: „Grundkurs Programmieren mit Delphi“, Vieweg+Teubner Verlag, Wiesbaden, 2006

[Ocze1996] Oczenski W., Werba A., Andel H.: „Atmen – Atemhilfen. Atemphysiologie und Beatmungstechnik“, Blackwell Wissenschaftsverlag, Berlin, 1996

[Pomb1993] Pomberger G., Blaschek G.: „Software Engineering“, Carl Hanser Verlag, München, 1993

[Psch2007] „Pschyrembel – Klinisches Wörterbuch“, de Gruyter Verlag, Berlin, 2007

[Rose2007] Rose L., Presneill J., Cade J.: “Update in computer-driven weaning from mechanical ventilation”, *Anaesthesia and Intensive Care,* Vol. 35, 213–221, 2007

[Russ1995] Russel S., Norvig P.: “Artificial Intelligence – A Modern Approach”, Prentice Hall, Englewood Cliffs, 1995

[Schi1998] Schildt G.-H., Kastner W.: „Prozessautomatisierung“, Springer Verlag, Wien, 1998

[Schm1995] Schmidt R., Thews G. (Hrsg.): „Physiologie des Menschen“, Springer Verlag, Berlin, 1995

[Schu1996] Schuh C., Hiesmayr M., Ehrngruber Th., Katz E., Neugebauer Th., Adlassnig K.-P., Klement E. P.: “Fuzzy knowledge-based weaning from artificial ventilation (Fuzzy KBWean)”, in Jamshidi, M., Fathi, M., Pierrot, F. (Eds.) *Proceedings of the World Automation Congress – WAC ’96*, 583–588, TSI Press, Albuquerque, 1996

[Schu1998] Schuh C.: „Wissensbasierte Entwöhnung vom Respirator unter Verwendung von Fuzzy- und Nichtfuzzy-Regelungsmodellen“, Dissertation, Technische Universität Wien, Wien, 1998

[Schu2004] Schuh C., Hiesmayr M., Kaipel M., Adlassnig K.-P.: “Towards an intuitive expert system for weaning from artificial ventilation”, in Dick, S., Kurgan, L., Musilek, P., Pedrycz, W., Reformat, M. (Eds.) *Proceedings of NAFIPS 2004, Annual Meeting of the North American Fuzzy Information Processing Society*, 1008–1012, IEEE, Banff, 2004

[Tehr2008] Tehrani F., Roum J.: “Intelligent decision support systems for mechanical ventilation”, *Artificial Intelligence in Medicine,* Vol. 44, 2008, doi:10.1016/j.artmed.2008.07.006.

[Zade1965] Zadeh L. A.: “Fuzzy Sets”, *Information and Control,* Vol. 8, 338–353, 1965

[Zimm1993] Zimmermann H.-J.: „Prinzipien der Fuzzy Logic“, *Spektrum der Wissenschaft 3/93*, 90–93, Spektrum der Wissenschaft Verlagsges. m. b. H., Heidelberg, 1993

Abkürzungen

AF	Atemfrequenz
AKH	Allgemeines Krankenhaus
AMV	Atemminutenvolumen
APRV	airway pressure release ventilation
ARDS	1. acute respiratory distress syndrome 2. adult respiratory distress syndrome
AZV	Atemzugvolumen
BIPAP	biphasic positive airway pressure
CMV	controlled mandatory ventilation
CoG	center of gravity
CPAP	continuous positive airway pressure
ERV	Exspiratorisches Reservevolumen
FC	fuzzy control
FRC	Funktionelle Residualkapazität
IMV	intermittent mandatory ventilation
IRV	1. Inspiratorisches Reservevolumen 2. inversed ratio ventilation
kbe	knowledge base executable
kbs	knowledge base source code
KBWean	knowledge based weaning
MoM	mean of maximum
mXPS	medizinisches Expertensystem
PEEP	positive endexpiratory pressure
PDMS	patient data management system
PICIS	patient integrated clinical information solutions
PS	pressure support
RV	Residualvolumen
SIMV	synchronized intermittent mandatory ventilation
TLC	total lung capacity
VAP	ventilator associated pneumonia
VC	vital capacity
VC_E	expiratory lung capacity
VC_I	inspiratory lung capacity
XPS	Expertensystem

Glossar

Siehe auch [Psch2007]

