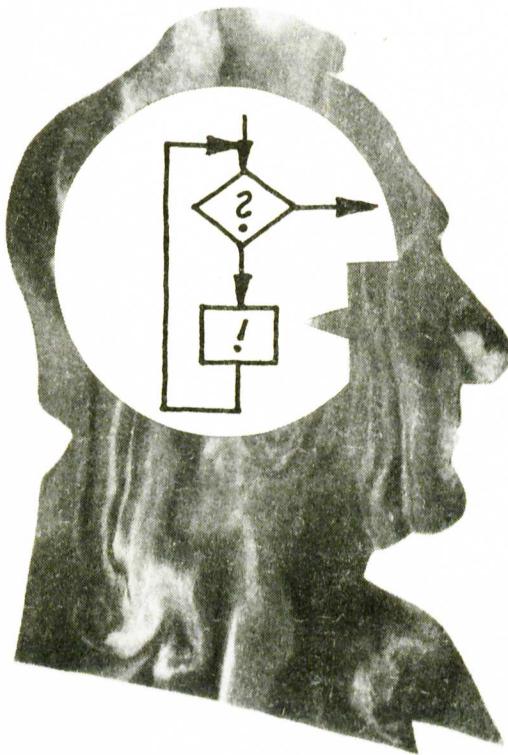


D 582/89 (3878) 500 II-16-8

Expertensysteme für die Nervenklinik



Schweriner BNK-Schriften
Heft 5
1989

Bezirksnervenklinik Schwerin
Direktor: OMR Prof.Dr.sc.med.K.Giercke

Expertensysteme für die Nervenklinik

Schweriner BNK-Schriften

Heft 5

1989

<u>Inhalt:</u>	Seite
Vorwort	3
0. Überblick	4
1. Theorie	
1.0. Übersicht	6
1.1. Wissen	
1.1.1. Objekte und Merkmale	8
1.1.2. Fakten und Regeln	10
1.1.3. Präzises und nicht-präzises Wissen	12
1.2. Schließen	
1.2.1. Vorwärtsschließen und Rückwärtsschließen	16
1.2.2. Tiefensuche und Breitensuche	18
1.2.3. Monotones und nicht-monotonen Schließen	20
1.3. Expertensysteme	
1.3.1. Aufgaben	22
1.3.2. Aufbau	24
1.3.3. Arten	26
2. Praxis	
2.0. Übersicht	28
2.1. Programmiersprachen	
2.1.1. BASIC	30
2.1.2. PASCAL	36
2.1.3. PROLOG	40
2.2. Programmiersysteme	
2.2.1. XPRO	46
2.2.2. GURU	48
2.2.3. RULEMASTER	50
2.3. Beispiele	
2.3.1. MYCIN	52
2.3.2. SPES	54
2.3.3. PPSA	56
3. Ausblick	
3.0. Übersicht	58
3.1. Software	58
3.2. Hardware	60
4. Quellen	62

Vorwort

"Wissenschaftlichkeit in der Arbeit stellt einen ganzen Komplex an Anforderungen, beginnend an die medizinische Forschung über das Niveau der Ausbildung und Erziehung, der Weiter- und Fortbildung bis hin zum wissenschaftlichen Leben in den Einrichtungen selbst. Dabei geht es stets um eine wissenschaftlich begründete Prophylaxe, Diagnostik, Therapie und Rehabilitation... Leitung und Planung erfordern aktuelle, komplexe und adäquate Informationen. Die Möglichkeiten der modernen Rechentechnik helfen und zwingen uns, diese Fragen zu lösen."

OMR Prof.Dr.sc.med.L.Mecklinger 1986

"Gefordert ist bei allen medizinischen Entscheidungen ein Höchstmaß an Wissenschaftlichkeit im Dienste des Patienten... Es ist notwendig, daß die Fragen der wissenschaftlichen Arzneimitteltherapie als eine ständige Schwerpunktaufgabe in die Leistungstätigkeit fest eingeordnet werden."

OMR Prof.Dr.sc.med.K.Thielmann, Minister für Gesundheitsw. 1989

Die Schweriner BNK-Schriften sind einer unserer Beiträge zur Erfüllung dieser Aufgaben.

Die Schrift "Expertensysteme für die Nervenklinik" erscheint
- als Heft 5 der "Schweriner BNK-Schriften"
- als Begleitmaterial für die Tagung der Arbeitsgemeinschaft "Organisation und Leitung psychiatrischer Einrichtungen", Arbeitsgruppe "Rechnerarbeitsplätze", in der Gesellschaft für Krankenhauswesen der DDR, veranstaltet vom 26. bis 28. September 1989 in Neuruppin
- als Lehrmaterial für den Lehrgang "Expertensysteme" der Bezirksakademie des Gesundheitswesens Schwerin, veranstaltet vom 28. bis 30. November 1989 in Schwerin
- als Informationsmaterial für die Nervenkliniken.

Die Schrift gibt einen ersten Einblick in Theorie und Praxis der Expertensysteme, einer neuen Art der Rechnernutzung, insbesondere zur Wissensverarbeitung für Diagnose und Therapie.

Wir wünschen uns fördernde Kritik.

Wir danken herzlich

- Dr.sc.techn.D.Herrig,
Bezirksnervenklinik Schwerin, Computerstation,
für Manuskriptentwurf und Manuskripterstellung
- Dipl.-Math.H.Wagner,
Büro für Sozialhygiene, EDV-Beauftragter des Bezirkssarztes,
für Manuskriptdurchsicht und Verbesserungsvorschläge
- Dipl.-Ing.F.Scharffenberg,
Ingenieurbetrieb für Anwendung der Mikroelektronik Schwerin,
für Konsultationen zu PROLOG
- stud.ing.H.Malberg,
Praktikant von der Technischen Hochschule Ilmenau,
für das Schreiben der PASCAL-Programme und der Blockbilder
- Frau S.Jakubczyk für das Tuschezeichnen der Bildvorlagen
- Frl. M.Herrig für den Entwurf des Titelbildes.

OMR Prof.Dr.sc.med.J.Giercke
Ärztlicher Direktor der BNK

0. Überblick

Eine Studie mit 1035 Patienten eines Krankenhauses ergab, daß 340 Patienten mit Antibiotika behandelt wurden, und das war

- bei 13 % berechtigt
- bei 21 % unentschieden
- bei 66 % unberechtigt, ja sogar ausgesprochen ungeeignet, zit. nach /3.13/.

Eine Studie in einer Nervenklinik ergab, daß die Neuroleptikadosierung in Chlorpromazin-Äquivalenten bei Schizophrenen

- in der Männerklinik bei 55 % über 1000 mg/Patient und Tag
- in der Frauenklinik bei 9 % über 1000 mg/Patient und Tag lag /2.1/.

Eine Studie mit dem Expertensystem MYCIN ergab, daß der Gutachterkommission von 80 Meningitis-Therapievorschlägen

- 52 Vorschläge von MYCIN
- 34 bis 50 Vorschläge von den behandelnden Ärzten
- 24 Vorschläge eines Medizinstudenten

akzeptabel erschienen (die Gutachter waren acht Experten, die bereits klinische Berichte über Meningitis-Therapien veröffentlicht hatten); nach neueren Studien erreicht MYCIN eine Treferrate von 90 %, s. /3.13//2.6//1.1/.

Bild 0.1.

In diesem Heft wird nun gezeigt, wie es prinzipiell möglich wird, den Arzt bei Diagnostik und Therapie mit Computerprogrammen zu unterstützen, vorrangig den angehenden Experten mit Wissen aus Expertensystemen zu unterstützen.

Kapitel 1 zeigt, was Wissen ist und wie aus Wissen weiteres, einigermaßen sicheres Wissen erschließbar ist.

