Lehrstuhl für Biomedizinische Technik
an der Rheinisch-Westfälischen Technischen Hochschule Aachen
Prof. Dr. rer. nat. G. Rau

Diplomarbeit

Manfred Jeusfeld
Matr.-Nr.: 121822

Thema: Entwurf und Realisierung eines Regelinterpreters und dessen Integration in die Benutzerschnittstelle eines Anästhesie-Informationssystems

Referent: Prof. Dr. rer. nat. G. Rau
Koreferent: Prof. Dr. rer. nat. W. Oberschelp
Lehrstuhl für Angewandte Mathematik, insbesondere Informatik
an der Rheinisch-Westfälischen Technischen Hochschule Aachen
Betreuer: Dipl.-Inform. H. Klocke

This is a reprint of my computer science diploma thesis at the Technical University of Aachen (RWTH). It was handed in on August 22, 1986. Images were scanned from a physical original. The thesis was done at the Helmholtz Institute for Biomedical Technology, situated inside the RWTH university hospital.
Manfred Jeusfeld, 2018-03-02

Hiermit versichere ich, daß ich die Arbeit selbständig verfaßt und keine anderen als die angegebenen Hilfsmittel benutzt habe.

Aachen, den 1986-08-22

... für meine Eltern

Ich danke allen Mitarbeitern der Abteilung Ergonomie des Helmholtz--Instituts Aachen und allen, die an dem Projekt beteiligt waren

Präludium (Fantasie) a-moll
BWV 922
J. S. Bach (1685-1750)

0. Einleitung

Thema. Die Arbeit steht in engem Zusammenhang mit dem am Helmholtz-Institut entwickelten Anästhesie-Informationssystem (AIS) zur Überwachung und Dokumentation von Narkoseverläufen. Das Informationssystem wird zur Zeit am Klinikum erprobt. Die Entscheidungen, die ein Anästhesist während einer Operation zur Steuerung des Patientenzustandes treffen muß, beruhen auf dem Erkennen von oft sehr komplexen Zusammenhängen zwischen einer Vielzahl physiologischer Parameter des Patienten und den therapeutischen Maßnahmen des Anästhesisten. Das AIS soll um Funktionen erweitert werden, die in der Lage sind, den Anästhesisten bei seinen Entscheidungen während der Operation zu unterstützen. Es ist geplant, das AIS dahingehend zu erweitern, daß bestimmte (automatisch erfaßte) Vitalparameter des Patienten (z. B. Blutdruck und Puls) vom AIS "beobachtet" und interpretiert werden können. Die medizinischen Wissensgrundlagen hierzu werden zur Zeit im Rahmen einer medizinischen Doktorarbeit in Form von Fakten und Regeln erarbeitet. Das Projekt wird in enger Zusammenarbeit mit der Abteilung für Anästhesiologie am Klinikum der RWTH Aachen durchgeführt.

In der Arbeit sollen folgende drei grundlegenden Komponenten für ein Anasthesie-Entscheidungsunterstützendes System (AES) entworfen und als experimentelle Version implementiert werden:

1. *Definition einer Syntax und Semantik zur Formulierung von Regeln und Fakten für die oben beschriebene Anwendung*
   Die Grundstruktur der Syntax soll eine "wenn ... dann ..."-Form haben. Durch die Verwendung einer möglichst einfachen und an die Problemstellung angepaßten Semantik soll auch ein mit Computersprachen wenig vertrauter Benutzer in der Lage sein, eine für ihn relevante Problemstellung oder einen Entscheidungsvorgang in annähernd natürlicher Sprache zu beschreiben. Auf Kontrollstrukturen sollte soweit wie möglich verzichtet werden, so daß die Formulierung von "Wissen" auf dieser Ebene weitgehend deklarativ erfolgen kann.

2. *Entwicklung eines datengesteuerten Inferenzmechanismus zur Interpretation von Regeln und Fakten aus der Wissensbasis*
   Der Schlußfolgerungsmechanismus muß deshalb datengesteuert arbeiten, damit das AES Überwachungsfunktionen ausführen kann, d. h. z. B.: das Empfangen einen Blutdruckwertes muß automatisch den Inferenzmechanismus anstoßen und damit eine (mögliche) Schlußfolgerung initiieren.

3. *Integration des AES in die AIS-Benutzerschnittstelle*
   Die entscheidungsunterstützenden Funktionen müssen an der Benutzerschnittstelle des AIS zur Verfügung gestellt werden. Weiterhin ist eine Erklärungskomponente zur Verfügung zu stellen, die dem Benutzer die Möglichkeit

gibt, diejenigen Regeln und Fakten, die zu einer Schlußfolgerung geführt haben, aus der Wissensbasis zu extrahieren und darzustellen.

Einführung. Im Rahmen dieser Arbeit ist der Entwurf und die Realisierung eines Regelinterpreters und dessen Integration in die Benutzerschnittstelle eines Anästhesie-Informationssystems durchzuführen. Das praktische Resultat soll ein Programm sein, das den Anästhesisten bei der Durchführung seiner Aufgaben hinsichtlich zu treffender Entscheidungen unterstützen kann. Der Schwerpunkt liegt in der Entwicklung geeigneter Werkzeuge, die es erlauben, medizinisches Wissens aus dem Bereich der Anästhesie zu formalisieren und damit rechnergestützte Entscheidungshilfen zu geben. Diese Aufgabe bedingt zunächst eine Analyse der Arbeit des Anästhesisten bei einer Operation sowie die Vorstellung des Anästhesie-Informationssystems (AIS), in das das praktische Ergebnis dieser Arbeit, nämlich das Anästhesie-Entscheidungsunterstützungssystem (AES) einzubetten ist. Die Untersuchung des Aufgabenbereichs des Anästhesisten kann hier nur aus Sicht des Informatikers beschrieben werden, eine annähernd umfassende Beschreibung würde den Rahmen dieser Arbeit sprengen und ist für die Beantwortung der hier zu erörternden Fragen nicht notwendig. Ebenso wird das AIS nur soweit zu beschreiben sein, wie es die Behandlung der durchzuführenden Einbettung erfordert. Nachdem das Umfeld dieser Arbeit vorgestellt sein wird, kann die gestellte Aufgabe genauer detailliert und in Einzelaufgaben unterteilt werden. Daran anschliessen wird sich die Entwicklung eines Lösungskonzeptes für diese Teilaufgaben. Dieses Konzept bildet dann die Grundlage für die Realisierung in Form eines Programmes oder Programmsystems. Ein Beispiel gegen Ende dieser Arbeit soll die Funktionalität des erstellten Programms auch in Hinsicht auf die gestellten Anforderungen erläutern. Den Schluß bildet die Diskussion der erarbeiteten Lösungskonzepte und deren Realisierung zusammen mit einem Vergleich mit anderen Arbeiten auf verwandten und entfernteren Gebieten. Ein Glossar soll die wichtigsten technischen und medizinischen Begriffe erläutern. Die Beschreibung des Aufgabenbereiches des Anästhesisten geht aus Gesprächen mit Wolfgang Jacob und Andreas Dawid hervor, die auch das medizinische Wissen zeitgleich mit dieser Diplomarbeit in ihren Dissertationen erarbeitet haben. Einige Eindrücke habe ich aus eigener Anschauung von Herzoperationen im Aachener Klinikum gewonnen.

1. Umfeld

Diese Diplomarbeit beinhaltet die Entwicklung eines Programmsystems, mit dessen Hilfe medizinisches Wissen aus der Anästhesie modelliert werden kann, und das in das schon bestehende Anästhesie-Informationssystem (AIS) einzubinden ist. So werde ich in diesem Kapitel diese beiden Gebiete, die das Umfeld der Arbeit bilden, vorstellen. Für die Modellierung des angesprochenen medizinischen Wissens ist eine zumindest ungefähre Vorstellung von den Entscheidungssituationen, denen der Anästhesist während einer Operation gegenübersteht, notwendig. Der Bereich des Wissenserwerbs wird im allgemeinen als der schwierigste in der Entwicklung wissensbasierter Systeme angesehen. Es kann hier nicht ein endgültiges Konzept erwartet werden, das alle Bereiche der Entscheidungsunterstützung in der Anästhesie abdeckt. Vielmehr ist das zu entwickelnde Anästhesie-Entscheidungsunterstützungssystem eine erste Annäherung an diese Aufgabe. Da die Formulierung des medizinischen Wissens in medizinischen Dissertationen zeitlich parallel zur Entwicklung des AES, das mit diesem Wissen arbeiten soll, läuft, ist zunächst in möglichst kurzer Zeit ein Prototyp zu entwickeln. So wird sich die Modellierung des Wissens parallel zu seiner Erhebung vollziehen. Dies bedingt eine gewisse Einschränkung; eine Beurteilung der Anwendbarkeit des zu entwickelnden Systems auf weitere Gebiete der Medizin ist aus der Sicht des Informatikers nur bedingt möglich.

1.1. Der Aufgabenbereich des Anästhesisten

Der Anästhesist bzw. die Anästhesistin ist während einer Operation für die Aufrechterhaltung der Lebensfunktionen des Patienten zuständig. Je nach Operationstyp wird die Funktion eines oder mehrerer Organe beeinträchtigt. Diese Beeinträchtigung kann sowohl durch den Eingriff selbst als auch durch die Narkosemethode bedingt sein, für deren Ablauf der Anästhesist verantwortlich ist. Die in diesem Kapitel umrissenen Tätigkeiten des Anästhesisten werden sich hauptsächlich an dem Ablauf von Herzoperationen insbesondere zur Legung eines Bypasses orientieren. Die Ausführungen orientieren sich an der Dissertation von Redecker [RED 85] über die Handlungs- und Entscheidungsläufe des Anästhesisten sowie Gespräche mit Wolfgang Jacob und Andreas Dawid ([JAC 86],[DAW 86]), die zeitgleich mit dieser Diplomarbeit ihre medizinischen Dissertationen am Helmholtz-Institut und der Abteilung für Anästhesiologie duchgeführt haben. Einige Kenntnisse stammen auch aus der persönlichen Teilnahme an Hezoperationen in einem Operationssaal des Aachener Klinikums.

Bild 1.1-1 zeigt ein Protokoll wie es von dem Anästhesie-Informationssystem erstellt wurde. Es zeigt im zweiten Fenster von oben den typischen Verlauf der Vitalparameter Blutdruck, Puls und der beiden Temperaturkurven für ösophageale

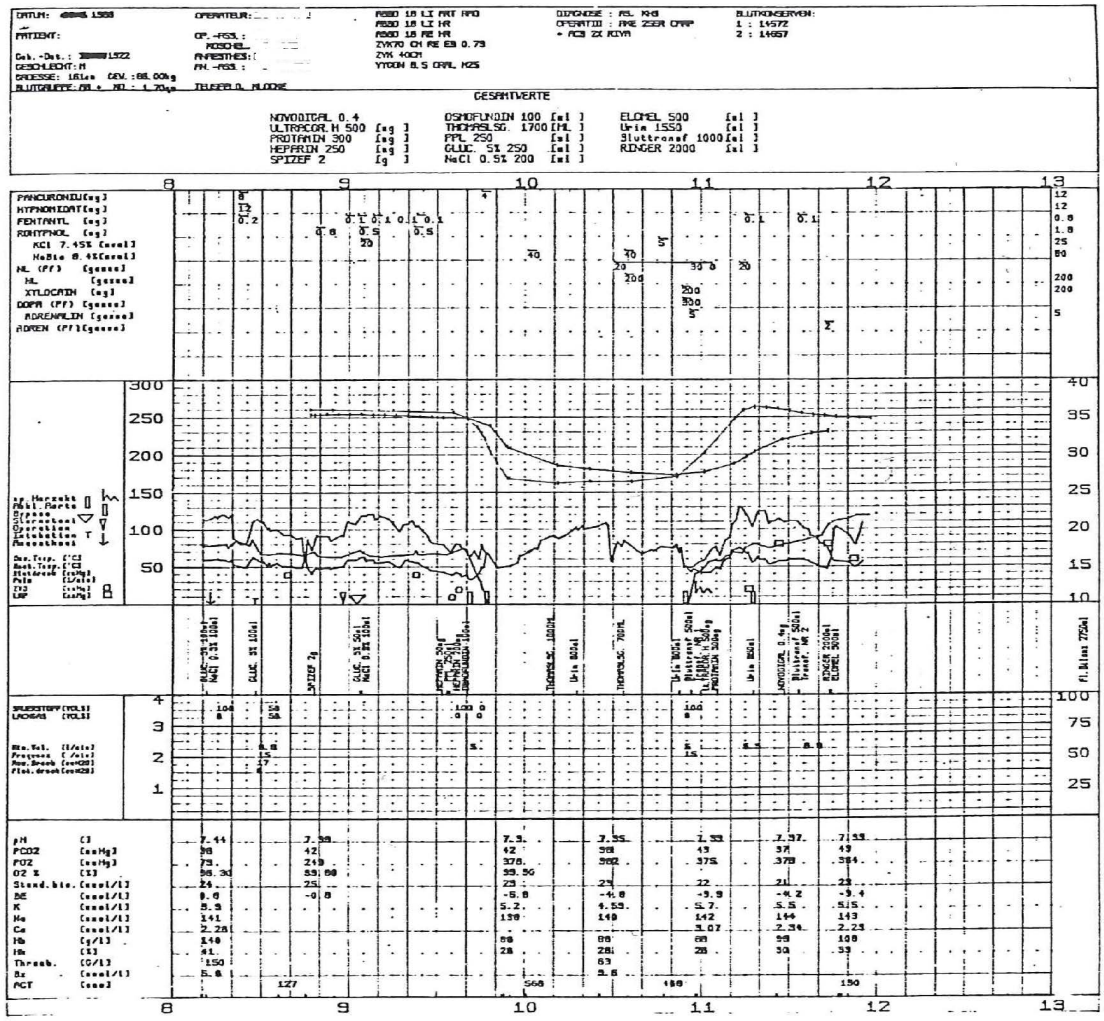

Bild 1.1-1: Anästhesie-Protokoll

(Außen-) und rektale (Körperkern-) Temperatur des Patienten. Das untere Fenster enthält die Laborwerte. Darüber kann man die verschiedenen Respiratoreinstellungen erkennen. Zwischen diesem Fenster und dem Vitalparameterfenster sind die Eintragungen für Verabreichungen von Medikamenten zu finden, wichtige Medikamente insbesondere Perfusorinfusionen sind im oberen Fenster aufgezeichnet. Ganz nach oben auf dem Formular sind Operationsdaten in Textform gesetzt.

Operationsereignisse wie z.B. *Operationsbeginn*, *Anästhesiebeginn* und *Intubation* sind im am unteren Rand des Vitalparameterfensters aufgeführt. Der Verlauf der Blutdruck- und Pulskurven zeigt folgende drei Phasen:

Phase 1 Zunächst wird die Operation durch die Legung von Meßfühlern und Kathetern vorbereitet. Oberer (systolischer) und unterer (diastolischer) Blutdruckwert sind deutlich unterschieden. Das Herz sorgt mit seinen Pumpbewegungen für den Transport des Blutes. Gegen 8:28 Uhr wird der Patient intubiert, d. h. das Mundstück des Beatmungsschlauches wird in die Luftröhre eingeführt (*Intubation*). Nach kurzer Beatmung mit 100 Prozent Sauerstoff wird der Patient mit einem Lachgas/Sauerstoff-Gemisch narkotisiert. Das Protokoll zeigt, daß die eigentliche Operation kurz vor 9 Uhr beginnt. Zehn Minuten später wird das Brustbein aufgetrennt (*Sternotomie*). Kurz vor Ende von Phase 1 wird der Patient wieder mit 100 Prozent Sauerstoff beatmet.

Phase 2 Das Herz wird vom Blutkreislauf abgeklemmt. Der Puls sinkt auf Null. In dieser Phase übernimmt die Herz-Lungen-Maschine die Funktionen sowohl des Herzens als auch der Lunge. Sie arbeitet nicht mit Pumpbewegungen sondern erzeugt einen kontinuierlichen Druck, so daß es für den Blutdruck nur noch einen Wert gibt. Da die Lungenfunktion von der Herz-Lungen-Maschine ausgeführt wird, entfällt die maschinelle Beatmung. Während des Herzstillstandes wird die Legung des Bypass vorgenommen. Das Blut wird durch die Herz-Lungen-Maschine auf eine Temperatur von etwa 27 Grad Celsius heruntergekühlt, um die die physiologischen Abläufe zu verlangsamen.

Phase 3 Das Herz wird wieder an den Blutkreislauf angeschlossen und zum Schlagen angeregt, wie die Pulskurve zeigt. Die Herz-Lungen-Maschine wärmt das Blut wieder auf und wird schließlich abgeklemmt (*Bypassende*). Die Beatmung erfolgt wieder über die Luftröhre.

Das Protokoll zeigt daneben eine Vielzahl weiterer Eintragungen insbesondere von Medikamenten und Laborwerten. Weitere Daten sind am Respirator, der die maschinelle Beatmung durchführt, und an einigen weiteren Geräten ablesbar. Sie bilden zusammen mit visuellen Eindrücken wie Gesichtsfarbe und Schweißbildung des Patienten die Grundlage für die Entscheidungen des Anästhesisten. Die Häufigkeit der Entscheidungssituationen hängt von der Operationsphase und der aktuellen Verfassung des Patienten ab. Ungünstige Werte führen im allgemeinen zu einer intensiveren Überwachung und Therapierung.

Wie der obigen kurzen Schilderung zu entnehmen ist, wird der Patient künstlich beatmet. Ein sogenannter Respirator führt die mechanische Lungenfunktion des Ein- und Ausatmens durch, indem er ein Gasgemisch über einen Tubus in die Lunge presst. Außerdem wird über ihn bei einer Lachgasnarkose dem Patienten das Narkosegas Lachgas (N_2O) zugeführt. Durch die Einstellung des Respirators muß eine ausreichende Sauerstoffversorgung und Kohlendioxidabfuhr sichergestellt werden. Der Anästhesist kann die Effektivität der Einstellung durch eine Anzahl von Meßwerten überprüfen und ggf. eine Veränderung vornehmen. Diese Parame-

ter werden zum Teil ständig gemessen, zum Teil werden sie aber auch in Laboruntersuchungen während der Operation ermittelt. Liegen einer oder mehrere dieser Parameter außerhalb eines Normbereiches, so ist vom Anästhesisten eine Maßnahme zu ergreifen, die zu einer Rückkehr des Parameters in seinen Normbereich führt. In der Regel stehen dem Anästhesisten mehrere Therapiemaßnahmen offen, die für die Abstellung der Abweichung geeignet sind. Die tatsächliche Wahl hängt einmal von den weiteren Umständen, also den restlichen Patientendaten und der Operationssituation, sowie möglicherweise von einer Präferenzliste des Anästhesisten ab.

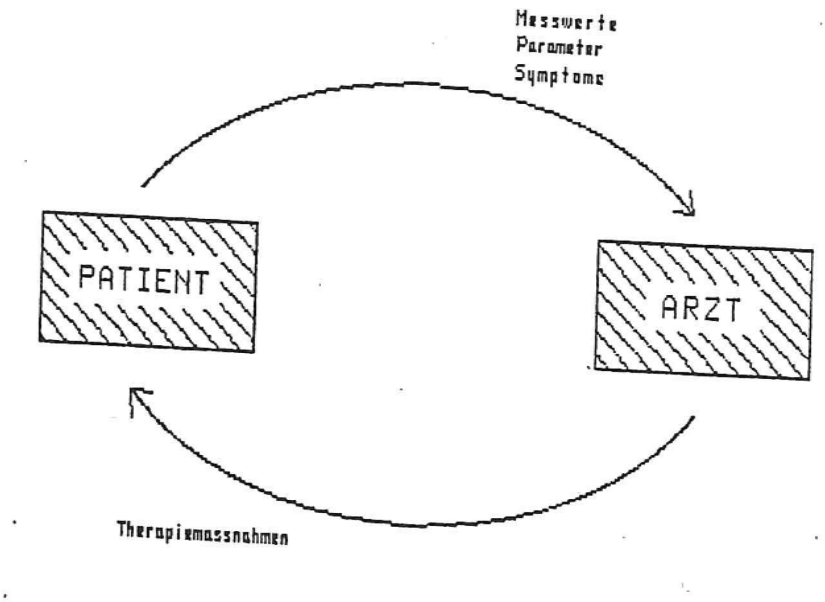

Bild 1.1-2: Entscheidungssituation des Anästhesisten

Bild 1.1-2 zeigt schematisch die Aufgabe des Anästhesisten. Er erhält vom Patienten Daten (Parameter, Meßwerte, Symptome) und fällt aufgrund dieser Daten und seines medizinischen Wissens Entscheidungen über die Ergreifung oder Nichtergreifung einer Therapie wie etwa einer veränderten Respiratoreinstellung oder einer Verabreichung eines Medikaments.

Die Einstellung des Respirators soll beispielhaft die Entscheidungsabläufe eines Anästhesisten während einer Operation am offenen Herzen verdeutlichen. Die Schilderung bezieht sich auf den am Aachener Klinikum eingesetzten Respiratortyp (*Siemens Servo Ventilator*). Bei unserem Operationstyp werden als Atemgase Sauerstoff und Lachgas verwendet. Am Respirator einstellbar sind nun folgende Parameter:

(1) *Frequenz*
   – die Anzahl der Atemstöße pro Minute
(2) *Minutenvolumen*

  - das Atemgasvolumen in Liter, das pro Minute in die Lunge des Patienten gepresst wird

(3) *PEEP*
  - der positive endexpiratorische Druck, d. h. der Druck, der auf die Rückleitung des Beatmungsschlauches gelegt wird

(4) *Seufzer*
  - tiefer Atemstoß

(5) *I/E*
  - Verhältnis der Dauer der Einatmungsphase zu der Dauer der Ausatmungsphase

(6) *Lachgas/Sauerstoff-Verhältnis*
  - Mischungsverhältnis der beiden Atemgase

Bei maschinellem Betrieb sind u. a. zwei Meßwerte am Respirator ablesbar:

(A) *Maximaldruck*
  - der maximale Druckwert, der sich während der Einatmungsphase aufbaut

(B) *Plateaudruck*
  - der Satteldruck vor der Ausatmungsphase

Der Respirator baut am Anfang eines Beatmungszyklus den Druck im Beatmungsweg bis zu einem Maximaldruck auf, der dann auf ein niedrigeres Niveau abfällt, dort einige Zeit verharrt und schließlich wieder auf Null (oder *PEEP*) sinkt. Der Verlauf der Druckkurve hängt ab vom aufgebauten Druck des Respirators und der Dehnbarkeit (*Compliance*) der Lunge sowie in geringem Maße der Zuleitungen ab.

Dem Anästhesisten stehen während der Operation einige Meßwerte zur Verfügung, die ihm ein Urteil darüber erlauben, ob die aktuelle Einstellung des Respirators die Erfüllung seiner Aufgabe gewährleistet. Eine zentrale Rolle spielen hier die Laborwerte, die im Laufe der Operation in unregelmäßigen Abständen aus Blutproben des Patienten gewonnen werden. Dies sind u. a.:

(a) PO_2
  - der Sauerstoffpartialdruck im Blut

(b) PCO_2
  - der Kohlendioxidpartialdruck im Blut

(c) $O_2\%$
  - der Sauerstoffsättigungsgrad des Blutes

(d) *pH*
  - die Wasserstoffionenkonzentration des Blutes

(e) *Standardbicarbonat*
  - "Pufferkapazität" des Blutes zur Aufnahme von sauren Stoffen

(f) *BE*
  - Basenüberschuß

Der PO_2-Wert ist ein Maß für die im Blut des Patienten enthaltene Sauerstoffmenge. Sinkt er unter einen gewissen Grenzwert, so muß der Anästhesist geeignete Maßnahmen treffen. Der Grund für einen zu niedrigen PO_2-Wert mag eine zu geringe Versorgung des Patienten mit Sauerstoff oder auch eine mangelnde Diffusion der Sauerstoffmoleküle durch die Membran der Lungenbläschen in das Blut sein. Der Anästhesist kann den Sauerstoffanteil an den Atemgasen erhöhen, falls dieser nicht schon 100 Prozent beträgt. Eine weitere mögliche Therapiemaßnahme ist die Erhöhung des positiven endexpiratorischen Drucks (*PEEP*), der als Gegendruck auf die Ausleitung des Beatmungsapparates gelegt wird und der einem Kollabieren der Lungenbläschen entgegenwirkt. Eine ähnliche Funktion erfüllt ein tiefer Atemstoß (*Seufzer*). All diese Maßnahmen sind natürlich nur möglich, wenn der Patient maschinell beatmet wird. Ist er an die Herz-Lungen-Maschine angeschlossen, so übernimmt diese die Versorgung des Blutes mit Sauerstoff. Zuständig für die Funktion der Herz-Lungen-Maschine ist der Kardiotechniker.

Die Laborwerte werden vom Anästhesisten je nach Zustand des Patienten vom Blutanalyselabor angefordert. Die Blutprobe wird von einem Mitarbeiter des Labors abgeholt, und die Ergebnisse liegen dem Anästhesisten nach etwa 10 bis 15 Minuten auf einem Papierformular vor.

1.2. Das Anästhesie-Informationssystem

Das Anästhesie-Informationssystem (AIS) ist ein rechnergestütztes System, das die Arbeit des Anästhesisten bei einer Operation erleichtern und die verschiedenen Apparaturen und Meßgeräte, mit denen der Anästhesist umgeht, integrieren soll. Es ist gekennzeichnet von einer auf die Anwendung im Operationssaal zugeschnittenen Benutzerschnittstelle, die physikalisch lediglich aus einem berührempfindlichen Farbbildschirm besteht. Das Konzept wurde ab etwa 1978 am Helmholtz-Institut Aachen in Zusammenarbeit mit der Abteilung für Anästhesiologie am Klinikum der RWTH Aachen erarbeitet, die Realisierung auf dem Rechner begann 1982, und seit 1985 findet die praktische Erprobung des Systems in einem Operationssaal für Herzoperationen des Klinikums statt. Dieses Kapitel soll das AIS in seinen Grundzügen vorstellen und bildet damit die Grundlage für die später vorzustellende Anbindung des Entscheidungsunterstüzungssystems an das AIS. Das AIS ist ein Projekt der Abteilung Ergonomie im Helmholtz-Institut Aachen. Bernotat und Rau ([BER 80], [RAU 82]) geben einen allgemeinen Überblick uber die Aspekte der Ergonomie in der Medizin insbesondere über den Stellenwert der Gestaltung der Mensch-Maschine-Schnittstelle bei technischen Geräten. Redecker [RED 85] und Trispel et al. ([TRI 82],[TKR 85]) heben die kognitiven Strukturen hervor, die Einfluß auf die Entwicklung des AIS nahmen. Klocke et al. [KLK 84] führen die Funktionalität des AIS vor und betonen

die Bedeutung der Überwindung der *Kommunikationsbarriere* zwischen Mensch und rechnergestützen Informationssystemen. Das Programm DMPROT wurde von Stefan Hoffmann im Rahmen seiner Diplomarbeit [HOF 85] am Helmholtz-Institut entwickelt. Ebenso entstand das Programm TMD als Teil der Diplomarbeit von Ralph Schlieter [SCH 85].

Die Anästhesisten sind verpflichtet, während der Operation ein Protokoll uber den Verlauf der Vitalparameter des Patienten sowie über alle verabreichten Medikamente und Geräteeinstellungen zu führen. Zur Zeit werden diese Aufzeichnungen noch von Hand auf einem Papierformular durch den Anästhesisten bzw. durch die Anästhesieschwester vorgenommen. Die Vitalparameter Blutdruck, Puls und Temperatur werden etwa alle fünf Minuten von einem Monitor abgelesen und in das Protokoll eingetragen.

Die Benutzerschnittstelle. Das AIS ermöglicht nun, daß alle vorzunehmenden Eintragungen auf einem berührempfindlichen Bildschirm durchgeführt werden. Die Vitalparameter werden automatisch erfaßt und vom System selbst eingetragen. Alle Daten werden auf dem gleichen Bildschirm dargestellt, auf dem sie auch eingetragen wurden. Bild 1.2-1 zeigt die Abschrift eines typisches Bildes des AIS, nämlich die Arbeitsseite 4. Man erkennt zwei sogenannte Meßwertfenster, die von z. T. beschrifteten Rechtecken umrahmt sind. Im oberen Meßwertfenster sind die Kurvenverläufe von Blutdruck, Puls, rektaler und oesophagealer Temperatur eingezeichnet. Sie beziehen sich auf die rechts und links von den Meßwertfenstern aufgetragenen Skalen. Die Zeitskala trennt das obere Fenster von dem unteren, in dem das Atemgasverhältnis dargestellt ist.

Die mit mit Zahlen beschrifteten "Flaggen" stehen für die verabreichten Mengen der Medikamente, deren Namen links von den Meßwertfenstern auf gleicher Höhe wie die entsprechenden Flaggen stehen. Die mit "0.4" beschriftete Flagge steht zum Beispiel für die um ca. 11:30 vorgenommene Verabreichung von 0.4 mg NOVODIGAL. Die aktuelle Zeit wird in Form einer senkrechten *Zeitlinie* dargestellt, deren digitaler Wert oberhalb der Zeitlinie aufgetragen ist. Der berührempfindliche Bildschirm ermöglicht es, daß auf ihm alle Eingaben des Benutzers durchgeführt werden können. Die Rechtecke um die Meßfenster herum sind sogennante *virtuelle Tasten*, d. h. ein Druck auf eine solche Taste ist eine Eingabe für das AIS, das bei Betätigung die entsprechende Aktion ausführt. Eine Farbkodierung der Tasten zeigt dem Benutzer an, welche der Tasten aktivierbar d. h. betätigbar ist. Es gibt drei Zustände[1] für die Tasten:

A. *nicht aktivierbar*
   - die Taste hat die Farbe Blau, ein Druck auf die Taste wird ignoriert

B. *aktivierbar*
   - die Taste hat die Farbe Hellblau, bei Betätigung wird die entsprechende Aktion ausgeführt

[1] Tatsächlich gibt es in der aktuellen Implementierung des AIS noch einen vierten Zustand, der jedoch nur für eine Spezialfunktion benötigt wird, und deshalb hier nicht von Interesse ist

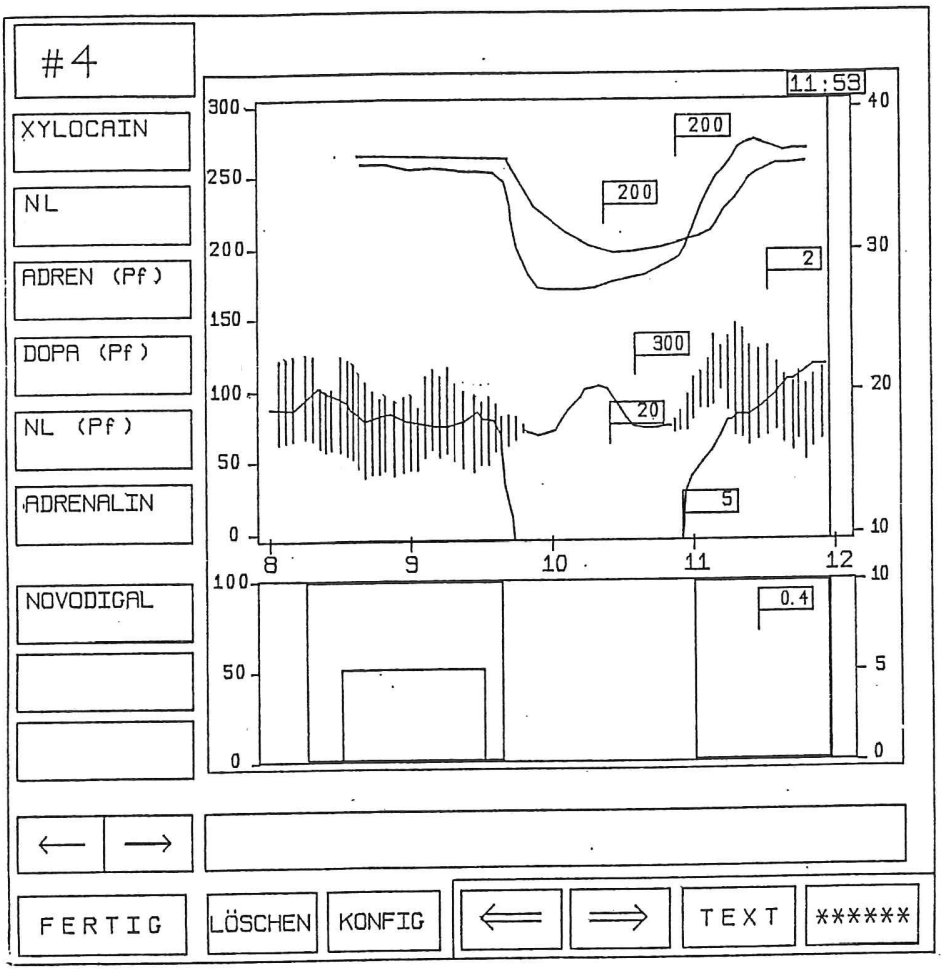

Bild 1.2-1: Arbeitsseite 4 des AIS

C. *aktiviert*
   - die Taste hat die Farbe Weiß, sie wurde zu einem früheren Zeitpunkt betätigt und kann auch in diesem Zustand betätigt werden

Diese Tastenzustände und ihre Farbkodierung sorgen dafür, daß der Benutzer nur syntaktisch korrekte Eingaben machen kann, und daß er in jeder Phase einen Überblick über die für den nächsten Schritt betätigbaren Tasten hat. Ein Beispiel soll die Wirkung dieser Benutzersteuerung verdeutlichen und zugleich weitere Bedienelemente des AIS vorstellen:

Aufgabe:

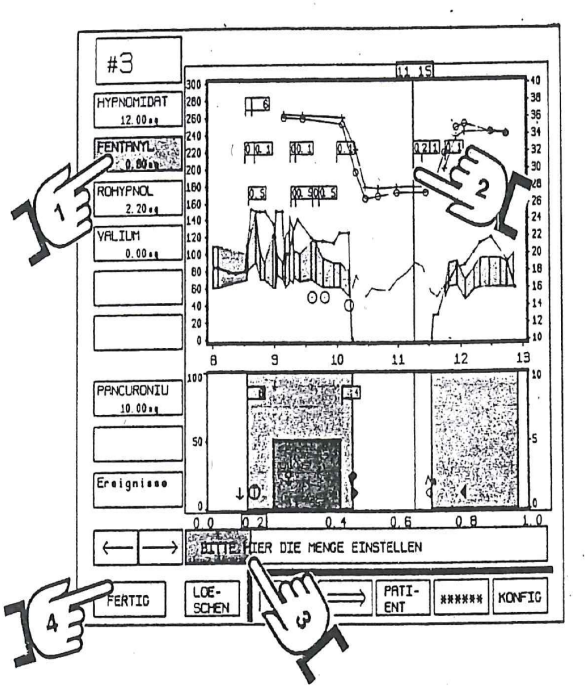

Bild 1.2-2: Eintragung eines Medikaments im AIS

Der Patient hat um 11:15 Uhr 0.2 *mg des Medikaments* FENTANYL *verabreicht bekommen, und dies soll mit Hilfe des AIS protokolliert werden. Ablauf der Eintragung (vgl. Bild 1.2-2):*

1. *Auswahl des Begriffs* FENTANYL *durch Betätigung der entsprechenden Taste*
   Die Taste FENTANYL wird auf *aktiviert* gesetzt, die Zeitlinie und der Mengenschieber auf *aktivierbar*, alle anderen Tasten wechseln in den Zustand *nicht aktivierbar*.
2. *Einstellung der Zeit durch Betätigung der Zeitlinie in einem der Meßfenster, falls die Eintragung nicht zur Echtzeit vorgenommen wird*
   Die Zeitlinie folgt der Bewegung des Fingers.
3. *Einstellung der Menge (*0.2 *mg) auf dem sog. Mengenschieber, das ist das langgezogene Rechteck unter dem unteren Meßfenster*
   Direkt über dem Mengenschieber folgt der Digitalwert der Bewegung des Fingers, der einstellbare Bereich ist dem Medikament angepaßt. In dem Meßfenster erscheint eine gestrichelte Flagge mit dem eingestellten Wert in der Höhe der Taste FENTANYL und in der Position der Zeitlinie. Die FERTIG-Taste wird auf *aktivierbar* gesetzt.
4. *Betätigung der FERTIG-Taste links unten auf der Bildseite*

Die Eintragung wird vom AIS auf einer Datei abgespeichert, die Flagge wird mit

durchgezogenen Linien dargestellt, und das System kehrt in den Ausgangszustand zurück.