Algorithmus (abgeleitet von *Muhammad Ibn Mûsâ Al-Chwârismî*, ca. 780 – ca. 850 n. Chr., arab. Mathematiker): Sammlung von Regeln, durch deren schrittweise Befolgung eine vorgegebene Aufgabe gelöst wird.
Alveolen: Lungenbläschen.
ARDS (engl. acute/adult respiratory distress syndrome): akutes Atemnotsyndrom, akutes progressives Lungenversagen, „Schocklunge". Akute »*respiratorische* »*Insuffizienz* durch diffuse Schädigung der »*alveo*»*kapillären* Membran.
Aspiration (lat. aspirare = anhauchen): Eindringen von festen oder flüssigen Stoffen in die Atemwege infolge fehlender Schutzreflexe.
Atelektase (griech. ατελης = unvollständig): nicht belüfteter Lungenabschnitt.

Basalmembran: lichtmikroskopisch homogenes Häutchen als Grenzfläche zwischen »*Epitheli*en bzw. »*Endotheli*en und Bindegewebe.
Beatmung: Einblasen von Atemgas (Raumluft, Sauerstoff, ...) in die Lungen durch Erzeugung eines Überdrucks. Beatmung wird bei ausgefallener oder gestörter Eigenatmung angewendet. Siehe »*Respirator.*

CCU (engl. cardiac care unit): Herzüberwachungsstation, »*Intensivstation* für Patienten mit »*kardial*en Erkrankungen.

Diagnose (griech. διαγνωσιζ = Entscheidung): Zuordnung einer gesundheitlichen Störung zu einem Krankheitsbegriff, im weiteren Sinne für ein »*Symptom* bzw. eine Vermutung (Verdachtsdiagnose).
Diffusion (lat. diffundere = ausgießen, verbreiten): Ausbreitung eines Stoffes bei Vorhandensein eines räumlichen Konzentrationsgefälles. Physiologisch: Stofftransport durch eine Membran.

Embolie: Verlegung eines Blutgefäßes durch ein im Blutstrom verschlepptes, nicht lösliches Gebilde (Blutgerinnsel, Fett, Luft, ...).
endo-: griech. ενδον = innen.
Endothel (griech. θηλειν = wachsen): einschichtige Zellauskleidung von Blutgefäßen und bestimmten Körperhöhlen.
Endotrachealtubus: »*Tubus* zum Einführen in die Luftröhre. Siehe »*Intubation*, »*Beatmung.*
Epithel (griech. επι = darauf, θηλειν = wachsen): geschlossener, ein- oder mehrschichtiger Zellverband, der innere oder äußere Körperoberflächen bedeckt.
Expertensystem (XPS): Computerprogramm für die Lösung von Aufgaben in einer bestimmten Problemdomäne, das sich dazu des Wissens von menschlichen Experten bedient. Dieses Wissen ist in der Wissensbasis des Expertensystems enthalten.
Exspiration (lat. exspirare = aushauchen): Ausatmung.
Extubation: Entfernen des »*Endotrachealtubus* aus der Luftröhre. Siehe »*Intubation.*

heuristisch: Bezeichnung für Mechanismen zur Problemlösung, die die Lösungssuche vereinfachen, aber keine Garantie für die Länge der Suche oder die Exaktheit des Ergebnisses liefern.
Hilus (lat. hilum = kleines Ding): Vertiefung an der Oberfläche eines Organs, wo strangförmig Gefäße, Nerven und Ausführungsgänge ein- bzw. austreten.

ICU (engl. intensive care unit): »*Intensivstation*, »*CCU.*
Indikation (lat. indicare = anzeigen): Grund zur Anwendung eines bestimmten diagnostischen oder therapeutischen Verfahrens in einem Krankheitsfall, der seine Anwendung hinreichend rechtfertigt. Siehe »*Kontraindikation.*
Inspiration (lat. inspirare = einhauchen): Einatmung.
insuffizient (lat. in- = un-, sufficiens = ausreichend): nicht ausreichend, ungenügend.
Insuffizienz (lat.): nicht ausreichende Funktion/Leistung eines Organs bzw. Organsystems.

Intensivstation (engl. ICU = intensive care unit): Station im Krankenhaus mit der personellen und apparativen Ausstattung zur Überwachung und Therapie von Patienten mit lebensbedrohlichen Verletzungen und/oder Erkrankungen. Siehe auch »*CCU.*
Interstitium: Raum zwischen den Organ»*parenchym*en, enthält Bindegewebe, Gefäße und Nerven.
Intubation: Einführen eines Kunststoffschlauches (»*Tubus*) in die Luftröhre. Dient zur Sicherung der Atemwege (»*Aspiration*sschutz) und ist Voraussetzung für maschinelle »*Beatmung*.
invasiv (lat.): eindringend. In der Medizin Bezeichnung für diagnostische oder therapeutische Verfahren, die den Körper verletzen oder auf sonstige Art und Weise beeinträchtigen.