Kapitel 2 erläutert die Programmiermittel zum Aufbau von Expertensystemen und bringt Beispiele,

Bild 0.2.

Kapitel 3 gibt einen Ausblick auf

- künftige Softwarekonzepte (Lernen, Diskutieren)
- künftige Hardwarekonzepte (neurale Netze).

Kapitel 4 enthält Quellen aus Philosophie, Medizin, Informatik und Statistik.

Expertensystem MYCIN hat folgende Teilziele		
Nr.		Teilziel
1		Patientendokumentation
2	/	Infektionssignifikanz
3		Erregeridentifikation
4		Medikamentenauswahl
5		Therapievorschlag

Bild 0.1

1	1	Theorie
		Was ist Wissen?
		Wie entsteht Wissen
		aus Wissen?
		Was wissen Experten?
1	2	Praxis
		Wie kommt Wissen
		- in den Computer
		- aus dem Computer,
		und zwar
		- mit Programmier-
		- sprachen
		- mit Programmier-
		- systemen?
		Wie entstehen
		Expertensysteme?

Bild 0.2

1. Theorie

1.0. Übersicht

Nach klassischer Auffassung besteht die objektive Realität, also unsere Welt

- aus unendlich vielen Objekten

- mit jeweils unendlich vielen Merkmalen,

von denen - je nach Situation - nur einige wesentlich sind.

Wissen besteht nun

- aus Fakten:

'Objekt O hat Merkmal M'

("Patient N.N. hat kurzwellige Pulshüllkurven", s. /2.5/)

- aus Regeln:

'Wenn Merkmal M, dann Objektklasse K'

("Wenn kurzwellige Pulshüllkurve, dann wahrscheinlich manisch-depressive Psychose").

Wissen kann sicher und unsicher sein:

- 'Objekt O hat Merkmal M'

(Sicherheit 1 oder 100 %)

- 'Objekt O hat vermutlich Merkmal M'

(Sicherheit beispielsweise 0.6 oder 60 %)

- 'Wenn Merkmal M, dann ziemlich sicher Objektklasse K'

(Sicherheit beispielsweise 0.9 oder 90 %).

Aus vorhandenem Wissen entsteht nun

- durch Schließen neues, aber noch ungetestetes Wissen

- durch Handeln neues, nun jedoch getestetes Wissen, das in diesem Lernprozeß das vorhandene Wissen bereichert, Bild 1.1.

Experten verfügen über Erfahrung mit

- sicheren und unsicheren Fakten

- sicheren und unsicheren Regeln

- monotonen und nicht-monotonen Schlüssen.

Expertensysteme sollen nun

- Fakten speichern (potentiell) und erfragen (aktuell)

- Regeln speichern (potentiell) und gewinnen (aktuell)

- Schlüsse ziehen und aus neuem Wissen lernen,

um so Anfänger, Fortgeschrittene, ja sogar Experten zu unterstützen.

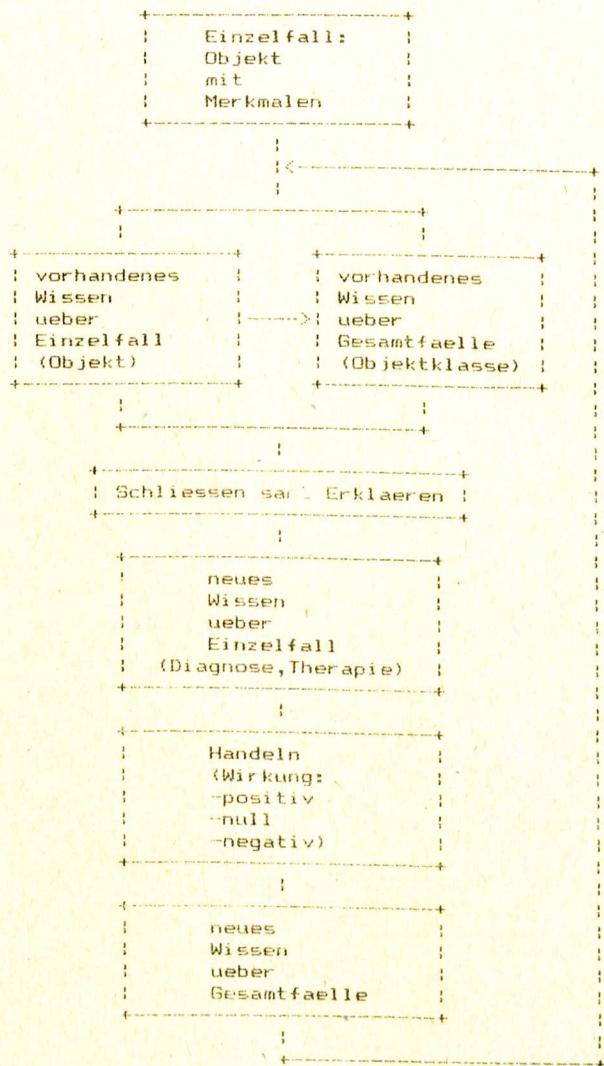


Bild 1.1

1.1. Wissen

1.1.1. Objekte und Merkmale

Objekte gleichen, ähneln oder unterscheiden sich in ihren Merkmalen.

Es gibt

- unendlich viele Objekte
- unendlich viele Merkmale je Objekt,
und aus systemtheoretischer Sicht
- können Objekte auch Merkmale höherer Objekte sein
- können Merkmale von Objekten niedere Objekte sein.

In allen praktischen Fällen sind jedoch

- nur einige Objekte
- nur einige Merkmale der Objekte, die wesentlichen Merkmale, interessant.

Es gibt zwei Arten Objekte, nämlich:

- die Verfahren, bei denen insbesondere die Zeit t wesentliches Merkmal ist
(Vorgang, nicht Zustand; dynamisches Objekt, Prozeß)
- die Gebilde, bei denen insbesondere der Raum x-y-z wesentliches Merkmal ist
(Zustand, nicht Vorgang; statisches Objekt, kurz: Objekt),
Bild 1.2.

Es gibt drei Arten Merkmale, nämlich

- die Bestandteile, selbst wieder Objekte, deshalb auch nullstellige Relationen (sprachlich meist Substantive)
- die Eigenschaften, die dem Objekt eigen sind, deshalb auch einstellige Relationen (sprachlich meist Adjektive)
- die Beziehungen zwischen Objekt und anderen Objekten, deshalb mehrstellige Relationen (sprachlich meist Präpositionen),
Bild 1.3.

Wenn nun Objekte mit Merkmalen vorliegen, wenn sich also "Sachen soundso verhalten", dann sind das

- in der objektiven Realität Sachverhalte
- in der subjektiven Realität Aussagen (gedankliche Abbilder)
- in der objektivierten Realität Sätze (sprachliche Abbilder),
Bild 1.4.

Objekte	
Verfahren (Zeit t; dynamisch)	Gebilde (Raum x,y,z; statisch)

Bild 1.2

Merkmale		
Bestandteile (nullstellig)	Eigenschaften (einstellig)	Beziehungen (mehrstellig)

Bild 1.3

objektive	subjektive	objektivierte
Realitaet		
Objekte (Sachen)	Begriffe	Worter (Termini)
Objekte mit Merkmalen (Sachverhalte)	Aussagen	Saetze

Bild 1.4

1.1.2. Fakten und Regeln

Der Sachverhalt, daß ein Objekt O ein Merkmal M hat, ist abbildbar

- als Faktenaussage, kurz: als Fakt, nämlich:
'Objekt O hat Merkmal M'
- als Regelaussage, kurz: als Regel, nämlich:
'Wenn Objekt O vorliegt, dann liegt auch Merkmal M vor'.