Schritte 2. und 3. können auch in umgekehrter Reihenfolge stattfinden und mehrfach hintereinander angewendet werden. Das AIS verfügt insgesamt über 14 Arbeitsseiten, auf denen jeweils bis zu 9 Begriffe konfiguriert sein können. Mit dem oben beschriebenen Eintragungszyklus werden praktisch alle Daten in das AIS eingetragen. Eine Ausnahme bilden die Patienten- und Textseiten, auf denen u. a. Patientenname, -alter und unformatierter Text mittels einer auf dem Bildschirm dargestellten Tastatur eingegeben werden können. Der Benutzer kann bereits eingetragene Daten editieren, indem er

1. den zu editierenden Begriff auswählt (z. B. FENTANYL),
2. sich mit den Pfeiltasten links neben dem Mengenschieber zu dem gewünschten Eintrag durchtastet,
3. die gewünschte Veränderung vornimmt (Zeit ändern, Wert ändern oder löschen) und
4. die Änderung mit der FERTIG-Taste abspeichert.

Der Benutzer hat jederzeit die Möglichkeit, den Interaktionszyklus abzubrechen, indem er den ausgewählten Begriff (hier FENTANYL) noch einmal betätigt.

Hard- und Softwarekomponenten. Das Programm AIS ist in der Programmiersprache PASCAL auf einem LSI-11/73-Rechner der Firma Digital Equipment Corporation (DEC) implementiert. Die LSI-11-Rechnerfamilie gehört zu den 16-Bit-Rechnern und ist voll kompatibel zu der PDP-11-Familie, die Anfang der siebziger Jahre von DEC als Nachfolger der PDP-8 entwickelt wurde. Kennzeichen dieser Rechner ist der für heutige Begriffe kleine Adreßraum von 64 KB, der von 16-Bit-Adreßregistern aufgespannt wird. Zu den Stärken des Rechners zählt die Anbindung externer Geräte.

Bild 1.2-2 zeigt schematisch die Rechnerkonfiguration des AIS, wie sie zur Zeit im Operationssaal ihre Verwendung findet. Die doppelt umrandeten Rechtecke stehen für physikalische Geräte, einfach umrandete Rechtecke darin stehen für Programme, die auf den Geräten (in diesem Fall Rechner) laufen. Schwarze Rechtecke am Rande der Geräte symbolisieren serielle Schnittstellen, über die die mit Kabel verbundenen Geräte miteinander kommunizieren. Umrandete Rechtecke an den Geräten stehen für Analogschnittstellen, über deren Leitungen analoge Signale (Spannungen) übertragen werden. Die Pfeile in dem Bild schließlich stellen einen wie auch immer gearteten Datenstrom dar. Das Bild zeigt im Zentrum den Rechner LSI-11/73, auf dem das Programm AIS läuft. Dieses steht zu seiner Linken mit dem Graphiksystem AED bzw. MGE in Verbindung. An das Graphiksystem ist der hochauflösende Farbbildschirm angeschlossen, auf dem die Bildseiten des AIS dargestellt werden. Dieser Bildschirm ist mit einer berührempfindlichen Folie (*TSD*) überzogen, die via Schnittstelle mit dem Rechner KXT gekoppelt ist. Der

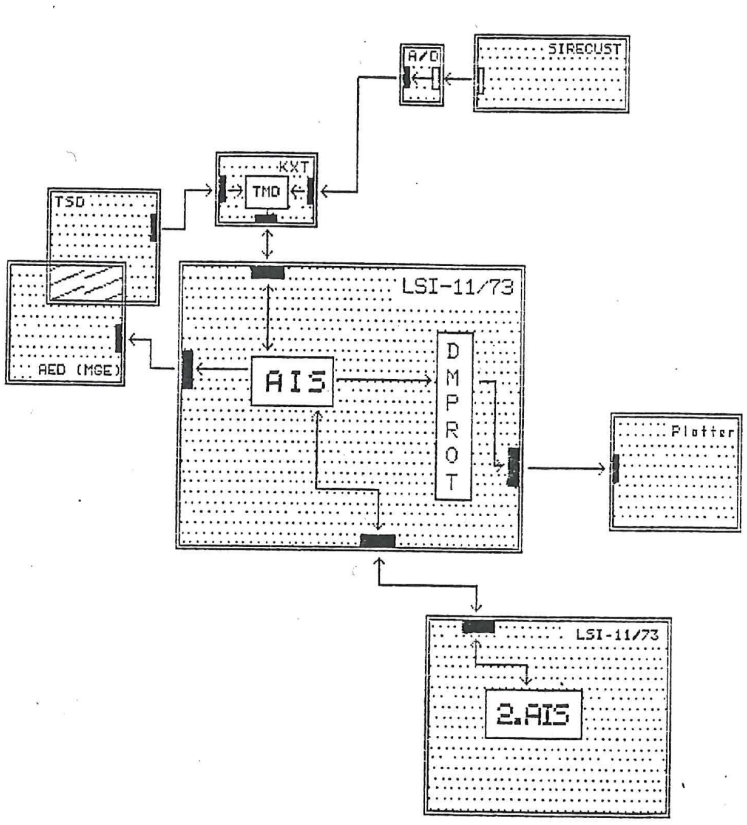

Bild 1.2-3: Rechnerkonfiguration des AIS

KXT ist ein Rechner, der mitsamt Speicher und Schnittstellen auf einer einzigen Karte untergebracht ist. Auf ihm läuft das Programm TMD, das zwei Aufgaben erfüllt, nämlich

(1) der Empfang von kartesischen Koordinaten vom TSD und die Weiterleitung der Information in Form von Tastennummern an das AIS und

(2) der Empfang von Daten des Monitors SIRECUST und die Weiterleitung an das AIS.

Punkt (2) betrifft die automatische Meßwerterfassung, die im AIS implementiert wurde. Der SIRECUST-Monitor ist ein weit verbreitetes Überwachungsgerät, das auch am Aachener Klinikum eingesetzt wird. Es erfaßt vom Patienten über Meßfühler Elektokardiogramm (EKG), Blutdruck und zwei verschiedene Temperaturen und stellt sie auf einem eingebauten Bildschirm in Kurvenform sowie als Digitalwerte dar. Aus dem EKG errechnet der SIRECUST die Herzfrequenz (Puls). Die Werte dieser Parameter werden über eine Analogschnittstelle und einen Analog-Digital-Wandler an das TMD-Programm übermittelt. Neben dem AIS-Programm läuft auf dem zentralen Rechner noch das Programm DMPROT.

Dieses Programm erhält vom AIS Daten über Eintragungen von Medikamenten und Vitalparametern und zeichnet sie in aufbereiteter Form auf einem Digitalplotter auf[2]. Schließlich kann das AIS auch noch mit einem zweiten identisch aufgebauten System kommunizieren.

Einen Einblick in das Zusammenwirken der Geräte mag die Implementierung der *virtuellen Tasten* geben, die im vorigen Abschnitt aus der Sicht des Benutzer beschrieben wurden. Eine *virtuelle Taste* hat zwei Aspekte, nämlich einerseits ihr Bild (in unserem Fall ein beschriftetes Rechteck) und andererseits ihre Funktion als Eingabelement ("gedrückt" oder "nicht gedrückt"). Der erste Aspekt wird durch eine entsprechende Darstellung auf dem Farbbildschirm durch das Graphiksystem nach Anweisung des AIS-Programmes realisiert. Die Realisierung der Eingabefunktion führt die berührempfindliche Folie zusammen mit dem auf dem KXT laufenden Programm TMD durch. Wenn vom AIS eine neue Bildseite aufgebaut wird, so teilt es dem TMD-Programm die Lage der in diesem Bild enthaltenen Tasten mit. Ein Druck auf eine Taste auf dem Bildschirm bewirkt die Übermittlung der kartesischen Koordinaten des Druckpunktes an TMD, dort den Vergleich mit den gespeicherten Tastenkoordinaten und die Übertragung der Nummer der gefundenen Taste an das AIS-Programm. Das AIS führt schließlich die mit der Taste verbundene Aktion aus. Der Eindruck des Benutzers, daß er es mit einer tatsächlichen Taste zu tun hat, wird noch dadurch verstärkt, daß das AIS bei Tastendruck den sofortigen Farbwechsel entsprechend der oben erwähnten Farbkodierung veranlaßt.

Zusammenfassung. Das AIS ist ein Hilfsmittel für den Anästhesisten, dessen Hauptcharakteristikum eine sehr effiziente Benutzerschnittstelle nach den Richtlinien ergonomischer Gestaltung ist. Es ermöglicht die gebündelte Darstellung einer Vielzahl von Informationen, wie sie bei Operationen anfallen, auf einem einzigen Bildschirm, der dank seiner Berührempfindlichkeit zugleich aus der Sicht des Benutzers das einzige Eingabemedium des Systems ist. Die automatische Meßwerterfassung entbindet den Anästhesisten bzw. die Anästhesieschwester von der stereotypen Tätigkeit der Aufzeichnung der Vitalparameterverläufe. Das vom AIS auf einem Digitalplotter erstellte Operationsprotokoll ersetzt das bisher ubliche handgeschriebene Papierprotokoll und ist diesem im Punkte Übersichtlichkeit weit überlegen. Daneben stellt es eine zusätzliche Sicherheit für den Anästhesisten dar, da er bei einem Ausfall des AIS das vom Plotter erstellte Protokoll von Hand weiterführen kann. Als Nachteil wäre der recht große Aufwand an Hardware und Software zu nennen, der im krassen Gegensatz zu der Einfachheit des Systems aus Benutzersicht steht und der eine weite Verbreitung zumindest in der geschilderten Konfiguration schon aus Kostengründen bis auf weiteres verhindert. Allerdings ist zu betonen, daß das AIS nicht entwickelt wurde, um den Anästhesisten ein neues Werkzeug in die Hand zu geben. Der Zweck war und

[2] s.a. Bild 1.1-1

ist vielmehr die Erforschung von Methoden, die die Arbeit des Anästhesisten bei Operationen erleichtern.

2. Aufgaben eines Entscheidungsunterstützungssystems in der Anästhesie

In dem vorigen Kapitel wurde das Umfeld vorgestellt, in dem diese Diplomarbeit angesiedelt ist. In der Darstellung des Aufgabenbereichs des Anästhesisten wurde beispielhaft der grobe Ablauf einer Herzoperation aufgeführt und ein Teil der Meßwerte und Symptome aufgezählt, die die Grundlage für die Entscheidungen bilden, die er im Laufe der Operation zur Aufrechterhaltung der Lebensfunktionen des Patienten trifft. Das am Helmholtz-Institut entwickelte Anästhesie-Informationssystem (AIS) wurde aus der Sicht des Benutzers und aus der Sicht der Implementierung betrachtet. Im Rahmen dieser Diplomarbeit soll nun ein Programm entwickelt werden, das das AIS um entscheidungsunterstützende Funktionen erweitert. In diesem Kapitel sollen die Möglichkeiten untersucht werden, die eine solche Unterstützung im Hinblick auf die Arbeit des Anästhesisten und unter der Randbedingung der Integrierung in das AIS bieten kann.

Wie aus der Ausführung des Themas in der Einleitung zu entnehmen ist, soll das zu implementierende Entscheidungsunterstützungssystem datengesteuert arbeiten. Dies bedeutet, daß das Entscheidungsunterstützungssystem Daten (etwa Werte für Parameter) erhält und daraus entsprechende Schlußfolgerungen (etwa Diagnosen) zieht. Die Benutzerschnittstelle soll weiterhin der Farbbildschirm sein. Bild 2-1 zeigt schematisch die Situation, die später gewissen Einfluß auf die Implementierung haben wird.

Eine übliche Beschreibung [ZIM 84] von Entscheidungssituationen ist die Angabe von (Welt-)Zuständen, möglichen Aktionen (Handlungen) und einer Abbildung von dem kartesischen Produkt dieser beiden Mengen in eine Menge von "Ergebnissen". Auf der Ergebnismenge kann man eine reellwertige Nutzenfunktion definieren, die es über dem Zustands- und Aktionenraum zu optimieren gilt, und an die im allgemeinen gewisse axiomatische Forderungen gestellt wird. Ein ahnlicher Ansatz findet sich auch in der Spieltheorie [RSZ 79]. Unsicherheiten uber den Zustand der Welt und das Eintreten von Ergebnissen werden durch Zufallsvariablen modelliert. Einer Übertragung dieser Methode auf die Entscheidungsunterstützung des Anästhesisten stehen eine Reihe von Hinderungsgründen entgegen:

1. Der (Welt-)Zustand wäre im wesentlichen mit dem Zustand des Patienten gleichzusetzen, und dieser ist unvergleichlich komplex zu dem, was sonst mit der oben umrissenen Methode behandelt wird.
2. Viele Therapiemaßnahmen (Aktionen) des Anästhesisten sind symptomatisch orientiert, der tatsächliche Patientenzustand ist i. a. nicht in vollem Umfang bekannt.
3. Die Wahl der Nutzenfunktion ("Verbesserung des Patientenzustandes") ist unklar.
4. Der Anästhesist trifft seine Entscheidungen im Operationssaal unter Zeit-

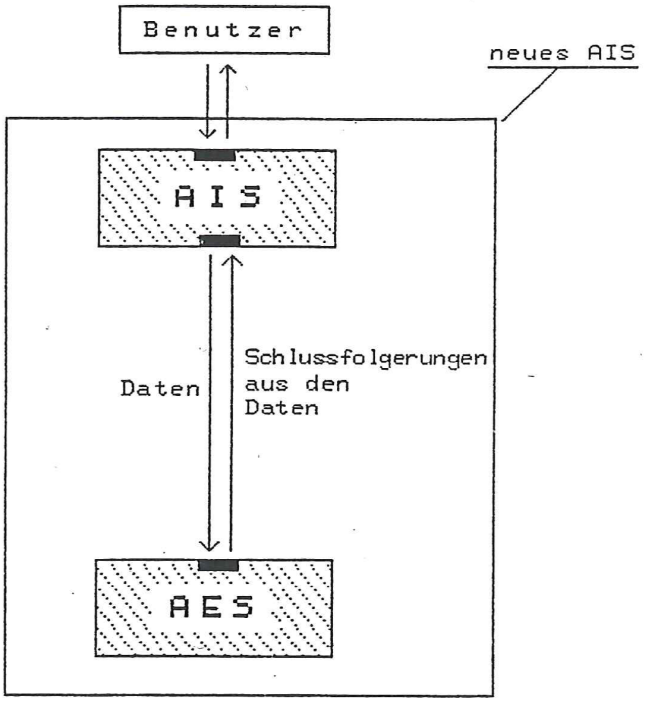

Bild 2-1: Stellung des AES zum AIS

druck, eine rechnergestützte Entscheidungshilfe müßte in kurzer Zeit Ergebnisse liefern, die für den Anästhesisten nachvollziehbar sind.

5. Das Entscheidungsverhalten ist von Anästhesist zu Anästhesist verschieden und ändert sich mit dem Fortschritt in der Medizin.

Es erscheint also nicht angemessen, ein exaktes Verfahren zu wählen, das dem Anästhesisten die "optimale" Entscheidung in einer gegebenen Situation (Patientenzustand) vorschlägt. Insbesondere ist es schwierig, wenn nicht unmöglich, dem Entscheidungsunterstüzungssystem alle Daten verfügbar zu machen, die für eine "optimale" Entscheidung erforderlich sind. Stattdessen wird sich das System, das hier hinsichtlich Entwurf und Realisierung vorzustellen ist, mit einem niedrigeren Anspruch begnügen müssen. Zu nennen sind hier das Stellen von einfachen Diagnosen und Therapievorschlägen, die Beobachtung von Vitalparameterverläufen sowie Rechenhilfen für den Anästhesisten. Das Konzept muß sich dabei an dem durch das AIS gegebene Umfeld orientieren.

Wie Bild 2-1 zeigt, wird das Entscheidungsunterstützungssystem (AES) seine Ausgaben (Diagnosen u. a.) allein aufgrund der vom AIS übertragenen Daten und des in ihm kodierten medizinischen Wissens treffen. In Kapitel 1.1. wurde bereits eine Auswahl dieser Daten angegeben. Man kann zwischen folgenden Daten

unterscheiden:

(a) *Vitalparameter*
- *Blutdruck, Puls, Temperatur, ZVD, LAP*

(b) *Laborwerte*
- z. B. *pH, BE, PO_2*

(c) *Geräteeinstellungen*
- z. B. *Beatmungsfrequenz, Minutenvolumen* beim Respirator

(d) *Patientenparameter*
- *Alter, Geschlecht, Gewicht, Größe, Körperoberfläche, Blutgruppe*

(e) *Verabreichungen von Medikamenten (Therapiemaßnahmen)*
- z. Zt. sind dem AIS etwa 200 Medikamente bekannt, bei Herzoperationen werden ca. 25 verschiedene Medikamente verwandt

(f) *Ereignisse*
- z. B. *Operationsbeginn, Intubation*

(g) *Weitere Meßwerte*
- z. B. *Maximaldruck* (angezeigt am Respirator)

Mit Ausnahme der automatisch erfaßten Vitalparameter werden alle Eingaben über die Benutzerschnittstelle des AIS vorgenommen (vgl. Kap. 1.2.). Die Häufigkeit der Eintragungen kann man am in Kapitel 1.1. abgebildeten Anästhesie-Protokoll ersehen (Bild 1.1-1).

Insbesondere bei den Meßwerten (Vitalparameter und Laborwerte) ist zu beachten, daß sie in dem Kontext des Meßzeitpunktes stehen. Das heißt praktisch, daß sie i. a. nur für eine gewisse Zeitspanne als gültig anzusehen sind. Spätestens beim Neueintrag eines Parameterwertes ist der alte Wert ungültig. Am deutlichsten wird dieser Zusammenhang bei den Vitalparametern. Sie werden vom AIS automatisch erfaßt und etwa alle zwei Minuten eingetragen. Diese dynamische Situation wird die Auswirkung haben, daß jede Ausgabe des AES zunächst nur für den Zeitpunkt ihrer Berechnung gilt. Neue Eintragungen von Daten durch das AIS können eine Situation schaffen, in der vorher gemachte Ausgaben *jetzt* falsch werden.

Es ist nicht zu erwarten, daß der Anästhesist während der Operation Zeit hat für einen intensiven Dialog mit dem System, um ihm Informationen einzugeben, die es nicht aus den standardmäßig eingetragenen Daten herleiten kann. Der Gebrauchswert des AES wird umso höher sein, je mehr Daten automatisch erfaßt und vom AES überwacht werden.

In Verbindung mit der Analyse des Umfeldes im vorigen Kapitel lassen sich folgende Spezifikationen für das Entscheidungsunterstützungssystem aufstellen:

(S1) Das Entscheidungsunterstützungssystem (AES) muß mit den Daten auskommen, die vom Anästhesisten bzw. der Anästhesieschwester zur Erstellung des Narkoseprotokolls sowieso über das AIS eingetragen werden.

(S2) Die Entscheidungshilfen des AES müssen, wenn sie Sinn haben sollen, in kurzer Zeit berechnet werden.

(S3) Jede Entscheidungshilfe muß begründbar und für den Anästhesisten nachvollziehbar sein.

(S4) Die Darstellung der Entscheidungshilfen erfolgt auf dem Bildschirm des AIS, jede Interaktion (z. B. zur Erklärung einer Entscheidungshilfe) mit dem AES erfolgt ebenfalls über die Benutzerschnittstelle des AIS.

Die Grundlage für die Entscheidungshilfen des Systems bildet medizinisches Wissen. In den folgenden Kapiteln wird zunächst ein Konzept für die Modellierung dieses Wissens zu erarbeiten sein, das es erlaubt schematische Schlußfolgerungsmechanismen darauf anzuwenden und somit die Grundlage für die Implementierung des Systems auf einem Rechner bildet. Die oben aufgeführten Spezifikationen werden Einfluß sowohl auf das Konzept als auch auf die Umsetzung desselben in Form der Implementierung haben. Punkt (S4) betrifft die Integration in das Anästhesie-Informationssystem (AIS) und aus Sicht des Entscheidungsunterstützungssystems eher eine technische Randbedingung. Sie bedeutet, daß ein Teil der Programmierarbeit im Rahmen dieser Arbeit in der Implementierungssprache des AIS – nämlich PASCAL – geleistet werden muß. Punkt (S1) drückt noch einmal aus, warum ein exaktes Verfahren mit einer "optimalen" Lösung hier nicht möglich erscheint. Die Effizienz (S2) wird am Schluß dieser Arbeit zu überprüfen sein. Es ist angestrebt, daß die *Wissensrepräsentation*, d. h. die Sprache, in der das Wissen formuliert wird, so allgemeinverständlich gehalten wird, daß der *Experte* (Anästhesist, Mediziner) selbst es in den Rechner eingeben kann *(Wissensakquisition)*, ohne Kenntnis von den genauen Abläufen zu haben, wie sie vom Rechner weiterverarbeitet und interpretiert wird. Die hier verwendeten Begriffe *Wissensrepräsentation*, *Wissensakquisition* und *Experte* stammen aus der Begriffswelt der *Künstlichen Intelligenz* [RIC 84] und spiegeln die Aufgaben wieder, die bei solchen Systemen immer wieder auftreten, die einen Experten auf seinem eigenen Gebiet unterstützen sollen, in dem eine exakte Lösung des Problems nicht bekannt ist oder als nicht praktikabel angesehen wird.

3. Lösungskonzept

Die Modellierung des medizinischen Wissens hat der Tatsache Rechnung zu tragen, daß Mediziner mit Rechnern und Programmiersprachen i. a. wenig vertraut sind. Der Idealfall der Wissensbeschreibung in natürlicher Sprache verbietet sich in unserem Fall wegen der inherenten Zweideutigkeiten und Kontextabhängigkeiten, die eine schematische Verarbeitung auf dem Rechner äußerst schwierig macht; tatsächlich ist das Verstehen natürlicher Sprache ein komplexes Forschungsgebiet innerhalb der *Künstlichen Intelligenz* [RIC 84] und noch weitgehend ungelöst. Als Alternative zur natürlichen Sprachen bieten sich formale Sprachen an, die eine strenge Form (Syntax) haben, und deren Bedeutung (Semantik) man von Zweideutigkeiten freihalten kann. Stoyan [STO 85] definiert die Entwicklung von Expertensystemen als Aufstellung der Syntax einer Sprache und die Implementierung eines *Interpreters*, der diese Sprache versteht. Es ist anzustreben, daß sich der Benutzer (Anästhesist) ein einfaches Modell von der Arbeitsweise dieses Interpreters machen kann, so daß er die Ergebnisse des Expertensystems nachvollziehen kann. Bild 3-1[3] zeigt den Zusammenhang graphisch.

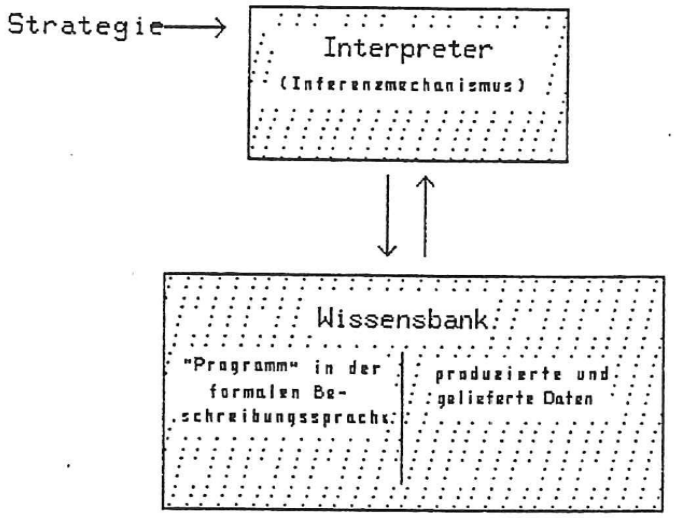

Bild 3-1: ein Modell für Expertensysteme

Man beachte die Ähnlichkeit mit Programmiersprachen. Die Bedeutung eines Programms kann u. a. mit der *operationellen Semantik* [IND 83] definiert werden, was nichts anderes als die Angabe eines abstrakten Interpreters für diese Sprache ist. Übersetzung von Programmen ist die Transformation des Programmtextes in

[3] nach Stoyan

einen Text einer anderen Sprache unter Bewahrung der Semantik. An unterster Stufe steht die Maschinensprache, die von der Zentraleinheit des Rechners direkt interpretiert wird.

Das hier zu entwickelnde Konzept zur Beschreibung von medizinischem Wissen im Bereich der Anästhesie und dessen schematische Auswertung auf dem Rechner wird sich an den Spezifikationen aus dem vorigen Kapitel orientieren. Die Beschreibung wird von einer formalen Sprache übernommen werden, und die Angabe der Bedeutung (Semantik) dieser Sprache wird gleichzeitig den Weg zu einer rechnergestützten Auswertung (Interpretation) weisen. Die Form der Beschreibungssprache hat beträchtlichen Einfluß auf die schon angesprochenen Aspekte der Wissensrepräsentation und Wissensakquisation. Es ist anzustreben, die Formulierung des Wissens von den technischen Aspekten der späteren Auswertung (*Kontrollstruktur*) freizuhalten.

Viele Systeme zur rechnergestützten Verarbeitung von Expertenwissen basieren auf einem *regelorientierten* Ansatz, der im folgenden Unterkapitel erläutert werden wird. Von grundlegender Bedeutung für wissensbasierte Systeme ist die Prädikatenlogik. Sie ist geeignet, alle Objekte, die hier von Interesse sein können (d. h. alle berechenbaren Objekte) zu beschreiben. Als Beispiel soll das Programm VM von Fagan ([FAG 80], [FAG 84]) den regelorientierten Ansatz erläutern. Außerdem sind von diesem System einige Anregungen für unser Problem zu erwarten, da es für eine ähnliche Anwendung, nämlich für die Überwachung der maschinellen Beatmung eines Patienten auf der Intensivstation, konzipiert wurde. Nach diesen Vorarbeiten kann das Lösungskonzept vorgestellt werden.

3.1. Regelorientierte Systeme

In den letzten Jahren ist eine unübersehbare Anzahl von Systemen entwickelt worden, denen alle das Ziel gemeinsam ist, die Arbeit eines (menschlichen) Experten auf einem klar umgrenzten Gebiet nachzubilden. Diverse Beispiele finden sich in Bereichen der medizinischen Diagnose, der Datenanalyse, des Entwurfs (z. B. von elektrischen Schaltungen), der Überwachung (etwa in Kernkraftwerken), der Planung (z. B. der Steuerung von intelligenten Robotern) und vielen weiteren Gebieten[4]. Meist werden solche Programme als *Expertensysteme* tituliert, womit i. a. eine klare Trennung von (dargestelltem) Wissen und Kontrollstruktur des Programmes, leichte Erweiterbarkeit und Modifizierbarkeit der Wissensbasis und das Vorhandensein einer Erklärungskomponente für die Ausgaben des Programms suggeriert werden soll. Inhalt dieses Kapitels ist die Vorstellung eines Konzeptes,

[4] eine recht ausführliche, allerdings leicht veraltete und auf den amerikanischen Markt bezogene Liste findet sich in dem Übersichtsartikel von Gevarter [GEV 83]

das sich bei der Entwicklung von Expertensystemen und vergleichbarer Systeme in der *Künstlichen Intelligenz* als sehr erfolgreich erwiesen hat: der *regelorientierte* Ansatz. Grundbegriffe wie *Graph, Knoten, Pfeil, Baum, Vorgänger, Abkömmling* usw. und grundlegende Algorithmen aus der *Graphentheorie* werden als bekannt vorausgesetzt. Für eine exakte Definition verweise ich auf das Vorlesungsskript von Möhring [MÖH 84].

Komponenten und Algorithmen. Nilsson [NIL 82] unterscheidet drei Hauptkomponenten, die sich in den meisten sog. *regelorientierten Systemen* finden:

(A) *der Zustand (globale Datenbasis),*
(B) *die Produktionsregeln* und
(C) *das Kontrollsystem.*

Der *Zustand* steht für die Problembeschreibung bzw. für den aktuellen Stand des Problemlösungsprozesses. *Produktionsregeln* bestehen aus einem Bedingungsteil (Test auf den Zustand) und einem Aktionsteil (Veränderung des Zustandes). Das *Kontrollsystem* legt die Abfolge der Anwendungen der Produktionsregeln fest insbesondere, wenn mehrere Produktionsregeln zutreffen. Die Komponenten *Zustand, Produktionsregeln* und *Kontrollsystem* werden hier nicht formal definiert; ihre Darstellung ist von Programm zu Programm verschieden. Vielmehr soll das Konzept vorgestellt werden, das der Klasse der regelorientierten Programme zugrunde liegt.

Ausgehend von diesen Begriffen stellt Nilsson den folgenden grundlegenden Algorithmus für regelorientierte Systeme auf:

```
Procedure PRODUCTION
1 DATA ← Anfangszustand
2 Wiederhole bis DATA die Endbedingung erfüllt:
3   begin
4     Wähle eine Regel R aus der Menge der Regeln
          die auf DATA anwendbar ist
5     DATA ← Ergebnis der Anwendung von R auf DATA
6   end
```

Mit *DATA* wird die aktuelle globale Datenbasis (Zustand) bezeichnet. Dieser Zustand wird mit einem Anfangszustand initialisiert. Dann werden solange die Produktionsregeln angewendet, bis der aktuelle Zustand eine *Endbedingung* erfüllt. Die Kriterien der Regelauswahl bleiben in dem nicht-deterministischen Algorithmus ungenannt. Sie werden in der Kontrollstruktur festgelegt. Grundsätzlich fordert man von einer Kontrollstruktur, daß sie zu einer Lösung (= Weg zu einem Zustand, der die Endbedingung erfüllt) führt, falls eine solche Lösung existiert.

Nilsson definiert auf der Grundlage der obigen Begriffe und des Algorithmus einen *Suchgraphen*, dessen Knoten mit den Zuständen (*globale Datenbasen*) und dessen Pfeile mit den Produktionsregeln identifiziert werden. Zwei Knoten sind

durch einen Pfeil verbunden, wenn eine Produktionsregel den einen korrespondierenden Zustand in den anderen überführt. Ausgezeichnet sind der Startknoten (Anfangszustand) und die Menge der Endknoten (Zustände, die die Endbedingung erfüllen). Gesucht ist ein Weg von dem Startknoten zu einem Endknoten. Die Suche in einem Graphen nach einem Weg zwischen zwei Knoten ist ein wohluntersuchtes Problem der Graphentheorie (vgl. Möhring [MÖH 84]). Allerdings werden üblicherweise nur endliche Graphen betrachtet, d. h. Graphen mit endlicher Knoten- und Pfeilmenge. Das folgende Beispiel eines regelorientierten Systems zeigt, daß wir es i. a. mit unendlichen Graphen zu tun haben:

Es sei

$$N = \{0, 1, 2, \ldots\}$$

die *Menge aller Zustände*,

$$P = \{p1 : (n \to n+1), p2 : (n \to n+2)\}$$

die (zweielementige) Menge der *Produktionsregeln* und

$$Z_0 = 0$$

der Anfangszustand. Die Menge der *Endzustände* sei

$$F = \{1\}.$$

Der Bedingungsteil der Produktionsregeln ist in diesem Fall leer. Sie sind so zu lesen, daß ihre Anwendung auf den Zustand $n \in N$ zu dem Zustand $n+1 \in N$ bzw. $n+2 \in N$ führt. Die einmalige Anwendung von $p1$ führt sofort zum Endzustand, während die Anwendung von $p2$ zum Zustand 2 führt, von wo aus es keinen Weg zurück nach 1 mehr gibt. Eine Suchstrategie (Kontrollstruktur), die etwa besagt, daß immer $p2$ angewendet werden soll, wenn beide Regeln zutreffen, wird zu einem nicht terminierenden Programm führen. Bild 3.1-1 zeigt dies an dem zugehörigen Suchgraphen.

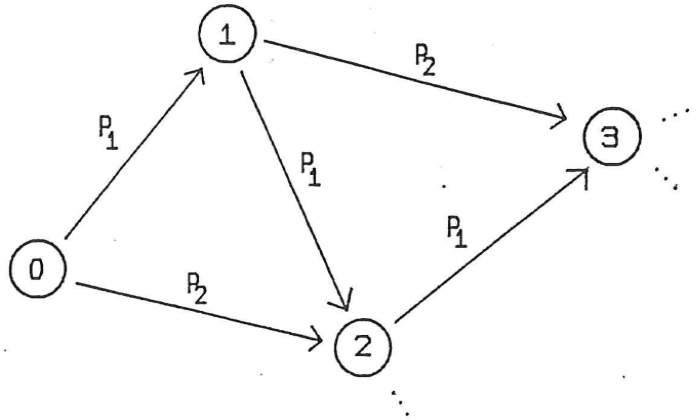

Bild 3.1-1: Graph zum Beispiel

In der Graphentheorie werden zwei Typen von Suchstrategien unterschieden:

1. *Depth-first-Suche* und
2. *Breadth-first-Suche.*

Möhring [MÖH 84] gibt die Grundideen der Algorithmen wie folgt an (für endliche Graphen):

Depth-first-Suche

- Wähle Anfangsknoten a, dann besuche einen Knoten b adjazent zu a, dann c adjazent zu b (und verschieden zu a), dann einen noch unbesuchten Knoten d adjazent zu c usw.
- Irgendwann erreicht man einen Knoten, dessen Nachbarn bereits alle besucht worden sind $\Rightarrow$ gehe zurück zum letzten besuchten Knoten und mache dort weiter
- Stoppe, falls Anfangsknoten wieder erreicht

Breadth-First-Suche

- Wähle einen Anfangsknoten und schreibe ihn auf eine (zunächst leere) Warteliste
- Nehme den ersten Knoten von der Warteliste, besuche alle seine Nachbarn und schreibe sie in der Reihenfolge der Besuche an das Ende der Warteliste
- Verfahre entsprechend mit dem jeweils ersten Knoten der Warteliste bis diese abgearbeitet ist

Bei unendlichen Graphen wird die *Depth-first-Suche* nicht notwendig einen Weg von dem Anfangsknoten (Anfangszustand) zu einem Endknoten (Endzustand) finden, falls dieser Weg existiert. Die *Breadth-first-Methode* garantiert dagegen das Auffinden eines existierenden Weges.