Kapillaren (lat. capillus = Haar): Haargefäße.
Kapnometrie: nichtinvasive Messung des Kohlendioxidgehalts in der Ausatemluft.
kardial: das Herz betreffend, vom Herzen ausgehend.
Kinematik (von griech. κινησιζ = Bewegung): Beschreibung der Beweglichkeit.
kollabieren (lat. collabor): zusammenfallen.
Kontraindikation (lat. contra = gegen): Gegenanzeige; Umstand, der die Anwendung eines bestimmten diagnostischen und/oder therapeutischen Verfahrens bei an sich gegebener »*Indikation* in jedem Fall (absolute K.) bzw. nur unter strenger Abwägung der Risiken (relative K.) verbietet.

Mortalität (lat. mortalitas = das Sterben): Sterblichkeit; das Verhältnis der Anzahl der Sterbefälle zum Durchschnittsbestand der betrachteten Population.

nosokomial (griech. νοσοκομειον = Krankenhaus): Bezeichnung für Krankheitserreger, deren Übertragung gleichzeitig mit der Behandlung und Pflege erfolgt, wobei die Verbreitungswahrscheinlichkeit durch Organisation und Bau des Krankenhauses beeinflusst wird.

Ödem (griech. οιδημα = Schwellung): Ansammlung von wässriger Flüssigkeit in Gewebespalten, z. B. von Haut und Schleimhaut.

Parenchym: Gesamtheit der spezifischen Zellen eines Organs, die dessen Funktion bedingen.
Perfusion (lat. perfundere = durchströmen): Durchströmung des Körpers bzw. von Organen mit Flüssigkeiten (Blut, ...).
Pulsoximetrie: transkutane Messung der arteriellen Sauerstoffsättigung.
Pneumonie: Lungenentzündung.

Respirator (lat. respirare = atmen): Beatmungsgerät; pneumatisch oder elektrisch angetriebenes Gerät zur maschinellen »*Beatmung*. In der »*Inspiration*sphase strömt das Atemgas durch Erzeugung eines Überdrucks in die Lungen; die »*Exspiration* erfolgt passiv durch die elastischen Rückstellkräfte der Lungen und des »*Thorax*.
respiratorisch: die Atmung betreffend.

Sedierung (lat. sedatio = Beruhigung): Dämpfung des Zentralnervensystems durch bestimmte Medikamente (Sedativa).
Splanchnikus (griech. σπλανχνον = Eingeweide): Kurzbezeichnung für den Eingeweidenerv (Nervus splanchnicus).
Symptom (griech. συμπτωμα = Begleiterscheinung): fassbares Krankheitszeichen.

Therapie (griech. θεραπεια = Pflege, Heilung): Behandlung von Krankheiten, Heilverfahren.
Thorax (griech. θωραξ): Brustkorb.
Trauma (griech. τραυμα): Verletzung, psychische oder physische Gewalteinwirkung.
Trigger (engl.): Auslöser.
Tubus (lat. Röhre): Rohr, meist aus Kunststoff, zum Einführen in die Atemwege. Siehe »*Intubation*, »*Beatmung.*

Weaning (engl. to wean = entwöhnen): im intensivmedizinischen Kontext: Entwöhnung beatmeter Patienten vom »*Respirator*.

XPS: Abkürzung für »*Expertensystem.*

Anhang

A – Physiologische Parameter
B – Format des Quelltextes von Fuzzy-Wissensbasen
C – Operatoren von FuzzyKBWean
D – SQL-Anweisungen zur Erzeugung der FuzzyKBWean-Datenbank

Anhang A – Physiologische Parameter

In diesem Kapitel sind die wichtigsten physiologischen Parameter beschrieben, die von FuzzyKBWean verwendet werden. Dabei wird zwischen Ein- und Ausgabedaten unterschieden. Erstere werden von den diversen Überwachungsgeräten geliefert und vermitteln ein Bild vom Patientenzustand, soweit dieser durch numerische Werte beschrieben werden kann. Letztere stellen jene Größen dar, die von FuzzyKBWean aktiv verändert werden können, es sind dies üblicherweise Stellgrößen für den Respirator.