Fakten sind Abbilder

- von unbedingten Sachverhalten
("Patient N.N. ist manisch-depressiv.")
- von Einzelheiten
("Patient N.N. hat kurzwellige Pulshüllkurven.").

Regeln sind Abbilder

- von bedingten Sachverhalten
("WENN Patient N.N. depressiv ist, DANN ist seine Mimik stark verarmt.")
- von Verallgemeinerungen
("WENN Patienten kurzwellige Pulshüllkurven haben,
DANN sind sie im allgemeinen manisch-depressiv."),
Bild 1.5.

Regeln können selbstverständlich mehrere Bedingungen enthalten, und zwar verknüpft durch

- logisches UND (und NICHT)
("WENN Patienten hirnorganisch unauffällig sind
UND
WENN sie depressive Stimmungsschwankungen haben
UND
WENN dafür NICHT äußere Anlässe vorliegen,
DANN sind sie sehr wahrscheinlich endogen depressiv.")
- logisches ODER (und NICHT)
("WENN bei Psychosen neurophysiologische Befunde erhoben wurden
ODER
WENN biochemische Befunde erhoben wurden,
DANN wird die Differentialdiagnostik erleichtert."),
Bild 1.6.

Fakten und Regeln sind also die wesentlichen Bestandteile des Wissens.

Aussagen	
Fakten	Regeln
ohne Bedingungen	mit Bedingungen

Bild 1.5

Verknuepfung von Aussagen					
UND			ODER		
Aus- sage1	Aus- sage2	Gesamt- aussage	Aus- sage1	Aus- sage2	Gesamt- aussage
wahr	wahr	wahr	wahr	wahr	wahr
wahr	falsch	falsch	wahr	falsch	wahr
falsch	wahr	falsch	falsch	wahr	wahr
falsch	falsch	falsch	falsch	falsch	falsch

Bild 1.6

1.1.3. Präzises und nicht-präzises Wissen

Die Aussage 'Objekt O hat Merkmal M'

deutet auf präzises Wissen hin;

die Aussage 'Objekt O hat vielleicht das Merkmal M'

deutet auf nicht-präzises Wissen hin.

Nicht-präzises Wissen ist

- vages Wissen ("Der Blutdruck ist ungefähr 23 kPa.")
- unvollständiges Wissen ("Der Blutdruck ist manchmal 23 kPa.")
- unsicheres Wissen ("Der Blutdruck ist ziemlich hoch.").

Vages und unvollständiges Wissen ist nur extern präzisierbar,
unsicheres Wissen ist rechner-intern behandelbar.

Unsicheres Wissen ist nämlich

- qualitativ (symbolisch, umgangssprachlich) darstellbar
("Der Blutdruck ist ziemlich hoch.")
- quantitativ (numerisch, algebraisch) darstellbar
("Der Blutdruck ist ziemlich hoch", und das bedeute
 - . bei 10 % der Aussagen einen Wert von p_s unter 21 kPa
 - . bei 30 % der Aussagen einen Wert von p_s zwischen 21 und 24
 - . bei 60 % der Aussagen einen Wert von p_s über 24 kPa).

Quantitative Mittel zur Darstellung unsicheren Wissens sind
vorzugsweise

- Konfidenzfaktoren nach SHORTLIFFE (1976)
- Evidenzwertpaare nach ROLLINGER (1983)
- Probabilitätswerte nach BAYES (1763!)
- Zugehörigkeitswerte nach ZADEH (1965).

Konfidenzfaktoren (Sicherheitsfaktoren, certainty factors CF)
Jeder Aussage wird ein Sicherheitsfaktor zwischen 0 und 1 oder
zwischen -1 und +1 zugeordnet,

Bild 1.7.

Der Gesamtsicherheitsfaktor ergibt sich dann

- bei Fakten in UND-Verknüpfung aus dem Minimalwert
- bei Fakten in ODER-Verknüpfung aus dem Maximalwert
- bei einer Regel aus dem Produkt = Bedingungssicherheit * Regelsicherheit
- bei mehreren Regeln beispielsweise aus dem Maximum,

Bild 1.8.

CF = 1.0 Aussage entspricht "voll" der Realität (wahr)

CF = 0.5 Aussage entspricht vielleicht der Realität

CF = 0.0 unbekannt

CF = -0.5 Aussage entspricht vielleicht nicht der Realität

CF = -1.0 Aussage entspricht nicht der Realität (falsch)

Bild 1.7

Beispiel: Verknüpfung von Konfidenzfaktoren

Regel 1:

WENN Kopf rot (CF11 = 0.75) Minimum:

UND Haut kühl (CF12 = 0.80) CF11 = 0.75

UND Übelkeit (CF13 = 0.90),

DANN Sonnenstich mit CFR1 = 0.95 ,

daraus CFG1 = CF11 * CFR1 = 0.75 * 0.95 = 0.71

Regel 2:

WENN Sonnenbad lang (CF21 = 0.80) Minimum:

UND Brechreiz groß (CF22 = 0.90), CF21 = 0.80

DANN Sonnenstich mit CFR2 = 0.85 ,

daraus CFG2 = CF21 * CFR2 = 0.80 * 0.85 = 0.68

Gesamtsicherheit als Maximum(CFG1,CFG2)

CFG = Maximum (0.71,0.68) = 0.71

Gesamtsicherheit als Differenz CFG1+CFG2 - CFG1*CFG2

CFG = 0.71+0.68-0.71*0.68 = 0.91

Bild 1.8

Evidenzwertpaare (Evidenzraum, s. /3.4/)

Jeder Aussage werden zwei Evidenzwerte zwischen 0 und 1 zugeordnet:

(0.0,0.0) unbekannt

(1.0,0.0) Aussage entspricht "voll" der Realität (wahr)

(0.0,1.0) Aussage entspricht nicht der Realität (falsch)

(1.0,1.0) Widerspruch

(0.5,0.0) es gibt Anzeichen/Argumente für Übereinstimmung,
jedenfalls keine dagegen

(0.5,0.5) es gibt Anzeichen/Argumente für Übereinstimmung,
es gibt auch welche dagegen.

Probabilitätswerte (bedingte Wahrscheinlichkeit, BAYES-Theorem)

Jeder Aussage wird die Wahrscheinlichkeit p ihres Auftretens zugeordnet, und die hängt von der

- Wahrscheinlichkeit $p(a)$ für die Aussage ohne Bedingungen
(a-priori-Wahrscheinlichkeit des Ereignisses)
- Wahrscheinlichkeit $p(b_i)$ für die Bedingungen ohne Aussage
- Wahrscheinlichkeit $p(b_i|a)$ für die Bedingungen der Aussage ab, Bild 1.9.

Zugehörigkeitswerte (Zugehörigkeitsfunktion, Fuzzy-Set, s. /4.1/)

Jeder Aussage werden einige Zugehörigkeitswerte zugeordnet, die Sicherheiten, Wahrscheinlichkeiten, Häufigkeiten o.ä. bedeuten, Bild 1.10.

Die Gesamtwerte ergeben sich dann beispielsweise

- bei UND-Verknüpfungen als
 - . Minimum oder Maximum(0, Summe-1)
 - . Produkt
- bei ODER-Verknüpfungen als
 - . Maximum oder Minimum(0, Summe)
 - . Differenz aus Summe und Produkt
 - . halbe Summe aus Maximum und o.a. Differenz.

Schwellwerte

Für das Bestehen oder Nichtbestehen einer Aussage wird oft eine Schwelle angegeben, beispielsweise 0.3

(ähnlich der Bewertung von Korrelationskoeffizienten, die unterhalb von 0.3 keinen Zusammenhang mehr anzeigen, s. z.B. /4.3/).