Die *Kontrollstruktur* regelorientierter Systeme legt die verwendete Suchmethode fest. Große Systeme, bei denen in jedem Zustand i. d. R. viele Produktionsregeln anwendbar sind, erfordern, um handhabbar zu bleiben, effizientere Suchmethoden. Meist ist man nicht nur an irgendeiner Lösung interessiert, sondern an einer Lösung mit geringsten "Kosten". Dazu definiert man auf der Pfeilmenge eine reellwertige *Kostenfunktion.* Die *Kosten* eines Weges sind als Summe der Kosten der enthaltenen Pfeile definiert. Bei Ansetzen von Einheitskosten für jeden Pfeil hat ein kostengünstigster Weg eine minimale Anzahl von Pfeilen, d. h. die Lösung im Algorithmus hat eine minimale Anzahl von Regelanwendungen im Algorithmus.

Heuristische Suche. Nilsson gibt unter dem Begriff *Heuristische Suche* eine allgemeine Vorgehensweise an, die in dem folgenden Algorithmus (aus [NIL 82]) dargestellt ist.

Procedure **GRAPHSEARCH**

1 Bilde einen *Suchgraphen* G, der nur aus dem Anfangsknoten s besteht. Setze s auf eine Liste *OPEN.*

2 Bilde eine Liste *CLOSED*, die zunächst leer ist.

3 *LOOP:* wenn *OPEN* leer, beende mit Mißerfolg
4 Wähle ersten Knoten von *OPEN*, entferne ihn von *OPEN* und schreibe ihn auf *CLOSED*. Nenne diesen Knoten n
5 Wenn n Endknoten, beende mit Erfolg und Lösung, die man durch Rückverfolgung der Zeiger von n nach s erhält (vgl. Schritt 7).
6 Expandiere den Knoten n durch Generieren einer Menge M, die alle Nachfolger von n enthält, die nicht Vorgänger von n sind. Erweitere den Suchgraphen G um diese Nachfolger von n.
7 Errichte von allen Elemente von M, die vorher noch nicht in G waren (d. h. entweder auf *OPEN* oder *CLOSED*), einen Zeiger nach G. Erweitere die Liste *OPEN* um diese Elemente. Entscheide für jedes Element aus M, das schon auf *OPEN* oder *CLOSED* war, ob sein Zeiger nach n umgeleitet werden soll oder nicht. Für jedes Element von M, das schon auf *CLOSED* war, entscheide für jeden seiner Abkömmlinge[5] in G, ob sein Zeiger umgeleitet werden soll oder nicht.
8 Sortiere die Liste *OPEN* nach einem gewissen *heuristischem* Kriterium.
9 Gehe nach *LOOP*

Neben dem *Suchgraphen* G baut der Algorithmus einen *Suchbaum* T auf, der durch die in Schritt 7 aufgestellten Zeiger definiert wird. Durch das Umleiten von Zeigern in Schritt 7 wird sichergestellt, daß im Fall des Auffindens einer Lösung (Schritt 5) diese Lösung geringste Kosten hat: Wenn ein Knoten erneut besucht wird, so wird der Zeiger von diesem Knoten umgeleitet, falls der neue Weg geringere Kosten hat. Die Liste *OPEN* enthält die Blätter des sukzessiv aufgebauten Suchbaums, die noch nicht *expandiert* wurden, während *CLOSED* die Nicht-Blätter bzw. die Blätter, die beim Expandieren keine Nachfolger hatten, beinhaltet.

Je nach Kriterium in Schritt 8 kann man unterschiedliche Suchverfahren erhalten. Ein Sortieren der Liste *OPEN* absteigend nach der Tiefe der enthaltenen Knoten im Suchbaum führt zur *Depth-first-Suche*, eine aufsteigende Anordnung resultiert in der *Breadth-first-Methode*. Nilsson nennt die Suchverfahren nach solchen willkürlichen Kriterien *uninformiert* im Gegesatz zu *heuristischen* Verfahren. Bei solchen Verfahren wird aus der Liste *OPEN* jeweils der "vielversprechendste" Knoten als nächster zu expandierender Knoten ausgewählt. Das Kriterium wird definiert durch eine sogenannte *Auswertungsfunktion*, die jedem Knoten des Suchgraphen eine reelle Zahl zuordnet. Im Idealfall ist der Wert der Funktion für einen Knoten n aus gleich der Summe der Kosten eines kostengünstigsten Weges vom Anfangsknoten s nach n und eines kostengünstigsten Weges von n zu einem Endknoten. Im allgemeinen hat man allerdings nicht genügend Information für die exakte Bestimmung der Auswertungsfunktion und ist deshalb auf eine Schätzung angewiesen.

[5] unmittelbarer oder mittelbarer Nachfolger

Im *GRAPHSEARCH*-Algorithmus kann für die Schätzung des kostengünstigsten Weges zu den Kandidatenknoten aus *OPEN* der aktuelle kostengünstigste Weg herangezogen werden. Die Schätzung der Wegekosten von den Kandidatenknoten nach einem Endknoten ist weit schwieriger, da dieser Teil des Graphen noch nicht besucht wurde und i. a. ja noch nicht einmal bekannt ist, ob ein solcher Weg existiert. Hier spielt *Heuristik* die wesentliche Rolle.

Die Implementierung eines regelorientierten Systems mit einer Kontrollstruktur die sich an *GRAPHSEARCH* anlehnt, wird u. U. aus Effizienzgründen nicht testen, ob ein durch Expansion erhaltener Knoten schon besucht wurde (Knoten entsprechen *Zuständen*, und diese können eine komplexe Struktur haben), so daß von dieser Seite weitere Probleme auftauchen. Üblicherweise wird in den Knoten nicht jedesmal der zugehörige *Zustand* abgespeichert, sondern nur seine *Differenz* zum Anfangszustand. Bei Verwendung einer Auswertungsfunktion ist darauf zu achten, daß die Zeit zur Berechnung der Funktionswerte nicht den dadurch erreichten Effizienzgewinn wieder wettmacht.

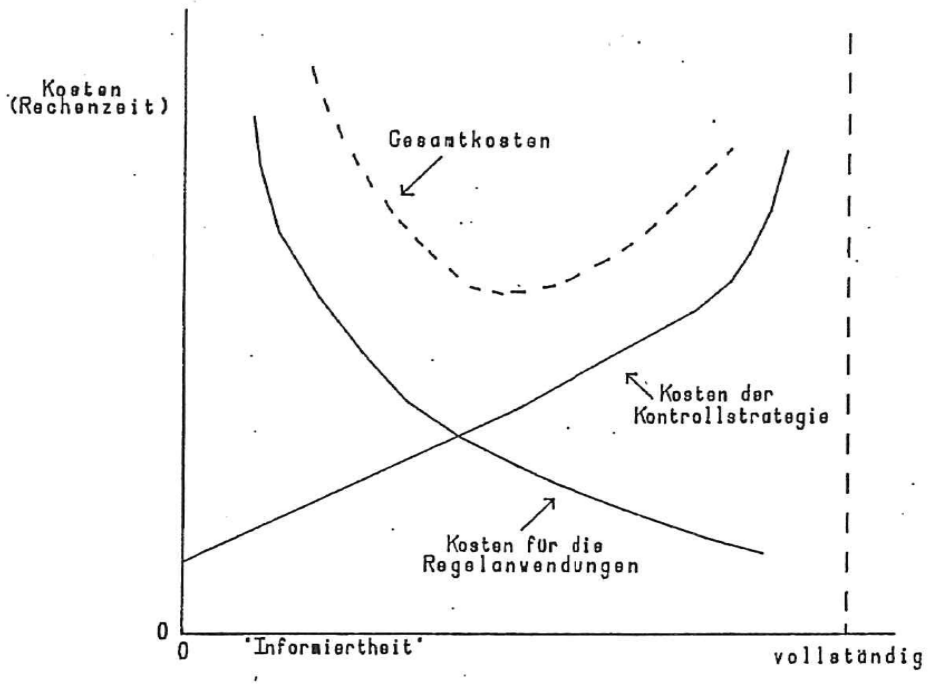

Bild 3.1-2: Zeitverhalten von regelorientierten Systemen

In Bild 3.1-2 (nach [NIL 82]) ist der Zusammenhang graphisch verdeutlicht. Je mehr Aufwand in die Beschaffung von Information über die "beste" anwendbare Produktionsregel gesteckt wird, desto geringer sind die Kosten der Regelanwendungen. Uninformierte Kontrollstrategien (Suchmethoden) haben i. a. hohe Kosten für die Regelanwendungen, da sie dazu neigen, Teile des Suchgraphen abzuarbeiten, die für die spätere Lösung nicht gebraucht werden. Total informierte Suchmethoden wenden immer die richtige Regel an, allerdings auf Kosten einer i.

a. teuren Informationsbeschaffung. Der richtige Weg — aus der Sicht der heuristischen Suche — liegt zwischen den Extremen und muß von Problem zu Problem neu ermittelt werden.

In der Praxis unterscheidet man oft zwischen *vorwärtsverkettenden* und *rückwärtsverkettenden* Systemen. *Vorwärtsverkettung* bedeutet das Ausgehen vom Anfangszustand (Startknoten) und die Anwendung der Produktionsregeln "von links nach rechts", bis ein Endzustand (Endknoten) erreicht wird. In *rückwärtsverkettenden* Systemen geht man von einem Endzustand aus und wendet die Produktionsregeln umgekehrt "von rechts nach links" an, bis der Anfangszustand erreicht wird. Die Unterschied zwischen den beiden Vorgehensweisen ist allerdings nur ein inhaltlicher und kein formaler, da sie durch Umbezeichnung ineinander uberführbar sind. In Expertensystemen wird die Menge der Produktionsregeln auch als *prozedurale Wissensbasis* bezeichnet.

Als Komplexitätsmaße für regelorientierte Systeme bieten sich Maße auf dem Suchbaum an. Nilsson gibt zwei Maße an:

(1) *Penetranz P*
$P = L/T$, wobei L die Länge des Weges zum Endknoten und T die Gesamtzahl der aufgesuchten Knoten

(2) *effektiver Verzweigungsfaktor B*
Verzweigungsfaktor (Anzahl der Nachfolger eines Knotens) des vollständig verzweigten Baumes, dessen Tiefe gleich der Länge des Weges zum Endknoten ist, und der genausoviel Knoten hat wie bei der Lösungssuche besucht wurden: $B + B^2 + \cdots + B^L = T$

Beide Maße können i. a. erst nach einer Berechnung angegeben werden.

Zusammenhänge. Wie oben erwähnt spielt der regelorientierte Ansatz eine wichtige Rolle in der Künstlichen Intelligenz. Das Konzept ist jedoch auch auf andere Gebiete anwendbar bzw. mit den dort angewandten Konzepten vergleichbar.

Ganz analoge Begriffe und Problemstellungen gibt es in der *Theorie der Automaten und formalen Sprachen* (vgl. [OBE 83]) und deren Anwendung in der *Syntaxanalyse* beim *Compilerbau* (vgl. [IND 85]). Den *Ersetzungsregeln* in *Chomsky-Grammatiken* entsprechen die *Produktionsregeln*, *Satzformen* entsprechen *Zuständen*[6] und das *Startsymbol* entspricht dem *Anfangszustand*. Ähnlich läßt sich die Analogie mit den Automaten angeben. Die Vorstellungen von *Vorwärts/Rückwärtsverkettung* finden sich im Compilerbau unter den Namen *Top-Down-* bzw. *Bottom-Up-Verfahren* wieder.

Die Verfahren der heuristischen Suche haben in der *Spieltheorie* (vgl. [RSZ 79]) und im *Operations Research* ([ZIM 78]) die gleiche Bedeutung. Gerade etwa

[6] Manchmal werden die Nicht-Terminalsymbole in Chomsky-Grammatiken auch als "Zustände" bezeichnet. Die Begriffsgleichheit sollte aber zu keiner Verwirrung führen.

bei Schachprogrammen ist man auf gute Heuristiken angewiesen, da der *Spielbaum* extrem stark verzweigt ist und daher ein vollständiges Abarbeiten zum Auffinden einer Gewinnstellung (=Endzustand) unmöglich ist.

Zusammenfassend ist der regelorientierte Ansatz als ein wichtiges Konzept zur Entwicklung von wissensbasierten Systemen zu bezeichnen. Er ermöglicht die Formulierung des Wissens in einer Vielzahl von *Produktionsregel*, die inhaltlich uber ihren Bedingungsteil miteinander vernetzt sein können. Für die Implementierung nach diesem Konzept stehen eine Reihe von Algorithmen, insbesondere aus der Graphentheorie, zur Verfügung.

3.2. Die Prädikatenlogik

Die *Prädikatenlogik* spielt in der *Künstlichen Intelligenz* eine ebenso bedeutende Rolle wie die vorher regelorientierten Systeme. Sie eignet sich besonders für die *formale* Beschreibung von Zusammenhängen. Die Definitionen und Ergebnisse in diesem Kapitel sind vorwiegend den Arbeiten von Richter ([RIC 78] und [RIC 81]) entnommen. Es kann hier keine vollständige Vorstellung des *Prädikatenkalküls* erwartet werden, sondern nur eine Rekapitulierung der wichtigsten Begriffe, die in den späteren Kapiteln benötigt werden. Für weitergehende Fragen sei der Leser auf die obigen Bücher verwiesen.

DEFINITION 1.
Eine (Relational)-Struktur *ist ein Tripel*

$$\underline{A} = < A, < f_i^A \mid i \in I_A >, < R_j^A \mid j \in J_A >> .$$

*Dabei ist A eine nichtleere Menge, I_A und J_A (beliebige) Indexmengen, $f_i^A : A^{n_i} \mapsto A$, $n_i \in N$ und $R_j^A \subseteq A^{m_j}$, $m_j \in N$. A heißt die*Trägermenge *von $\underline{A}$, die f_i^A heißen* Operationen *und die R_i^A* Relationen *von $\underline{A}$.*

DEFINITION 2.
$\sigma(\underline{A}) = << n_i \mid i \in I_A >, < m_j \mid j \in J_A >>$ *heißt* Signatur *von $\underline{A}$.*

Die Signatur $\sigma = << n_i \mid i \in I >, < m_j \mid j \in J >>$ sei im folgenden fest vorgegeben. $Var := \{x_0, x_1, x_2, \ldots\}$ sei eine abzählbare Menge, deren Elemente *Variablen* genannt werden. Für jedes $i \in I$ sei ein Element f_i, für jedes $j \in J$ ein Element P_j gegeben. P_j heißt *Prädikat*, f_i *Funktionszeichen.* Im Falle $n_i = 0$ nennt man f_i *Konstantenzeichen.*

DEFINITION 3.

Terme *sind wie folgt definiert:*

i) alle Variablen und Konstantenzeichen sind Terme;

ii) wenn $t_1, \ldots, t_{n_i}$ Terme sind, so ist auch $f_i(t_1, \ldots, t_{n_i})$ ein Term;

iii) Terme sind nur mit i) und ii) herstellbar.

Setzt man $f_i^T(t_1, \ldots, t_{n_i}) = f_i(t_1, \ldots, t_{n_i})$, so bilden die Terme eine Algebra, die Termalgebra $\underline{T}$. Ausgehend von diesen Termen definiert man die *prädikatenlogischen Formeln* mit den logischen Zeichen (aussagenlogischen Operationen) $\wedge$ ("und"), $\vee$ ("oder"), $\neg$ ("nicht"), $\rightarrow$ ("wenn-so"), $\exists$ ("existiert") und $\forall$ ("für alle"):

DEFINITION 4.

Formeln *sind wie folgt definiert:*

i) wenn P_j ein Prädikat ist und $t_1, \ldots, t_{m_j}$ Terme sind, so ist $P_j(t_1, \ldots, t_{m_j})$ eine Formel, auch Atomformel *genannt;*

ii) sind φ und ψ Formeln, so sind $(\neg\varphi)$, $(\varphi \wedge \psi)$, $(\varphi \vee \psi)$, $(\varphi \rightarrow \psi)$ Formeln, für jede Variable x auch $(\forall x\varphi)$ und $(\exists x\varphi)$.

iii) Formeln lassen sich nur aus i) und ii) gewinnen.

Damit ist die Syntax der Sprache der Prädikatenlogik (erster Stufe) definiert. Zur Definition der Semantik wird im folgenden eine Relationalstruktur

$$\underline{A} = << A, < f_i^A \mid i \in I >, < R_j^A \mid j \in J >>$$

zugrunde gelegt. Man sagt "f_i^A ist die Interpretation von f_i" und "R_j^A ist die Interpretation von P_j". Formal wird dies durch sog. *Belegungs-* und *Auswertungsfunktionen* definiert:

DEFINITION 5.

Sei $\underline{A}$ eine Relationalstruktur; dann ist eine Belegung der Variablen *eine Abbildung $u : Var \mapsto A$.*

Eine Belegung läßt sich eindeutig zu einem *Homomorphismus*

$$\hat{u} : \underline{T} \mapsto < A, < f_i^A \mid i \in I >>$$

auf der Termalgebra fortsetzen. $\hat{u}$ bildet die Grundlage für die *Auswertungsfunktion*, die den Formeln aus der Sprache der Prädikatenlogik die *Wahrheitswerte* 0 oder 1 zuordnet. Sie wird so definiert werden, daß sie der Semantik der **klassischen Logik** entspricht. Zunächst ist allerdings noch eine technische Definition notwendig.

DEFINITION 6.

Seinen u,u' zwei Belegungen $Var \mapsto A$.
Dann bedeutet $u =_{x_k} u'$, daß $u(x_j) = u'(x_j)$ für alle $j \neq k$.

DEFINITION 7.

Sei u eine Belegung in $\underline{A}$ und $\hat{u}$ die homomorphe Fortsetzung auf die Termalgebra. Dann wird die Auswertungsfunktion *v_u von der Menge der Formeln in die Boole'sche Algebra $\{0,1\}$ wie folgt erklärt:*

a) $v_u(P_j(t_1,\ldots,t_{m_j}) = 1$ *genau dann, wenn* $R_j^A(\hat{u}(t_1),\ldots,\hat{u}(t_{m_j}))$;
b) für die aussagenlogischen Operationen wird v_u homomorph fortgesetzt:
$v_u(\neg\varphi) = 1$, *wenn* $v_u(\varphi) = 0$ *und umgekehrt,*
$v_u(\varphi \wedge \psi) = 1$, *wenn* $v_u(\varphi) = 1$ *und* $v_u(\psi) = 1$, *sonst 0,*
$v_u(\varphi \vee \psi) = 1$, *wenn* $v_u(\varphi) = 1$ *oder* $v_u(\psi) = 1$, *sonst 0,*
$v_u(\varphi \rightarrow \psi) = 1$, *wenn* $v_u(\varphi) = 0$ *oder* $v_u(\psi) = 1$, *sonst 0;*
c) $v_u(\forall x_k\varphi) = 1$ *genau dann, wenn* $v_{u'}(\varphi) = 1$ *für alle u' mit* $u =_{x_k} u'$;
d) $v_u(\exists x_k\varphi) = 1$ *genau dann, wenn ein u' mit* $u =_{x_k} u'$ *und* $v_{u'}(\varphi) = 1$ *existiert.*

Man sagt, "*u erfüllt φ (in $\underline{A}$)*", falls $v_u(\varphi) = 1$. φ ist "wahr in $\underline{A}$", falls φ von allen Belegungen erfüllt wird. $\underline{A}$ heißt dann auch "Modell für φ". $\underline{A}$ heißt "Modell für Σ", falls $\underline{A}$ Modell für alle $\varphi \in \Sigma$ ist. φ ist "falsch" in $\underline{A}$, wenn keine Belegung in $\underline{A}$ die Formel φ erfüllt. Des weiteren bezeichnet man Formeln als "Sätze", falls in ihnen alle Vorkommen von Variablen mit $\forall$ bzw. $\exists$ quantifiziert sind.

DEFINITION 8.

Eine Formelmenge Σ impliziert *eine Formel φ* semantisch, *in Zeichen $\Sigma \models \varphi$, falls jedes Modell $\underline{A}$ für Σ auch Modell für φ ist.*

Damit ist der semantische Folgerungsbegriff für die Prädikatenlogik erster Stufe auf der Grundlage der klassischen Boole'schen Algebra eingeführt. Die Auswertungsfunktion v_u beschreibt im aussagenlogischen Teil nichts anderes als die bekannten Wahrheitstafeln. Man zeigt leicht, daß in einer gegebenen Struktur alle Sätze entweder wahr oder falsch sind.

Neben dem semantischen Folgerungsoperator $\models$ definiert man den syntaktischen Ableitungsoperator $\vdash$ durch Angabe eines *Kalküls* in Form von *Axiomen* und *Regeln*. Ein kompletter Kalkül angeben kann [RIC 78] entnommen werden. Beispielhaft seien ein Axiom und eine Regel zitiert:

Axiom (A1)

$$\varphi \rightarrow \varphi$$

Regel (*Modus ponens*)

$$\frac{\varphi,\ \varphi \to \psi}{\psi}$$

Die Regeln eines Kalküls kann man "von oben nach unten" lesen, was im Fall des *Modus pones* heißt, daß man mit den Formeln φ, $\varphi \to \psi$ zur Formel ψ ubergehen darf. Man kann aber auch eine Regel von "unten nach oben" lesen, was als "Suche nach einem Beweis" für eine vorgegebene Formel zu bezeichnen ist. Richter [RIC 78] zeigt für den durch den *Hilberttypkalkül* definierten syntaktischen Ableitungsoperator, daß er korrekt ist, d. h. jede in ihm abgeleitete Formel ist in jeder Struktur $\underline{A}$ wahr. Der *Gödel'sche Vollständigkeitssatz* schließlich besagt, daß der semantische Folgerungsoperator $\models$ und der syntaktische Ableitungsoperator $\vdash$ ubereinstimmen.

Der Wahrheitsbegriff läßt sich also durch rein syntaktische Zeichenersetzungsregeln axiomatisieren. *Automatische Beweisverfahren* bauen auf diesem Ergebnis auf und verwenden spezielle Kalküle, um für eine eingegebene Formel zu entscheiden, ob sie wahr ist oder nicht. Allerdings kann man keine allzu hohen Erwartungen an diese Verfahren knüpfen, wie der folgende Satz besagt.

Satz 1. *Der Prädikatenkalkül ist unentscheidbar.*

Man muß sich also auf entscheidbare Unterklassen von Formeln beschränken und hat es auch hier häufig mit Effizienzproblemen zu tun. Ausdrücklich erwähnt sei hier der *Resolutionskalkül*, der mit einer einzigen Regel, der *Resolutionsregel*, auskommt.

Resolution. Das *Resolutionsprinzip* wurde 1965 von J. A. Robinson formuliert. Es kennt nur eine Regel, die auf einer speziellen Normalform für logische Formeln arbeitet. Dieser kurze Abriß der Resolution orientiert sich an dem Vortrag von Prof. Thomas im Rahmen des PROLOG-Seminars im WS83/84 (vgl. [THF 84]). Zunächst ist die Normalform der sog. *Klausen* zu erklären.

Definition 9.
Klausen *sind Formeln der Gestalt*

$$B_1, \ldots, B_m \leftarrow A_1, \ldots, A_n,$$

wobei die A_i und B_i Atomformeln oder negierte Atomformeln sind.

Eine solche Klause steht als Abkürzung für die logische Formel

$$\forall x_1 \cdots \forall x_k (A_1 \wedge \ldots \wedge A_n \to B_1 \vee \ldots \vee B_m)$$

oder auch

$$\forall x_1 \cdots \forall x_k (B_1 \vee \ldots \vee B_m \vee \neg A_1 \vee \ldots \vee \neg A_n),$$

wobei $x_1, \ldots, x_k$ alle in den A_i, B_i vorkommenden Variablen umfaßt. Die vorkommenden Atomformeln und negierten Atomformeln in der letzteren Darstellung heißen auch *Literale*, manchmal unterschieden in *positive* (für nicht negierte Atomformeln) und *negative* (für negierte Atomformeln) Literale.

Bei $m \leq 1$ spricht man von *Hornklausen*. Die Klause mit $m = n = 0$ heißt *leere Klause* (in Zeichen $\square$) und ist per Konvention in jedem Modell falsch. Der folgende Satz zeigt die Allgemeinheit dieser Normalform.

SATZ 2.

Für jede Formel φ aus der Spache der Prädikatenlogik erster Stufe gibt es eine Menge Φ von Klausen, so daß gilt:

φ erfüllbar $\Leftrightarrow$ Φ erfüllbar.

Die Menge Φ ist berechenbar (unter Verwendung der sog. *Skolemisierung*). Im folgenden werden Klausen in der Form

$$(B_1 \vee \ldots \vee B_m \vee \neg A_1 \vee \ldots \vee \neg A_n)$$

angegeben und mit der Menge

$$\{B_1, \ldots, B_m, \neg A_1, \ldots, \neg A_n\}$$

identifiziert. Alle vorkommenden Variablen sind per Konvention $\forall$-quantifiziert.

Zur Angabe der Resolutionsregel ist noch die Definition des *allgemeinsten Unifikators* notwendig.

DEFINITION 10.

a) *Ein* Unifikator *für zwei Formeln A_1und A_2 ist eine Substitution θ mit $A_1\theta = A_2\theta$*

b) *Ein* allgemeinster Unifikator *zu A_1,A_2 ist ein Unifikator θ_0, für den gilt: falls ein weiterer Unifikator θ für A_1, A_2 existiert, so gibt es ein θ' mit $\theta = \theta_0\theta'$.*

Der allgemeinste Unifikator ist effektiv berechenbar. Ein Algorithmus zur Berechnung wurde zuerst von Robinson (vgl. [HOG 84]) angegeben und später weiter verbessert. Damit läßt sich die Resolutionsregel wie folgt angeben:

$$\frac{C_1, C_2}{(C_1\theta \setminus \{A\theta\}) \cup (C_2\theta \setminus \{\neg B\theta\})},$$

wenn C_1, C_2 Klausen, A kommt in C_1 vor, $\neg B$ kommt in C_2 vor, A und B unifizierbar mit allgemeinsten Unifikator θ.

Zusammenfassung. Die Prädikatenlogik eignet sich als universelle Beschreibungssprache für mathematische Aussagen. Wegen der Übereinstimmung des semantischen Folgerungsoperators und des syntaktischen Ableitungsoperators können Fragen nach dem Zutreffen (Erfüllbarkeit) von Formeln rein syntaktisch formuliert und im Rahmen eines entsprechenden korrekten und vollständigen Kalküls für gewisse entscheidbare Klassen von Formeln beantwortet werden. Die Regeln in einem solchen Kalkül sind vergleichbar mit den *Ersetzungsregeln* der im vorigen Kapitel erwähnten *Chomsky-Grammatiken* und zumindest in der Art der schematischen Anwendung auch mit den *Produktionsregeln* in regelorientierten Systemen, in denen die Frage, ob der Endzustand erreicht werden kann, ebenfalls i. a. unentscheidbar ist.

Die Resolutionsmethode auf der Basis der Normalform der *Klausen* ist eine elegante Methode für ein automatisiertes Beweisverfahren. Unter weiterer Einschränkung der Formeln und mit einer speziellen Suchstrategie fand sie Eingang die Programmiersprache PROLOG, die bei Abhandlung der Implementierung des AES noch näher zu beschreiben sein wird.

3.3. Ein Beispielsystem: VM

In den vorigen Kapiteln wurden zwei theoretische Konzepte vorgestellt, die immer wieder Eingang in Systeme finden, die mit Expertenwissen manipulieren. Es soll nun ein System vorgestellt werden, das sowohl einen gewissen Einblick in die Verwirklichung solcher Konzepte gibt, als auch von der Problemstellung her uber eine starke Verwandtschaft mit unserer Aufgabe verfügt.

Das System heißt VM (*Ventilator Manager*) und wurde Ende der siebziger Jahre von L. Fagan ([FAG 80], [FAG 84]) entwickelt. Es hat die Aufgabe, die maschinelle Beatmung von Patienten in der Intensivstation zu überwachen, die am Herzen operiert wurden, und das Pflegepersonal durch entsprechende Hinweise auf unvorhergesehene Ereignisse aufmerksam zu machen.

Eine wesentliche Teilaufgabe ist es dabei, Fehler der Meßgeräte und des Lebenserhaltungssystems zu erkennen und zu melden. VM ist so angelegt, daß es praktisch ohne Interaktion mit dem Benutzer arbeitet.

Fagan faßt die Aufgaben von VM wie folgt zusammen:

- Meß- und Gerätefehler erkennen,
- ungünstige Ereignisse im Mensch-Maschine-System erkennen,
- den physiologischen Zustand des Patienten zusammenfassen,
- Vorschläge zur Anpassung der Therapie gemäß dem aktuellen Patientenzustand und den therapeutischen Langzeitzielen machen und
- eine Menge von patientenspezifischen Erwartungen und Zielen für die spätere Auswertung durch das Programm führen.

Der letzte Punkt ist eher programmtechnischer Natur. Die Ausgaben von VM basieren auf ca. 30 Meßwerten vom Patienten bzw. vom Respirator und werden auf einem normalen Terminal neben dem Patienten dargestellt.

Das medizinische Wissen von VM wird in Form der im Kapitel 3.1 vorgestellten Produktionsregeln kodiert und vom MYCIN-Inferenzmechanismus[7] interpretiert. MYCIN ist Mitte der siebziger Jahre von E. Shortliffe an der Stanford Universität entwickelt worden und dient zur Diagnose- und Therapiehilfe bei Infektionskrankheiten. Die benötigten Daten werden in einem Frage/Antwortspiel (*Konsultation*) vom Arzt eingegeben. Das medizinische Wissen von MYCIN ist in Form von ca. 400 Produktionsregeln abgespeichert.
Fagan betont zwei wesentliche Unterschiede seiner Anwendung zu MYCIN:

- ▷ VM soll in einer dynamischen Umgebung arbeiten, d. h. der Kontext, in dem die zu interpretierenden Daten zu sehen sind, ändert sich ständig.
- ▷ Es soll kein bzw. nur ein minimaler Dialog mit dem Benutzer (Pflegepersonal) stattfinden.

Beide Unterschiede haben Einfluß auf die Implementierung. Punkt 2 bedeutet, daß die Schlußfolgerungen von VM allein aufgrund der Meßwerte getroffen werden. Kontextspezifische Information wie etwa die Beatmungsart muß VM aus diesen Daten folgern. Das oberste therapeutische Ziel ist, den Patienten schrittweise von der maschinellen Beatmung zu lösen. Dazu kann der Respirator je nach Patientenzustand auf verschiedene Arbeitsweisen eingestellt werden:

(1) *kontrollierte zwangsweise Beatmung*
- dem Patienten wird in festen Intervallen Luft in die Lunge gepreßt

(2) *assistierende Beatmung*
- der Respirator preßt Luft in die Lunge, wenn der Patient versucht einzuatmen

(3) *T-Stück-Beatmung*
- die mechanische Beatmung ist ausgeschaltet, aber der Patient bleibt weiter intubiert

(4) *Extubation*
- die Beatmungsschläuche werden entfernt, der Patient atmet sebständig

In der Regel kommt der Patient in der Beatmungsart (1) vom Operationssaal in die Intensivstation. Bei Verbesserung kann auf die jeweils nächste Beatmungsart umgeschaltet werden. Eine Verschlechterung des Patientenzustandes muß die evtl. vorher vorgenommene Lockerung der Beatmungsart wieder zurückgenommen werden. VM empfiehlt eine solche Veränderung, das Bedienungspersonal muß sich allerdings nicht an eine solche Empfehlung halten. Die aktuelle Beatmungsart leitet VM selbständig aus den automatisch erfaßten Meßwerten ab.

[7] dieser wird auch EMYCIN genannt

Implementierung. Eine wichtige Eigenschaft von VM ist die *symbolische* Behandlung von Meßwerten. Den Patientenparametern etwa werden die Charakterisierungen "ideal", "annehmbar", "keinesfalls annehmbar" und "unmöglich" zugeordnet. Diesen Namen entsprechen Bereiche auf dem Wertebereich der Meßwerte. Bild 3.3-1 zeigt den Zusammenhang dieser Intervalle. Die Charakterisierung "unmöglich" bedeutet, daß ein Meß- oder Gerätefehler vorliegt.

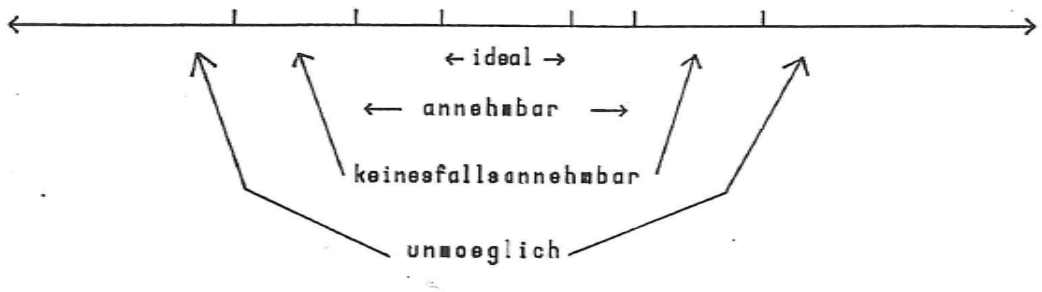

Bild 3.3-1: Bereiche für Parameter

VM ordnet jedem neuankommenden Meßwert eine solche symbolische Charakterisierung gemäß den aktuellen Bereichsgrenzen zu, und arbeitet dann mit diesen Namen anstatt mit den Originaldaten. Die Bereichsgrenzen können sich aufgrund etwa einer Veränderung der Beatmungsart verschieben, ohne daß sich deshalb die Produktionsregeln, die mit den Bereichen arbeiten, ungültig werden. Fagan nennt die Bereiche *Erwartungen*: die Erwartung, in welchem Bereich ein Meßwert in der nächsten Zeit liegen wird. Die Veränderung der Bereichsgrenzen wird durch Produktionsregeln vorgenommen.

Die folgende Produktionsregel (aus [FAG 80]) zeigt dies beispielhaft:

```
 INITIALIZING-RULE: INITIALIZE.V-RETURN
Definition:  INITIALIZE RETURN TO VOLUME ventilation
APPLIES-TO: VOLUME
IF
  M: PATIENT TRANSITIONED FROM T-PIECE TO VOLUME
THEN SYSTEM ASSUMES
  S: PATIENT STARTING CONTROLLED VOLUME VENTILATION
  C:                    [---- Acceptable range------]
  C:           very              [-- ideal --]          very
  C:           low      low      min      max      high   high
  C:           ----      ----     ----     ----     ----   ----
  E: SYS       +       + 110    +       +       + 150  +      +
  E: DIA       +       + 60     +       +       + 95   +      +
  E: MAP       + 60    + 75     + 80    + 95    + 110  + 120  +
  E: HEART RATE+       + 60     +       +       + 110  +      +
```

...