Eingabedaten

pO_2

Bezeichnung: Sauerstoff-Partialdruck
Als Partialdruck bezeichnet man den Teildruck eines Gases in einem Gasgemisch.
Maßeinheit: mmHg
Messung: Blutgasanalyse
Werte: < 50 mmHg : lebensbedrohlich niedrig
< 60 mmHg : sehr niedrig
$\geq$ 60 und < 80 : niedrig
$\geq$ 80 und $\leq$ 100 mmHg : optimal
> 100 mmHg : zu hoch

pCO_2

Bezeichnung: Kohlendioxid-Partialdruck
Maßeinheit: mmHg
Messung: Blutgasanalyse
Werte: < 30 mmHg : zu niedrig
< 35 mmHg : niedrig
zwischen 35 und 45 mmHg : normal
> 45 mmHg : hoch
> 55 mmHg : zu hoch

SpO_2

Bezeichnung: prozentuelle Sauerstoffsättigung
Maßeinheit: %
Messung: Pulsoximeter
Werte: < 85% : lebensbedrohlich niedrig
< 95% : niedrig
um 97% : normal
> 98% : hoch

$EtCO_2$

Bezeichnung: endtidales Kohlendioxid
Maßeinheit: mmHg
Messung: Kapnometer

$EtCO_2$ ist oft niedriger als pCO_2. Aus diesem Grund verwendet FuzzyKBWean die **Differenzkalibrierung**, bei der nach jeder Blutgasanalyse die Differenz

$$delta = pCO_2 - EtCO_2$$

gebildet wird. Die Variable *delta* wird dann in der Zeit bis zur nächsten Blutgasanalyse zum permanent messbaren $EtCO_2$ addiert und pCO_2 daraus rechnerisch ermittelt.

TV_E

Bezeichnung: Atemzugvolumen
Maßeinheit: l

Vrate

Bezeichnung: kontrollierte Atemfrequenz
Maßeinheit: 1/min
Werte: 8/min : Untergrenze
< 10/min : niedrig
zwischen 10/min und 15/min : Normalbereich
12/min : Standard
> 15/min : erhöht
> 20/min : hoch
> 30/min : sehr hoch

Die kontrollierte Atemfrequenz ist in der Regel etwas niedriger als die natürliche Atemfrequenz des Patienten.

I:E

Bezeichnung: Verhältnis von Inspirations- zu Exspirationszeit
Werte: 1:2 : normal (außer bei APRV)
sonstige: 1:2, 1:1, 2:1, 3:1

PIP

Bezeichnung: peak inspiratory pressure, oberes Druckniveau
Maßeinheit: mbar
Werte: ≤ 12 mbar : niedrig
> 12 und ≤ 20 mbar : normal
> 20 und ≤ 28 mbar : hoch
> 28 und ≤ 40 mbar : sehr hoch
> 40 und ≤ 60 mbar : extrem hoch
> 30 mbar : aggressive Beatmung
> 35 mbar : sehr gefährlich, nur in Ausnahmefällen

PEEP

Bezeichnung: positive endexpiratory pressure, unteres Druckniveau
Maßeinheit: mbar
Werte: ≤ 4 mbar : niedrig
> 4 und ≤ 6 mbar : normal
> 6 und ≤ 10 mbar : hoch normal
> 10 und ≤ 18 mbar : hoch
> 18 und ≤ 25 mbar : sehr hoch
3 mbar : physiologischer Wert
5 mbar : Standardwert
> 12 mbar : aggressive Beatmung

FiO_2

Bezeichnung: Inspiratorische prozentuelle Sauerstoffkonzentration
Maßeinheit: %
Werte: 21% : Sauerstoffkonzentration in der natürlichen Luft
< 40% : anzustreben, um Extubation zu ermöglichen
> 60% : toxisch, bei aggressiver Beatmung

Ausgabedaten

PIP, PEEP
Für die Änderungen an den Druckniveaus werden oft verschiedene Strategien verfolgt.

FiO_2
Die Änderungen an der inspiratorischen Sauerstoffkonzentration werden in Fünfer- oder Zehnerschritten vorgenommen, da geringere Änderungen ohne Wirkung sind.