Beispiel: Ermittlung von Probabilitätswerten

WENN gestörter Schlaf (Schlafstörung)
UND schwankende Stimmung (depressive Tagesschwankung),
DANN endogene Depression mit $p = 0.63$,
weil

- $p(a) = p(\text{Diagnose}) = p(\text{endogene Depression}) = 0.006$
(also a priori 0.6 % der Bevölkerung)
- $p(b_1) = p(\text{Symptom1}) = p(\text{gestörter Schlaf}) = 0.200$
- $p(b_2) = p(\text{Symptom2}) = p(\text{schwankende Stimmung}) = 0.030$
- $p(b_1, a) = p(S.1 \mid D.) = p(\text{ges. Sl.} \mid \text{Depression}) = 0.900$
- $p(b_2, a) = p(S.2 \mid D.) = p(\text{sch. St.} \mid \text{Depression}) = 0.700$

und bei unabhängigen Merkmalen

$$p = p(a) * \frac{p(b_1 \mid a)}{p(b_1)} * \frac{p(b_2 \mid a)}{p(b_2)} * \dots$$

$$p = 0.006 * \frac{0.900}{0.200} * \frac{0.700}{0.030} * \dots$$

$$p = 0.006 * 4.5 * 23.3 = \underline{\underline{0.63}}$$

Bild 1.9

Beispiel: Zuordnung von Zugehörigkeitswerten

Patient N.N. ist nervös,

nämlich:

0.1 oder 10 % : FPI1-Skala hat Stanine-Wert 6

0.2 oder 20 % : FPI1-Skala hat Stanine-Wert 7

0.7 oder 70 % : FPI1-Skala hat Stanine-Wert 8 oder 9

(FPI-Test s. z.B. in /3.5/).

Bild 1.10

1.2. Schließen

1.2.1. Vorwärtsschließen und Rückwärtsschließen

Es gibt - nach der Richtung Start-Ziel - zwei Arten von Schlußstrategien:

- das Vorwärtsschließen oder Vorwärtsverketten, bei dem
 - . für vorhandene Fakten und Bedingungen (z.B. Symptome)
 - . die zutreffenden Regeln oder Hypothesen (z.B. Diagnosen) gesucht werden

("Weil der Patient N.N. unter Schlafstörungen und Stimmungsschwankungen leidet,
ist eine endogene Depression zu vermuten.")
- das Rückwärtsschließen oder Rückwärtsverketten, bei dem
 - . für vorhandene Regeln oder Hypothesen (z.B. Diagnosen)
 - . die zutreffenden Fakten und Bedingungen (z.B. Symptome) gesucht werden

("Der Patient N.N. hat vielleicht eine endogene Depression,
zu erfragen ist, ob er unter Schlafstörungen und Stimmungsschwankungen leidet."),

Bild 1.11.

Vorwärtsverkettung (forward-chaining) ist also

- datengesteuert ("Schlafstörungen sind Merkmal von ...")
- ereignisgesteuert (erst Ereignis, dann Ergebnis)
- bedingungsgesteuert (WENN ..., DANN ...; "Weil ...") *
- induktiv gesteuert (vom Besonderen zum Allgemeinen,
vom Konkreten zum Abstrakten)
- bottom-up-gesteuert (von unten nach oben),

Bild 1.12.

Rückwärtsverkettung (backward-chaining) ist dagegen

- zielgesteuert ("Endogene Depression hat als Merkmale...")
- erwartungsgesteuert (erst hypothet. Ergebnis, dann Ereignis)
- folgegesteuert (DANN ..., WENN ...; "..., weil ...")
- deduktiv gesteuert (vom Allgemeinen zum Besonderen,
vom Abstrakten zum Konkreten)
- top-down-gesteuert (von oben nach unten),

Bild 1.13.

Experten nutzen Vorwärts- und Rückwärtsverkettung,

Bild 1.14.

Beispiel: Vorwärtsschließen und Rückwärtsschließen

Regeln	vorwärts	rückwärts
	Start: A (Fakt)	Ziel: E bewiesen
WENN A, DANN B	1: B bewiesen	4: A beweisen ! ^
WENN B, DANN C	2: C bewiesen	3: B beweisen !
WENN C, DANN D	3: D bewiesen	2: C beweisen !
WENN D, DANN E	4: E bewiesen v	1: D beweisen ! ,
	Ziel: E bewiesen	Start: E (Hypothese)

Bild 1.11

Algorithmus zum Vorwärtsschließen (NILSON, vereinfacht)

1. Sammle Fakten F.
2. Bis Fakten F zum Ziel Z führen, tue folgendes:
 - 2a. Wähle eine Regel R, deren Bedingungsteil (WENN ...) durch F erfüllbar ist.
 - 2b. Nutze den Aktionsteil (DANN ...) nun mit als Fakt F'.

Bild 1.12

Algorithmus zum Rückwärtsschließen (BUCHANAN, vereinfacht)

1. Sammle Regeln R, aus denen das Ziel Z ableitbar ist.
2. Bis Ziel Z erreicht ist, tue folgendes:
 - 2a. Wähle eine Regel R, deren Bedingung (WENN ...) ein Zwischenziel Z' ist.
 - 2b. Wenn Z' beweisbar ist, dann nutze R, sonst führe Algorithmus mit Z' aus.

Bild 1.13



Bild 1.14

1.2.2. Tiefensuche und Breitensuche

Es gibt - nach der Lage der Zwischenziele - zwei Arten von Suchstrategien:

- die Tiefensuche (depth first search), bei der
 - . zunächst eine Regel vollständig
 - . danach die folgende Regelgenutzt wird

("Hat Patient N.N. Schlafstörungen und Stimmungsschwankungen?"

"Hat Patient N.N. also ein depressives Grundmuster?",
danach eventuell

"Hat Patient N.N. kurzwellige Pulshüllkurven?";
daraus dann

"Patient N.N. ist wahrscheinlich endogen depressiv.")

- die Breitensuche (breadth first search), bei der
 - . zunächst von Regeln mindestens eine Bedingung, ein Teilziel
 - . danach die folgenden Teilzielebetrachtet werden

("Hat Patient N.N. Schlafstörungen, Stimmungsschwankungen,
kurzwellige Pulshüllkurven?",
daraus dann

"Patient N.N. ist wahrscheinlich endogen depressiv."),

Bild 1.15.

Schluß- und Suchstrategien sind kombinierbar:

- Vorwärtsschließen und Tiefensuche
(zunächst viele Fakten unter einem Aspekt sammeln, z.B.
psychopathologische Fakten)
- Vorwärtsschließen und Breitensuche
(zunächst viele Fakten unter vielen Aspekten sammeln, z.B.
psychopatholog., neurophysiologische, biochemische Fakten)
- Rückwärtsschließen und Tiefensuche
(zunächst eine Hypothese unter einem Aspekt beweisen)
- Rückwärtsschließen und Breitensuche
(zunächst eine oder mehrere Hypothesen unter mehreren
Aspekten zu beweisen beginnen),

Bild 1.16.