Die Produktionsregeln von VM haben ein festes Format. Die erste Zeile ordnet die Regel einem Typ oder einer Klasse zu. Die zweite Zeile (`Definition`) enthält in Freitext die Bedeuutung der Regel. Die dritte Zeile (`APPLIES-TO`) bezeichnet den Kontext, in dem die Regel anzuwenden ist. Hier ist der Kontext "volumengesteuerte Beatmung", das ist Beatmungsart (2). Das Schlüsselwort "`IF`" leitet den Bedingungsteil (Prämisse) der Regel ein. Im Bedingungsteil findet ein Test auf den *Zustandsraum* statt (vgl. Kap. 3.1). Jeder Einzelbedingung wird ein "`M:`" vorangestellt. In unserem Fall gibt es nur eine Einzelbedingung. Einzelbedingungen in der Prämisse können durch "und", "oder" sowie durch weniger naheliegende Operatoren wie z. B. "mindestens 2 Einzelbedingungen treffen zu" verknüpft werden. Der "dann"-Teil (Aktionsteil) der Regel besteht zunächst aus einem *Statement*, dessen Text auf das Terminal ausgegeben wird. Darauf folgen 4 Kommentarzeilen. Die Bereichsdefinitionen werden durch die mit "`E:''` beginnenden Zeilen festgelegt Die Originalregel in VM hat noch 10 weitere Bereichsdefinitionen, die hier weggelassen sind.

Offenbar wird durch diese Regel der Übergang von Beatmungsart (3) nach (1) beschrieben. Der Aktionsteil verändert den Zustandsraum des Systems dahingehend, daß die Zahlenwerte für die symbolischen Bereiche der Meßwerte aktualisiert werden. Als Seiteneffekt wird eine Ausgabe auf das Terminal veranlaßt ("`PATIENT STARTING CONTROLLED VOLUME VENTILATION`"). Ein zweites Beispiel zeigt den Aufbau einer sogenannten Statusregel:

```
 STATUS-RULE: Status.Hypertension
Definition:  Identify Hypertension
APPLIES-TO: VOLUME CMV ASSIST T-PIECE
IF
  M: MAP IS HIGH
  M: SYS IS HIGH
THEN
  I: THERE IS HYPERTENSION
  S: Hypertension
```

Diese Regel erkennt Bluthochdruck. Wenn der mittlere Blutdruck (MAP) und der systolische Blutdruck hoch sind, dann wird die Interpretation "`HYPERTENSION`" in den Zustandsraum eingetragen und "`Hypertension`" auf das Terminal ausgegeben. Daneben gibt es noch sog. *Transitionsregeln* zur Feststellung des aktuellen Therapiezustandes (insbesondere Beatmungsart), *Therapieregeln* zur Ausgabe einer Therapieempfehlung und *Instrumentenregeln* zur Erkennung von Meßfühler- und Gerätefehlern. Diese Kategoriesierung der Produktionsregeln ist zum einen ein willkommenes Hilfsmittel zur Strukturierung, zum anderen bietet sie die Möglichkeit zur Einschränkung des Suchraumes: wenn Therapieempfehlungen berechnet werden sollen, müssen nur die Therapieregeln betrachtet werden.

Die genaue Syntax der Produktionsregeln ist hier nicht von Interesse. Sie werden von einem Übersetzungsprogramm in ein *internes* Format für VM umgewandelt.

Der generelle Ablauf eines Ableitungszyklus in VM ist so:

① Validierung der Daten
② Kontextbestimmung
③ Setzen der Erwartungen
④ Bestimmung des physiologischen Status
⑤ Therapieempfehlungen

Für jeden Schritt gibt es eine eigene Kategorie von Produktionsregeln. Der Zyklus wird bei jedem Eintrag neuer Meßwerte neu angestossen.

Wie oben erwähnt, baut VM auf dem Inferenzmechanismus von MYCIN auf. Es kann hier nicht der Aufbau des Zustandsraums von VM in allen Einzelheiten beschrieben werden. Klar ist aber, daß zunächst alle aktuellen Meßwerte abgespeichert sind. Fagan gebraucht den Begriff *Parameter* und meint damit einmal diese Meßwerte, aber insbesondere auch aus diesen Meßwerten abgeleitete Parameter, z. B. die oben erwähnte "Hypertension". Falls "Hypertension" abgeleitet wurde, so wird für diesen Parameter der Wert "`PRESENT`" im Zustandsraum abgespeichert. Später aufgerufene Regeln können den Wert von "Hypertension" dann mit der Bedingung "`HYPERTENSION IS PRESENT`" abfragen. Man beachte, daß dies die einzige Art der inhaltlichen Verkettung von Produktionsregeln ist: Produktionsregeln testen in ihrem Bedingungsteil den Zustandsraum (d. h. Abfragen auf abgespeicherte Daten) und Verändern im Aktionsteil den Zustandsraum (d. h. Abspeichern neuer Daten).

Das Kontrollsystem arbeitet mit folgenden Listen, die für jeden Parameter vorliegen (nicht alle Listen gleichzeitig):

- ▷ Liste der Namen der Regeln, in denen dieser Parameter benutzt wird
- ▷ Liste der Namen der Regeln, in denen dieser Parameter berechnet wird
- ▷ Liste der Namen der Regeln, in denen Erwartungen für diesen Parameter berechnet werden
- ▷ Zeitspanne, in der dieser Parameter als gültig anzusehen ist
- ▷ Zeit, zu der die letzte Berechnung des Parameters gemacht wurde

Die Listen werden von dem oben erwähnten Übersetzungsprogramm berechnet. Aufbauend auf diesen Daten sind eine Reihe von Suchstrategien denkbar. Fagan gibt als Stichworte *Vorwärtsverkettung* (vgl. Kap. 3.1) und *datengesteuerter Inferenzmechanismus* an. Diese Sprechweise ist inhaltlich dadurch gerechtfertigt, daß dem System Meßwerte eingegeben werden und daraus die ableitbaren Parameter berechnet werden. Die Steuerung erfolgt über die oben aufgeführten Listen. Eine ausgeklügelte Suchstrategie findet in VM keine Verwendung, da die Anzahl der Produktionsregeln recht klein ist. Fagan berichtet von etwa 60 Regeln, die in jedem Zyklus angestoßen werden.

Zusammenfassung. VM ist ein Programm, daß eine klar umgrenzte Aufgabe hat: die Überwachung der maschinellen Beatmung von Patienten auf der Intensivstation. Seine Hilfen werden auf einem Terminal ausgegeben und gründen allein auf den automatisch erfaßten Meßwerten.

Fagan stellt die integrierende Funktion von VM heraus. Während der Arzt bzw. das Pflegepersonal ohne VM eine Vielzahl von Meßwerten zu beobachten haben, die zudem in verschiedenen Sitationen unterschiedliche Bedeutung haben können, übernimmt VM die Interpretation der Daten auf niedriger Ebene und meldet nur "interessante" Ereignisse. Leider kann hier nicht über praktische Erfahrungen mit VM berichtet werden, da das System bis 1980, das ist der in [FAG 80] beschriebene Stand, noch nicht zum praktischen Einsatz kam. VM ist eindeutig den in Kapitel 3.1 erklärten regelorientierten Systemen zuzurechnen. Ein Vergleich mit den in Kapitel 2 aufgestellten Anforderungen an das AES zeigt eine recht große Übereinstimmung mit VM. Hier sind die dynamische Situation und das datengesteuerte Arbeiten zu nennen. Die symbolische Verarbeitung der Parameterbereiche verdient besondere Beachtung. Durch sie wird es möglich, von einem "hohen Blutdruck" oder einem "normalen Puls" zu sprechen. Dies kommt der Denkweise der medizinischen Experten, mit denen zusammen die Wissensbasis als Ansammlung von Produktionsregeln formuliert wurde, sehr entgegen. Außerdem erleichtert diese Einrichtung das Ziel, die Produktionsregeln möglichst allgemein zu halten.

3.4. Eine Beschreibungssprache für Anästhesiewissen: AES/L

Die voranstehenden Kapitel leisteten einige Vorarbeit für die Entwicklung des Lösungskonzeptes. Die regelorientierten Systeme sind ein weit verbreiteter Ansatz, um Expertenwissen zu formalisieren und auf dem Rechner auszuführen. Das Programm VM zeigt die Anwendbarkeit auf eine Problemstellung, die mit der unseren sehr verwandt ist. Auf der anderen Seite ist die Prädikatenlogik eine Sprache, die sich zur Beschreibung von Zusammenhängen bewährt hat, und deren Bedeutung in den Grundzügen allgemein bekannt ist. Beiden Ansätzen ist gemeinsam, daß sie sich zur Formulierung von Wissen und dessen Auswertung auf einem *Interpreter* (Inferenzmechanismus, Kalkül) eignen.

Dieses Kapitel beinhaltet die erste Grundentscheidung auf dem Weg zur Lösung der gestellten Aufgabe:

Das medizinische Wissen wird in einer eigenständigen formalen Sprache kodiert.

Diese Sprache soll unabhängig von der späteren Implementierung und weitgehend selbsterkärend sein. Zur Rechtfertigung dieser Entscheidung kann man

anführen:

- ▷ für die spätere Eingabe des medizinischen Wissens muß der Benutzer nur die Syntax und Semantik *dieser* Sprache kennen,
- ▷ die Sprache legt fest, was überhaupt mit dem Entscheidungsunterstützungssystem geleistet werden kann,
- ▷ Vorhersagen über die Auswirkung von in der Sprache formulierten Sätzen können ohne Wissen über die tatsächliche Auswertung gemacht werden und
- ▷ die Sprache kann an die Denkweise der Mediziner angepaßt werden.

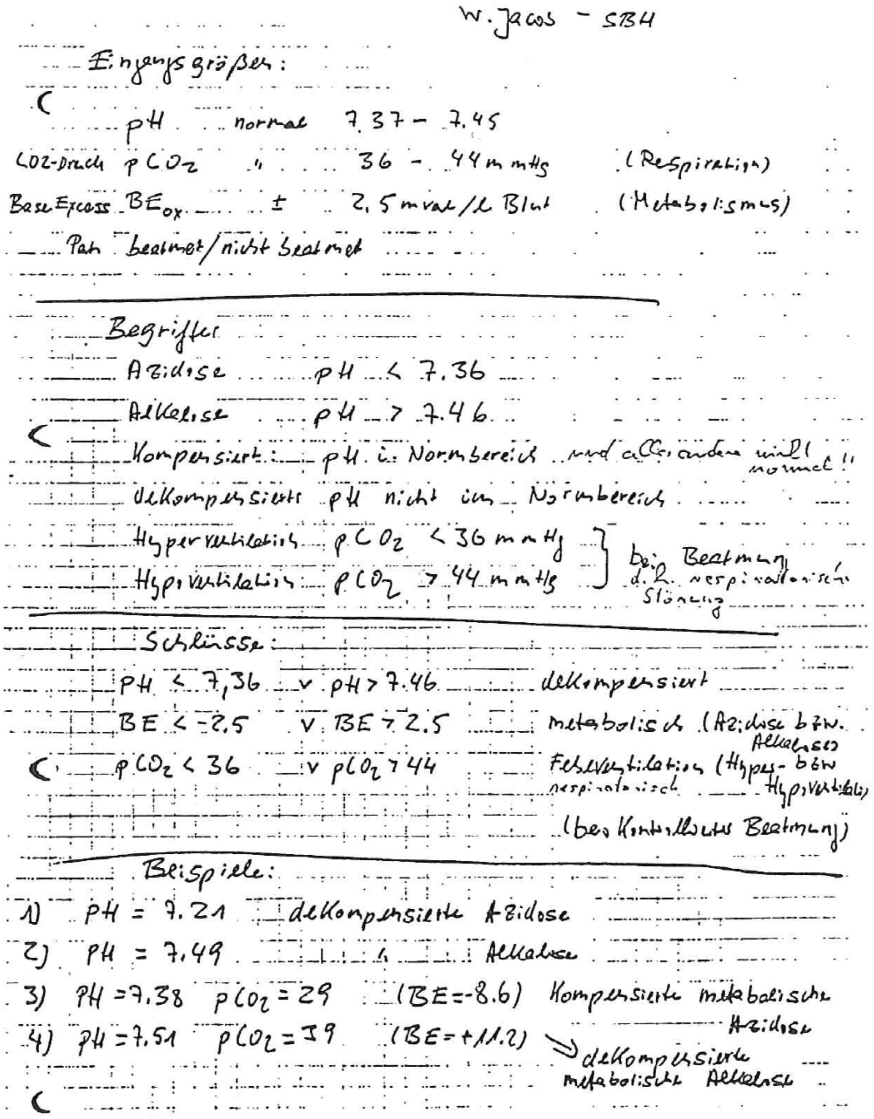

Bild 3.4-1: Regeln für den Säurestatus

Bild 3.4-1 ist eine Kopie der ersten Formulierung medizinischen Wissens von

Wolfgang Jacob über den Säurestatus des Blutes (vgl. [JAC 86]). Die Bedeutung der verwendeten Begriffe pH, PCO_2 und BE kann dem Glossar (Kap. 8) entnommen werden. Am Anfang werden Normalbereiche für die drei Meßwerte definiert. Davon ausgehend werden die Begriffe *Azidose, Alkalose* usw. erklärt. Man spricht also von einer Azidose, falls der pH-Wert im Blut kleiner als 7.36 ist. Für eine Formalisierung der Beschreibung gibt es eine Fülle von Möglichkeiten. Wenn nur solche einfachen Zusammenhänge beschrieben werden sollen, reicht die folgende Sprache:

Azidose: $pH < 7.36$
Alkalose: $pH > 7.46$
usw.

Offenbar können mit einer solchen Sprache lediglich intervallische Zusammenhänge beschrieben werden. Man kann aber schon jetzt die Beobachtung machen, daß die in Bild 3.4-1 enthaltene Information in der Form "*Aussage* trifft zu, wenn *Bedingung* erfüllt ist" darstellbar ist.

Dies motiviert motiviert das erste Konstrukt unserer Sprache, die im folgenden AES/L heissen soll und anhand des obigen Beispiels informell entwickelt wird:

wenn *Bedingung* dann *Folgerung*.

Beispeilsweise kann man damit die Definition der Azidose ausdrücken:

wenn $pH < 7.36$ dann "Azidose".

In der Vorstellung des VM-Systems wurde bereits die Nützlichkeit von der Verwendung von symbolischen Namen für Intervalle erwähnt. Tatsächlich taucht dieses Konzept in Bild 3.4-1 wieder auf, und zwar für die Normalbereiche der drei Laborwerte. Es erscheint also angemessen, dafür ein Konstrukt in AES/L vorzusehen:

Parameter ist *Eigenschaft* innerhalb A bis B.

Damit läßt sich unser Beispiel eleganter formulieren:

pH ist normal innerhalb 7.36 bis 7.45.[8]
wenn pH ist unter normal dann "Azidose".

Die Formalisierung von z. B. "kompensierte metabolische Azidose" bereitet noch Schwierigkeiten, da hier offenbar mehrere Bedingungen erfüllt sein müssen. Der Operator "und" leistet hier Abhilfe:

wenn pH ist normal und PCO_2 ist normal und
BE ist unter normal dann "kompensierte metabolische Azidose".

[8] die Zahl 7.37 in Bild 3.4-1 ist falsch

Analog ist der Operator "oder" zu verwenden. Damit ist die Grundidee von AES/L informell eingeführt. Die Anlehnung an die natürliche Sprache suggeriert bereits die beabsichtigte Bedeutung, die allerdings erst im nachfolgenden Kapitel definiert werden wird.

Syntax. Die formale Definition der Syntax kann im Appendix (Kap. 7) nachgelesen werden. An dieser Stelle werden die Konstrukte der Sprache weiterhin anhand von Beispielen vorgestellt. Die gelegentliche Andeutung der Bedeutung geschieht aus Gründen der Motivation. Das Einrücken der Satzbestandteile in den Beispielen ist keine Vorschrift der Sprache, es soll vielmehr die Struktur der Sätze betonen.

Zunächst ist anzumerken, daß die obigen *Regeln* für "Azidose" und "kompensierte metabolische Azidose" in AES/L mit einem *Regeltyp* und einer Nummer versehen werden, etwa:

```
diagnoseregel 1 heisst
   wenn
     pH ist unter normal
   dann
     "Azidose".
```

Der Regeltyp ist in diesem Fall also "diagnoseregel" und die Nummer "1". In AES/L gibt es noch die Regeltypen "therapieregel" und "hilfsregel". Als Nummer ist jede natürliche Zahl zugelassen. Wie man der formalen Definition der Syntax in Kapitel 7 ersieht, unterscheiden sich *Diagnoseregeln, Therapieregeln* und *Hilfsregeln* prinzipiell nur durch den vorangestellten Regeltyp. Einer Bereichsdefinition wie oben für den *pH*-Wert eingeführt, kann auch eine Bedingung vorangestellt werden:

```
wenn
   'Alter' =< 40
dann
   'Puls' ist akzeptabel innerhalb 70 bis 140.

wenn
   `Alter' > 40
dann
   `Puls' ist akzeptabel innerhalb 80 bis 120.
```

Offenbar wird hier der Bereich "akzeptabel" für den Parameter 'Puls' in Abhängigkeit vom Patientenalter verschieden definiert. Die Bedingung besteht jeweils aus einem einzelnen Vergleich. Weitere in AES/L zugelassene Vergleichsoperatoren sind "=", "\ =", "<" und ">=". Im Beispiel wird der Parameter 'Alter' mit einer Zahl verglichen. Im Allgemeinfall kann auf der rechten Seite eine komplexe *Formel* stehen. Formeln werden mit Zahlen, Parametern und Variablen unter Verwendung der arithmetischen Operatoren "+", "−", "∗", "/" und "^" (*Potenz*) sowie der Funktionsnamen "exp" (Exponentialfunktion),

“ln” (natürlicher Logarithmus), “log” (Zehnerlogarithmus), “log2” (Zweierlogarithmus), “sqrt” (Quadratwurzel) und “abs” (Absolutbetrag) gebildet. Das folgende Beispiel zeigt die Verwendung von Formeln und führt ein neues Konstrukt von AES/L ein:

```
wenn
   'Geschlecht' = "weiblich"
dann
   'O2-Grenzwert' := 108.86 − 0.26 ∗ 'Alter'
                     −7.30 ∗ ('Gewicht')/('Laenge' − 1) − 15.10.
```

Zunächst sieht man, daß Parameter nicht nur mit *Zahlen* verglichen werden können, sondern auch mit Zeichenketten. Das Konstrukt selbst dient der Definition des *Hilfsparameters* ‘O2-Grenzwert’. Der Name ‘O2-Grenzwert’ kann wie alle anderen Parameternamen verwendet werden, also auch in anderen *Sätzen* auftauchen. Das nächste Beispiel enthält die vorhin erwähnten *Variablen*.

```
wenn
   'Blutdruck' = A:B
dann
   'syst.Blutdruck' = A.
```

Variablen beginnen in AES/L mit einem Großbuchstaben oder dem Zeichen “_”. Die folgende Therapieregel soll die Verwendung von Argumenten in der *Folgerung* und den Gebrauch von Folgerungen in einer Bedingung erläutern:

```
therapieregel 12 heisst
   wenn
      "Hypoventilation" und
      'AZV' ist unter 'im AZVmax-Bereich'
   dann
      "Minutenvolumen erhoehen auf ", 'AZVmax'∗'Frequenz'.
```

Bisher war die *Folgerung* eine mit ‘"’ umrahmte Zeichenkette. Jetzt können wir an den Anfang, an das Ende und in der Mitte Argumente in Form von Formeln einfügen. Man erkennt, daß die “Hypoventilation”, die z. B. mit einer Diagnoseregel eingeführt sein mag, auf einfache Weise in einer Bedingung auftauchen kann. Weitere wichtige Konstruktionen sind die “zeitlichen Aussagen” im Bedingungsteil:

```
diagnoseregel 24 heisst
   wenn
      'mittl.Blutdruck' ist ueber 'normal fuer HLM' seit 5 min
   dann
      "mittl.Blutdruck ist zu hoch".
```

Analog können auch Aussagen über den zeitlichen Verlauf von Folgerungen formuliert werden, etwa:

```
... "Azidose" seit 25 min ...
```

Ähnlich wie Hilfsparameter werden *Dosierempfehlungen* geschrieben:

```
empfohlen 'EVIPAN' := 5 * 'Gewicht'.

wenn
   'Plat.druck' =< 20 und 'Rect.Temp.'  =< 37.5
dann
   empfohlen 'Min.Vol.'  := 0.0672 * 'KO' * 'EV' +
          'Frequenz' * (0.05 * 'KO' + 0.10).
```

Der letztere Satz zeigt an, daß Dosierempfehlungen auch Bedingungen vorangestellt sein können.

AES/L ermöglicht die Formulierung von *Gültigkeitsdauern* und *Ungültigkeitsdefinitionen*, womit der dynamischen Situation Rechnung getragen wird:

```
'Blutdruck' ist gueltig für 3 min.

'BE' ist gueltig bis veraendert.

wenn
   'Geschlecht' = "maennlich" und
   "Hypoventilation" seit 10 min
dann
   'PO2' ist gueltig fuer 'Alter'/50 min.

'PO2' ist ungueltig
   wenn 'Frequenz', 'Min.Vol.'  eingetragen wird.
```

Damit sind alle Konstrukte von AES/L anhand von Beispielen eingeführt. Die meisten Beispiele sind der Datei 'aesreg.reg' im Anhang entnommen.

Der oben mehrfach zitierte Begriff *Parameter* wird syntaktisch exakt in Kapitel 7 definiert. Er ist entweder ein sogenannter *Atomname* oder ein Atomname, der dem Wort "gesamtmenge" folgt. Die Bedeutung von Parametern wird ausführlich im nächsten Kapitel erläutert, doch sei an dieser Stelle darauf hingewiesen, daß sich die in einem *Satz* der Sprache AES/L vorkommenden Parameter eindeutig bestimmen lassen (s. Kap 3.5).

In Richtung auf die spätere Verwendung wird ein Text in AES/L *Wissensbasis* genannt. Die Menge der *Folgerungen* ist als Menge aller aus dem Nichtterminalsymbol $<Folgerung>$ ableitbaren Zeichenketten definiert. Analog kann man von der Menge der *Formeln*, der Menge der *Bedingungen*, der Menge der *Variablen* usw. sprechen.

Bei der Beschreibung der Implementierung wird eine zusätzliche semantische Randbedingung an Texte in AES/L gestellt werden. Dieses Vorgehen kann in Analogie zur Übersetzung von Programmiersprachen (s. [IND 85]) gesehen werden: nicht jeder Programmtext, der unbeanstandet durch die Phase der syntaktischen Analyse läuft, besteht auch die semantische Analyse.

3.5. Die Interpretation von AES/L

Bei der Einführung der Sprache AES/L anhand von Beispielen wurde Betonung auf die suggerierte Bedeutung der Sprachkonstrukte gelegt. Einige Teile der Sprache sind hinsichtlich ihrer Bedeutung aber noch unklar geblieben. Der Sinn erschließt sich erst, wenn ihre Beziehung zum Umfeld offengelegt ist.

Am Anfang dieses Kapitels wird daher noch einmal rekapituliert, welche Zusammenhänge der realen Welt mit AES/L beschrieben werden sollen, und welche Rolle das AIS in dieser Sichtweise spielt. Danach wird die Darlegung der *Semantik* von AES/L keine großen inhaltlichen Schwierigkeiten mehr bereiten.

Die Welt des AES. In Kapitel 2 wurden die im AIS eingetragenen Daten in die Kategorien *Vitalparameter, Laborwerte, Geräteeinstellungen, Patientenparameter, Medikamentenverabreichungen, Ereignisse* und *weitere Meßwerte* eingeteilt. Diese Daten decken einen guten Teil der Informationen ab, die der Anästhesist zum Treffen seiner Entscheidungen heranzieht. Ab jetzt werden alle Mitglieder der Kategorien wieder einheitlich betrachtet und *Welt-Parameter* genannt. Die wichtigste gemeinsame Eigenschaft der Parameter ist ihre Quantifizierbarkeit, d. h. es kann ihnen ein (meist reeller) Wert zugeordnet werden. Beispiele für *Welt-Parameterwerte* sind:

$$
\begin{aligned}
Puls &= 107.4 \\
Blutdruck &= 140:85 \\
Geschlecht &= männlich \\
VALIUM &= 10 \\
Minutenvolumen &= 6.8
\end{aligned}
$$

Die physikalischen Einheiten der Welt-Parameter sind weggelassen. Die Beispiele zeigen einen verschiedenen Grad der zeitlichen Veränderung. Während *Puls* und *Blutdruck* sich ständig ändern, bleibt das *Geschlecht* des Patienten immer gleich. Die Verabreichungsmenge des Medikamentes *VALIUM* ist i. a. von Verabreichung zu Verabreichung verschieden. Die Einstellung des *Minutenvolumens* am Respirator wird nach Bedarf dem Patientenzustand angepaßt.

Grundsätzlich muß man also davon ausgehen, daß sich die Welt-Parameterwerte mit der Zeit ändern. Eine Beurteilung des Patientenzustandes ist also im zeitlichen Zusammenhang der Welt-Parameterwerte zu sehen, sie kann veraltet sein, wenn sich ein in sie eingehender Welt-Parameterwert ändert.

Die Welt des AES ist demnach die Zusammenfassung aller *aktuellen* Welt-Parameterwerte. Die Einschränkung des AES auf quantifizierbare Welt-Parameter, genauer: auf die im AIS eingetragenen Welt-Parameter, bewirkt eine Einschränkung der möglichen Entscheidungshilfen.

Die Semantik von AES/L. Die Verwendung der Partikel "und", "oder" und "wenn ... dann ..." in AES/L hat dem Leser wahrscheinlich schon die beabsichtigte Bedeutung offenbart:

Die Sätze von AES/L sind als prädikatenlogische Formeln zu lesen.

Es reicht damit, die genaue *Lesart* zu definieren und eine *Relationalstruktur* für AES/L anzugeben.

Die Lesart wird durch Angabe einer Transformation der AES/L-*Sätze* in prädikatenlogische Formeln definiert. Die Sprache der Prädikatenlogik sei wie in Kapitel 3.2 definiert. Sie enthalte zusätzlich die Funktionzeichen für die arithmetischen Operationen ("+", "−", "∗", "/", "^") sowie die Funktionszeichen "exp", "ln", "log", "log2", "sqrt" und "abs". Des weiteren seien das Gleichheitsprädikat "=" und die Vergleichsprädikate "<", ">", "≤", "≥" fest vorgegeben. Als Konstantenzeichen (0-stellige Funktionszeichen) seien die in AES/L ableitbaren Parameter- und Bereichsnamen sowie die Namen *in, unter, über, nicht-unter, nicht-über* enthalten. Schließlich seien das "immer wahre" nullstellige Prädikat *true* und das zweistellige Prädikat *apw* für den "aktuellen Parameterwert" vorgegeben. Die Bedeutung von *apw* wird später erläutert.
Ferner seien

$$
\begin{aligned}
L &= \textit{Menge der prädikatenlogischen Formeln} \\
T &= \textit{Menge der Terme von L (vgl. Kap. 3.2)} \\
Var &= \textit{Menge der Variablen von L } (x_1, x_2, \ldots) \\
Pn &= \textit{Menge der Prädikatsnamen von L} \\
T \supseteq LFor &= \textit{Menge der arithmetischen Ausdrücke in L} \\
S &= \textit{Menge der Sätze von AES/L} \\
Rt &= \textit{Menge der Regeltypen in AES/L} \\
Fol &= \textit{Menge der Folgerungen in AES/L} \\
Num &= \textit{Menge der Regelnummern in AES/L} \\
V &= \textit{Menge der Variablen in AES/L} \\
For &= \textit{Menge der Formeln in AES/L} \\
Bed &= \textit{Menge der Bedingungen in AES/L} \\
Par &= \textit{Menge der Parameter in AES/L.} \\
Ber &= \textit{Menge der Bereichsnamen in AES/L.}
\end{aligned}
$$

Die Hilfsfunktionen $\underline{e}$ und $\underline{v}$ sollen die Definition der Transformation erleichtern und sind wie folgt definiert:

$$\underline{e} : S \mapsto Pot(Par)$$

$\underline{e}(x)$ ist die Menge der in x vorkommenden Parameter

$\underline{v} : Par \cup V \mapsto Var$

$\underline{v}(x)$ sei ein jeweils neuer Variablenname

Die Hilfsfunktionen $\underline{transapw}$, $\underline{transf}$ und $\underline{te}$ sind wie folgt erklärt:

$\underline{transapw} : Pot(Par) \mapsto L$
$\underline{transapw}(\{p_1, \ldots, p_n\}) = apw(p_1, \underline{v}(p_1)) \wedge \ldots \wedge apw(p_n, \underline{v}(p_n))$
$\underline{transapw}(\emptyset) = true$

$\underline{transf} : Fol \mapsto Pn$ injektiv

$\underline{te} : For \mapsto LFor$
formt die Formeln aus AES/L in entsprechende Ausdrücke der Sprache L um, indem es die vorkommenden Parameter P_i durch ihre Variablennamen $\underline{v}(P_i)$ und die arithmetischen Operatoren/Funktionen durch ihre "Spiegelbilder" in L ersetzt

Die gesuchte Transformation wird jetzt induktiv über den syntaktischen Aufbau in AES/L definiert. Alle Variablen in den prädikatenlogischen Formeln sind als $\forall$-quantifiziert anzusehen. Die nachfolgend verwendeten Metavariablen stehen für

P : Parametername aus Par
B : Bereichsname aus Ber
F, F_1, F_2 : Formel aus For
$BED, BED1, BED2$: Bedingung aus Bed
$FOLG$: Folgerung aus Fol
Y : Variable aus V
D : Regeltyp aus Rt
Nr : Regelnummer aus Num.

$\underline{trans} : S \mapsto L$

$x =$ D Nr heisst wenn BED dann FOLG
$\Rightarrow \underline{trans}(x) = \underline{transapw}(\underline{e}(x)) \wedge \underline{trans}(BED) \rightarrow \underline{transf}(FOLG, D, Nr)$

$x =$ P ist B innerhalb F_1 bis F_2
$\Rightarrow \underline{trans}(x) = \underline{transapw}(\underline{e}(x)) \rightarrow ber(P, B, \underline{te}(F_1), \underline{te}(F_2))$

$x =$ wenn BED dann P ist B innerhalb F_1 bis F_2
$\Rightarrow \underline{trans}(x) = \underline{transapw}(\underline{e}(x)) \wedge \underline{trans}(BED) \rightarrow ber(P, B, \underline{te}(F_1), \underline{te}(F_2))$

$x = \quad P := F$
$\Rightarrow \underline{trans}(x) = \underline{transapw}(\underline{e}(x)) \rightarrow apw(P, \underline{te}(F))$

$x =$ wenn BED dann $P := F$
$\Rightarrow \underline{trans}(x) = \underline{transapw}(\underline{e}(x)) \wedge \underline{trans}(BED) \rightarrow apw(P, \underline{te}(F))$

$x =$ empfohlen $P := F$
$\Rightarrow \underline{trans}(x) = \underline{transapw}(\underline{e}(x)) \rightarrow empf(P, \underline{te}(F))$

$x =$ wenn BED dann empfohlen $P := F$
$\Rightarrow \underline{trans}(x) = \underline{transapw}(\underline{e}(x)) \wedge \underline{trans}(BED) \rightarrow empf(P, \underline{te}(F))$

$x =$ BED1 und BED2
$\Rightarrow \underline{trans}(x) = \underline{trans}(BED1) \wedge \underline{trans}(BED2)$

$x =$ BED1 oder BED2
$\Rightarrow \underline{trans}(x) = \underline{trans}(BED1) \vee \underline{trans}(BED2)$

$x = FOLG$
$\Rightarrow \underline{trans}(x) = \underline{transf}(FOLG)$

$x = \quad P = F$
$\Rightarrow \underline{trans}(x) = \quad \underline{v}(P) = \underline{te}(F)$

$x = \quad P < F$
$\Rightarrow \underline{trans}(x) = \underline{v}(P) < \underline{te}(F)$

$x = \quad P > F$
$\Rightarrow \underline{trans}(x) = \underline{v}(P) > \underline{te}(F)$

$x = \quad P =\!< F$
$\Rightarrow \underline{trans}(x) = \underline{v}(P) \leq \underline{te}(F)$

$x = \quad P >\!= F$
$\Rightarrow \underline{trans}(x) = \underline{v}(P) \geq \underline{te}(F)$

$x = \quad Y = F$
$\Rightarrow \underline{trans}(x) = \quad \underline{v}(Y) = \underline{te}(F)$

$x = \quad Y < F$
$\Rightarrow \underline{trans}(x) = \underline{v}(Y) < \underline{te}(F)$

$x = \quad Y > F$
$\Rightarrow \underline{trans}(x) = \underline{v}(Y) > \underline{te}(F)$

$x = \quad Y =\!< F$
$\Rightarrow \underline{trans}(x) = \underline{v}(Y) \leq \underline{te}(F)$

$x = \quad Y >\!= F$
$\Rightarrow \underline{trans}(x) = \underline{v}(Y) \geq \underline{te}(F)$

$x =$ P ist B
$\Rightarrow \underline{trans}(x) = ch(P, in, B)$

$x =$ P ist unter B
$\Rightarrow \underline{trans}(x) = ch(P, unter, B)$

$x =$ P ist über B
$\Rightarrow \underline{trans}(x) = ch(P, uber, B)$

$x =$ P ist nicht B
$\Rightarrow \underline{trans}(x) = ch(P, nicht, B)$

$x =$ P ist nicht unter B
$\Rightarrow \underline{trans}(x) = ch(P, nicht\text{-}unter, B)$

$x =$ P ist nicht über B
$\Rightarrow \underline{trans}(x) = ch(P, nicht\text{-}über, B)$

Das oben verwendete Prädikat *ch* (für "charakterisiere") läßt sich mit Hilfe der schon bekannten Prädikate definieren.

$$\forall x_1 \forall x_2 \forall x_3 \forall x_P \forall x_B \; apw(x_P, x_1) \wedge ber(x_P, x_B, x_2, x_3) \wedge x_1 \geq x_2 \wedge x_1 \leq x_3 \rightarrow ch(x_P, in, x_B)$$

$$\forall x_1 \forall x_2 \forall x_3 \forall x_P \forall x_B \; apw(x_P, x_1) \wedge ber(x_P, x_B, x_2, x_3) \wedge x_1 < x_2 \rightarrow ch(x_P, unter, x_B)$$

$$\forall x_1 \forall x_2 \forall x_3 \forall x_P \forall x_B \; apw(x_P, x_1) \wedge ber(x_P, x_B, x_2, x_3) \wedge x_1 > x_3 \rightarrow ch(x_P, uber, x_B)$$

$$\forall x_1 \forall x_2 \forall x_3 \forall x_P \forall x_B \; apw(x_P, x_1) \wedge ber(x_P, x_B, x_2, x_3) \wedge (x_1 < x_2 \vee x_1 > x_3) \rightarrow ch(x_P, nicht, x_B)$$

$$\forall x_1 \forall x_2 \forall x_3 \forall x_P \forall x_B \; apw(x_P, x_1) \wedge ber(x_P, x_B, x_2, x_3) \wedge x_1 \geq x_2 \rightarrow ch(x_P, nicht\text{-}unter, x_B)$$

$$\forall x_1 \forall x_2 \forall x_3 \forall x_P \forall x_B \; apw(x_P, x_1) \wedge ber(x_P, x_B, x_2, x_3) \wedge x_1 \leq x_3 \rightarrow ch(x_P, nicht\text{-}über, x_B)$$

Die logischen Formeln für *ch* sind wie folgt zu lesen:

> Das Prädikat *ch* mit den Argumenten x_P (Parametername), *in* (Bereichsmodifikator) und x_B (Bereichsname) ist wahr, <u>wenn</u> $apw(x_p, x_1)$ (d. h. der aktuelle Parameterwert von x_P ist x_1) <u>und</u> $ber(x_P, x_B, x_2, x_3)$ (d. h. der Bereich x_B für den Parameter x_P geht von x_2 bis x_3) <u>und</u> $x_1 \geq x_2$ <u>und</u> $x_1 \leq x_3$.