Andere Parameter des Beatmungsgerätes bleiben in der aktuellen Konfiguration von FuzzyKBWean fix eingestellt:

Vrate = 15/min
I:E = 3:1 (entspricht APRV-BIPAP)

Anhang B – Format des Quelltextes von Fuzzy-Wissensbasen

Abschnitt [knowledge base]
Enthält allgemeine Angaben zur Wissensbasis.
KBName=<Name der Wissensbasis>
KBDescription=<Beschreibung der Wissensbasis>
OutOfBoundsColor=<Farbcode für Werte, die außerhalb ihres Gültigkeitsbereiches liegen>

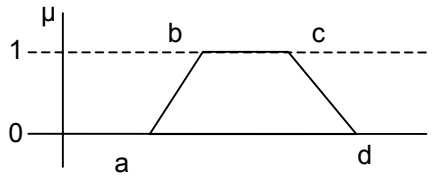
Abschnitt [variables]
Enthält eine Liste der Variablen:
<Variablenname>=

Abschnitt [rules]
Enthält eine Liste der Regelnamen:
<Regelname>=

Abschnitt [v_<Variablenname>]
Enthält die Eigenschaften der Variable „<Variablenname>“:
vdesc=<Beschreibung der Variable>
vtype=<Variablentyp (0=numerisch, 1=linguistisch)>
vdir=<Datenrichtung (0=input, 1=output)>
vunit=<Maßeinheit>
vdelta=<Schrittweite bei grafischer Darstellung>
vlow=<Untergrenze bzw. „I“ für „unbegrenzt“>
vup=<Obergrenze bzw. „I“ für „unbegrenzt“>

Abschnitt [c_<Variablenname>]
Enthält eine Liste der Werte der Variable „<Variablenname>“ (Fuzzy-Mengen oder linguistische Terme) und ihrer Eigenschaften:
<Wertename>=<aI>|<aC>|<a>|<bC>|<b>|<c>|<cC>|<d>|<dC>|<bI>|
<ForeColor>|<BackColor>|<Beschreibung>

<a>, <b>, <c>, <d> bezeichnen die Eckpunkte der trapezförmigen Fuzzy-Menge



μ … Zugehörigkeitsfunktion

<aC>, <bC>, <cC>, <dC>	:	geben an, ob der Eckpunkt zum Intervall gehört oder nicht
<aI>, <bI>	:	geben an, ob die Intervallseite unendlich ist oder nicht
<ForeColor>	:	Vordergrundfarbe (für interne Verwendung)
<BackColor>	:	Hintergrundfarbe (für interne Verwendung)
<Beschreibung>	:	Beschreibung des Wertes

Wenn der Wert ein linguistischer Term ist, sind alle Eigenschaften, mit Ausnahme der Beschreibung, irrelevant und daher auf „0“ gesetzt.

Abschnitt [r_<Regelname>]
Enthält den Text der Regel „<Regelname>“. Jede Zeile wird durch eine Zeilennummer, beginnend bei 0, eingeleitet.

Abschnitt [t_<Regelname>]
a=<Konklusion der Regel in Textform>
p=<Priorität der Regel>

Abschnitt [a_<Regelname>]
Enthält eine Liste der von dieser Regel gesetzten Stellgrößen.
<Stellgröße 1>=<Stellwert 1>
:
<Stellgröße m>=<Stellwert m>

Abschnitt [e_<Regelname>]
Enthält die Erklärung zur Regel „<Regelname>“. Jede Zeile wird durch eine Zeilennummer, beginnend bei 0, eingeleitet.