Beispiel: Tiefensuche und Breitensuche

Regeln

WENN A UND B, DANN C

WENN C, DANN E

WENN D, DANN E

Tiefe

1: A,B : C

2: C : E

(3: D : E)

Breite

1: A,C,D : E

(2: B : C)

(3: C : E)

Bild 1.15

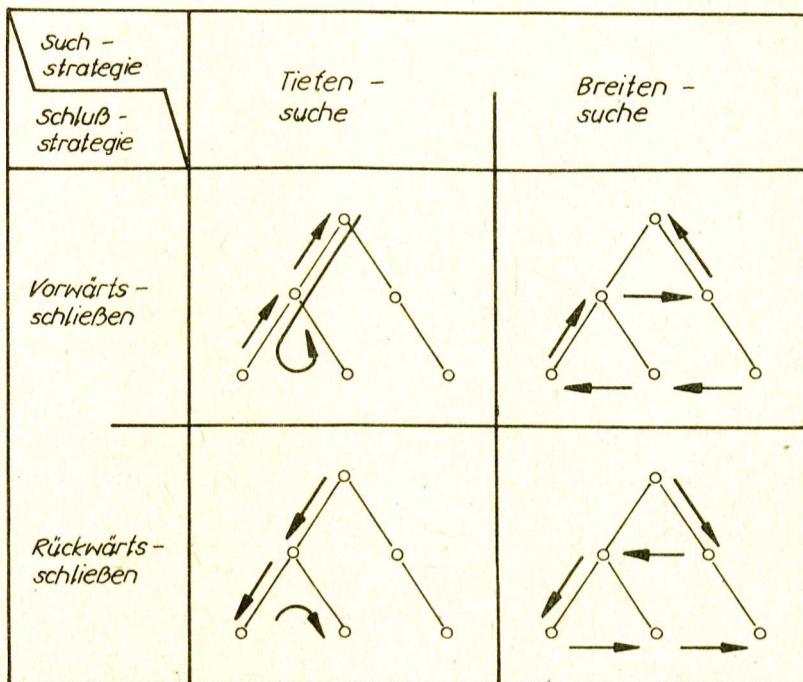


Bild 1.16

1.2.3. Monotones und nicht-monotones Schließen

Es gibt – nach der Beeinflussung des Wissens und des Schließens – zwei Schlußarten:

- das monotone Schließen, bei dem
 - . mit der Anzahl der Fakten das Wissen monoton wächst
 - . erschlossene Fakten nicht revidierbar sind
 - ("Patient N.N. hat depressive Stimmungsschwankungen -
das kann auf eine endogene Depression hindeuten."
"Patient N.N. hat Schlafstörungen -
das erhärtet die Annahme einer endogenen Depression.")
 - das nicht-monotone Schließen, bei dem
 - . mit der Anzahl der Fakten das Wissen nicht (monoton) wächst
 - . erschlossene Fakten revidierbar sind
 - ("Patient N.N. hat Stimmungsschwankungen und Schlafstörungen -
das deutet auf eine endogene Depression hin."
"Patient N.N. zog kürzlich vom Dorf ins Stadtzentrum -
das kann mindestens die Schlafstörungen erklären und weicht
die Annahme einer endogenen Depression auf."),

Bild 1.17.

Es ist zwar leicht, den Wert eines Merkmals zu ändern, es ist aber schwierig, alle Abhängigkeiten von dem geänderten Wert zu überschauen !

Experten kombinieren nun

- Vorwärts- und Rückwärtsverkettung
 - Tiefen- und Breitensuche
 - monotonen und nicht-monotonen Schließen.

Beispiel:

- Kennen vieler Diagnosen
 - Stellen einiger Fragen (Vorwärtsverkettung !)
 - Bilden einiger Hypothesen (Breitensuche,
Rückwärtsverkettung !)
 - Stellen gezielter Fragen (Tiefensuche !)
 - Verändern der Hypothesen (nicht-monotones Schließen)
 - Stellen gezielter Fragen
 - Bestätigen bzw. Widerlegen der Diagnosen,
Bild 1.18 nach /3.14/.

Beispiel: Monotones Schließen und nicht-monotonen Schließen

Regeln	monoton	nicht-monoton
WENN A UND B, DANN D	1: A,B : D	1: A,B : D
WENN A UND C, DANN NICHT D (nicht zulässig)		2: A,C : \bar{D}

Bild 1.17

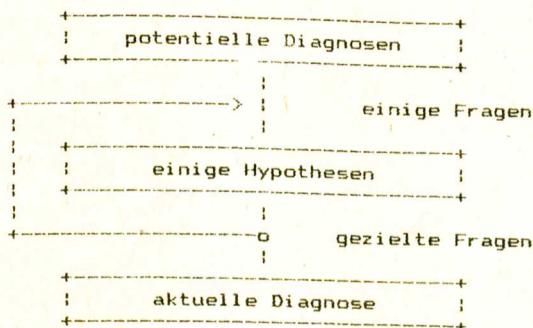


Bild 1.18

1.3. Expertensysteme

1.3.1. Aufgaben

Expertensysteme sind Systeme aus

- Hardware, und zwar mindestens einem guten Personalcomputer
- Software, und zwar mindestens zum
 - . Speichern von Wissen
 - . Ziehen von Schlüssen
 - . Geben von Erklärungen, warum und wie aus dem Wissen die Schlüsse gezogen worden sind.

Expertensysteme sollen auf ausgewählten Gebieten

- die Anfänger und Fortgeschrittenen auf dem Weg zu Kompetenten oder gar zu Gewandten und Experten unterstützen
- die Gewandten und Experten bei ihrer Arbeit unterstützen,
Bild 1.19 nach /1.1/.

Expertensysteme

- können nach Ansicht scharfsinniger Kritiker nie das Niveau von Gewandten und Experten erreichen:

"Wir haben gesehen, daß Computer in der Realität eher verstandesmäßig kalkulieren - wie Menschen, denen jede Erfahrung fehlt. Erst umfassendere menschliche Erfahrungen jedoch ermöglichen ein wirkliches Know-how - eine weit überlegene, intuitive Art, Probleme anzugehen, eine Art, die Regeln-befolgende Computer nicht imitieren können.",
DREYFUS in /1.1/

- sollten nach Ansicht der meisten Menschen nie Gewandte oder Experten ersetzen:

"Damals hatte uns alle die Wahnnidee gepackt, eine universell programmierbare Maschine zu bauen, eine Art Mikrokosmos zu schaffen, in dem alles erfaßbar und berechenbar ist. Doch Jahrzehntelange Erfahrungen und Enttäuschungen sowie die vielen Diskussionen über 'Computer und menschliches Denken' drängten mir den Schluß auf, daß die Probleme weder technischer noch mathematischer Natur sind. Sie können nicht dadurch gelöst werden, daß wir Fragen stellen, die mit 'können' beginnen. Die Grenzen in der Anwendung von Computern lassen sich letztlich nur als Sätze angeben, in denen das Wort 'sollten' vorkommt.",
WEIZENBAUM in /1.5/.

Expertensysteme sind einsetzbar

- beim Analysieren ("Etwas ist ..."), also beispielsweise als Diagnosesysteme zur Unterstützung von Medizinern
- beim Synthesisieren ("Etwas sei ..."), also beispielsweise als Projektierungssysteme zur Unterstützung von Technikern.

Relation:	Regel - Situation	Regeln - Ziel	Subjekt - Objekt
Tätigkeit:	Regel- anwendung	Ziel- planung	Objekt- betrachtung
Stufe:			
1. Anfänger	kontext- frei, situations- unabhängig	nein	einzelheitlich (kognitiv)
2. Fortge- schrittenener	kontext- haft, situations- abhängig	nein	einzelheitlich (kognitiv)
3. Kompetenter	kontext- haft, situations- abhängig	ja, durch Auswahl (Metaregeln)	einzelheitlich beeinflussen, ganzheitlich Ergebnisse bewerten
4. Gewandter, Profi	kontext- haft, situations- abhängig	ja, aus Erfahrung	einzelheitlich/ ganzheitlich beeinflussen, ganzheitlich Ergebnisse bewerten
5. Experte	kontext- haft, situations- abhängig	ja, aus Erfahrung	ganzheitlich (kognitiv, emotional): intuitiv

Bild 1.19

1.3.2. Aufbau

Expertensysteme bestehen aus Hardware und Software.