Die weiteren Formeln für *ch* sind analog zu lesen. Man beachte, daß das Prädikat *ber* durch die Transformation der *Bereichsdefinitionen* bestimmt wird.

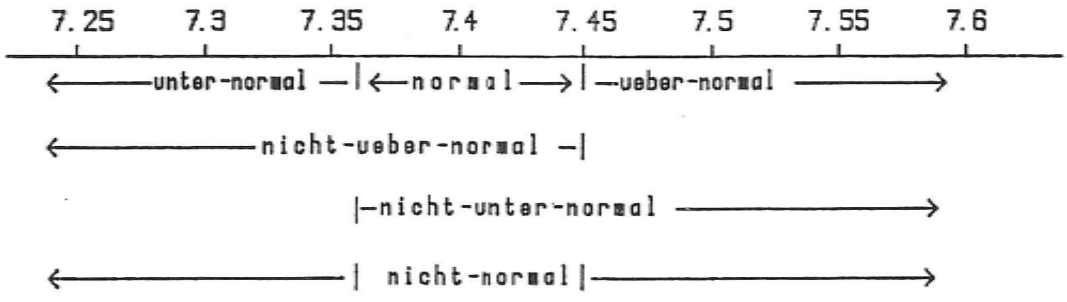

Bild 3.5-1: Normalbereich von pH

Bild 3.5-1 zeigt die Bedeutung des Prädikats *ch* auf einer Werteskala am Beispiel des Normalbereiches für den Parameter pH.

Damit ist die Transformation im Kern beschrieben. Die Semantik einer *Wissensbasis* in AES/L ist gegeben durch die Semantik der daraus mittels *trans* gebildeten Formelmenge vereinigt mit den Formeln für das Prädikat *ch*. Die Semantik dieser Formelmenge wiederum gründet auf der unten beschriebenen Relationalstruktur insbesondere auf der Relation APW. Die Bedeutung der Konstrukte für die *Gültigkeitsdauern* und *Ungültigkeitsdefinitionen* in AES/L wird an späterer Stelle angegeben. Das Gleiche gilt für die Konstrukte "*Parameter* ist bereits definiert" und "*Parameter* ist noch undefiniert".

Die informelle Definition einiger Hilfsfunktionen geschah aus Gründen der Übersichtlichkeit.

Es bleibt noch die Aufgabe, die Semantik der so erhaltenen prädikatenlogischen Formeln zu definieren. Dies geschieht hier durch die Umschreibung der beabsichtigten Relationalstruktur. Eine Relationalstruktur wurde in Kapitel 3.2 als ein Tripel

$$\underline{A} = < A, < f_i^A \mid i \in I_A >, < R_j^A \mid j \in J_A >>$$

definiert.

Zunächst seien in $\underline{A}$ die üblichen Interpretationen der arithmetischen Operationen und Funktionen enthalten: "+", "−", "∗", "/" werden durch die Addition, Substraktion, Multiplikation und Division interpretiert. "exp" steht für die Exponentialfunktion, "ln" für den natürlichen Logarithmus, "log" für den Zehner-Logarithmus, "log2" für den Zweier-Logarithmus, "sqrt" für die Quadratwurzelfunktion und "abs" für die Absolutsbetragsfunktion. Alle Operatoren und Funktionen arbeiten auf den reellen Zahlen. Für das oben eingeführte Gleichheitsprädikat und die Vergleichsprädikate gebe es entsprechende Relationen. Alle Zahlen in den prädikatenlogischen Formeln werden als reelle Zahlen interpretiert.

In unserem Fall wird die Trägermenge A die Menge der möglichen Parameterwerte umfassen. Definiere die Menge M von *Namen* und die Menge W von

Werten:

$$M = \textit{Menge von Namen}$$
$$W := \bigcup_{i=1}^{k} W_0^i, \quad k \in N,\ W_0 = R \cup M$$

Die Konstantenzeichen für die Parameter- und Bereichsnamen sowie für die *Bereichsmodifikatoren* "in", "unter", "über", "nicht" "nicht-unter" und "nicht-uber" können durch sich selbst interpretiert werden, sind also in M und damit in W enthalten. Das Gleiche gilt für die nicht reellwertigen Parameterwerte wie "weiblich" usw.. Die Dimensionalität von W soll zur Modellierung zusammengesetzter Parameterwerte wie der des Blutdrucks dienen.

Nun setze die Trägermenge A der zu beschreibenden Relationalstruktur:

$$A \supseteq W.$$

Damit läßt sich der Begriff *Welt-Parameter* durch eine Relation modellieren. Die Relation heiße APW für "aktueller Parameterwert". Sie ist definiert durch:

$$APW \subseteq M \times W$$
$$(p, r) \in APW \Leftrightarrow \text{"der aktuelle Wert des Parameters } p \text{ ist } r\text{"}.$$

Der Relation APW entspricht das Prädikat apw, das bei der Definition der Lesart der *Sätze* von AES/L benötigt wird. Der Inhalt der Relation APW ist als "Schnappschuß" eines Weltzustandes zu betrachten.

Beispiele. Einige Beispieltransformationen soll die Wirkung der Transformationen verdeutlichen. Die Beispielsätze sind Kapitel 3.4 entnommen, die Quantifizierungen der Variablen werden jetzt ausgeschrieben.

Beispiel 1

AES/L:

```
diagnoseregel 1 heisst
   wenn
     'pH' ist unter normal
   dann
     "Azidose".
```

Prädikatenlogik:

$$true \wedge ch(pH, unter, normal) \rightarrow p_{d1}$$

Der *Folgerung* "Azidose" entspricht also das Prädikat p_{d1}. Der Parametername '*pH*' und der Bereichsname '*normal*' sind wie oben erwähnt Bestandteil der Sprache L. Das Prädikat *true* ist an dieser Stelle eigentlich überflüssig, es wird aus technischen Gründen durch $\underline{transapw}$ eingebracht.

Beispiel 2

AES/L:

```
pH ist normal innerhalb 7.36 bis 7.45.
```

Prädikatenlogik:

$$true \rightarrow ber(pH, normal, 7.36, 7.45)$$

Das Prädikat *ber* wird durch die aus den Bereichsdefinitionen transformierten logischen Formeln definiert. Die reellen Zahlen sind als Konstantenzeichen Bestandteil der Sprache L.

Beispiel 3

AES/L:

```
therapieregel 12 heisst
   wenn
      "Hypoventilation" und
      'AZV' ist unter 'im AZVmax-Bereich'
   dann
      "Minutenvolumen erhoehen auf ", 'AZVmax'*'Frequenz'.
```

Prädikatenlogik:

$$\forall x_1 \forall x_2 \; apw(AZVmax, x_1) \wedge apw(Frequenz, x_2) \wedge p_{Hyp} \wedge$$
$$ch(AZV, unter, \text{'}im\ AZVmax\text{-}Bereich\text{'}) \rightarrow p_{t12}(x_1 * x_2)$$

Man beachte, daß die arithmetischen Operatoren in der AES-Regel formal von denen in der logischen Formel zu unterscheiden sind und diese wiederum sind von den zugehörigen Funktionen "Addition" usw. in der Relationalstruktur zu trennen. Die Hilfsfunktionen haben in diesem Beispiel folgende Werte:

$e(\ldots) = \{AZVmax, Frequenz\}$
$\underline{v}(AZVmax) = x_1$
$\underline{v}(Frequenz) = x_2$
$\underline{transf}(\text{"}Hypoventilation\text{"}) = p_{Hyp}$
$\underline{transf}(\text{"}Minutenvolumen\ erhoehen\ auf\ \ldots\text{"}) = p_{t12}(x_1 * x_2)$

Beispiel 4

AES/L:

```
wenn
```

```
   'Geschlecht' = "weiblich"
dann
   'O2-Grenzwert' := 108.86 − 0.26 ∗ 'Alter'
                     −7.30 ∗ ('Gewicht')/('Laenge' − 1) − 15.10.
```

Prädikatenlogik:

$$\forall x_1 \forall x_2 \forall x_3 \forall x_4 \; apw(Geschlecht, x_1) \wedge apw(Alter, x_2) \wedge$$
$$apw(Gewicht, x_3) \wedge apw(Laenge, x_4) \wedge x_1 = weiblich$$
$$\rightarrow apw(O2\text{-}Grenzwert, 108.86 - 0.26 * x_2 - 7.30 * x_3/(x_4 - 1) - 15.10)$$

Benutzungshinweise. Mit AES/L wird später die Wissensbasis formuliert, mit der das lauffähige Entscheidungsunterstützungssystem AES arbeitet. Da das AES alle Parameterwerte vom AIS erhalten wird, sind sinnvollerweise die Parameternamen wie im AIS zu schreiben und ggf. in Hochkommas zu setzen (vgl. Syntax von *Parametern* in AES/L). Wenn in einer Bedingung die Mengen von Medikamentenverabreichungen abgefragt werden soll, so ist häufig die die letzte verabreichte Menge interessant, sondern die bisher verabreichte Gesamtmenge. Dies kann in AES/L durch das Präfix "gesamtmenge" vor dem Medikamentennamen ausgedrückt werden. Die Gesamtmengen von Medikamenten werden durch das AIS von jeher aufsummiert und gespeichert. Ihre Übertragung zum AES wird daher keine Schwierigkeiten machen.

Eine interessante Anwendung von Folgerungen mit Argumenten sei hier am Beispiel erklärt:

```
hilfsregel 756 heisst
   wenn
      a > 10 und
      'K' ist normal
   dann
      "Alpha-X-Koeffizient ist ", 'K'/(a-1).

diagnoseregel 99 heisst
   wenn
      "Alpha-X-Koeffizient ist ",X und
      'Puls' > X
   dann
      "Puls zu hoch".
```

In der Diagnoseregel wird die Hilfsregel mit der Variablen X benutzt. Allerdings ist zumindest in diesem Fall eine *Parameterdefinition* für den 'Alpha-X-Koeffizienten' eleganter:

```
wenn
  a > 10 und
  'K' ist normal
dann
  'Alpha-X-Koeffizient' := 'K'/(a-1).

diagnoseregel 99 heisst
   wenn
      'Puls' > 'Alpha-X-Koeffizient'
   dann
      "Puls zu hoch".
```

Zusammenfassung. Dieses Kapitel hatte die Definition der Semantik der Beschreibungssprache AES/L zur Aufgabe. Dies wurde gelöst durch Angabe einer Transformation von *Sätzen* der Beschreibungssprache in prädikatenlogische Formeln. Diese sind in der Mathematik wohluntersucht und zumindest in ihrem aussagenlogischen Teil in ihrer Gültigkeit und Nachprüfbarkeit auch unter Nichtmathematikern anerkannt. Die Betrachtung der durch *trans* erzeugten Formeln zeigt, daß sie schon fast die Form von Klausen, sogar Horn-Klausen haben (vgl. Kap. 3.2). Diese Eigenschaft wird bei dem Rechnen mit den Formeln noch von großem Nutzen sein, da ein recht effizienter Kalkül, der Resolutionskalkül, für solche Formeln existiert. Der Umfang der Definition der Funktion *trans* sollte nicht von der Ähnlichkeit der *Sätze* von AES/L mit logischen Formeln ablenken. Die *Partikel*[9] "wenn ... dann ...", "und", "oder" sind direkt mit den Symbolen "$\rightarrow$ (Implikation)", "$\wedge$" und "$\vee$" zu assozieren. Die Besonderheit von AES/L, nämlich die Verwendung von Parameternamen, wenn ihr aktueller Wert gemeint ist, wurde durch das Prädikat *apw* erklärt. Der Sinn des Konstruktes der *Bereichsdefinitionen* in AES/L ist an sich nur zur besseren Lesbarkeit der *Sätze* gedacht, es konnte durch das Prädikat *ch* auf recht einfache Weise in die Prädikatenlogik abgebildet werden.

Das Rückgrat der Sprache besteht zweifelsohne aus den "wenn ... dann ..."-Konstrukten. Der zugehörige logische Operator "$\rightarrow$" wird von Lenk [LEN 82] als der Inbegriff des schematischen Vorgehens bezeichnet.

3.6. Konzept eines lauffähigen Systems

Der Schlüssel zur Definition der Bedeutung der Beschreibungssprache AES/L war die Lesart als prädikatenlogische Formeln und die Einführung einer Relation *APW* für den "aktuellen Parameterwert". Diese Sichtweise reicht vollkommen

[9] Bezeichnung von Lenk [LEN 82]

zum Verständnis der im vorigen Kapitel erklärten Sprachkonstrukte von AES/L. In der Praxis ist der aktuelle Wert eines Parameters i. a. nicht zu jeder Zeit bekannt. So werden die Laborwerte etwa alle 10 Minuten aus einer Blutprobe bestimmt und sind streng genommen nur für den Zeitpunkt der Blutprobenentnahme zutreffend.

Das angestrebte Entscheidungsunterstützungssystem soll dennoch mit einer in AES/L geschriebenen Wissensbasis arbeiten können. In diesem Kapitel wird ein Weg zur Annäherung an die nicht verfügbare Relation APW und zur Implementierung des AES aufgezeigt werden. Parallel dazu wird die Bedeutung der im vorigen Kapitel nicht erklärten Konstrukte definiert.

Der zweistelligen Relation APW entspricht das Prädikat $apw(x, y)$ mit der Bedeutung: "der aktuelle Wert des Parameters x ist y". Die Relation ist als "Schnappschuß" der Welt zu verstehen, in dem alle Parameterwerte gleichzeitig aufgenommen werden. Wie in Kapitel 2 dargelegt, ist das zu implementierende AES allein auf die im AIS eingetragenen Parameter angewiesen. Im AIS werden Parameter mit einem Wert und einer Zeit eingetragen, z. B. "Puls = 120 um 10:25". Eine Eintragung im AIS besteht also aus dem Parameternamen, dem Wert und der Zeit, zu dem dieser Wert bestimmt wurde. Damit ist ein Prädikat

$$pw(x, y, z)$$

mit der Interpretation "der Wert des Parameters x ist y zur Zeit z" realisierbar. Setzt man zudem ein Prädikat $zeit(x)$ mit der Interpretation "die aktuelle Uhrzeit ist x" als gegeben voraus, so ist folgende Annäherung an apw möglich:

$$\forall x \forall y \forall z \; pw(x, y, z) \wedge zeit(z) \rightarrow apw'(x, y).$$

Allerdings ist diese Lösung nicht praktikabel, da alle in einer AES-Regel vorkommenden Parameter zur gleichen Zeit eingetragen sein müssen, damit die Regel erfüllt ist. Dies zeigt ein kleines Beispiel mit den Parametern a und b:

```
hilfsregel 17 heisst wenn a < b dann "a ist kleiner b".
```

Dem entspricht die prädikatenlogische Formel mit apw':

$$\forall x_a \forall x_b \; apw'(a, x_a) \wedge apw'(b, x_b) \wedge x_a < x_b \rightarrow p_{h17}.$$

Sind die Zeiten für die Parameter a und b verschieden, so trifft die Regel nicht zu. Es ist also ein anderer Weg zu finden. Das Problem ist, daß für das "Schnappschuß"-Prädikat apw jeder Parameterwert zu jeder Zeit bekannt sein muß, aber nur die Werte an einigen Stützstellen verfügbar sind. Eine Lösung für dieses Problem ist die Verlängerung des letzten Parameterwertes auf die Zeitspanne bis zur nächsten Stützstelle oder bis zu dem Zeitpunkt, an dem aufgrund gewisser inhaltlicher Eigenschaften des Parameters der letzte Wert als veraltet anzusehen ist. Gerade dies wird durch das Konstrukt

Parameter ist gueltig *Dauer*

in AES/L ausgedrückt wird. Beispielsweise kann man so die *Gültigkeitsdauer* eines Blutdruckwertes definieren:

```
'Blutdruck' ist gueltig fuer 3 min.
```

Die Intention des Konstruktes ist auszudrücken, daß spätestens nach 3 Minuten der zuletzt gemessene Blutdruckwert ungültig ist. Andere Parameter haben andere Gültigkeitsdauern:

```
'Alter' ist gueltig bis veraendert.
```

Hiermit wird zum Ausdruck gebracht, daß der eingetragene Wert für das Patientenalter solange gilt, bis es korrigiert wird. Die Lesart der obigen Beispiele in der Prädikatenlogik ist

$true \rightarrow gd(Blutdruck, 3)$ bzw. $true \rightarrow gd(Alter, \infty)$.

In AES/L können solchen *Gültigkeitsdefinitionen* auch Bedingungen vorangestellt werden, etwa:

```
wenn
  'Alter' > 40 und pH ist normal
dann
  'Puls' ist gueltig fuer 1 min.
```

Dies liest sich prädikatenlogisch:

$\forall x_A \; apw(Alter, x_A) \wedge x_A > 40 \wedge ch(pH, in, normal)$
$\rightarrow gd(Puls, 1)$.

Mit dem so gewonnenen neuen Prädikat gd (für "Gültigkeitsdauer") kann das Prädikat apw'' als Annäherung an apw definiert werden:

$\forall x_p \forall x \forall y \forall z_p \forall z \; pw(x_p, x, z_p) \wedge gd(x_p, y) \wedge zeit(z) \wedge z \leq z_p + y$
$\rightarrow apw''(x_p, x)$,

d. h. "der aktuelle Wert der Parameters x_p ist x, wenn x der Parameterwert von x_p zur Zeit z_p war und die Gültigkeitsdauer noch nicht überschritten ist".

Ist die Gültigkeitsdauer für einen Parameter nicht explizit definiert, so soll "bis verändert" angenommen werden. Ein weiteres Konstrukt handelt ebenfalls von der Gültigkeit von Parametern:

Parameter ist ungueltig wenn *Parameterliste* eingetragen wird.

Ein gutes Beispiel für die Anwendung einer solchen *Ungültigkeitsdefinition* ist der Sauerstoffpartialdruck PO_2 im Blut. Der Anästhesist hat die Aufgabe, die Sauerstoffversorgung des Patienten auf einem ausreichenden Niveau zu halten. Als Indikator dient ihm u. a. der PO_2-Wert. Ist er zu niedrig, so kann er z. B. die

Einstellung am Respirator verändern. Ist eine solche Veränderung vorgenommen, so ist der Meßwert PO_2 auf jeden Fall als ungültig anzusehen, da sie den Meßwert beeinflußt. In diesem Beispiel würde man in AES/L etwa formulieren:

```
'PO2' ist ungueltig wenn
'Frequenz', 'Min.Vol.'  eingetragen wird.
```

Frequenz und *Minutenvolumen* sind Einstellungen am Respirator.

Damit ist die Annäherung an das Prädikat *apw* abgeschlossen. Das um die Ungültigkeitsdefinition erweiterte Prädikat soll apw''' heißen und bildet die Grundlage für die Implementierung.

Implementierungskonzept. Es liegt auf der Hand, das AES als eigenständiges Programmsystem zu verwirklichen, das eine genau definierte Schnittstelle zum AIS hat. Um die Antwortzeiten des AIS auf Benutzereingaben nicht zu verschlechtern, wird die Implementierung auf einem eigenem Rechner stattfinden. Dies bedeutet, daß ein Kommunikationsprotokoll AIS – AES aufgestellt werden muß. Alle Daten (Parameterwerte), die im AIS eingetragen werden, müssen an das AES geschickt werden. Damit steht das Prädikat *pw* prinzipiell dem AES zur Verfügung. Die Lesart der Sprache AES/L als Spezialfall der Prädikatenlogik legt nahe, daß das AES mit den in prädikatenlogische Formeln übersetzten *Sätzen* der Wissensbasis arbeitet. Wie schon erwähnt, haben diese Formeln praktisch schon die Form von Horn-Klausen, so daß der Resolutionskalkül zur Auswertung der Formeln anwendbar ist. Damit kann man folgende Teilaufgaben einer möglichen Implementierung unterscheiden:

1. Entgegennahme von im AIS eingetragenen Parameterwerten und Anpassung der Relation PW für das Prädikat *pw*
2. Ableitung von Schlußfolgerungen, die sich aus den neuen Parameterwerten und der übersetzten Wissensbasis ergeben
3. Ausgabe der zutreffenden Schlußfolgerungen an das AIS

Als Text der Ausgabe kann der "dann"-Teil der AES-Regeln genommen werden. Ein Beispiel zeigt die angestrebte Funktionsweise.

In der Wissensbasis seien die *Sätze*

```
diagnoseregel 1 heisst
  wenn
    pH ist unter normal
  dann
    "Azidose".
pH ist normal innerhalb 7.36 bis 7.45.
```

Die Eingabe im AIS sei: "pH=7.33 um 10:34". Damit trifft die Regel für "Azidose" zu, und es kann der Text "Azidose" an das AIS übertragen werden.

Bedeutung der weiteren Konstrukte in AES/L. Bisher nicht erklärt sind die *Bedingungen*

"*Parameter* ist *Eigenschaft* seit *Formel* min" und

"*Folgerung* seit *Formel* min".

Das AES wird sich zur Implementierung derselben Aufzeichnungen über das Zutreffen von *Folgerungen* und *Charakterisierungen* machen. Es wird hier keine formale Definition der Bedeutung gegeben, da diese Konstrukte aus dem normalen Rahmen von AES/L herausfallen: man spricht über Prädikate. Stattdessen soll es mit Worten definiert werden:

- Das fragliche Prädikat selbst muß zutreffen,
- es muß zu einem früheren Zeitpunkt zugetroffen haben,
- zwischen diesem Zeitpunkt und der aktuellen Zeit darf niemals sein Nichtzutreffen festgestellt worden sein und
- zwischen dem Zeitpunkt und der aktuellen Zeit müssen mindestens soviel Minuten liegen, wie *Formel* angibt.

Ebenfalls aus dem Rahmen fallen die *Bedingungen*

"*Parameter* ist noch undefiniert" und

"*Parameter* ist bereits definiert".

In diesem Fall werden Aussagen über das Prädikat *pw* gemacht. Die erste Bedingung ist wahr, wenn noch kein Parameterwert für *Parameter* vom AIS eingetragen wurde, die zweite ist wahr, wenn bereits ein Parameterwert eingetragen wurde.

In AES/L kann eine Parameterdefinition anstatt mit ":=" auch mit "=" formuliert werden. Der Grund für diese Unterscheidung liegt in der in Kapitel 4 vorzustellenden Implementierung. Eine Parameterdefinition ist mit "=" durchzuführen, wenn der Wert des zu definierenden Parameters keine reelle Zahl ist.

Beispiel

```
wenn
   'Alter' < 12
dann
   'Pat' = "Kind".
```

Die Konstrukte für die *Dosierempfehlungen* werden in ein Prädikat

$$empf(x, y)$$

(vgl. Kap. 3.5) mit der beabsichtigten Bedeutung "die Dosierempfehlung für das Parameter x ist y" transformiert. Natürlich sollen in AES/L nur Dosierempfehlungen für Medikamente definiert werden. Die Implementierung des AES wird dann dafür sorgen, daß die in $empf(x, y)$ berechneten Werte an das AIS übertragen werden, um dort in geeigneter Weise dargestellt zu werden.

Die Verwendung von Regeltypen (diagnoseregel, therapieregel, hilfsregel) in AES/L dient hauptsächlich der Strukturierung der Wissensbasis. In der Logik unterscheiden sich die drei Regeltypen überhaupt nicht. Lediglich bei der Ausgabe an das AIS werden sie sich unterscheiden: die Texte der "dann"-Teile der zutreffenden Diagnoseregeln werden vor denen der Therapieregeln ausgegeben, zutreffende Hilfsregeln werden nicht ausgegeben. Die Numerierung identifiziert eine Regel eindeutig, was insbesondere für die Implementierung wichtig ist. Die an das AIS ausgegebenen Texte sollen ab jetzt *Entscheidungshilfen* heißen.

Zusammenfassung. Die prädikatenlogische Sicht auf AES/L eröffnete im vorigen Kapitel eine elegante Möglichkeit zur Definition der Semantik der wichtigsten Konstrukte von AES/L. In diesem Kapitel wurde der Weg aufgezeigt, wie man von der Interpretation der Beschreibungssprache zu einem System gelangen kann, das mit einer *Wissensbasis* in AES/L arbeiten kann. Wesentlich war dabei die Ersetzung des in unserer Anwendung nicht verfügbaren Prädikats *apw* für den "aktuellen Parameterwert" durch eine Annäherung, die auf der Betrachtung von Gültigkeitsdauern für die Parameter basiert.

4. Realisierung des Anästhesie-Entscheidungs-unterstützungssystems

In Kapitel 3 standen formale Begriffe und Konzepte im Vordergrund. Nun ist eine etwas pragmatischere Sicht angebracht. Das medizinische Wissen wird in der Sprache AES/L formuliert und als normale Textdatei dem Rechner zur Verfügung gestellt. Eine solche Textdatei wird die Wissensgrundlage für das Entscheidungsunterstützungssystem bilden. In Kapitel 3.6 wurde bereits der Weg zur Auswertung einer derartigen Wissensbasis angedeutet: die Umformung in äquivalente prädikatenlogische Formeln, für die es Kalküle zum Rechnen (Beweisen) gibt. Die herausgestellte Hornklausen-Form der Formeln ist kein Zufall gewesen, sondern sie wurde absichtlich im Hinblick auf die gewählte Implementierungssprache herbeigeführt. Diese Sprache ist die Programmiersprache PROLOG.

Die Prozeduren von PROLOG sind Hornklausen, die nach dem Resolutionskalkül (vgl. Kap. 3.2) ausgewertet werden. Es bietet sich an, die in AES/L geschriebene Wissensbasis in PROLOG zu übersetzen, um sie dann in einem Laufzeitsystem zur Ausführung zu bringen. Das Laufzeitsystem wird die Entgegennahme der Daten vom AIS, die Anwendung der Wissensbasis und die Ausgabe der Entscheidungshilfen an das AIS zur Aufgabe haben. Nach einer kurzen Vorstellung der Programmiersprache PROLOG wird die Übersetzung der Wissensbasis von AES/L nach PROLOG Thema von Kapitel 4.1 sein. Danach wird die Implementierung des Laufzeitsystems vorgestellt. Kapitel 4.3 behandelt dann die Anbindung an das AIS. Die Effizienzsteigerung des Systems durch *Lemma-Generierung* wird in Kapitel 4.4 besprochen und den Abschluß bildet die Beschreibung einer Erklärungskomponente in Kapitel 4.5. Die Details der Implementierung können den kommentierten Programmausdrucken im Anhang entnommen werden.

Logikprogrammierung. Die Geschichte der Logikprogrammierung (vgl. [KOW 85], [HOG 84]) begann in den frühen fünfziger Jahren mit Versuche zur Mechanisierung von mathematischen Beweisen auf dem Rechner. Der Durchbruch gelang aber erst Anfang der siebziger Jahre, als eine effiziente Anwendung des Resolutionsprinzips von Robinson (vgl. [ROB 65]) auf eine eingeschränkte Klasse von logischen Formeln, den *Horn-Klausen*, entdeckt wurde.

Ein erster Interpreter, der mit diesen Klausen nach dem Resolutionsprinzip arbeitete, wurde 1972 von Roussel,Colmerauer et al. implementiert. Damit war die Programmiersprache PROLOG (*Programming in Logic*) erfunden. Da die Resolutionsregel (vgl. Kap. 3.2) Freiheitsgrade hinsichtlich der zu resolvierenden Formeln hat, ist für einen Interpreter eine Auswertungsstrategie erforderlich. Die meisten PROLOG-Interpreter arbeiten nach einer Auswertungsstrategie, die stichwortartig beschrieben ist durch

a) *Top-down-Strategie*
   - dem Interpreter wird eine Zielklause vorgegeben, die er zu beweisen hat,

b) *lineare Input-Resolution*
   - die Resolutionsregel wird auf die Anwendung "Elemente der Zielklause gegen die Klausen des Programms" beschränkt,

c) *Depth-First-Suche*
   - aus der Liste der zu beweisenden Ziele wird immer die am Anfange stehende als nächstes der Resolutionsregel unterworfen.

Die Klausen eines PROLOG-Programms werden auch *Regeln* oder *Prozeduren* genannt. Sie werden in der Form

$$A \leftarrow B_1, \ldots, B_n$$

notiert und als "A wenn $B1$ und ... und B_n" gelesen. In der prozeduralen Sicht (vgl. [HOG 84]) entspricht A dem Prozedurkopf und $B_1, \ldots, B_n$ dem Prozedurrumpf, der aus lauter Prozeduraufrufen besteht. Falls $n = 0$ nennt man die Regel auch *Fakt*. Eine *Zielklause* wird in der Form $\leftarrow G_1, \ldots, G_n$ geschrieben. Für eine Einführung in PROLOG verweise ich auf die entsprechende Literatur, insbesondere auf das Bücher von Hogger [HOG 84] und Clocksin/Mellish [C&M 81].

Heute gibt es PROLOG für beinahe jede Rechnergröße, vom 8-Bit-Rechner bis zum Großrechner. Bei dieser Diplomarbeit wird der PROLOG-Interpreter der Firma InterFace GmbH (s. [IFA 86], [PCS 84a]) mit der Bezeichnung "IF/Prolog" in der Version $V3.0$ auf einem Rechner CADMUS 9200 der Firma PCS eingesetzt. Die Entwicklung läuft unter dem Betriebssystem MUNIX (s. [PCS 84b]), einer UNIX-Portierung. IF/Prolog V3.0 folgt dem Sprachstandard, der in dem Buch von Clocksin/Mellish vorgeschlagen wird. Prädikatsnamen sind *Atome*, d. h. sie beginnen mit einem Kleinbuchstaben gefolgt von beliebig vielen Buchstaben und Ziffern, oder sie sind beliebige Zeichenketten, die in 'Hochkommas' gesetzt sind. Variablen beginnen mit einem Großbuchstaben oder dem Zeichen "_" gefolgt von beliebig vielen Buchstaben und Ziffern. Das auf einer Standardtastatur nicht verfügbare Zeichen "$\leftarrow$" wird durch ":-" ersetzt.

```
geschwister(X,Y) :-
   eltern(X,Vater,Mutter),eltern(Y,Vater,Mutter).
```

'X', 'Y', 'Vater' und 'Mutter' sind Variablen, die wie üblich als $\forall$-quantifiziert anzusehen sind.

Zusätzlich erlaubt IF/Prolog V3.0 das Rechnen mit reellen Zahlen und hat die Möglichkeit der Compilation von Teilen des PROLOG-Programmes.

4.1. Übersetzung von AES/L

Die Wahl der Programmiersprache PROLOG ermöglicht eine sehr elegante Lösung des Problems der Auswertung einer in AES/L geschriebenen Wissensbasis: die Übersetzung in PROLOG-Regeln. Die Übersetzung kann sich dabei im wesentlichen an die Funktion *trans* aus Kapitel 3.5 halten. Diese Übersetzung soll natürlich von einem Programm ausgeführt werden, so daß der Benutzer nur die Sprache AES/L und die Kommandos kennen muß, um die Wissensbasis in aquivalente PROLOG-Regeln zu übersetzen, mit denen später das AES arbeiten wird.

Das Übersetzungsprogramm selbst ist ebenfalls in PROLOG implementiert. Es ist stark von der *funktionalen Definition der Codegenerierung* aus der Vorlesung "Compilerbau II" von Prof. Indermark [IND 85] inspiriert, die fast unverändert ein PROLOG-Programm umgesetzt werden kann.

Syntaxanalyse. Dieser Teil des Übersetzungsprogramms wird zur Gänze von PROLOG selbst übernommen. Es ist lediglich die Definition von sog. *Operatoren* notwendig, wie sie nachfolgend abgebildet ist:

```
:- op(1155, xfx, 'heisst').
:- op(1150, fx, 'wenn').
:- op(1150, xfx, 'wenn').
:- op(1100, fx, 'weiterhin').
:- op(1100, xfx, 'eingetragen').
:- op(935, xfx, 'innerhalb').
:- op(1050, xfx, 'dann').
:- op(1040, xfx, 'bis').
:- op(1031, xfy, 'oder').
:- op(1030, xfy, 'und').
:- op(1005, xfx, 'nr').
:- op(940, xfx, 'ist').
:- op(933, xfx, 'seit').
:- op(925, xf, 'min').
:- op(930, xfx, 'fuer').
:- op(910, fx, 'noch').
:- op(910, fx, 'bereits').
:- op(900, fx, 'nicht').
:- op(850, fx, 'unter').
:- op(850, fx, 'ueber').
:- op(700, xfx, ':=').
:- op(550, xfy, ':').
:- op(400, fx, 'diagnoseregel').
```

```
:- op(400, fx, 'therapieregel').
:- op(400, fx, 'hilfsregel').
:- op(195, fx, 'empfohlen').
:- op(190, fx, 'gesamtmenge').
```

Die arithmetischen Operatoren '+', '−', '∗','/', '^' und die Vergleichsoperatoren '=' '<', '>', '>=' und '=<' sowie der Operator ',' sind bereits in IF/Prolog vordefiniert.

Die oben stehenden Operatordeklarationen sind der Datei 'aespar.op' im Anhang entnommen. Das erste Argument enthält die Priorität (Bindungsstärke) des zu definierenden Operators, das zweite den Operatortyp (Präfix, Infix oder Postfix) und das dritte den neuen Operator selbst. Mit diesen Operatoren kann man *Terme* aufbauen, z. B.:

```
a und 27.5
wenn A dann B
'Erwin' ist 'Schlosser'
X < Y usw..
```

Wie man sieht, sind nicht alle mit den Operatoren erzeugbaren Terme in der zu übersetzenden Sprache AES/L. Allerdings sind alle *Sätze* der Sprache AES/L solche Operatorterme. Operatorterme existieren nur außerhalb des PROLOG-Interpreters. Beim Einlesen werden sie ein ein internes Format umgewandelt, das *Struktur* heißt. Das Beispiel einer Diagnoseregel soll die Beziehung 'Operatorterm zu Struktur' zeigen.

Operatorterm:

```
diagnoseregel 4097 heisst
  wenn a<b und c ist unter normal
  dann "Alles klar".
```

Struktur:

```
heisst(
  diagnoseregel(4097),
  wenn(dann(
    und('<'(a,b), ist(c,unter(normal))),
    "Alles klar")))
```

Das Einlesen von Termen und deren Umwandlung in Strukturen übernimmt die fest eingebaute PROLOG-Prozedur 'read' (vgl. [C&M 81]). Damit kann schon eine PROLOG-Regel angegeben werden, die die Übersetzung aller in einer Datei enthaltenen AES/L-*Sätze* in PROLOG steuert:

```
compile(InputFile, OutputFile) :-
  see(InputFile),
  tell(OutputFile),
  repeat,
```

```
  read(ARegel),
  translate(ARegel),
  ARegel = end_of_file,
  seen,
  told.
```

Die Variable 'InputFile' enthält den Namen der Textdatei, auf dem die Wissensbasis in der Sprache AES/L steht; die Ausgabe wird auf die Textdatei, deren Name in 'Outputfile' steht, geschrieben. Nach dem Öffnen der beiden Dateien, werden solange Terme von der Eingabedatei gelesen und auf die Ausgabedatei ubersetzt, bis das Ende der Eingabedatei erreicht ist. Danach werden die beiden Dateien geschlossen.