Beispiel für eine (fiktive) Fuzzy-Wissensbasis

```
[knowledge base]
KBName=TestBase
KBDescription=Wissensbasis für Testzwecke
OutOfBoundsColor=0

[variables]
SpO2=
FiO2=
verbale_Reaktion=

[rules]
OXY_1=

[v_SpO2]
vdesc=Sauerstoffsättigung
vtype=0  ; numerisch
vdir=0   ; input
vunit=%
vdelta=1
vlow=0
vup=100

[v_FiO2]
vdesc=inspiratorischer Sauerstoffanteil
vtype=0  ; numerisch
vdir=0   ; output
vunit=
vdelta=0.1
vlow=0
vup=1

[v_verbale_Reaktion]
vdesc=verbale Reaktion des Patienten
vtype=1  ; linguistisch
vdir=0   ; input
vunit=
vdelta=1
vlow=0
```

```
vup=0

[c_SpO2]
low=[|0|[|0|94|]|96|]|0|0|niedrige Sättigung
normal=[|94|[|96|100|]|100|]|0|0|Normalbereich

[c_FiO2]
low=[|0|[|0|0.2|]|0.21|]|0|0|niedriger O2-Anteil
normal=[|0.2|[|0.21|0.21|]|0.22|]|0|0|Normalbereich
high=[|0.21|[|0.22|1|]|1|]|0|0|hoher O2-Anteil

[c_verbale Antwort]
keine=[|0|[|0|0|]|0|]|0|0|keine verbale Antwort auf äußeren Reiz
unverständlich=[|0|[|0|0|]|0|]|0|0|nur unverständliche Antwort
inadäquat=[|0|[|0|0|]|0|]|0|0|
verwirrt=[|0|[|0|0|]|0|]|0|0|
orientiert=[|0|[|0|0|]|0|]|0|0|

[r_OXY_1]
0=-in
1=- SpO2
2=- SpO2\low

[t_OXY_1]
a=Pulsoximeter überprüfen !  Oder doch Hypoxie ?
p=1

[a_OXY_1]
FiO2=+0.1

[e_OXY_1]
0=Bei fallender O2-Sättigung zuerst Fingersensor
1=des Pulsoximeters überprüfen.
2=Dann Hypoxie ausschließen.
3=Auf klinische Zeichen achten: Zyanose, ...
```

Anhang C – Operatoren von FuzzyKBWean

Logische Operatoren

Diese Operatoren verwenden logische Ausdrücke als Operanden, also beliebig verschachtelte Konstrukte, die in ihrer Gesamtheit ein „Boole'sches" Ergebnis (WAHR oder FALSCH) liefern.

and

Syntax: and <Logischer Ausdruck> <Logischer Ausdruck> { <Logischer Ausdruck> }
Ergebnis: Logischer Ausdruck
Funktion: Verknüpft zwei bis 255 logische Ausdrücke mittels eines logischen UND ($\wedge$). Das Ergebnis der Operation ist WAHR, wenn alle Operanden WAHR ergeben, sonst FALSCH

or

Syntax: or <Logischer Ausdruck> <Logischer Ausdruck> { <Logischer Ausdruck> }
Ergebnis: Logischer Ausdruck
Funktion: Verknüpft zwei bis 255 logische Ausdrücke mittels eines logischen ODER ($\vee$). Das Ergebnis der Operation ist WAHR, wenn mindestens ein Operand WAHR ergibt, sonst FALSCH

not

Syntax: not <Logischer Ausdruck>
Ergebnis: Logischer Ausdruck
Funktion: Negiert den als Operanden angegebenen logischen Ausdruck ($\neg$). Das Ergebnis der Operation ist WAHR, wenn der Operand FALSCH ergibt und FALSCH, wenn der Operand WAHR ergibt.

Vergleichsoperatoren

Als numerischen Ausdruck bezeichnet man einen beliebig verschachtelten Ausdruck, der nach seiner Auswertung eine Zahl ergibt, z. B. 14/2, (x+y)*2, ... oder eine Zahl selbst, z. B. 3,14, 28, ...
Ein linguistischer Ausdruck ist ein Ausdruck, der nach seiner Auswertung eine Zeichenfolge („string") ergibt, z. B. „Hallo, Patient !", „Weaning" + „ forever !"

<, <=, =, >=, >, <>

Syntax: <Operator> <Numerischer Ausdruck> <Numerischer Ausdruck>
Ergebnis: Logischer Ausdruck
Funktion: Diese Operatoren vergleichen zwei numerische Ausdrücke auf „kleiner als" (<), „kleiner oder gleich" (<=), „gleich" (=), „größer oder gleich" (>=), „größer als" (>) oder „ungleich" (<>). Das Ergebnis der Operation ist WAHR, wenn der Vergleich zutrifft, sonst FALSCH.

in

Syntax: in <Numerischer Ausdruck> <Fuzzy-Menge>
Ergebnis: Logischer Ausdruck, Zugehörigkeitsgrad des numerischen Ausdrucks zur Fuzzy-Menge
Funktion: Überprüft, ob ein numerischer Ausdruck in einer Fuzzy-Menge liegt. Das Ergebnis der Operation ist wahr, wenn das der Fall ist, sonst FALSCH. Zusätzlich liefert der in-Operator auch den Zugehörigkeitsgrad des numerischen Ausdrucks zur Fuzzy-Menge, eine reelle Zahl zwischen 0 und 1.

is

Syntax: is <Linguistischer Ausdruck> <Linguistischer Ausdruck>
Ergebnis: Logischer Ausdruck
Funktion: Liefert WAHR, wenn zwei linguistische Ausdrücke identisch sind, sonst FALSCH.