Expertensysteme im engen Sinne haben als Softwarekomponenten mindestens

- die Wissenskomponente (Wissensbasis)
- die Schlußkomponente (Regelinterpreter, Inferenzmechanismus), und zwar als weitgehend getrennte Komponenten
(die Änderung der Wissenskomponente sollte keine Änderung der Schlußkomponente bewirken).

Wünschenswert sind weiterhin

- eine Wissenserwerbskomponente (Wissensakquisition)
- eine Kommunikationskomponente (Dialogkomponente), und zwar als weitgehend menschengerechte Komponenten
(die Sprache des Expertensystems sollte der Sprache des Benutzers nahekommen).

Expertensysteme im weiten Sinne haben weitere Softwarekomponenten zur Lösung von Aufgaben und Problemen, beispielsweise

- Datenbasen mit strukturierten Fakten von Objekten
- Programmabasen mit üblichen Prozeduren der Statistik,
Bild 1.20.

Expertensysteme sind aufbaubar

- mit Programmiersprachen, und zwar mit
 - . algorithmischen, prozedurorientierten, operativen Sprachen, z.B. BASIC und PASCAL
(wenig geeignet)
 - . nicht-algorithmischen, objektorientierten, deklarativen S., z.B. LISP und PROLOG, s. /3.16/ und /3.8/
(besser geeignet)
- mit Programmiersystemen, und zwar mit
 - . deduktiven Rahmensystemen (shells, Schalen), z.B. XPRO und GURU, s. /3.2/
(Regeln werden außerhalb des Systems gewonnen)
 - . induktiven Rahmensystemen (shells, Schalen), z.B. 1stCLASS und RULEMASTER, s. /3.2/
(Regeln werden auch im System aus Beispielen gewonnen).

	klassische Datenverarbeitung	moderne Wissensverarbeitung
Themen	Aufgaben, wohl- strukturiert	Probleme, kaum strukturiert
Wissen	vollstaendig formatisierbar	unvollstaendig formatisierbar
Steuerung	direkt abhaengig vom Wissen	nur indirekt abhaengig vom Wissen

Bild 2.1

Symptome	Diagnose
WENN Kopf blass UND Haut kuehl UND Puls schnell	DANN wahrscheinlich Hitze- erschoepfung
WENN Kopf rot UND Haut heiss UND Temperatur	DANN wahrscheinlich Hitz- schlag
WENN Kopf rot UND Haut kuehl UND Brechreiz	DANN wahrscheinlich Sonnen- stich

Bild 2.2

2.1. Programmiersprachen

2.1.1. BASIC

Programmelemente

BASIC (Beginner's All-purpose Symbolic Instruction Code, KEMENY und KURTZ 1964) ist eine einfache algorithmische (prozedurorientierte, operative) Programmiersprache mit Programmelementen für

- Anfang (meist REM) und Ende (END)
- Dimensionierung von Feldern (DIM) / Setzen von Werten (LET)
- Eingabe (INPUT) und Ausgabe (PRINT)
- Datenspeichern (DATA) / Datenlesen (READ) im Programm
- Zeichenkettenzerlegung (LEFT, MID, RIGHT)
- Programmsteuerungen
 - . unbedingter Sprung: immer (GOTO)
 - . bedingter Sprung: wenn ... dann ... sonst ...
(IF ... THEN ... ELSE ...)
 - . Schleife : für ... bis ... nächstes
(FOR ... TO ... NEXT),

anwendbar auf

- Ganzzahlen (Kennzeichen: %)
- Dezimalzahlen (Kennzeichen: ohne)
- Zeichenketten (Kennzeichen: \$ oder o) (STRING), Bild 2.3.

Programmstruktur

BASIC stellt das "Wie", nicht das "Was" in den Vordergrund. BASIC unterstützt aber keinen Programmierstil.

Empfehlenswert ist folgender Stil:

- Szenenkonzept, also beispielsweise
 - . Szene 0: Vorspann (möglichst ab Zeile 100, mit DIM)
 - . Szene 1: Eingabe (möglichst ab Zeile 1000)
 - . Szene m: Verarbeitung (möglichst ab Zeile m*1000)
 - . Szene n: Ausgabe (möglichst ab Zeile n*1000)
 - . Szene E: Ende (möglichst ab Zeile 10 000)
 - . dann Unterprogramme und Daten
- Verwendung von GOTO nur zum Szenenanfang/zu Nachbarzeilen
- strukturierte (eingerückte) Schreibweise.

1. Rechenanweisungen

LET	es sei ... (kann auch weggelassen werden)
+ - * /	wie in Mathematik: Punkt- vor Strichrechnung
^	Potenz; größere Bindung als Punktrechnung
()	wie in Mathematik, sonst von links nach rechts
= < >	wie in Mathematik
ABS	Absolutwert
SIN COS usw.	wie in Mathematik
AND OR	wie in Logik
RND	Erzeugung von Zufallszahlen 0...1

2. Eingabe-Ausgabe-Anweisungen

INPUT	Eingabe
PRINT	Ausgabe (Bildschirm), nur PRINT ist Leerzeile
LPRINT	Ausgabe (Drucker, L wie line)
TAB(position)	Tabulator, z.B. PRINT TAB(5); "ab 5. Position"

3. Steueranweisungen

IF...THEN...ELSE	wenn...dann...sonst
FOR i ... NEXT i	für i...bis...dann weiter mit nächstem i
GOTO zeilennr.	gehe zum Befehl mit der Zeilennummer...
GOSUB zeilennr.	gehe zum Unterprogramm mit der Zeilennummer...
RETURN	kehre zurück aus dem Unterprogramm

4. Zeichenkettenanweisungen

VAL(K\$)	Zahl aus den Ziffern in der Zeichenkette K\$
STR\$(Z)	Zeichenkette aus der Zahl Z
LEN(K\$)	Länge der Zeichenkette K\$
LEFT\$(K\$, N)	linker Teil der Zeichenkette K\$, n lang
MID\$(K\$, M, N)	mittlerer Teil der Zeichenkette K\$, ab Stelle m
RIGHT\$(K\$, N)	rechter Teil der Zeichenkette K\$, n lang

5. Dateianweisungen

OPEN	Datei öffnen; "I" sequentiell, "R" wahlfrei
CLOSE	Datei schließen, auch in END enthalten
FIELD	Eingabe- oder Ausgabefeld formieren
PUT	Satz auf Datei ausgeben (Diskette)
GET	Satz von Datei eingeben (Diskette)

6. Rahmenanweisungen

REM	Bemerkung (rem wie remark)
END	Ende

Bild 2.3

Programm

EXPPRO1 ist ein kleines BASIC-Programm, das
- bei einigen Symptomen (hier nur 3)
- eine oder keine Diagnose (hier aus 3)
vorschlagen soll, Bild 2.4, s.a. /3.6/.

Szene 0 ist der Vorspann:

- Zeile 150: Dimensionierung eines zweidimensionalen Feldes für D Diagnosen mit jeweils S Symptomen und einer Diagnosebezeichnung, also S+1
- Zeile 180: Bildschirmlöschen wird vorbereitet (sonst CLS)
- Zeilen 190 bis 210: Festlegen der 3 Symptome mit ihren Werten, z.B. Kopf: normal, blass, rot
- Zeilen 200 bis 260: zwei Schleifen zum Füllen der Feldelemente mit Daten aus der "Wissensbasis" (ab Zeile 11 020).