Wegen der oben abgebildeten Operatordeklarationen kann die Prozedur 'read' in AES/L geschriebene *Sätze* einlesen und in das Strukturformat von PROLOG uberführen. Eine solche Struktur ist eine Art komprimierter *Syntaxbaum* für den eingelesenen *Satz*. Die Prozedur 'translate' wird dann nur solche Terme in PROLOG übersetzen, die aus der Sprache AES/L stammen. 'translate' wird sich wie gesagt an die Funktion *trans* aus Kapitel 3.5 orientieren. Die Details der Programmierung können der Datei 'aespar.pro' im Anhang entnommen werden. An dieser Stelle sollen zwei Prozeduren aus dem Übersetzungsprogramm 'aespar.pro' genügen, um einen Eindruck von dem Progammstil zu vermitteln.

```
translate(diagnoseregel N heisst
          wenn BEDINGUNG dann FOLGERUNG) :-
  integer(N),
  validateConclusion(FOLGERUNG, d, N, TFOLGERUNG, FTERM),
  writeConclusion(TFOLGERUNG),
  writeString(" :- "),
  nl,tab(3),
  extractLGW(BEDINGUNG und FTerm, f(TFOLGERUNG), Nliste),
  transCond(BEDINGUNG, Nliste),
  transCformulas(FTerm, NListe, lhs),
  write_TIassign,
  write('.'),
  nl, nl,
  !.

transCond(BEDINGUNG1 und BEDINGUNG2, Nliste) :-
  transCond(BEDINGUNG1, Nliste),
  write(','),
  nl,tab(3),
  transCond(BEDINGUNG2, Nliste),
  !.
```

Die erste Prozedur übersetzt eine Diagnoseregel in PROLOG. Man beachte, daß in dem Argument des Prozedurkopfes die syntaktische Form von Diagnoseregeln fast unverändert übernommen ist (vgl. Kap. 7). Wird diese Prozedur etwa mit der Diagnoseregel 4097 von oben angewendet, so wird die Variable N an die Zahl 4097, die Variable BEDINGUNG an den Term "a<b und c ist unter normal" und die Variable FOLGERUNG an "Alles klar" gebunden. In dem Prozedurrumpf wird zunächst getestet, ob die Regelnummer N wie in der Syntax von AES/L verlangt eine natürliche Zahl ist. Danach wird die syntaktische Form von FOLGERUNG überprüft und bei Erfolg in eine interne Darstellung zerlegt. Danach wird der Code für die äquivalente PROLOG-Regel generiert.

Die abgebildete Prozedur für 'transCond' behandelt die Übersetzung von durch "und" verknüften Bedingungen. Die Einzelheiten sollen hier nicht weiter interessieren. Die Übersetzung der obigen Diagnoseregel 4097 lautet dann:

```
folgerung('Alles klar', d, 4097, []) :-
  letztergueltigerWert(a, P0),
  letztergueltigerWert(b, P1),
  P0 .< P1,
  charakterisiere(c, unter normal).
```

Das Prädikat 'letztergueltigerWert' übernimmt die Rolle von apw''' aus Kapitel 3.6, das Prädikat 'charakterisiere' steht für das Prädikat *ch* aus Kapitel 3.5. Ferner heißt das Prädikat *ber* von Kapitel 3.5 jetzt 'bereich', das Prädikat *empf* 'empfehlung' und das Prädikat *gd* 'gueltigkeitsdauer'. Der eindeutige Prädikatsname für den "dann"-Teil der Regeln wird durch "folgerung('Alles klar',d,4097,[])" dargestellt. Das erste Argument enthält den Namen des "dann"-Teils, die nächsten beiden Argumente ergeben eine eindeutige Kennzeichnung der Regel und das letzte Argument enthält die in der Originalregel vorkommenden Argumente des "dann"-Teils und ist in unserem Fall leer. Im Laufzeitsystem sind die Prädikate 'letztergueltigerWert', 'charakterisiere' und die Vergleichsoperatoren '. <', '. >' usw. für reelle Zahlen zur Verfügung zu stellen. Die "gepunkteten" Vergleichsoperatoren tragen möglichen Ungenauigkeiten beim Rechnen mit reellen Zahlen Rechnung. Sie sind wie folgt definiert:

$$
\begin{aligned}
x \mathbin{.<} y &\Leftrightarrow x < y + \epsilon \\
x \mathbin{.>} y &\Leftrightarrow x > y - \epsilon \\
x \mathbin{.=} y &\Leftrightarrow x \geq y - \epsilon \wedge x \leq y + \epsilon \\
x \mathbin{.<>} y &\Leftrightarrow x < y - \epsilon \vee x > y + \epsilon \\
x \mathbin{.>=} y &\Leftrightarrow x \geq y - \epsilon \\
x \mathbin{.=<} y &\Leftrightarrow x \leq y + \epsilon
\end{aligned}
$$

Der Wert von ϵ kann der Datei 'aesdp.pro' entnommen werden, in der die gepunkteten Vergleichsoperatoren in PROLOG für das Laufzeitsystem implementiert sind.

Der Abhängigkeitsgraph. Das im nächsten Kapitel vorzustellende Laufzeitsystem des AES wird Informationen darüber benötigen, in welche Regeln welche Parameter eingehen. Diese Information wird von dem Übersetzungsprogramm berechnet. Für jeden zu übersetzenden *Satz* der Sprache AES/L ist eindeutig berechenbar, von welchen Prädikaten der übersetzte Satz abhängt. In unserem Beispiel:

lgw(a) $\longrightarrow$ f(d,4097)
"der letzte gültige Wert des Parameters a geht in die Diagnoseregel 4097 ein"
lgw(b) $\longrightarrow$ f(d,4097)
lgw(c) $\longrightarrow$ f(d,4097)
gd(a) $\longrightarrow$ f(d,4097)
"die Gültigkeitsdauer von a geht in die Diagnoseregel 4097 ein"
gd(b) $\longrightarrow$ f(d,4097)
gd(c) $\longrightarrow$ f(d,4097)
be(c,normal) $\longrightarrow$ f(d,4097)
"der Bereich 'normal' für den Parameter c geht in die Diagnoseregel 4097 ein"

Die symbolisch bezeichneten Prädikate bilden also die Knoten des Abhängigkeitsgraphen. Zwei Knoten A und B sind verbunden, wenn es einen AES/L-*Satz* gibt, der das Prädikat B in Abhängigkeit von A definiert. Man beachte, daß in Kapitel 3.6 das Prädikat *gd* in die Definition von apw'' eingeht. Folgerichtig gehen die Gültigkeitsdauern auch in die Diagnoseregel 4097 ein. Das Gleiche gilt für das Prädikat *ber* für die Bereichsdefinition. Der Graph wird schrittweise bei der Übersetzung aufgebaut. Nach Übersetzung des letzten *Satzes* der Wissensbasis berechnet das Programm für jeden Parameter folgende Listen aus dem Abhängigkeitsgraphen:

(1) *Diagnoseliste*
- enthält die Köpfe aller übersetzten Diagnoseregeln, in die der Parameter unmittelbar oder mittelbar eingeht

(2) *Therapieliste*
- enthält die Köpfe aller übersetzten Therapieregeln, in die der Parameter unmittelbar oder mittelbar eingeht

(3) *Hilfsregelliste*
- enthält die Köpfe aller übersetzten Hilfsregeln, in die der Parameter unmittelbar oder mittelbar eingeht

(4) *Empfehlungsliste*
- enthält die Köpfe aller übersetzten Dosierempfehlungen, in die der Parameter unmittelbar oder mittelbar eingeht

(5) *Parameterliste*
- enthält die Namen der Parameter, in die der Parameter unmittelbar oder mittelbar eingeht

Graphentheoretisch bedeutet diese Berechnung das Auffinden aller Wege von den Parameterknoten lgw(.) zu den Knoten f(.,.), emp(.) und lgw(.). Beim Durchrechnen des Abhängigkeitsgraphen wird gleichzeitig festgestellt, ob in der Wissensbasis eine rekursive Definition eines Prädikats auftaucht. Dies ist verboten und wird als Fehler gemeldet.
Beispiel:

```
hilfsregel 123 heisst
  wenn
    a ist normal und b < c+3*t
  dann
    "a,b,c,t sind ok".

wenn
  "a,b,c,t sind ok"
dann
  a ist normal innerhalb 10 bis 15.
```

Damit ist auch die am Ende von Kapitel 3.4 erwähnte Randbedingung für die Sprache AES/L erklärt. Die drei letzten Listen sind nur für die in Kapitel 4.4 besprochene Lemma-Generierung wichtig. In unserem Beispiel würde die Diagnoseliste für den Parameter a etwa so aussehen:

```
diagnoseListe(a,
    [ ..., folgerung('Alles klar',d,4097,[]), ...]).
```

Zusätzlich erstellt das Übersetzungsprogramm sogenannte Ungültigkeitslisten, die den Ungültigkeitsdefinitionen in AES/L entsprechen. Beispiel:

```
ungueltigkeitsListe('Frequenz', ['PO2','PCO2']).
```

Die Parameter 'PO2' und 'PCO2' sind also als ungültig anzusehen, wenn der Parameter 'Frequenz' verändert wird. Eine ausführliches Beispiel wird im Anhang durch die Dateien 'aesreg.reg' und 'aesreg.txt' gegeben. 'aesreg.reg' ist die in AES/L geschriebene Wissensbasis und 'aesreg.txt' ist deren Übersetzung in PROLOG. Das in PROLOG geschriebene Übersetzungsprogramm steht in der Datei 'aespar.pro' und ist ebenfalls im Anhang zu finden. Man wird feststellen, daß die Übersetzung tatsächlich etwas anders aussieht wie hier beschrieben. Der Grund dafür ist die in Kapitel 4.4 behandelte Lemma-Generierung.

Benutzen des Übersetzungsprogrammes. Das Übersetzungsprogramm heißt 'aespar.pro' und wird mit Hilfe des PROLOG-Interpreters zum Ablauf gebracht. Das Kommando 'translreg' bewirkt die Übersetzung der Wissensbasis in der Datei 'aesreg.reg' in die PROLOG-Datei 'aesreg.txt' (vgl. Datei 'translreg' im Anhang). Ohne die Kommandodatei 'translreg' kann die Übersetzung wie folgt durchgeführt werden:

```
$ifprolog -dbs 500k
IF/Prolog Version 3.0 pf Thu May 29 19:02:55 EST 1986
Copyright (C) 1984, 1985, 1986 InterFace Computer GmbH

?- load('aespar.sem').
yes

?- compile('aesreg.reg', 'aesreg.txt').
... informative Ausgaben über den Stand der Übersetzung

?- end_of_file.
$
```

Mit dem Befehl

```
load('aespar.sem')
```

wird die das semi-compilierte Übersetzungsprogramm (vgl. Kap. 5) in den Interpreter geladen. Alternativ kann auch das Quellprogramm mit

```
consult('aespar.pro')
```

eingelesen werden. Der Befehl

```
compile('aesreg.reg', 'aesreg.txt')
```

veranlaßt dann die eigentliche Übersetzung. Natürlich können auch andere Dateinamen angegeben werden. Mit dem Dateinamen 'user' wird in PROLOG das Terminal bezeichnet. Der Befehl

```
compile(user,user)
```

ermöglicht das interaktive Arbeiten mit dem Übersetzungsprogramm: die Wissensbasis wird Satz für Satz über die Terminaltastatur eingegeben, nach jedem eingegeben Satz erscheint dessen Übersetzung auf dem Terminal. Die Eingabe von `end_of_file` bzw. *Ctrl-Z* schließt die Eingabedatei 'user'.

Für die im Anhang abgedruckte Datei 'aesreg.reg' benötigt das Programm etwa fünf Minuten. Ein guter Teil der Zeit wird dabei von dem Durchrechnen des Abhängigkeitsgraphen verbraucht. Syntaktische Fehler werden erkannt und auf dem Bildschirm angezeigt.

4.2. Das Laufzeitsystem des AES

Nachdem die Wissensbasis in PROLOG übersetzt ist, ist sie auch im Prinzip ausführbar. Es fehlt nur noch die *Umgebung*, die die Prädikate 'letztergueltiger-Wert', 'charakterisiere' usw. zur Verfügung stellt. Das Prädikat 'charakterisiere'

wurde bereits in Kapitel 3.5 unter dem Namen *ch* definiert. Das Prädikat 'letzterGueltigerWert' kann aus dem Prädikat *apw''* bzw. *apw'''* aus Kapitel 3.6 gewonnen werden.
Das *Laufzeitsystem* des AES soll nun folgende Aufgaben erfüllen:

1. Entgegennahme und Abspeichern der Daten (Parameterwerte) vom AIS,
2. Bereitstellung der Prädikate, die in der übersetzten Wissensbasis nachgefragt werden,
3. Erstellung von Schlußfolgerungen aus den Daten unter Zuhilfenahme der übersetzten Wissensbasis und
4. Ausgabe der zutreffenden Regeln an das AIS zur dortigen Darstellung.

Das Konzept zu Punkt 1 wurde in Kapitel 3.6 behandelt. Die vom AIS kommenden Daten werden aus dem Parameternamen, seinem Wert und der Eintragungszeit bestehen. Das Laufzeitsystem wird diese Daten entgegennehmen und als PROLOG-Fakten in der Form 'letzterWert(Parameter,Wert,Zeit)' abspeichern. Damit ist das Prädikat *pw* aus Kapitel 3.6 realisiert. Darauf aufbauend kann das Prädikat 'letztergueltigerWert' definiert werden:

```
letztergueltigerWert(Param, Wert) :-
  letzterWert(Param, Wert, EZ),
  not ungueltig(Param),
  aktuelleZeit(AZeit),
  gueltigkeitsdauer(Param, bisveraendert),
  !.

letztergueltigerWert(Param, Wert) :-
  letzterWert(Param, Wert, EZ),
  not ungueltig(Param),
  aktuelleZeit(AZeit),
  gueltigkeitsdauer(Param, GD),
  numeric(GD),
  ZeitDifferenz is AZeit-EZ,
  ZeitDifferenz =< GD,
  !.
```

Die erste Prozedur behandelt den Fall, daß die Gültigkeitsdauer des Parameters, dessen letzter gültiger Wert nachgefragt wird, "bis verändert" ist (vgl. Kap. 3.6). In diesem Fall ist der letzte gültige Wert immer der letzte Wert, falls dieser nicht durch die Eintragung eines anderen Parameters als ungültig anzusehen ist, was durch die Ungültigkeitsdefinitionen in der Sprache AES/L spezifiziert werden kann (vgl. Kap. 3.6). Abgefragt wird dies mit "not ungueltig(Param)". Die zweite Prozedur behandelt den Fall, daß die Gültigkeitsdauer des Parameters eine Zahl ist. Die aktuelle Zeit wird von dem Laufzeitsystem als das Maximum der Eintragungszeiten der Parameterwerte vom AIS angenommen und in dem Fakt

'aktuelleZeit(X)' abgespeichert. Dies geschieht immer dann, wenn ein neuer Parameterwert vom AIS zum AES geschickt wird.

Zur Erstellung der Schlußfolgerungen werden die Diagnose- und Therapielisten aus der übersetzten Wissensbasis herangezogen. Jedesmal, wenn ein neuer Parameterwert vom AIS kommt, merkt sich das AES die Diagnose- und Therapieregeln, in die der neue Parameterwert eingeht. Auf einen Befehl des AIS hin werden alle bis dahin als zu überprüfend markierte Regeln ausgeführt.

Die Texte der "dann"-Teile der zutreffenden Regeln können an das AIS als Antwort zurückgeschickt werden. Die angesprochenen Texte befinden sich jeweils in dem ersten Argument des Prädikats 'folgerung' (vgl. Kap. 4.1). Der genaue Ablauf kann den Programmausdrucken im Anhang entnommen werden, insbesondere in 'aesmain.pro', 'aesfg.pro' und 'aeslgw.pro'.

Ein/Ausgabeverhalten. Nach dem Starten des PROLOG-Interpreters und dem Einlesen der Module des Laufzeitsystems und der übersetzten Wissensbasis kann das AES mit dem Kommando 'main.' gestartet werden (vgl. Kap. 5). Es meldet sich mit dem Prompt "AES–" und zeigt damit, daß es bereit für Eingaben ist.
Die wichtigsten Eingaben sind:

(1.) `eintrage(`*Parameter*`,` *Wert*`,` *Zeit*`)` und
(2.) `folgerung.`

Mit dem ersten Kommando kann das AIS seine Parameterwerte eintragen und mit dem zweiten Kommando kann es die daraus folgenden Diagnosen und Therapieempfehlungen anfordern.
Ferner gibt es die Kommandos:

(3.) `erklaere(`*Typ*`,` *Nr*`,` *Zeit*`)` ,
(4.) `ftext(`*Typ*`,` *Nr*`,` *Zeit*`)` und
(5.) `stop.`

Die Kommandos (3.) und (4.) werden im Kapitel über die Erklärungskomponente erläutert. Das Kommando 'stop' beendet das AES. Alle Eingaben werden mit der PROLOG-Prozedur 'read' eingelesen, alle Ausgaben werden mit den Ausgabeprozeduren von PROLOG vorgenommen. Wird das AES von einem Terminal gestartet, so erfolgt die Ein- und Ausgabe auch von diesem Terminal. Dies ist besonders sinnvoll zum Testen des AES. Einzelheiten über die Ein/Ausgabeprozeduren des AES können in der Datei 'aesio.pro' im Anhang nachgelesen werden.

4.3. Anbindung an das AIS

Das AES ist wie in Kapitel 4 erläutert auf einem eigenen Rechner, dem CADMUS 9200, realisiert. Eine Implementierung auf dem Rechner des AIS erscheint aus drei Gründen als nicht empfehlenswert:

A) durch das AES würden die Antwortzeiten des AIS verschlechtert,
B) der AIS-Rechner hat einen sehr kleinen Adreßraum, der das Arbeiten mit großen PROLOG-Programmen fast unmöglich macht und
C) für den CADMUS ist ein PROLOG-Interpreter mit Arithmetik für reelle Zahlen verfügbar.

Daher ist ein Schema der Kommunikation zwischen AIS und AES zu entwickeln. Danach kann die Einbindung des AES in die Benutzerschnittstelle des AIS besprochen werden.

Kommunikation. Wie in Kapitel 4.2 erläutert, kann das AES auch ohne das AIS von einem Terminal aus betrieben werden. Dies ist möglich, weil das AIS aus der Sicht des AES tatsächlich die Rolle eines Terminals spielt.

Dies bedeutet: Eingaben werden als Zeichenketten an die Terminalschnittstelle des CADMUS übertragen, und Ausgaben des AES werden vom AIS von der gleichen Schnittstelle gelesen. Für das AES bedeutet diese Art der Lösung einen minimalen Aufwand, da es wie gewohnt die Eingaben von seinem "Terminal" erhält und dorthin auch seine Ausgaben schickt. Die Realisierung der Kommunikation auf der Seite des AIS ist allerdings etwas komplizierter. Zunächst sollen jedoch die technischen Einzelheiten der Verbindung der beiden Rechner beschrieben werden.

Eine serielle Schnittstelle des AIS-Rechners (LSI-11/73) wird über eine serielle Leitung mit einer sog. Terminalschnittstelle des AES-Rechners (CADMUS) verbunden. Die Adresse der Schnittstelle auf der LSI-11/73 ist 177240 oktal, die Übertragungsrate ist 9600 bps. Bei Einsatz des AIS im Operationssaal wird der AIS-Rechner über das Datenleitungsnetz des Klinikums mit dem CADMUS verbunden, der in den Räumen der Abteilung Ergonomie des Helmholtz-Instituts steht. Bild 4.3-1 zeigt die Rechnerkonfiguration des AIS unter Hinzunahme des AES. Es entspricht Bild 1.2-3 aus Kapitel 1.2 mit Ausnahme des AES.

Module auf der AIS-Seite. Beim Starten des AIS schickt es Kommandos über die serielle Leitung, um sich in den Rechner einzuschalten und das AES zu starten. Die Einschaltprozedur heißt LOGAES.PAS und ist im Anhang abgedruckt. Die Kommandos können dieser Prozedur oder dem Beispiel in Kapitel 5 entnommen werden. Sie wird wie alle hier besprochenen PASCAL-Prozeduren in das AIS-Programm eingebunden. Nach dem Starten des AES meldet sich dieses mit seinem Prompt "AES–" und zeigt damit, daß es bereit für Eingaben ist.

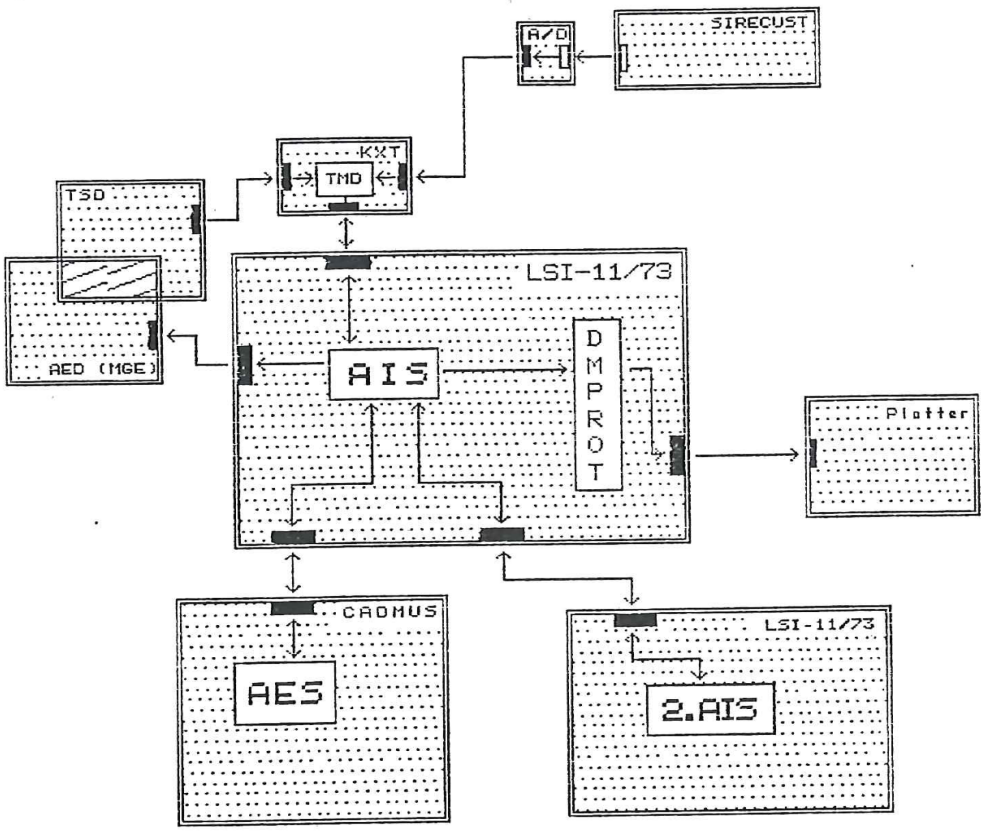

Bild 4.3-1: AIS mit AES

Das AIS erkennt den Zustand des AES mit Hilfe der Prozedur CHKAES.PAS. Diese Prozedur liest das letzte vom AES gekommene Zeichen aus der verbindenden Schnittstelle und weist im Abhängigkeit von diesem Zeichen der globalen Variablen 'AESstate' den Zustand zu. Falls CHKAES das Zeichen "–", das letzte Zeichen des AES-Prompts, in der Schnittstelle liest, so weist es der Variablen 'AESstate' den Zustand 'AESready' zu. Der Zustand 'AESworking' besagt, daß das AES nicht bereit ist für Eingaben. Er wird zugewiesen, wenn keines der Zeichen ">", "_", "^", "#", "&" oder "%" in der Schnittstelle steht. All diese Zeichen bis auf "–" werden vom AES auf die Schnittstelle geschrieben, wenn es etwas für das AIS übertragen will. Sie dienen der Synchronisation mit dem AIS, damit es in die entsprechende Prozedur zum Empfangen der Texte verzweigen kann. Die Bedeutung der einzelnen Zeichen steht in CHKAES.PAS. An dieser Stelle soll die Erklärung des Zeichens ">" genügen. Wenn das AES eine Entscheidungshilfe (=Liste von zutreffenden Diagnosen und Therapievoschlägen) berechnet hat, so schreibt es das Zeichen ">" auf die Terminalschnittstelle zum AIS. Die ständig aufgerufene Prozedur CHKAES.PAS erkennt das Zeichen und setzt

`AESstate := AESwaitingfortransmit''.`

Daraufhin verzweigt das AIS zur Prozedur RAPANS.PAS, die für das Einlesen der Entscheidungshilfe zuständig ist. RAPANS schreibt als erstes die Zeichenkette "send." und das ASCII-Zeichen *carriage return* auf die Schnittstelle. Auf dieses Kommando hin sendet das AES den Text der Entscheidungshilfe. Zur leichteren Weiterverarbeitung im AIS ist dieser Text mit Sonderzeichen in Einzelteile zerlegt (vgl. RAPANS.PAS, 'aesio.pro'). Er wird in der sogenannten Meldezeile des AIS-Bildschirms dargestellt (s. Kap. 5). Außerdem merkt sich das AIS von jeder Einzelfolgerung in der Antwort den Regeltyp ("d" für Diagnoseregel, "t" für Therapieregel) und die Regelnummer, die mit dem Text der Einzelfolgerungen vom AES übertragen werden. Daneben wird vom AES die Folgerungszeit für jede Folgerung übertragen. Diese Information wird für die in Kapitel 4.5 vorzustellende Erklärungskomponente des AIS benötigt.

Das Zeichen "^" wird in ähnlicher Weise für die synchronisierte Übertragung der Dosierempfehlungen benutzt. Dosierempfehlungen werden durch die Prozedur GETSUG.PAS eingelesen und weiterverarbeitet. Die restlichen Zeichen werden für die Erklärungskomponente benötigt.

Datenübertragung zum AES. Alle Eintragungen des AIS werden nicht direkt zum AES übertragen, sondern zunächst auf eine Pufferdatei ('PL.DAT') geschrieben. Die Datei hat den Datentyp "file of PL_record" und kann an beliebiger Stelle beschrieben oder gelesen werden. Die Datei wurde nicht im Rahmen dieser Diplomarbeit eingerichtet, sondern besteht schon für die Kommunikation des AIS mit dem Protokollierungsprogramm DMPROT (s. [HOF 85]). Die Datei PL.DAT wird als zirkuläre Warteliste verwaltet mit einem Lese- und Schreibzeiger. Jedesmal, wenn das AES bereit ist und die Warteliste nicht leer ist, wird das nächste Datum zum AES übertragen. Der genauere Ablauf kann den Dateien TASK3.PAS, GETPDF.PAS, PROTPL.PAS, PLSPEC.PAS, TOAES.PAS und GOAES.PAS entnommen werden, die jedoch nicht alle im Rahmen dieser Diplomarbeit entwickelt wurden, sondern an die zusätzlichen Anforderungen anzupassen waren. Das AIS schickt die Parameterwerte in dem Format

`eintrage(`*Parameter*`,` *Wert*`,` *Zeit*`).`

zum AES. Nach einer Gruppe von 'eintrage'-Befehlen folgt der Befehl 'folgerung.', woraufhin das AES die aus den Daten folgenden Diagnosen und Therapieempfehlungen berechnet. Ein Beispiel wird in Kapitel 5 gegeben.

4.4. Lemma-Generierung

Das in Kapitel 4.2 beschriebene Laufzeitsystem ist bereits in der Lage, das Problem der Ableitung von zutreffenden Diagnosen und Therapieempfehlungen aus der übersetzten Wissensbasis und den vom AIS gelieferten Parameterwerten zu lösen. Der scheinbare Widerspruch zwischen der Anforderung, aus Daten Schlußfolgerungen abzuleiten (Datenorientiertheit), und der Top-down-Strategie des PROLOG-Interpreters wurde mit Hilfe der vom Übersetzungsprogramm 'aespar.pro' berechneten Diagnose- und Therapielisten gelöst. In dieser Form hat das AES die Eigenschaft, daß es nicht *lernfähig* ist, d. h. ein PROLOG-Prädikat, das an verschiedenen Stellen aufgerufen wird, muß jedesmal neu bewiesen werden. Dies führt zu unnötigen mehrfachen Berechnungen. Das Konzept der *Lemma-Generierung* ist geeignet, hier Abhilfe zu schaffen (vgl. [HOG 84]). Die Idee ist einfach: ein bewiesenes Prädikat wird mit seinen Argumentbelegungen in die PROLOG-Datenbasis als Fakt eingetragen. Ein erneuter Aufruf des Prädikats kann dann auf dieses Fakt zugreifen.
Eine einfache PROLOG-Regel realisiert die Lemma-Generierung:

```
gib(X) :- lemma(X),!.
gib(X) :- call(X),asserta(lemma(X)).
```

In den PROLOG-Regeln schreibt man dann anstatt "p(X)" "gib(p(X))" für alle Prädikate "p", deren Berechnung mit Lemma-Generierung optimiert werden soll.

Wegen der dynamischen Situation ist die Realisierung beim AES etwas komplizierter. Prädikate, die von Parameterwerten abhängen, können falsch werden, wenn die Werte sich ändern oder sind als falsch anzusehen, wenn einer der eingehenden Parameter seine Gültigkeitsdauer überschreitet. Lemmas, die aufgrund neuer Parameterwerte oder einer Überschreitung der Gültigkeitsdauer eines eingehenden Parameters veraltet sind, müssen also gelöscht werden. Die Lösung für das Problem ist die Hinzufügung eines Argumentes für die Eintragungszeit und die Gültigkeitsdauer in jedem Prädikat, das von Parameterwerten abhängt. Dies fängt an bei dem Prädikat 'letztergueltigerWert', das nun wie folgt aussieht:

```
letztergueltigerWert(Param, Wert, (EZ,bisveraendert) ) :-
  letzterWert(Param, Wert, EZ),
  not ungueltig(Param),
  aktuelleZeit(AZeit),
  gueltigkeitsdauer(Param, bisveraendert),
  !.

letztergueltigerWert(Param, Wert, (EZ,GD) ) :-
  letzterWert(Param, Wert, EZ),
```

```
  not ungueltig(Param),
  aktuelleZeit(AZeit),
  gueltigkeitsdauer(Param, GD),
  numeric(GD),
  ZeitDifferenz is AZeit-EZ,
  ZeitDifferenz =< GD,
  !.
```

Alle anderen Prädikate erben ihre Eintragungszeit und Gültigkeitsdauer letztlich von 'letztergueltigerWert'. Das Beispiel der Diagnoseregel 4097 aus Kapitel 4.1 soll dies exemplarisch zeigen. Es wird jetzt für die Lemma-Generierung wie folgt von 'aespar.pro' übersetzt:

```
folgerung('Alles klar', d, 4097, [], TI) :-
  gib(letztergueltigerWert(a, P0, TI0)),
  gib(letztergueltigerWert(b, P1, TI1)),
  P0 .< P1,
  gib(charakterisiere(c, unter normal, TI2)),
  assign_min([TI0,TI1,TI2],TI).
```

Bei Aufruf von "folgerung('Alles klar',d,4097,[],TI)" werden die Variablen TI0 und TI1 mit der Eintragungszeiten und Gültigkeitsdauern der Parameter 'a' bzw. 'b' belegt. TI2 erhält die Eintragungszeit und die Gültigkeitsdauer, die sich aus den in das Prädikat 'charakterisiere' eingehenden Parameter (vgl. 'aesdp.pro') berechnet. Die Eintragungszeit des Prädikats 'folgerung' ergibt sich dann als das Maximum der Eintragungszeiten in den TI0, TI1 und TI2, die Gültigkeitsdauer wird so bestimmt, daß das Prädikat ungültig wird, wenn eines der eingehenden Prädikate ungültig wird (vgl. 'assign_min' in 'aesdp.pro').

Zur Zeit wird die Lemma-Generierung für die Prädikate

'letztergueltigerWert',
'charakterisiere' und
'folgerung'

eingesetzt. Neben der Effizienzerhöhung hat sie den Vorteil, daß man jederzeit die gerade zutreffenden Folgerungen (Diagnosen und Therapieempfehlungen) als Lemmas verfügbar hat. Die Definition des 'gib'-Prädikats kann in 'aeslemma.pro' im Anhang nachgelesen werden. Dort sind auch die Vorkehrungen zum Löschen veralteter Lemmas erklärt. Man beachte, daß nur solche Prädikate als Lemmas abgespeichert werden, die mindestens einmal in einer Berechnung benötigt werden. Wird ein neuer Parameterwert vom AIS eingetragen, so werden alle Lemmas, die von diesem Wert abhängen aus der Datenbasis entfernt. Bei jedem Zugriff auf ein Lemma wird getestet, ob es veraltet ist (vgl. Datei 'aeslemma.pro'). Veraltete Lemmas werden ebenfalls aus Speicherplatzgründen aus der Datenbasis entfernt. Wenn auf ein veraltetes Lemma nicht mehr zugegriffen wird, so verbleibt es in der Datenbasis. Dies ist sowohl vom Standpunkt der Implementierung als auch

aus prinzipiellen Überlegungen vertretbar, da es ja weiterhin richtig ist, daß das abgespeicherte Lemma zu seinem Beweiszeitpunkt wahr war.

Die Daten in den entfernten Lemmas werden vom AES zum Aufbau von Informationen über den zeitlichen Verlauf von Prädikaten genutzt. Mit Hilfe dieser Informationen werden die Konstrukte

Parameter ist *Eigenschaft* seit *Formel* min und
Folgerung seit *Formel* min

implementiert. Die Einzelheiten können der Datei 'aesverlauf.pro' im Anhang entnommen werden.

4.5. Die Erklärungskomponente

In der Einleitung dieser Diplomarbeit wird bei den Ausführungen zum Thema ausdrücklich die Implementierung einer Erklärungskomponente erwähnt. Die Erklärung der Entscheidungshilfen ist ein unverzichtbares Mittel, um überhaupt eine Akzeptanz bei den Anwendern des Systems, den Anästhesisten, erwarten zu können. Die Erklärungen sind wie die Entscheidungshilfen an der Benutzerschnittstelle des AIS zur Verfügung zu stellen.

Auf der Seite des AES stehen die Prozeduren zur Erklärung der Entscheidungshilfen in der Datei 'aestracer.pro'. Sie werden durch das Kommando

`erklaere(`*Typ*`,` *Nr*`,` *Zeit*`)`

vom AIS aus angesprochen. 'Typ' ist dabei der Typ der zu erklärenden Entscheidungshilfe ("d" für Diagnoseregeln, "t" für Therapieregeln), 'Nr' ist die Nummer der Regel und 'Zeit' ist die Zeit, zu der die mit Typ und Nummer eindeutig bezeichnete Regel bewiesen wurde. Alle drei Informationen werden vom AES dem AIS beim Übertragen der Entscheidungshilfen mitgeteilt (vgl. Kap. 4.3). Das Kommando veranlaßt das AES, zunächst den Text der spezifizierten Regel in der in AES/L geschriebenen Wissensbasis zu suchen und zu übertragen. Danach wird mit einer Rückverfolgung der übersetzten Regel die Begründung für das Zutreffen bzw. Nichtzutreffen der Regel generiert und zum AIS übertragen.