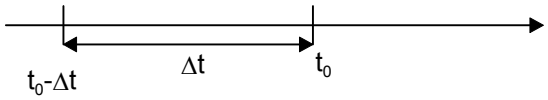
Arithmetische Operatoren

+, -, *, /

Syntax: <Operator> <Numerischer Ausdruck> <Numerischer Ausdruck>
Ergebnis: reelle Zahl
Funktion: Diese Operatoren führen die Rechenoperationen Addition (+), Subtraktion (-), Multiplikation (*) und Division (/) auf den beiden Operanden aus.

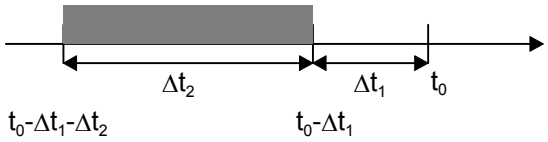
ago

Syntax: ago <Variable> <Δt (Numerischer Ausdruck)>
Ergebnis: reelle Zahl
Funktion: Der ago-Operator ermittelt den in der Vergangenheit liegenden Wert einer Variable. Der numerische Ausdruck gibt an, um wieviele Minuten Δt vom augenblicklichen Zeitpunkt t_0 an gerechnet der gesuchte Wert zurückliegt.



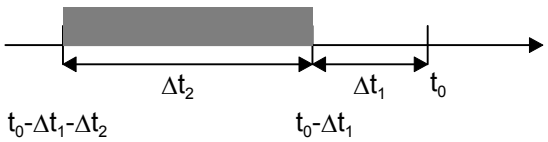
mean

Syntax: mean <Variable> <Δt_1 (Numerischer Ausdruck)> <Δt_2 (Numerischer Ausdruck)>
Ergebnis: reelle Zahl
Funktion: Der mean-Operator berechnet den arithmetischen Mittelwert aus den in einem vergangenen Intervall liegenden Werten einer Variable. Die numerischen Ausdrücke Δt_1 und Δt_2 legen, in Minuten gerechnet, die Position des Intervalls fest.



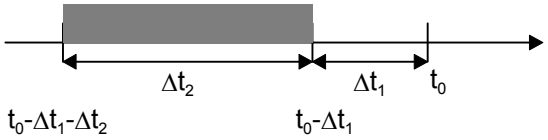
stdDevAbsolute

Syntax: stdDevAbsolute <Variable> <Δt_1 (Numerischer Ausdruck)> <Δt_2 (Numerischer Ausdruck)>
Ergebnis: reelle Zahl
Funktion: stdDevAbsolute berechnet die absolute Standardabweichung aus den in einem vergangenen Intervall liegenden Werten einer Variable. Die numerischen Ausdrücke Δt_1 und Δt_2 legen, in Minuten gerechnet, die Position des Intervalls fest.



stdDevRelative

Syntax: stdDevRelative <Variable> <Δt_1 (Numerischer Ausdruck)> <Δt_2 (Numerischer Ausdruck)>
Ergebnis: reelle Zahl
Funktion: stdDevRelative berechnet die prozentuelle Standardabweichung aus den in einem vergangenen Intervall liegenden Werten einer Variable. Die numerischen Ausdrücke Δt_1 und Δt_2 legen, in Minuten gerechnet, die Position des Intervalls fest.



diffAbs

Syntax: diffAbs <Numerischer Ausdruck> <Numerischer Ausdruck>
Ergebnis: reelle Zahl
Funktion: Dieser Operator berechnet den Absolutwert der Differenz zwischen den beiden numerischen Ausdrücken, also den Betrag, um den sie sich unterscheiden. Das Ergebnis ist immer positiv.

diffRel

Syntax: diffRel <Numerischer Ausdruck> <Numerischer Ausdruck>
Ergebnis: reelle Zahl
Funktion: Dieser Operator berechnet, um welchen Betrag, in Prozent ausgedrückt, sich die beiden numerischen Ausdrücke unterscheiden. Das Ergebnis ist immer positiv. Der Prozentbetrag wird auf der Basis des ersten Ausdruckes berechnet.