Szene 1 ist die Eingabeszene:

- Zeile 1020: Bildschirmlöschen, Ausgabe der Überschrift
- Zeilen 1030 bis 1050: Schleife zum Zeigen der möglichen Symptome (Zeilen 190 bis 210)
- Zeile 1070: Eingabe des Nutzers (1, 2 oder 3), wenn er von den Symptomen 1, 2 oder 3 im aktuellen Fall etwas weiß
- Zeile 1090: Rücksprung bei falscher Eingabe, z.B. 4 statt 1
- Zeilen 1100 bis 1110: Ausgabe der gewählten Symptomzeile und Eingabe des aktuellen Merkmals, z.B. rot.

Szene 2 ist die Verarbeitungsszene:

- Zeilen 2030 bis 2080: Vergleich der jeweils ersten drei Zeichen von potentiellem und aktuellem Symptom, Speichern der Anzahl übereinstimmender Werte für jede Diagnose in M(I)
- Zeilen 2090 bis 2130: Ermitteln der Maximalzahl an Übereinstimmungen ME. Der Vergleich heißt Matching.

Szene 3 ist die Ausgabeszene:

- Zeile 3020: Ausgabe der Maximalzahl an Übereinstimmungen
- Zeilen 3030 bis 3060: Ausgabe der Diagnosevorschläge.

Szene E ist die Endeszene:

- Zeilen 10 030 bis 10 040: Wenn der Nutzer noch weitere Merkmale kennt, dann wird bei Zeile 1000 fortgesetzt, sonst ist bei Zeile 10 060 das Programm beendet.

Expertensystem-Erarbeiter			
Wissen aus		I Wissen aus	
Informatik		I Fachgebiet	
Y		Y	
Programmiermittel Wissenerwerbskomponente			
Prozedur-	Schluss-	Wissens-	Daten-
basis	komponente	komponente	basis
	(Regel- interpret)	(Fakten/ Regeln)	
Kommunikationskomponente			
Aufgaben	^	I Loesungen mit	
		Y Loesungsweg	
Expertensystem-Benutzer			

Bild 1.20

1.3.3. Arten

Expertensysteme sind klassifizierbar

- nach dem Aufbau (Form)
- nach dem Aufwand (Umfang)
- nach der Anwendung (Inhalt).

Nach dem Aufbau unterscheidet man

- aus der Sicht des Schließens
 - . induktive und deduktive Systeme (Regelgewinnung)
 - . Vorwärts- und Rückwärtssysteme (Regelschlußstrategie)
 - . Tiefen- und Breitensuchsysteme (Regelsuchstrategie)
- aus der Sicht der Wissensdarstellung (Wissensrepräsentation)
 - . regelbasierte Systeme (Wissen prozedural)
 - . nicht-regelbasierte Systeme (deklarativ), vorzugsweise netz-, frame- oder script-orientiert
(Frame: Tabelle aus Merkmalen (slots) und Werten (subslots))
(Script: Text für Szenen mit Eingang, Ausgang, Ablauf), Bilder 1.21 und 1.22
- aus der Sicht der Wissenskomplexität
 - . Schrifttext-, Grafiktext-, Raum-, Raumzeitsysteme
(z.B. Situationserkennungssysteme für Roboter).

Nach dem Aufwand unterscheidet man

- elementare Systeme und komplexe Systeme.

Nach der Anwendung unterscheidet man

- aus der Sicht der Tätigkeitsart
 - *. Arbeitssysteme (Beratungs- und Handlungssysteme)
 - . Lehrsysteme (Tutorsysteme) und Spielsysteme
- aus der Sicht der Erkenntnisart
 - . Analysesysteme und Synthesesysteme
- aus der Sicht des Erkenntnishorizonts
 - . Analysesysteme für Istzustände
(z.B. Diagnosesysteme in Technik und Medizin)
 - . Analysesysteme für Sollzustände
(z.B. Prognosesysteme für Ökonomie und Ökologie)
 - . Synthesesysteme für Sollzustände als Rechtzeitsysteme
(z.B. Projektierungssysteme für Technik und Informatik)
 - . Synthesesysteme für Sollzustände als Echtzeitsysteme
(z.B. Kontroll- und Regelsysteme in der Produktion).

Beispiel: Klinik-Frame

Lage: Schwerin-Sachsenberg, Rostock-Gehlsdorf, ...
Raum: Einzelgebäude im Park, Gebäudekomplex, ...
Größe: 1, 5, 10, 25 ha (Standard=Default= 5 ha)
Struktur: Aufnahme/Entlassung
Psychiatrische Kliniken
Neurologische Kliniken
Ambulanzen
Laboratorien
Apotheke ...
Aufgabe: Diagnostizieren/Therapieren

```
graph TD; A[Struktur] --> B[Aufnahme/Entlassung]; C[Aufgabe] --> D[Aufnahme/Entlassung]
```

Aufnahme/Entlassung
Lage: ...
Raum: ...
Größe: ...
Struktur: ...
Aufgabe: ...
(Funktion)

Bild 1.21

Beispiel: Klinik-Script

Rolle: Patient
Eingang: krank
Ausgang: gesund
Ablauf: 6 Szenen

Szene 1: in die Klinik gehen
Wohnung verlassen
Verkehrsmittel benutzen
Klinik betreten

Szene 2: in der Aufnahme/Entlassung anmelden
...

Szene 6: aus der Klinik gehen
Klinik verlassen
Verkehrsmittel benutzen
Wohnung betreten

Bild 1.22

2. Praxis

2.0. Übersicht

Expertensysteme sind wissensbasierte Systeme.

Datenverarbeitungssysteme und Wissensverarbeitungssysteme unterscheiden sich in folgenden Merkmalen:

- Datenverarbeitungssysteme
 - betreffen vorwiegend stark strukturierte, vollständig formal beschreibbare Themen (Aufgaben im engeren Sinne: "Ermittlung psychiatrischer Befunde aus Pulshüllkurven")
 - vermischen weitgehend Wissenskomponente und Schlußkomponente (Abhängigkeit: Programmsteuerung vom Regelwissen)
- Wissensverarbeitungssysteme
 - betreffen vorwiegend schwach strukturierte, unvollständig formal beschreibbare Themen (Probleme im engeren Sinne: "Ermittlung psychiatrischer Diagnosen")
 - trennen weitgehend Wissenskomponente und Schlußkomponente (Unabhängigkeit),

Bild 2.1.

Wichtige Programmiermittel sind

- für Datenverarbeitungssysteme
 - die prozeduralen Programmiersprachen,
z.B. BASIC und PASCAL zum Programmieren im Kleinen
 - die klassischen Rahmensysteme zum Programmieren im Großen
- für Wissensverarbeitungssysteme
 - die deklarativen Programmiersprachen,
z.B. LISP und PROLOG zum Programmieren im Kleinen
 - die modernen Rahmensysteme zum Programmieren im Großen,
z.B. XPRO (elementar, deduktiv) und
RULEMASTER (komplex, induktiv).

An einem einfachen Musterbeispiel - bewußt nicht aus dem psychiatrischen Bereich, sondern Diagnosevorschläge bei Hitzeeinwirkung - wird nun gezeigt, wie Expertensysteme aufbaubar sind,
Bild 2.2 nach /3.7/.

Es folgen einige Praxisbeispiele aus dem neurologischen und psychiatrischen Bereich.