Auf der Seite des AIS ist die Prozedur TASK2.PAS für das Zusammenspiel mit 'aestracer.pro' und die Interaktion mit dem Benutzer zuständig. Die Erklärungskomponente ist vollständig in die Benutzerschnittstelle des AIS integriert: sie bildet eine eigene sog. *Bildseite*, auf der berührempfindliche Tasten für die Eingaben des Benutzers zur Verfügung stehen. Für die *Erklärungsseite* reichen drei Tasten: eine Pfeiltaste, um die zu erklärende Entscheidungshilfe auszuwählen, eine Taste, um die Erklärung vom AES anzufordern, und eine Taste, um die Erklärungsseite zu verlassen. Die Erklärungstexte werden in zwei Fenstern

der Erklärungsseite dargestellt. Die aktuell zur Erklärung ausgewählte Einzelfolgerung erscheint über diesen Fenstern und wird zusätzlich in der Meldezeile des AIS hervorgehoben (vgl. Bild 5-3 in Kap. 5). Die Interaktion auf der Erklärungsseite wird in Kapitel 5 erklärt.

Neben den aktuell im AIS dargestellten Entscheidungshilfen können auch alte Entscheidungshilfen erklärt werden. Alle Entscheidungshilfen (ohne Dosierempfehlungen) des AES werden vom AIS auf einer Datei mit Typ, Nummer und Zeit abgespeichert. Zum Aufsuchen der Einträge wurde die Prozedur PTASK6.PAS entwickelt. Will man alte Einträge erklärt haben, so konfiguriert man den Begriff "–AES–" auf einer der Arbeitsseiten des AIS (vgl. Kap. 1.2). Jede Entscheidungshilfe wird mit einem kleinen Strich in dem Meßwertfenster markiert und kann mit den Pfeiltasten über der FERTIG-Taste aufgesucht werden (siehe Bild 1.2-1) aufgesucht werden. Da die Entscheidungshilfen nur mit Typ, Nummer und Zeit abgespeichert sind, muß der zugehörige Text vom AES erfragt werden. Dies geschieht mit dem Kommando "ftext(Typ,Nr,Zeit)". Die Erklärung der aufgesuchten Entscheidungshilfe kann dann wie in Kapitel 5 erläutert durch Drücken des "Titelrechtecks"angefordert werden.

5. Ein Beispiel mit Rechenzeiten

In den vorigen Kapiteln ist das Konzept und die Realisierung des Anästhesie-Entscheidungsunterstützungssystems vorgestellt worden. Das medizinische Wissen wird in einer Sprache formuliert, deren Syntax so gewählt wurde, daß der durch die Schlüsselwörter suggerierte *Sinn* mit der formalen Semantik, die sich an dem Prädikatenkalkül orientiert, weitgehend übereinstimmt. Die Implementierung erfolgte in der Programmierspache PROLOG. Da das AES auf einem eigenem Rechner läuft, war ein Weg zu finden, auf dem das AIS mit dem AES kommunizieren kann. Es stellte sich heraus, daß eine einfache, jedoch sehr flexible Möglichkeit die ist, daß sich das AIS zu dem AES-Rechner (CADMUS) wie ein interaktiver Benutzer verhält, d. h. technisch: das AIS ist an eine Terminalschnittstelle des CADMUS angeschlossen.

Dieses Kapitel soll einen Eindruck von der Funktion des AES vermitteln. Zunächst wird das Verhalten des AES aus der Sicht des CADMUS vorgestellt. Dazu werden die Ein- und Ausgaben des AES, wie sie auf der erwähnten Terminalschnittstelle erscheinen, wiedergegeben. Verschiedene Ausführungszeiten sollen einen Eindruck von der Leistungsfähigkeit des Systems, insbesondere der vorgenommenen Optimierungen geben. Das Verhalten des AES aus der Sicht des AIS wird anhand einiger Abbildungen erläutert.

Ein-/Ausgabeverhalten. Im folgenden wird der Verlauf einer Sitzung mit dem AES mittels eines gekürzten Protokolls wiedergegeben. `Eingaben` in das AES und Ausgaben des AES werden durch Schrifttypen unterschieden. Kürzungen werden durch kursiv geschriebene *Erläuterungen* vermerkt. Sonderzeichen werden durch eckig geklammerte Zahlen dargestellt, z.B. "[1]" für das ASCII-Zeichen Nummer 1, die mit "←—ⓘ" markierten Stellen beziehen auf die später anzugebenen Ausführungszeiten. Die Funktion und Bedeutung der Ein- und Ausgaben wurde im vorigen Kapitel erläutert. Die programmtechnischen Einzelheiten können den Ausdrucken im Anhang entnommen werden. Die dort ganz am Anfang aufgeführte Datei 'aes.doc' enthält eine Kurzbeschreibung der Module und kann somit als Wegweiser für den interessierten Leser dienen.

```
hello.
Password: carriage-return
Login incorrect
login: ais
Password: ...

... Begrüßung durch den CADMUS-Rechner

$date
Wed Jul 16 10:45:25 MET 1986
$startaes
IF/Prolog Version 3.0 pf Thu May 29 19:02:55 EST 1986
```

Copyright (C) 1984, 1985, 1986 InterFace Computer GmbH
?- ['aes.txt'].
... Meldungen von IF/Prolog über eingelesene Dateien
yes

?- main.
AES-eintrage('Geschlecht', weiblich, 510.77).
AES-eintrage('Alter', 50.00, 510.94).
AES-eintrage('Laenge', 170.00, 511.07).
AES-eintrage('Gewicht', 60.00, 511.30).
^send.
[2]HEPARIN 180[3]
... weitere Dosierempfehlungen

AES-eintrage('KO', 1.77, 511.30).
^send.
[2]SUCCINYL 70.8[3]
AES-folgerung.
>send.
[22][2]Empfohlenes Min.Vol. = 6.92 bei Frequenz=15, [3]t[0]15[1]511.3
[2]Empfohlenes Min.Vol. = 6.35 bei Frequenz=12, [3]t[0]16[1]511.3[4]
Zeit: 9 sec ←—①
AES-eintrage('KO', 1.78, 511.97).
^send.
[2]SUCCINYL 71.2[3]
AES-folgerung.
>send.
[22][2]Empfohlenes Min.Vol. = 6.95 bei Frequenz=15, [3]t[0]15[1]511.97
[2]Empfohlenes Min.Vol. = 6.38 bei Frequenz=12, [3]t[0]16[1]511.97[4]
Zeit: 0 sec ←—②
AES-eintrage('pH', 7.34, 512.68).
AES-eintrage('PCO2', 39, 512.68).
AES-eintrage('BE', -2.9, 512.68).
AES-folgerung.
>send.
[22][2]Azidose, [3]d[0]1[1]512.68
[2]dekomp. metabolische Azidose [3]d[0]7[1]512.68[4]
Zeit: 2 sec
AES-erklaere(d,7,512.68).
.#send.
[2]diagnoseregel 7 heisst
wenn
BE ist unter normal und
pH ist nicht fastnormal

```
  dann
    "dekomp. metabolische Azidose"[3]

&send.
[2]BE = -2.9 (8:32)
BE normal : -2.5 ... 2.5
pH fastnormal : 7.35 ... 7.45
pH = 7.34 (8:32)
==> Regel trifft zu[3]
AES-stop.

Echte Ableitungen  : 19
Zugriffe auf Fakten : 265
Veraltet : 0
Weggelegt : 3
31% of database used
?- end.

$login
login:
```

Zum Verständnis der Ausgaben des AES (Folgerungen und Dosierempfehlungen) möge man in der Datei 'aesreg.reg' im Anhang nachschauen, die die Wissensbasis im Klartext enthält. Die Eingaben in das AES verstehen sich jeweils inklusive *carriage-return* (ASCII-Zeichen Nummer 13). Wenn das AIS gestartet wird, versucht es, über die verbindende serielle Leitung das AES auf dem CADMUS in Gang zu bringen. Dazu schickt es zunächst

```
hello.
```

zum CADMUS. Im Normalfall ist auf der angeschlossenen Terminalstelle kein Benutzer eingeschaltet, so daß der CADMUS das "`hello.`" als Benutzerkennzeichnung (*Log-In*) interpretiert und das Passwort erfragt. An dieser Meldung erkennt das AIS, daß die Einschaltprozedur vollständig durchgeführt werden muß, und uberspringt das Passwort durch die Eingabe eines *carriage-return*. Dies veranlaßt den CADMUS, erneut die Benutzerkennzeichnung zu erfragen. Das AIS gibt nun die korrekten Wörter an, und der CADMUS meldet sich mit '`$`', dem sogenannten *Shell-Prompt*, der anzeigt, daß der Rechner bereit ist zur Entgegennahme von Kommandos. Das AIS erfragt zunächst das Datum und die Uhrzeit[10] mittels

```
date
```

,

startet den PROLOG-Interpreter mit dem Kommando

```
startaes
```

,

[10] Der CADMUS ist mit einer batteriegepufferten Echtzeituhr ausgerüstet

befiehlt das Einlesen der Programmmodule und der Wissensbasis mit

```
['aes.txt']
```

und startet schließlich das AES mit der Eingabe

```
main.   .
```

Daraufhin meldet sich das AES mit seinem Prompt ("AES–"), der analog zum *Shell-Prompt* anzeigt, daß es jetzt bereit ist zur Entgegennahme von Eingaben und Befehlen.

In der hier abgebildete Sitzung werden als erstes die Patientenparameter *Geschlecht, Alter, Länge* und *Gewicht* eingegeben. Das erste Argument des Befehls

```
eintrage(X,Y,Z)
```

enthält den Parameternamen, der zweite den Parameterwert und das dritte die Uhrzeit in Minuten, zu der der Parameterwert im AIS eingegeben wurde (510.77 entspricht etwa 8:31). Die Eintragung des *Patientengewichts* und der *Körperoberfläche* (`'KO'`) führt zu einer Reihe von Dosierempfehlungen des AES, von denen hier nur die Empfehlungen für *Heparin* und *Succinyl* wiedergegeben sind. Das AES teilt dem AIS durch das Zeichen "^" mit, daß es eine Dosierempfehlung ubertragen will. Das AIS fragt ständig die Schnittstelle zum AES ab und verzweigt in eine entsprechende Prozedur, wenn das Zeichen "^"in der Schnittstelle steht. In dieser Prozedur schickt es dann

```
send.
```

zum CADMUS, woraufhin das AES die Dosierempfehlung überträgt. Die Sonderzeichen "[2]" und "[3]" klammern den *Nutztext* ein, um die Weiterverarbeitung im AIS zu erleichtern. Der Befehl

```
folgerung.
```

stößt das AES zur Ableitung aller jetzt zutreffenden Folgerungen an, die von den eben eingetragen Parametern abhängen. In diesem Fall treffen zwei nahezu gleichlautende Therapieregeln über eine Empfehlung für die Einstellung des *Minutenvolumens* zu. Analog wie bei den obigen Dosierempfehlungen erfolgt die Synchronisation mit dem AIS, diesmal mit dem Zeichen ">". Die Einteilung der Antwort des AES ist wie folgt:

- Die ganze Meldung ist mit den ASCII-Zeichen "[22]" und "[4]" eingeklammert.
- Der Text einer einzelnen Folgerung beginnt mit "[2]" und endet mit "[3]".
- Nach dem Zeichen "[3]" folgt die Identifikation des Regeltyps, und zwar:
  'd' für Diagnoseregeln und
  't' für Therapieregeln.
- Danach kommt das Zeichen "[0]", das von der Regelnummer gefolgt wird.
- Das Zeichen "[1]" schließlich kündigt die Folgerungszeit (in Minuten) an.

Das strenge Format ist notwendig, damit das AIS die Antwort weiterverarbeiten kann (vgl. Datei RAPANS.PAS im Anhang). In unserem Fall sind also die Therapieregeln 15 und 16 gefolgert worden (vgl. Datei 'aesreg.reg' im Anhang).

Die nächste Eintragung betrifft noch einmal die *Körperoberfläche*. Da sie mittelbar in die obigen beiden Folgerungen eingeht, werden diese noch einmal abgeleitet. Man beachte die erheblich verminderte Rechenzeit.

Die Eintragung der Laborwerte *pH, PCO_2* und *BE* führen zu den Diagnosen "Azidose" (Diagnoseregel 1) und"dekomp. metabolische Azidose" (Diagnoseregel 7). Der Befehl

```
erklaere(d,7,512.68).
```

fordert die Erklärung der zuletzt genannten Diagnose an. Dazu wird zunächst der Regeltext ausgegeben, wie er in der Wissensbasis 'aesreg.reg' steht. Danach folgt die Erklärung der Regel. Die Synchronisation erfolgt in diesem Fall über die Zeichen "#" und "&".

Der Befehl

```
stop.
```

beendet den AES-Lauf. Es wird noch eine Statistik ausgegeben, die den Nutzeffekt der Lemma-Generierung (vgl. Kap. 4.4.) angibt. Die Eingabe

```
end.
```

beendet den PROLOG-Interpreter und das Kommando

```
login
```

beendet die Sitzung auf dem CADMUS.

Ausführungszeiten. An dieser Stelle soll ein Eindruck von der Effizienz des erstellten Systems und der implementierten Optimierungen[11] gegeben werden. Auf eine formale Bestimmung der *Zeitkomplexität* (vgl. Thomas [THO 84]) wird hier aus folgenden Gründen verzichtet:

(1.) Das AES ist in PROLOG geschrieben; in PROLOG wird ein Algorithmus zum einen durch die Regeln in dem Programm, zum anderen durch die Auswertungsstrategie des verwendeten PROLOG-Interpreters bestimmt (vgl. Hogger [HOG 84]).

(2.) Es ist nicht klar, was als Programm zu gelten hat, und was den Daten zuzurechnen ist.

(3.) Die Maße der Größe des Suchbaums (vgl. Nilsson [NIL 82]) erfordern Wissen uber die Art des Suchbaums. Die jetzige Wissensbasis ist allerdings parallel mit dieser Diplomarbeit in Form von medizinischen Dissertationen ([JAC 86], [DAW 86]) erarbeitet worden und für solche Untersuchungen noch zu klein.

[11] Optimierung hier im Sinne von Effizienzerhöhung

(4.) Die hier vorgestellte Implementierung des AES ist ein erster Schritt am Helmholtz-Institut in Richtung Entscheidungsunterstützung. Bei der Implementierung wurde Wert auf die schnelle Bereitstellung eines Prototypen gelegt, der als Diskussionsgrundlage für weitere Entwicklungen dienen soll.

Shapiro [SHA 82] weist auf die enge Beziehung zwischen Komplexitätsmaßen auf *Logikprogrammen* und solchen auf *Alternierenden Turingmaschinen.* Für die Komplexität von Logikprogrammen definiert er drei Maße auf deren Beweisbäumen (*refutation trees*) und zeigt deren Zusammenhang mit Maßen auf Alternierenden Turingmaschinen. Dazu zeigt er, daß eine Alternierende Turingmaschine ein Logikprogramm simulieren kann und umgekehrt. Die Resultate sind allerdings hier nur von theoretischer Bedeutung, da die Bestimmung der von ihm vorgeschlagenen Komplexitätsmaße für Logikprogramme wegen (3.) kaum möglich erscheint.

Stattdessen werde ich die Antwortzeiten für zwei spezielle Anfragen des AIS an das AES angeben. Dieser Ansatz läßt zwar keine allgemeinen Aussagen über die Effizienz des dem AES zugrunde liegenden Algorithmus zu, gibt aber einen praktischen Eindruck von der Leistungsfähigkeit des AES. In der oben abgebildeten Sitzung sind die Stellen, auf die hier Bezug genommen wird, mit "⟵①" bzw. "⟵②" markiert. Die Eingabe, die zu der markierten Ausführungszeit geführt hat, ist jeweils der vorstehende Befehl "`folgerung.`". Das Beispiel ist so gewählt, daß in beiden Fällen die zwei "rechenintensivsten" Regeln in der Wissensbasis uberprüft werden. Im ersten Fall wird die Zeit für die erstmalige Ableitung der zwei Folgerungen gemessen, im zweiten Fall die Zeit für die erneute Ableitung, nachdem sich der Wert für einen der eingehenden Parameter (`'KO'`) geändert hat. In der nachstehenden Tabelle (Bild 5-1) werden die Ergebnisse für verschiedene Konfigurationen des AES wiedergegeben.

Die Konfigurationen betreffen die Möglichkeiten der

(a) *Compilation*: der hier verwendete PROLOG-Interpreter bietet die Möglichkeit, einzelne Module in Maschinencode (Voll-Compilation) bzw. in einen Zwischencode (Semi-Compilation) zu übersetzen; in der im Anhang abgedruckten Version des AES sind etwa 80 % des Programms voll- bzw. semicompiliert, die Wissensbasis ist nicht compiliert (vgl. Datei 'aes.txt' im Anhang)

(b) *Lemma-Generierung*: in Kapitel 4 hinreichend beschrieben

(c) *Statistik*: das AES-Programm führt über den Nutzeffekt der implementierten Lemma-Generierung eine Statistik.

Das auffälligste Resultat ist wohl der Effekt der Lemma-Generierung. Sie reduziert die Zeit für die erstmalige Ableitung auf weniger als die Hälfte und die Zeit für die zweite Ableitung auf weniger als eine Sekunde. Bemerkenswert ist auch der zumindest in diesem Beispiel geringe Einfluß der Compilation. Die Konfiguration "*compiliert,ohne Lemma-Gen.*" zeigt sogar eine unerwartete Anomalie: sie ist langsamer als die darüberstehende nicht compilierte Version. Die um etwa 4 Sekunden kürzeren Ausführungszeiten der zweiten Ableitung bei den zwei ersten Konfigurationen erklären sich aus der Tatsache, daß bei der Eingabe ① insgesamt

Typ	Eingabe	Zeit (sec)
nicht compiliert ohne Lemma-Gen.	①	25
	②	21
compiliert ohne Lemma-Gen.	①	27
	②	23
nicht compiliert mit Lemma-Gen. mit Statistik	①	11
	②	<1
compiliert mit Lemma-Gen. mit Statistik	①	9
	②	<1
nichtcompiliert mit Lemma-Gen. ohne Statistik	①	9
	②	<1
compiliert mit Lemma-Gen. ohne Statistik	①	7
	②	<1

Bild 5-1: Ausführungszeiten für zwei Anfragen

22 Regeln aus der Wissensbasis überprüft werden und bei der Eingabe ② nur noch zwei.

Leider bietet die aktuell verwendete Version des PROLOG-Interpreters nicht mehr die Möglichkeit, auf den eingebauten *Hash-Mechanismus*[12] Einfluß zu nehmen. Es wäre interessant, die Auswirkungen verschiedener Zugriffsarten auf die Datenbasis im Zusammenspiel etwa mit der Lemma-Generierung zu sehen. Die in Datei 'aesdecl.txt' im Anhang aufgeführten Einstellungen des Hash-Mechanismus hatten unter IF/Prolog V2.4 den Erfolg der Reduzierung der Rechenzeit auf etwa die Hälfte.

Die in der Tabelle dokumentierte geringe Auswirkung der Compilation steht in krassem Gegensatz zu der Leistungsfähigkeit der Compilation, wenn sie anhand eines sog. *Benchmark-Tests* zur Bestimmung der Maßzahl LIPS (*logic inferences per second*) des Interpreters gemessen wird. Mit LIPS wird die Geschwindigkeit angegeben, mit der zwei Listen konkateniert werden (nach Cuadrado [CUA 85b]). So bedeutet eine Angabe von 1000 LIPS für einen PROLOG-Interpreter, daß er in einer Sekunde Listen der Länge 1000 konkatenieren kann. IF/Prolog V3.0 schafft interpretiert 500 LIPS und compiliert 25000 LIPS. Implementierungen auf Spezialrechnern bringen es auf bis zu 425000 LIPS ([CUA 85a], [HOG 84]). Naturgemäß wird mit solchen einfachen Benchmark-Tests nur ein Teilaspekt der Leitungsfähigkeit des untersuchten Interpreters gemessen wird. Der Grund für das in unserem Beispiel festgestellte "Versagen" der Compilation liegt an der Tat-

[12] Das Prädikat "sethashaccess" war in IF/Prolog V2.4 noch verfügbar, nicht mehr in IF/Prolog V3.0

sache, daß hier wohl die extensive Suche in der Datenbasis den Hauptanteil an der Rechenzeit ausmacht.

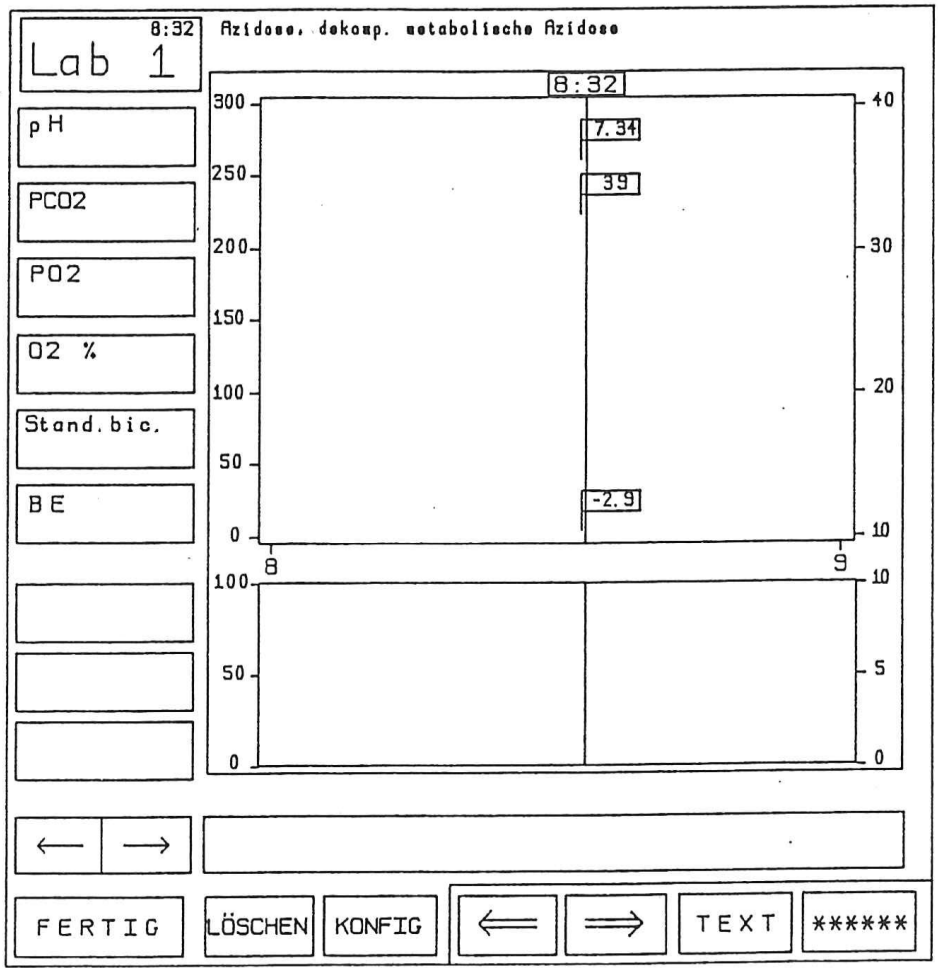

Bild 5-2: Darstellung von Folgerungen im AIS

Einbindung in das AIS. Die am Anfang dieses Kapitels wiedergegebene Sitzung geschah aus der Sicht des AES. Es führt alle Ein- und Ausgaben mit dem ihm vom Betriebssystem zugeordneten Lese- und Schreibgeräten durch, also standardmäßig mit dem angeschlossenen Terminal (= *Tastatur* + *Bildschirm*). Tatsächlich kann man interaktiv von einem Terminal mit dem AES kommunizieren, um z. B. neue Regeln oder Änderungen im Programm zu testen. Das AIS verhält sich aus der Sicht des AES genau wie ein Terminal, d. h. es macht Eingaben (Senden von Zeichen) und akzeptiert Ausgaben (Empfangen von Zeichen). Für den Benutzer des AIS ist dieser Aspekt völlig verborgen. Seine Eintragungen uber die Benutzerschnittstelle des AIS werden vom Programm ohne sein Zutun in dem korrekten Format (`'eintrage(X,Y,Z)'`) an das AES übertragen. Die vom AES berechneten Ausgaben werden über den in Kapitel 4 dargelegten Mechanismus zum AIS zurückgeschickt und dort auf dem Farbbildschirm dargestellt. Aus

der Sicht des Benutzer bedeutet dies, daß einige Sekunden nach seiner Eingabe die daraus folgenden Diagnosen und Therapieempfehlungen auf dem Bildschirm erscheinen. Bild 5-2 zeigt dies am Beispiel der Laborwerte pH,PCO_2 und BE.

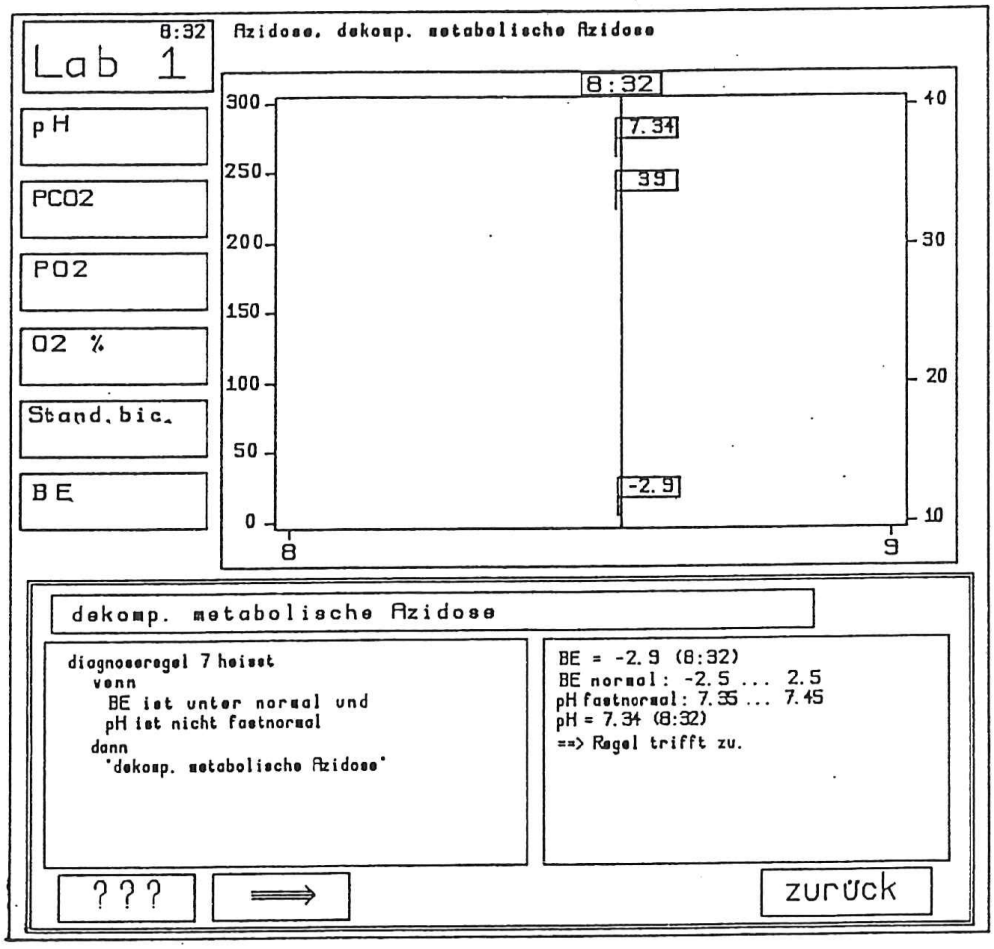

Bild 5-3: Erklärungsseite des AIS

Die Eintragungen führen zu den Diagnosen "Azidose" und "dekomp. metabolische Azidose". Sie werden über dem oberen Meßfenster dargestellt. Die Zeit, zu der diese Diagnosen gemacht wurden, steht links daneben in dem mit "Lab 1" beschrifteten *Titelrechteck*. Die korrespondierenden Eingaben in das AES stehen in der zweiten Hälfte der oben wiedergegebenen Sitzung. Der Benutzer hat die Möglichkeit, sich über die Benutzerschnittstelle des AIS eine Erklärung für die vom AES gemachten Schlußfolgerungen geben zu lassen. Dazu drückt er auf das Titelrechteck, was ihn zu der *Erklärungsseite* des AIS führt. Bild 5-3 zeigt die Erklärungsseite, die in die untere Hälfte der aktuellen Arbeitsseite eingeblendet wird. Der Benutzer kann mittels der Pfeiltaste die ihn interessierende Folgerung auswählen, um dann mit der Taste "???" links daneben deren Erklärung anzufordern. Die Erklärung besteht aus dem Regeltext, der im linken Fenster darge-

stellt wird, und aus der Begründung für das Zutreffen der Regel im rechten Fenster. Die Taste “zurück” bewirkt die Rückkehr zur aktuellen Arbeitsseite (Bild 5-2).

6. Zusammenfassung und Diskussion

In dieser Diplomarbeit war ein System zu entwickeln, das die Möglichkeit entscheidungsunterstützender Funktionen für das Anästhesie-Informationssystem (AIS) schafft.

Dazu wurde die Beschreibungssprache AES/L entworfen, in der das für die Entscheidungshilfen notwendige medizinische Wissen formuliert werden kann. Die Sprache wurde soweit als möglich von den technischen Details der Implementierung freigehalten. Oberstes Ziel war die leichte Lesbarkeit der *Sätze* von AES/L aus der Sicht des medizinischen Anwenders. Die Schlüsselwörter der Sprache sind so gewählt, daß die *Sätze* zwar nicht ganz die Syntax von Sätzen der deutschen Sprache haben, ihr jedoch nahe kommen. Zur formalen Definition der Bedeutung der Sprache wurde eine Transformation in prädikatenlogische Formeln angegeben. *Sätze* in AES/L sind also logische Aussagen, die eine an die Anwendung angepaßte Syntax haben.

Zur Realisierung wurde die Programmiersprache PROLOG herangezogen, deren Prozeduren logische Formeln, genauer: Horn-Klausen, sind. Der Hauptgrund für die Wahl von PROLOG war die Möglichkeit der Übersetzung der in AES/L geschriebenen Wissensbasis in PROLOG-Regeln, die direkt mit einem PROLOG-Interpreter ausgeführt werden können. Bei der Entwicklung des Übersetzungsprogramms konnte auf die zur Definition der Bedeutung von AES/L benötigte Transformationsfunktion zurückgegriffen werden. Zur Ausführung gelangt die übersetzte Wissensbasis in einem Laufzeitsystem, das die Umgebung, insbesondere die Parameterwerte, zur Verfügung stellt. Das Laufzeitsystem sorgt auch für die Kommunikation mit dem AIS, das die Parameterwerte liefert und dafür Entscheidungshilfen erwartet.

Zur Anbindung des AES in das AIS mußte ein Kommunikationsschema konzipiert und entwickelt werden. Dazu wurde eine Reihe von neuen Prozeduren für das PASCAL-Programm AIS geschrieben. Viele bestehende Prozeduren mußten an die neuen Anforderungen angepaßt werden. Die Entscheidungshilfen in Form von Diagnosen und Therapieempfehlungen werden auf dem Farbbildschirm des AIS dargestellt. Eine eigens für das AIS entwickelte Bildseite ermöglicht die Erklärung der vom AES kommenden Entscheidungshilfen.

Die praktische Anwendung des Systems fand parallel zu seiner Entwicklung statt. In der hier beschriebenen Ausbaustufe konnten in AES/L alles in den medizinischen Dissertationen ([JAC 86], [DAW 86]) erarbeitete Wissen formuliert und mit dem Laufzeitsystem zur korrekten Auswertung gebracht werden. Die Erprobung im Operationssaal ergab, daß das AES die vom AIS übermittelten Daten in hinreichend kurzer Zeit verarbeiten kann. Da das AES nur mit den sowieso im AIS eingetragenen Daten arbeitet, entsteht für den Benutzer des AIS kein zusätzlicher Aufwand. Die Entscheidungshilfen des AES kommen scheinbar spontan und vermitteln dem Benutzer den Eindruck eines "mitdenkenden" Systems. Dieser Eindruck wird verstärkt, wenn die Ausgaben des Systems aufgrund

automatisch erfaßter Meßwerte entstehen.

6.1. Vergleich mit anderen Arbeiten

Das AES heißt zwar Entscheidungsunterstützungssystem, es ist aber aufgrund der Problemstellung am ehesten mit Expertensystemen zu vergleichen. Die strikte Trennung des Expertenwissens von dem ausführenden Programm findet sich auch in dem System VM (vgl. Kap. 3.3). VM ist ein regelorientiertes System, d. h. das Wissen wird in Form von sog. *Produktionsregeln* formuliert. Produktionsregeln haben die Form

wenn *Bedingung* dann *Aktion*.

Der Bedingungsteil ist ein Test auf den *Zustandsraum*, der Aktionsteil verandert den Zustandsraum (vgl. Kap. 3.1). Zum Verstehen der Produktionsregeln ist also eine Vorstellung über den Zustandsraum notwendig.

Im Gegensatz dazu liegt die Bedeutung der Sprache AES/L allein in der Prädikatenlogik begründet. Insbesondere sind die "dann"-Teile der Regeln nicht als Ansammlungen von Anweisungen zur Zustandsänderung anzusehen, sondern sind die Konklusionen innerhalb von Implikationen. Dennoch kann man das AES auch als regelorientiertes System ansehen, wenn man die Arbeitsweise des PROLOG-Interpreters in Betracht zieht. Dem *Zustand* in regelorientierten Systemen entspricht die Liste der noch zu beweisenden Unterziele bei einem Beweisvorgang des AES. Den Produktionsregeln entsprechen die spezialisierten Resolutionsregeln, die entstehen, wenn man die Resolutionsregel (vgl. Kap. 3.2) für jede PROLOG-Regel aus der übersetzten Wissensbasis einzeln formuliert. Der Bedingungsteil entspricht dem Test, ob das Prädikat der PROLOG-Regel in der Zielklause vorkommt, der Aktionsteil ist der Ersatz des Prädikats in der Zielklause durch den Rumpf der PROLOG-Regel. Der Suchstrategie in einem regelorientierten System entspricht die (feste) Suchstrategie von PROLOG, der Anfangszustand ist mit dem Zielklause und der Endzustand mit der leeren Klause zu vergleichen.

Ein inhaltlicher Vergleich mit VM zeigt eine Reihe von Gemeinsamkeiten. Als erstes sind die symbolischen Bereiche für Parameterwerte zu nennen. Sie erhöhen sowohl die Lesbarkeit als auch die Allgemeinheit der Regeln. Aussagen über den zeitlichen Verlauf von Schlußfolgerungen (z. B. `"Azidose" seit 15 min`) sind ebenfalls in beiden Systemen möglich. Die Fähigkeit von VM, den aktuellen Patientenzustand zusammenzufassen, ist als Nebenprodukt der Lemma-Generierung im AES leicht und effizient zu implementieren.

Die Vorgehensweise der Übersetzung der Wissensbasis in ein internes Format ist bei VM wie beim AES zu finden. VM berechnet auch Listen für jeden Parameter, in welche Regeln er eingeht, um die Suchstrategie zu steuern. Dagegen ist

die Suchstrategie beim AES prinzipiell dem PROLOG-Interpreter überlassen.[13] Es ist durchaus denkbar, daß das AES auf einem PROLOG-Interpreter mit einer anderen als der in Kapitel 4 beschriebenen Auswertungsstrategie zum Ablauf gebracht wird.

Eine logikerhaltende Optimierung in Form der Lemma-Generierung führt de facto zu einer Mischung zwischen Bottom-Up- und Top-down-Berechnung (vgl. Ausführungen von Hogger [HOG 84] zu diesem Thema).