Kontrolloperatoren

hasFiredRule

Syntax: hasFiredRule <Regel> <Δt (Numerischer Ausdruck) | „ever“>
Ergebnis: Logischer Ausdruck
Funktion: liefert WAHR, wenn die Regel innerhalb des Zeitraumes vom augenblicklichen Zeitpunkt t_0 bis einschließlich vor t_0-Δt Minuten (bzw. „ever“ = irgendwann) mindestens einmal gefeuert hat, sonst FALSCH.

hasNotFiredRule

Syntax: hasNotFiredRule <Regel> <Δt (Numerischer Ausdruck) | „ever“>
Ergebnis: Logischer Ausdruck
Funktion: liefert WAHR, wenn die Regel innerhalb des Zeitraumes vom augenblicklichen Zeitpunkt t_0 bis einschließlich vor t_0-Δt Minuten (bzw. „ever“ = irgendwann) nicht gefeuert hat, sonst FALSCH.

hasFiredGroup

Syntax: hasFiredGroup <Gruppe> <Δt (Numerischer Ausdruck) | „ever“> <„all“ | „any“>
Ergebnis: Logischer Ausdruck
Funktion: liefert WAHR, wenn alle („all“) oder mindestens eine („any“) Regel einer Gruppe innerhalb des Zeitraumes vom augenblicklichen Zeitpunkt t_0 bis einschließlich vor t_0-Δt Minuten (bzw. „ever“ = irgendwann) mindestens einmal gefeuert hat, sonst FALSCH.

hasNotFiredGroup

Syntax: hasNotFired <Regel> <Δt (Numerischer Ausdruck) | „ever“> <„all“ | „any“>
Ergebnis: Logischer Ausdruck
Funktion: liefert WAHR, wenn alle („all“) oder mindestens eine („any“) Regel einer Gruppe innerhalb des Zeitraumes vom augenblicklichen Zeitpunkt t_0 bis einschließlich vor t_0-Δt Minuten (bzw. „ever“ = irgendwann) nicht gefeuert hat, sonst FALSCH.

isValid

Syntax: isValid <Variable>
Ergebnis: Logischer Ausdruck
Funktion: liefert WAHR, wenn die als Operand angegebene Variable einen gültigen Wert beinhaltet, sonst FALSCH.

isNotValid

Syntax: isNotValid <Variable>
Ergebnis: Logischer Ausdruck
Funktion: liefert WAHR, wenn die als Operand angegebene Variable einen ungültigen Wert beinhaltet, sonst FALSCH.

Anhang D – SQL-Anweisungen zur Erzeugung der FuzzyKBWean-Datenbank

```
CREATE DATABASE <Dateiname> USER <LoginName> PASSWORD <Passwort>;

CREATE TABLE PATIENTS (
   FAMNAME VARCHAR(30),
   NAME VARCHAR(30),
   BIRTHDATE CHAR(10),
   GENDER CHAR(1),
   S_DATE CHAR(10),
   S_TIME CHAR(5),
   BEDNUMBER SMALLINT,
   KNOWBASE VARCHAR(255),
   E_DATE CHAR(10),
   E_TIME CHAR(5));

CREATE TABLE DAT_TEMPLATE (
   D_DATE CHAR(10) NOT NULL,
   D_TIME CHAR(5) NOT NULL,
   :
<Parameter>
   :
   PRIMARY KEY(D_DATE, D_TIME)
);

CREATE TABLE CHG_TEMPLATE (
   C_DATE CHAR(10) NOT NULL,
   C_TIME CHAR(5) NOT NULL,
   PIP_PRP CHAR(6),
   PIP_EFF CHAR(6),
   PEEP_PRP CHAR(6),
   PEEP_EFF CHAR(6),
   FIO2_PRP CHAR(6),
   FIO2_EFF CHAR(6),
   COMMENT BLOB SUB_TYPE TEXT SEGMENT SIZE 80,
   PRIMARY KEY(CH_DATE, CH_TIME)
);

CREATE TABLE CON_TEMPLATE (
   RULE VARCHAR(120) NOT NULL,
   LF_DATE CHAR(10),
   LF_TIME CHAR(5),
   CONCLUSION BLOB SUB_TYPE TEXT SEGMENT SIZE 80,
   EXPLANATION BLOB SUB_TYPE TEXT SEGMENT SIZE 80,
   PRIMARY KEY(RULE)
);

CREATE TABLE RUL_TEMPLATE (
   R_DATE CHAR(10) NOT NULL,
   R_TIME CHAR(5) NOT NULL,
   RULE VARCHAR(120) NOT NULL,
   PRIMARY KEY(R_DATE, R_TIME, RULE)
);
```