In dem von Futó et al. [FDS 78] entwickelten *Question-Answering*-System uber die Wechselwirkung von verschiedenen Medikamenten wird das medizinische Wissen direkt in PROLOG kodiert. Dieser Weg wäre zwar auch bei der Entwicklung des AES gangbar gewesen, er hat aber in unserer Anwendung zwei entscheidende Nachteile:

1. Um das medizinische Wissens zu erweitern, muß man zumindest Grundkenntnisse von PROLOG haben.
2. PROLOG ist eine Programmiersprache und in seiner Ausdrucksstärke viel mächtiger als es für die Formulierung des Wissens benötigt wird. Eine unkontrollierte Verwendung etwa logikfremder Bestandteile von PROLOG zerstört den gewünschten deklarativen Charakter der Wissensbasis.

Im Gegensatz dazu ist die Sprache AES/L eine Beschränkung (=Anpassung) auf die Mittel, die zur Beschreibung der Problemzusammenhänge benötigt werden. Das Verbot jeglicher Rekursion in AES/L (s. Kap. 4.1) ist in diesem Sinne zur Verhinderung des "Mißbrauchs" von AES/L als Programmiersprache zu sehen. Zudem können dadurch Endlosschleifen aufgrund rekursiver Regeln ausgeschlossen werden.

Die Tatsache, daß im AES gemachte Schlußfolgerungen aufgrund neuer Daten falsch werden können, erinnert an die sogenannte *nicht-monotone* Logik (vgl. [MCD 80]). In dieser Logik ist es erlaubt, daß eine Formel, die aus einer Formelmenge Σ ableitbar war, nicht mehr aus einer vergrößerten Formelmenge $\Sigma \cup \Delta$ ableitbar ist.

Trotzdem arbeitet das AES nicht nach den Regeln der nicht-monotonen Logik. Wenn das AES vom AIS den Befehl '`folgerung`' zur Ableitung von Schlußfolgerungen bekommt, so vollzieht sich die Berechnung ganz und gar nach den Regeln der klassischen monotonen Logik. Beweist das AES etwa das Zutreffen einer Diagnose, so gibt es an der Tatsache des Zutreffens in dem Kontext der *aktuellen Parameterwerte* nichts zu zweifeln. Dies drückt sich in der Implementierung besonders bei dem Punkt Lemma-Generierung aus: bewiesene Prädikate werden zusammmen mit der Zeit ihrer Ableitung und der Gültigkeitsdauer als Lemmas abgespeichert. Eine gemachte Schlußfolgerung mag zu einem späteren Zeitpunkt ungültig werden, es bleibt aber weiterhin richtig, daß sie zu ihrem Beweiszeitpunkt wahr war. Die Eintragung neuer Parameterwerte durch das AIS

[13] Gerade deshalb konnte die Sprache AES/L von Kontrollstruktur freigehalten werden.

bedeutet für die Logik des AES, daß eine neue Relation *APW* für den aktuellen Parameterwert (vgl. Kap. 3.5) aufgebaut wird. Gerade diese ständige Veränderung der Welt (der Parameterwerte) ist das dynamische Element unserer Anwendung, das sowohl bei der Interpretation von AES/L als auch bei der Implementierung des AES zu berücksichtigen war.

In Anbetracht der Gemeinsamkeiten des AES mit dem System VM könnte man einwenden, daß die Anpassung von VM an unsere Problemstellung oder die Verwendung eines sogenannten *Expertensystem-Shells* (=Expertensystem ohne Wissensbasis) auf einfachere Weise zu schnelleren Ergebnissen geführt hätte. Dazu ist anzumerken, daß man durch bestehende Systeme auf deren Ausdrucksmittel beschränkt wird, die fast nie mit den eigenen Anforderungen kongruent sind. Außerdem zeigt diese Diplomarbeit exemplarisch die Anwendbarkeit der Prädikatenlogik auf dem Gebiet der Entscheidungsunterstützung. Sie ist damit eine Alternative zu den weitverbreiteten regelorientierten Systemen.

6.2. Grenzen und mögliche Erweiterungen

Eine gewünschte Grenze des Systems ist die Ausdrucksstärke der Beschreibungssprache. Komplexe Diagnosen, die von einer Vielzahl von Symptomen mit gewissen Wahrscheinlichkeiten abhängen, sind wohl nur schwer in AES/L formulierbar. Zudem ist zu beachten, daß vielfach die vom AIS gelieferte Datenmenge für komplexe Aussagen nicht reichen wird. Auch für den Fall, das die Datenmenge reichen sollte, ist Vorsicht vor komplexen Aussagen geboten, da eine solche Entscheidungshilfe für den Anästhesisten in kurzer Zeit nachvollziehbar sein sollte.

Die Anwendung des AES ist m. E. verstärkt in der Überwachung automatisch erfaßter Meßwerte zu suchen, da die Beschreibung von abnormalen Werten i. a. vergleichsweise einfach ist. Ausdrücklich sei hier auf die Möglichkeit hingewiesen, daß die *Bereichsdefinitionen* in AES/L von Bedingungen abhängig gemacht werden können. So kann die Bedeutung von "pH ist normal" für verschiedene Situationen verschieden definiert werden. Hilfreich für die Erkennung von Situationen dürften die *Operationsereignisse* wie *Operationsbeginn, Anästhesiebeginn* usw. sein, die allerdings leider noch nicht korrekt vom AIS übertragen werden.

Für ein erweitertes Anwendungsgebiet mag durchaus der Sprachumfang um Konstrukte vergrößert werden, die eine leichtere Formulierung der Zusammenhänge auf den zusätzlichen Gebieten erlauben. Der jetzige Sprachumfang entstand aus der Notwendigkeit zur Formulierung von Wissen hauptsächlich aus dem Bereich der Interpretation von Laborwerten.

Falls die Wissensbasis sehr groß werden sollte, kann das Übersetzungsprogramm 'aespar.pro' als zu langsam empfunden werden. 'aespar.pro' wurde als *schneller Prototyp* entwickelt, bei dem die Effizienz nur eine untergeordnete Rolle

spielte. Einer späteren Implementierung des Übersetzungsprogramms nach den Grundsätzen des Compilerbaus steht jedoch nichts entgegen. Eine sehr große Wissensbasis könnte auch einen effizienteren Zugriff auf die Regeln erfordern. Falls dazu nicht der in IF/Prolog mögliche Hash-Zugriff reicht, muß über eine andere Abspeicherung der Regeln nachgedacht werden. Hier bieten sich Suchbäume mit den entsprechenden Algorithmen ([WIR 79]) an.

Ein erheblicher Zeitgewinn kann erwartet werden, wenn die Lemma-Generierung auf das Abspeichern von nicht zutreffenden Prädikaten ausgedehnt wird. Die "gib"-Prozedur aud Kapitel 4.4 sähe dann etwa so aus:

```
gib(X) :- nlemma(X),!,fail.
gib(X) :- lemma(X),!.
gib(X) :- call(X),asserta(lemma(X)),retract(nlemma(X)),!.
gib(X) :- asserta(nlemma(X)),!,fail.
```

Allerdings wird durch eine solche Erweiterung der Lemma-Generierung sowohl der Speicherbedarf für die Lemmas (und damit der Suchraum zum Auffinden der Lemmas) als auch der Aufwand zum Entfernen von nicht mehr zutreffenden Lemmas steigen.

Eine Alternative zu die weitgehend durch den PROLOG-Interpreter bestimmte Auswertungsstrategie ist die Implementierung der *heuristischen Suche* (vgl. Kap. 3.1). Der dazu notwendige Suchgraph entspricht praktisch dem in Kapitel 4.1 eingeführten Abhängigkeitsgraphen. Allerdings müßte das heuristische Kriterium erst noch erarbeitet werden. In dieser Diplomarbeit wurde die Verfahren der heuristischen Suche nicht verwendet, da der Implementierungsaufwand beträchtlich steigen würde. Außerdem hat sich die Auswertungsstrategie des PROLOG-Interpreters zusammen mit der Lemma-Generierung bisher als hinreichend schnell erwiesen. Falls das Prädikat 'sethashaccess' von IF/Prolog (vgl. Kap. 5) wieder verfügbar sein sollte, so kann mit der sorgfältigen Gestaltung des Hash-Zugriffs (nach heuristischen Kriterien !) auf PROLOG-Regeln das jetzige Verfahren auch für große Wissensbasen handhabbar gehalten werden.

Die aktuelle Implementierung des Sprachpartikels "oder" wird möglicherweise als inkorrekt empfunden. Dies sei an einem Beispiel erläutert. Gegeben sei die AES-Regel

```
diagnoseregel 11011 heisst
  wenn
    a < 1 oder b < 17
  dann
    "Regel erfuellt".
```

Die aktuelle Version des Übersetzungsprogramms macht daraus folgende PROLOG-Regel:

```
folgerung('Regel erfuellt', d,11011,[],TI) :-
  gib(letzterguelltigerWert(a,P0,LTI0)),
  gib(letzterguelltigerWert(b,P1,LTI1)),
  (P0 .< 1 ; P1 .< 17),
  assign_min([LTI0,LTI1],TI).
```

Bekanntlich wird durch ';' in PROLOG das logische "oder" bezeichnet. Wie man sieht, müssen zum Zutreffen der Regel für beide Parameter 'a' und 'b' gültige Parameterwerte definiert sein. Wenn dies als unsauber angesehen wird, so kann durch folgende Modifizierung der Übersetzung von *Bedingungen* Abhilfe erreicht werden:

(1) Die Bedingung wird in die disjunktive Normalform überführt.
(2) Für jedes Monom der disjunktiven Normalform wird eine eigene Übersetzung der Regel mit dem jeweiligen Monom als Bedingungsteil generiert.

Die obige Diagnoseregel würde dann so übersetzt:

```
folgerung('Regel erfuellt', d,11011,[],TI) :-
  gib(letzterguelltigerWert(a,P0,LTI0)),
  P0 .< 1,
  TI = LTI0.

folgerung('Regel erfuellt', d,11011,[],TI) :-
  gib(letzterguelltigerWert(b,P0,LTI0)),
  P0 .< 17,
  TI = LTI0.
```

Das PROLOG-Oder ';' wird also nicht mehr benötigt. Es wird durch die Übersetzung in mehrere PROLOG-Regeln ersetzt.

Der Vererbungsmechanismus für die Eintragungszeit und Gültigkeitsdauer (vgl. Kap. 4.4) kann grundsätzlich auch für die Attributierung der Prädikate mit Maßzahlen gebraucht werden. Als Beispiele sind Wahrscheinlichkeiten und Unsicherheitsfaktoren zu nennen. Die Einführung der *Fuzzy Logic* (vgl. [ZIM 84]) ist allerdings nicht ohne weiteres möglich, da dafür die logischen Operatoren ersetzt werden müssen.

Das AES wurde zwar für die Entscheidungsunterstützung des Anästhesisten bei einer Operation entwickelt, es ist jedoch auch für Anwendungsgebiete geeignet, in denen aus einer Menge von (automatisch erfaßten) Daten Schlußfolgerungen gezogen werden sollen. Ein Beispiel ist die Überwachung von Meßwerten einer Maschine.

7. Appendix: Syntax von AES/L

Das Appendix enthält die formale Definition der Sprache AES/L. Sie ist in einer Variante der *Backus-Naur-Form* angegeben (vgl. [A&U 72]). Nichtterminalsymbole sind mit "<" und ">" geklammert. Die Symbole "⟶" und "|" gehören zur Metasprache dieser Variante der Backus-Naur-Form. Alle anderen Zeichen sind Terminalsymbole. Wie üblich werden Kommentare nicht als Bestandteil der Spache angesehen. In der aktuellen Implementierung des AES werden Texte in AES/L durch das PROLOG-Programm 'aespar.pro' übersetzt, was dazu führt, daß in AES/L in der gleichen Weise wie in PROLOG kommentiert werden darf (siehe [IFA 86]). Das Nichtterminalsymbol < *carriage-return* > soll das ASCII-Zeichen Nummer 13 bezeichnen.

< *Wissensbasis* > ⟶ < *Satz* >< *Terminator* > |
< *Satz* >< *Terminator* > < *Wissensbasis* >

< *Satz* > ⟶ < *AES-Regel* > | < *Parameterdef* > | < *Dosierempf* > |
< *Bereichsdef* > | < *GDdef* > | < *UGdef* >

< *AES-Regel* > ⟶ < *Regeltyp* > < *natZahl* > **heisst wenn** < *Bedingung* >
dann < *Folgerung* >

< *Parameterdef* > ⟶ < *Parameter* > **:=** < *Formel* > |
wenn < *Bedingung* > **dann** < *Parameter* > **:=** < *Formel* > |
< *Parameter* > = < *Parameterwert* > |
wenn < *Bedingung* > **dann** < *Parameter* > =
< *Parameterwert* >

< *Dosierempf* > ⟶ **empfohlen** < *Parameter* > **:=** < *Formel* > |
wenn < *Bedingung* > **dann**
empfohlen< *Parameter* > **:=** < *Formel* >

< *Bereichsdef* > ⟶ < *Parameter* > **ist** < *Bereich* > **innerhalb** < *Formel* > **bis**
< *Formel* > | **wenn** < *Bedingung* > **dann** < *Parameter* >
ist < *Bereich* > **innerhalb** < *Formel* > **bis** < *Formel* >

< *GDdef* > ⟶ < *Parameter* > **ist gueltig** < *Dauer* > | **wenn** < *Bedingung* >
dann < *Parameter* > **ist gueltig** < *Dauer* >

< *UGdef* > ⟶ < *Parameter* > **ist ungueltig wenn** < *Parameterliste* >
eingetragen wird

< *Regeltyp* > ⟶ **diagnoseregel** | **therapieregel** | **hilfsregel**

$$
\begin{aligned}
<Bedingung> \longrightarrow\ & <Einzelbedingung> \mid (<Bedingung>) \mid \\
& <Bedingung> \textbf{ und } <Bedingung> \mid \\
& <Bedingung> \textbf{ oder } <Bedingung>
\end{aligned}
$$

$$
\begin{aligned}
<Einzelbedingung> \longrightarrow\ & <Folgerung> \mid <Folgerung> \textbf{ seit } <Formel> \textbf{ min} \mid \\
& <Parameter> \textbf{ ist bereits definiert} \mid \\
& <Parameter> \textbf{ ist noch undefiniert} \mid \\
& <Parameter> \textbf{ ist } <Eigenschaft> \mid \\
& <Parameter> \textbf{ ist } <Eigenschaft> \textbf{ seit } <Formel> \textbf{ min} \mid \\
& <Parameter> <Vergleichsop> <Wert> \\
& <Variable> <Vergleichsop> <Wert>
\end{aligned}
$$

$$
<Dauer> \longrightarrow \textbf{bis veraendert} \mid \textbf{fuer } <Formel> \textbf{ min}
$$

$$
<Parameterliste> \longrightarrow <Parameter> \mid <Parameter> \textbf{ , } <Parameterliste>
$$

$$
<Parameter> \longrightarrow <Atomname> \mid \textbf{gesamtmenge } <Atomname>
$$

$$
<Variable> \longrightarrow <Variablenname>
$$

$$
<Wert> \longrightarrow <Parameterwert> \mid <Formel> \mid <Variable>
$$

$$
<Folgerung> \longrightarrow <Folgerungsteil>
$$

$$
\begin{aligned}
<Folgerungsteil> \longrightarrow\ & <String> \mid <Folgerungsteil> \textbf{ , } <Formel> \mid \\
& <Formel> \textbf{ , } <Folgerungsteil> \mid \\
& <Folgerungsteil> \textbf{ , } <Formel> \textbf{ , } <Folgerungsteil>
\end{aligned}
$$

$$
\begin{aligned}
<Formel> \longrightarrow\ & <Zahl> \mid <Parameter> \mid <Variable> \mid (<Formel>) \mid \\
& <Vorzeichen><Formel> \mid <Funktion>(<Formel>) \\
& <Formel> <arithOp> <Formel> \mid
\end{aligned}
$$

$$
<Parameterwert> \longrightarrow <Zahl> \mid <String> \mid <Wert> : <Wert>
$$

$$
<Bereich> \longrightarrow <Atomname>
$$

$$
\begin{aligned}
<Eigenschaft> \longrightarrow\ & <Bereich> \mid \textbf{unter } <Bereich> \mid \textbf{ueber } <Bereich> \mid \\
& \textbf{nicht } <Bereich> \mid \textbf{nicht unter } <Bereich> \mid \\
& \textbf{nicht ueber } <Bereich>
\end{aligned}
$$

$$
<Atomname> \longrightarrow <Kleinbuchstabe><Name> \mid \text{'}Zeichenkette\text{'}
$$

$$
<Variablenname> \longrightarrow\ <Großbuchstabe><Name> \mid \_<Name>
$$

$$
\begin{aligned}
<String> &\longrightarrow \text{"} <Zeichenkette> \text{"} \\
<Zahl> &\longrightarrow <natZahl> \mid <Vorzeichen><natZahl> \mid <reelleZahl> \mid \\
&\quad\ <Vorzeichen><reelleZahl> \\
<natZahl> &\longrightarrow <Ziffer> \mid <Ziffer><natZahl> \\
<reelleZahl> &\longrightarrow <natZahl> \mid <natZahl>.<natZahl> \mid \\
&\quad\ <Ziffer>.<natZahl>\,\mathbf{E}\,<Vorzeichen><natZahl> \\
<Name> &\longrightarrow <NZeichen> \mid <NZeichen><Name> \\
<Zeichenkette> &\longrightarrow <Zeichen> \mid <Zeichen><Zeichenkette> \\
<Buchstabe> &\longrightarrow <Kleinbuchstabe> \mid\ <Großbuchstabe> \\
<NZeichen> &\longrightarrow <Buchstabe> \mid <Ziffer> \mid \_ \\
<Zeichen> &\longrightarrow <Buchstabe> \mid <Ziffer> \mid <Sonderzeichen> \\
<Funktion> &\longrightarrow \mathbf{exp} \mid \mathbf{ln} \mid \mathbf{log} \mid \mathbf{log2} \mid \mathbf{sqrt} \mid \mathbf{abs} \\
<Vorzeichen> &\longrightarrow\ + \mid - \\
<Ziffer> &\longrightarrow \mathbf{0} \mid \ldots \mid \mathbf{9} \\
<Kleinbuchstabe> &\longrightarrow \mathbf{a} \mid \ldots \mid \mathbf{z} \\
<Großbuchstabe> &\longrightarrow \mathbf{A} \mid \ldots \mid \mathbf{Z} \\
<Sonderzeichen> &\longrightarrow \mathbf{\&} \mid \mathbf{\%} \mid \ldots \mid \mathbf{)} \\
<Vergleichsop> &\longrightarrow\ = \mid \backslash = \mid < \mid > \mid =< \mid >= \\
<arithOp> &\longrightarrow\ + \mid - \mid * \mid / \mid \hat{} \\
<Terminator> &\longrightarrow .<carriage\text{-}return>
\end{aligned}
$$

8. Glossar

In diesem Glossar werden die technischen und medizinischen Begriffe erläutert, die in dieser Diplomarbeit eine Rolle spielen. Die (unvollständige) Definition der medizinischen Begriffe mag eine Hilfe zum Lesen der Wissensbasis (Datei 'aesreg.reg' in Anhang) sein.

ACT
: Blutgerinnungszeit ("activated clotting time"), liegt normalerweise bei ca. 125 sec und wird bei Betrieb der Herz-Lungen-Maschine durch Medikamentierung auf etwa 500 sec erhöht

AED
: Farbgraphiksystem der Firma Advanced Electronics Design mit einer Auflösung von 512×512 Bildpunkten

AES
: Anästhesie-Entscheidungsunterstützungssystem, das praktische Resultat dieser Diplomarbeit in Form eines PROLOG-Programms

AES/L
: Sprache der Wissensbasis des AES (vgl. Kap. 3.4 u. 7)

AIS
: einmal Bezeichnung für das gesamte Anästhesie-Informationssystem, im speziellen Bezeichnung für das PASCAL-Programm (vgl. Kap. 1.2)

BE
: Basenüberschuß ("base excess"), gibt an, wieviel Lauge man dem Blut zuführen muß, um den pH-Wert normal zu machen, kann aus pH und Stand.bic. errechnet werden; Laborwert

Blutdruck
: arterieller Blutdruck, gemessen in der *arteria radialis* am Handgelenk; man unterscheidet systolischen Blutdruck (Spitzendruck), diastolischen Blutdruck (Minimaldruck) und Mitteldruck; wird ständig von SIRECUST erfaßt und an das AIS übermittelt

CADMUS
: hier CADMUS 9230; Rechner der Firma PCS, auf dem das AES implementiert ist; Prozessor: Motorola MC68010, Betriebssystem: MUNIX (eine UNIX-Portierung)

Digitalplotter
: Ausgabegerät, mit dessen Hilfe das AIS ein Protokoll des Operationsverlaufs erstellt; zur Zeit wird ein Gerät der Firma Gould (Colorwriter 6300) eingesetzt

EKG

Elektrokardiogramm; Verlauf der Herzströme, gemessen mit drei Elektroden auf der Haut; dargestellt auf SIRECUST

Elektrolyte

im Blut gelöste Stoffe: Natrium (Na), Kalium (K), Kalzium (Ca) und Blutzucker (Bz)

Hb

Hämoglobin; Molekül in den roten Blutkörperchen, das die Sauerstofftransportkapazität bestimmt; Laborwert

Hk

Hämatokrit; Verhältnis der Blutkörperchen zum Flüssigkeitsvolumen des Blutes; Laborwert

KXT

Ein-Karten-Rechner der Firma DEC, der die automatische Meßwerterfassung und die Betreibung des TSD für das AIS durchführt

LAP

linksarterieller Druck, gemessen im linken Vorhof des Herzens; wird nur bei Herzklappenoperationen benötigt; wird ständig von SIRECUST erfaßt und an das AIS übertragen

LSI-11/73

16-Bit-Rechner der Firma DEC, auf dem das AIS implementiert ist; kompatibel mit PDP-11

MGE

am Helmholtz-Institut entwickeltes Farbgraphiksystem; weitgehend kompatibel mit AED

osophageale Temperatur

periphere Temperatur, genauer: Temperatur in der Speiseröhre; wird ständig vom SIRECUST erfaßt und an das AIS übertragen

$O_2\%$

Sauerstoffsättigungsgrad des Blutes; Laborwert

PEEP

positiver endexpiratorischer Druck; der Restdruck im Beatmungsschlauch, wenn der Patient ausgeatmet hat; wird am Respirator eingestellt

PCO_2

Kohlendioxidpartialdruck, Maß für das im Blut enthaltene Kohlendioxid; wie PO_2 ein Blutgas und im Labor bestimmt

pH
: Wasserstoff-Ionen-Konzentration im Blut, bestimmt zusammen mit BE und PCO_2 den Säure-Basen-Status des Blutes; Laborwert

PO_2
: Sauerstoffpartialdruck, Maß für den im Blut enthaltenen Sauerstoff; Laborwert

Puls
: hier eigentlich Herzfrequenz, wird aus dem EKG vom SIRECUST errechnet und an das AIS übertragen

rektale Temperatur
: Körperkerntemperatur; vom SIRECUST erfaßt und an das AIS übertragen

SIRECUST
: Überwachungsgerät (Monitor) der Firma Siemens, wie es am Aachener Klinikum für die Erfassung der Vitalparameter des Patienten eingesetzt wird; steht über den KXT mit dem AIS in Verbindung

Standardbicarbonat
: Maß für die "Pufferkapazität" des Blutes zur Aufnahme saurer Stoffe; Laborwert

Thrombozyten
: Blutplättchen (bewirken Blutgerinnung); Laborwert

TSD
: berührempfindliche Folie der Firma TSD Display Products Canada Inc., die die Funktion der virtuellen Tasten des AIS ermöglicht

Vitalparameter
: Meßwerte, die "lebenswichtige Hinweise" Hinweise über den Patientenzustand geben können [RED 85], also z.B. Blutdruck, Puls, Temperatur, ZVD

ZVD
: zentralvenöser Druck, wird in der Hohvene direkt vor dem Herzen gemessen; vom SIRECUST erfaßt und ans AIS übertragen

9. Literatur

[AED 83] **Advanced Electronics Design Inc.**
AED512/767 Color Graphics Terminals — User Manual
Part No. 990003-01 Rev-A
Advanced Electronics Design, California, USA, 1983

[A&U 72] **Aho,A.V., Ullman,J.D.**
The Theory of Parsing, Translation and Compiling
Vol. I: Parsing
Prentice-Hall, 1972

[BER 80] **Bernotat,R., Rau,G.**
Ergonomics in Medicine
in: Reul,H., Ghista,D.N., Rau,G. (eds.)
Perpectives in Biomechanics
Harwood Acad.Publ., New York, 1980, S.381-398

[CMC 82] **Clark,K.L., McCabe,F.G.**
PROLOG: a language for implementing expert systems
in: Hayes,J.E., Mitchie,D., Pao, Y.-H. (eds.)
Machine Intelligence Vol.10, 1982
Ellis Horwood Ltd.,Chichester,England, S.455-470

[C&M 81] **Clocksin,W.F., Mellish,C.S.**
Programming in Prolog
Springer-Verlag, 1981

[CUA 85a] **Cuadrado,C., Cuadrado,J.**
Prolog Goes to Work
BYTE Vol.10, No.8, August 1985, S.151-158

[CUA 85b] **Cuadrado,J.**
Private Kommunikation mit dem Autor
August 1985

[DAW 86] **Dawid,A.**
Beurteilung und Optimierung der Funktionen des Anästhesie-Informationssystems (AIS) in der Hand des Anästhesisten
med. Dissertation (in Bearbeitung), RWTH Aachen

[DEC 83] **Digital Equipment Corporation**
RT-11 V5, Vol.1-3
DIGITAL Software, Maynard, Massachusetts, USA, 1983

[DUD 80] **Dudenredaktion (Hrsg.)**
DUDEN — Die Rechtschreibung
18. Auflage, Bibl. Inst., 1980

[DZI 85] **Dzierzanowski,J., Bourne,J., Shiavi,R., Sandell,H., Guy,D.**
GAITSPERT: An Expert System for the Evaluation of Abnormal Human Locomotion Arising from Stroke
IEEE Transactions on Biomedical Engineering, Vol. BME-32, No.11, Nov. 1985, S.935-941

[FAG 80] **Fagan, L.**
VM: Representing Time-dependent Relations in a Medical Setting
Stanford University, USA, 1980

[FAG 84] **Fagan,L., Kunz,J., Feigenbaum,E., Osborn,J.**
Extensions to the Rule-Based Formalism for a Monitoring Task
in: Buchanan,B., Shortliffe,E.
Rule-Based Expert Systems
Addison-Wesley, 1984, S.397-423

[FDS 78] **Futó,I., Darvas,F., Szeredi,P.**
The Application of Prolog to the Development of QA and DBM Systems
in: Gallaire,H., Minker,J. (eds.)
Logic and Data Bases
Plenum Proc., 1978

[GEV 83] **Gevarter,W.B.**
Expert systems: limited but powerful
IEEE spectrum, August 1983, S.39-45

[HOF 85] **Hoffmann,S.**
Entwurf und Realisierung eines interaktiven Dokumenteneditors für ein klinisches Informationssystem
Diplomarbeit RWTH Aachen, 1985

[HOG 84] **Hogger,J.H.**
Introduction to Logic Programming
A.P.I.C. Studies in Data Processing No.21
Academic Press London, 1984

[H&U 79] **Hopcroft,J.E., Ullman,J.D.**
Introduction to Automata Theory, Languages and Computation
Addison-Wesley, 1979

[IND 83] **Indermark,K.**
Semantik von Programmiersprachen
private Mitschrift der Vorlesung im SS83, RWTH Aachen, 1983

[IND 85] **Indermark,K.**
Compilerbau I+II
private Mitschrift der Vorlesungen im SS 84 bzw. WS84/85,
RWTH Aachen, 1985

[IND 82] **Indermark,K., Fehr,E.**
λ-Kalkül und LISP
Ausarbeitungen des Proseminars im SS82, RWTH Aachen, 1982

[IFA 86] **InterFace Computer GmbH**
– IF/Prolog Version 3.0 Users Guide Rev.1.0
– IF/Prolog Version 3.0 Reference Manual Rev.1.0
InterFace Computer GmbH München, 1986

[JAC 86] **Jacob,W.**
Wissensgrundlagen für ein entscheidungsunterstützendes Anästhesie-Informationssystem
med. Dissertation (in Bearbeitung), RWTH Aachen

[J&W 78] **Jensen,K., Wirth,N.**
User Manual and Report – Second Edition
Springer-Verlag, 1978

[KLK 84] **Klocke,H., Trispel,S., Rau,G.**
Entwicklung einer Mensch-Rechner Schnittstelle unter Berücksichtigung ergonomischer Gesichtspunkte
Angewandte Informatik 5/84, 1984, S.197-208

[KTR 86] **Klocke,H., Trispel,S., Rau,G., Hatzky,U., Daub, D.**
An Anesthesia Information System for Monitoring and Record Keeping during Surgical Anesthesia
J. Clinical Monitoring 10/86 (erscheint im Oktober 1986)

[KOW 85] **Kowalski,R.A.**
Logic Programming
BYTE Vol.10, No.8, August 1985, S.161-177

[LEN 82] **Lenk,H.**
Zur Frage der apriorischen Begründbarkeit und Kennzeichnung der logischen Partikeln
in: Gethmann,C.F. (Hrsg.)
Logik und Pragmatik
Suhrkamp Taschenbuch Wissenschaft 399
Suhrkamp Verlag, 1982

[LÖB 82] **Löbner,H.**
Konzept, Entwicklung und Implementierung eines Makro-Assemblers für das Farbgraphiksystem MGE
Diplomarbeit RWTH Aachen, 1982

[MCD 80] **McDermott,J., Doyle,J.**
Non-Monotonic Logic I
Artificial Intelligence Vol. 13 (1-2), 1980, S.41-72

[MÖH 84] **Möhring,R.H.**
Graphentheorie
Skript zur Vorlesung im SS84, RWTH Aachen, 1984

[NIL 82] **Nilsson,N.J.**
Principles of Artificial Intelligence
Springer-Verlag, 1982

[OBE 83] **Oberschelp,W.**
Automatentheorie und formale Sprachen
private Mitschrift der Vorlesung im WS82/83, RWTH Aachen, 1983

[OBE 84] **Oberschelp,W.**
Datenbanken
private Mitschrift der Vorlesung im SS84, RWTH Aachen, 1984

[ORE 81] **Oregon Software Inc.**
PASCAL-2 – Version 2.0 for RT-11 — User Manual
Oregon Software, Portland, Oregon, USA, 1981

[ORE 83] **Oregon Software Inc.**
PASCAL-2 V2.1 for RT-11 — User Manual
Second Edition, Update No.2
Oregon Software, Portland, Oregon, USA, 1983

[PCS 84a] **Pheriphere Computersysteme GmbH**
PROLOG Manual
Doc.-No. G0930223, PCS München, 1984

[PCS 84b] **Pheriphere Computer Systeme GmbH**
MUNIX Manual, Vol.Ia: Commands
Doc.-No. G0929509, PCS München, 1984

[PPS 81] **Plessey Peripheral Systems**
PT-100B Video Terminal Manual
Part No. MA100264-B
Plessey Peripheral Systems, Irvine, California, USA, 1981

[RAU 82] **Rau,G., Trispel,S.**
Ergonomic Design Aspects in Interaction between Man and Technical Systems in Medicine
Med.Progr.Technol. 9, S.153-159, Springer-Verlag, 1982

[RSZ 79] **Rauhut,B., Schmitz,N., Zachow,E.-W.**
Spieltheorie
Leitfäden der Angewandten Mathematik und Mechanik, Band 49
Teubner Studienbücher Stuttgart, 1979

[RED 85] **Redecker,T.**
Untersuchung von Handlungs- und Entscheidungsabläufen des Anästhesisten zur Strukturierung eines Informationssystems unter Berücksichtigung ergonomischer Aspekte
Centaurus-Verlagsgesellschaft Pfaffenweiler, 1985

[RIC 78] **Richter,M.M.**
Logikkalküle
Leitfäden der Angewandten Mathematik und Mechanik, Band 43
Teubner Studienbücher Stuttgart, 1978

[RIC 84] **Richter,M.M.**
Methoden der künstlichen Intelligenz
private Mitschrift der Vorlesung im WS83/84, RWTH Aachen, 1984

[RIC 85] **Richter,M.M.**
Funktionale Programmiersprachen
private Mitschrift der Vorlesung im SS85, RWTH Aachen, 1985

[RIC 86a] **Richter,M.M.**
Logikprogrammierung
private Mitschrift der Vorlesung im WS85/86, RWTH Aachen, 1986

[RIC 86b] **Richter,M.M.**
Abstraktion in der Künstlichen Intelligenz
erscheint in: Gatzemeier,M.
Aspekte der Abstraktionstheorie
Aachener Schriften zur Wissenschafttheorie, Logik und Logikgeschichte, Heft 3

[RIC 81] **Richter,M.M., Schlösser,B., Schwarz,D.**
Modelltheorie
Schriften zur Informatik und Angewandten Mathematik,
RWTH Aachen, 1981

[ROB 65] **Robinson, J.A.**
A machine-orientet logic based on the resolution principle
Journal of the ACM, 12, S.23-41

[SCH 85] **Schlieter,R.**
Entwicklung eines Daten-Ein/Ausgabe-Prozessors für ein Anästhesie-Informationssystem
Diplomarbeit RWTH Aachen, 1985

[SHA 82] **Shapiro,E.Y.**
Alternation and the Computational Complexity of Logic Programs
Proceedings of the First International Logic Programming Conference, 14.-17. Sept. 1982, Faculté de Sciences de Luminy, Marseille, France

[STO 85] **Stoyan,H.**
Programmierstile in der Künstlichen Intelligenz und ihre Symbiose in Expertensystemen
Vortrag RWTH Aachen, 27.11.1985

[S&H 85] **S&H Computer Systems Inc.**
TSX-Plus Reference Manual, Fifth Edition
S&H, Nashville, Tennessee, USA, 1985

[THO 84] **Thomas,W.**
Schnelle Algorithmen
Skript zu Vorlesung im SS84, RWTH Aachen, 1984

[THF 84] **Thomas,W., Fehr,E.**
PROLOG und Logikprogrammierung
Ausarbeitungen des Seminars im WS83/84, RWTH Aachen, 1984

[TRI 82] **Trispel,S., Klocke,H., Günther,K., Rau,G., Schlimgen,R., Redecker,T.**
Operation Simulation for the Evaluation and Improvement of a Medical Information System
Proceedings "Analysis, Design and Evaluation of Man-Machine Systems" Baden-Baden, 27.-29. Sept. 1982, S.233-239

[TKR 85] **Trispel,S., Klocke,H., Rau,G., Schlimgen,R.**
Towards a Coherent Structure of the Anesthesist-Computer Interface
in: Osswald,P.M. (ed.)
Computers in Critical Care and Pulmonary Medicine
Springer-Verlag, 1985, S.29-37

[WIR 79] **Wirth,N.**
Algorithmen und Datenstrukturen
Leitfäden der Angewandten Mathematik und Mechanik, Band 31
Teubner Studienbücher Stuttgart, 1979

[ZIM 78] **Zimmermann,H.-J., Hamacher,H.**
Operations Research I
Skriptum zur einführenden Vorlesung
4. Auflage, RWTH Aachen, 1978

[ZIM 84] **Zimmermann,H.-J.**
Entscheidungstheorie Scriptum
Vorlesungsskript zur gleichnamigen Vorlesung im WS84/85
2. Auflage, RWTH Aachen, 1984