PROGRAM EXPDAT; - 37 -

(-----Deklarationsteil-----)

```
CONST ZD = 100;
TYPE EXPDAT = RECORD
  D : ARRAY [1..ZD] OF STRING [20];
  M1 : ARRAY [1..ZD] OF STRING [15];
  M2 : ARRAY [1..ZD] OF STRING [15];
  M3 : ARRAY [1..ZD] OF STRING [30];
  MZ : ARRAY [1..ZD] OF INTEGER;
END;
```

VAR NAME : STRING[15];
 DIA : EXPDAT;
 DN : FILE OF EXPDAT;
 I : INTEGER;
 EIN : CHAR;

(-----Anweisungsteil-----)

```
BEGIN
  CLRSCR;WRITELN;WRITELN;
  WRITE ('Geben Sie bitte den Dateinamen ein! ');READ (NAME);
  WRITELN;
  ASSIGN (DN,NAME);
  REWRITE (DN);
  FOR I:=1 TO ZD DO
  BEGIN
    WITH DIA DO
    BEGIN
      D[I]:='';M1[I]:='';M2[I]:='';M3[I]:='';MZ[I]:=0;
    END;
    END;
    I:=0;EIN:='j';
    BEGIN
      WRITELN;
      WHILE (EIN = 'j') OR (EIN = 'J') DO
      BEGIN
        I:=I+1;
        WRITE ('Diagnose: ');READLN (DIA.D[I]);
        WITH DIA DO
        BEGIN
          WRITE ('Kopf : ');READLN (M1[I]);
          WRITE ('Haut : ');READLN (M2[I]);
          WRITE ('sonst: ');READLN (M3[I]);
          MZ[I]:=0;READ (MZ[I]);
        END;
        WRITELN;
        END;
        WRITE ('weiter ? (j): ');READLN (EIN);
        WRITELN;
      END;
      WRITE (DN,DIA);
    END;
  CLOSE (DN);
END.
```

Bild 2.6

Bild 2.7 ↓

D(I)	M1(I)	M2(I)	M3(I)	MZ(I)	I
Hitzeerschoepfung	blass	kuehl	schneller Puls	0	1 1
Hitzschlag	rot	heiss	hohe Temperatur	0	1 2
Sonnenstich	rot	kuehl	Brechreiz	0	1 3

Programm

EXPPRO1 ist ein kleines Turbo-PASCAL-Programm, das

- bei einigen Symptomen (hier nur 3)
 - eine oder keine Diagnose (hier aus 3)
- vorschlagen soll, Bild 2.8.

Das Programm beginnt mit PROGRAM.

Deklarationsteil:

- Deklaration der Konstanten ZD (Anzahl der Diagnosen)
- Deklaration der Wissensbasis (Typ RECORD, s. Progr. EXPDAT)
- Deklaration der Variablen.

Anweisungsteil:

- Bildschirmlöschen (CLS)
- Eingabe des Dateinamens und Zuordnung (ASSIGN, Diskette)
- Öffnung der Datei (RESET)
- Einlesen der Datensätze von der Diskette (bis EOF READ)
- Ausgabe des Menüs und Eingabe vom Nutzer (READLN)
- Vergleich der Dateiwerte ($M(I)$) mit den Eingabewerten (W)
- Zählen der Übereinstimmungen (MZ)
- Ermitteln der Maximalzahl an Übereinstimmungen (ME)
- Ausgabe der Diagnose-Vorschläge mit der Maximalzahl an Übereinstimmungen
- Schließen der Datei (CLOSE).

Das Programm endet mit END.

Programmnutzung

Der Nutzer

- gibt den Dateinamen ein, hier: EXPDAT
- gibt aktuelle Symptome ein, z.B. rot, kuehl, Brechreiz
- erhält auf dem Bildschirm den Diagnose-Vorschlag, z.B. Sonnenstich bei 3 übereinstimmenden Symptomen.

Programmkritik

Für das Turbo-PASCAL-Programm EXPPRO1 liegt eine getrennte "Wissensbasis" vor. Ansonsten hat das PASCAL-Programm ähnliche Nachteile wie das BASIC-Programm.

Für den Aufbau von Expertensystemen eignet sich PROLOG besser als BASIC oder PASCAL.

```

PROGRAM EXPPRO1; { Expertensystemversuch in TURBO-PASCAL }           - 39 -
{-----Deklarationsteil-----}
CONST   ZD          =100;
TYPE    { wie bei EXPDAT }
VAR     W1,W2          :STRING [15];
       W3,NAME        :STRING [30];
       DIA            :EXPDAT;
       DN             :FILE OF EXPDAT;
       I,ME           :INTEGER;
{-----Anweisungsteil-----}
BEGIN
  CLRSCR;WRITELN;WRITELN;
  WRITE (' Eingabe des Dateinamens: ');READ (NAME);
  ASSIGN (DN,NAME);
  {$I-}  RESET  (DN)  {$I+};                                     { Datei-Eroeffnung }
  BEGIN
    CLRSCR;
    WHILE NOT EOF (DN) DO
    BEGIN
      READ (DN,DIA);                                         { Einlesen der Datei }
      WRITELN;WRITELN;
      WRITELN (' Geben Sie Merkmalwerte an ! ');           { Eingabeaufforderung }
      WRITELN (' ~~~~~~');WRITELN;
      WRITE (' Kopf : normal,blass,rot                      : ');READLN (W1);
      WRITE (' Haut : normal,kuehl,heiss                     : ');READLN (W2);
      WRITE (' sonst: hohe Temp.,schneller Puls,Brechreiz: ');READLN (W3);
      I:=0;ME:=0;
      FOR I:=1 TO ZD DO
      BEGIN
        WITH DIA DO
        BEGIN
          IF D[I]<>'' THEN
          BEGIN
            IF M1[I]=W1 THEN MZ[I]:=MZ[I]+1;                   { Vergleich }
            IF M2[I]=W2 THEN MZ[I]:=MZ[I]+1;
            IF M3[I]=W3 THEN MZ[I]:=MZ[I]+1;
            IF MZ[I]>ME THEN
            BEGIN
              ME:=MZ[I];
            END;
          END;
        END;
        WRITELN;
        IF ME > 0 THEN
        BEGIN
          FOR I:=1 TO ZD DO
          BEGIN
            IF DIA.MZ[I]=ME THEN WRITELN (' Diagnose-Vorschlag: ',DIA.D[I])
          END;
        END
        ELSE
        BEGIN
          WRITELN (' Diagnose-Vorschlag: keiner');
        END;
        WRITELN (' uebereinstimmende Symptome: ',ME);
        CLOSE (DN);
      END;
    END;
  END;

```

2.1.3. PROLOG

Programmelemente

PROLOG (Programming in Logic, COLMERAUER 1972, Kernsprache des japanischen Projekts "Fünfte Rechnergeneration" 1981, hier Turbo Prolog der Fa. BORLAND) ist eine einfache nicht-algorithmische (objektorientierte, deklarative) Programmiersprache mit Programm-elementen für

- Deklarationen (domains, z.B. char, string, symbol)
- Prädikate (predicates, z.B. diagnose(symbol))
- Klauseln (clauses für Fakten und Regeln, allgemein:
dann ... wenn ...
(... if ...), statt if auch :-)
- Eingabe (read) und Ausgabe (write)
- Datenbankaufbau (database)
- Datenspeicherung (assert) und Datenlöschen (retract)
- Ziele (goal),
anwendbar auf
 - Ganzzahlen (integer)
 - Dezimalzahlen (real)
 - Einzelzeichen (char, z.B. 'j')
 - Zeichenketten (string, z.B. "Diagnosevorschlag: ")
 - Symbole (symbol, z.B. schneller_Puls (Anfang stets klein)).Hinzu kommen einige prozedurale, operative Prädikate, z.B.
- fail (zum Erzwingen des sog. Backtracking, des Nutzens weiterer Regeln, nachdem eine Regel gescheitert ist)
- cut (!, zum Verhindern des Backtracking),
Bild 2.9.

Programmstruktur

PROLOG stellt das "Was", nicht das "Wie" in den Vordergrund.
Turbo Prolog fordert folgenden Programmierstil:

- (evtl. Compilerbefehle zur Programmübersetzung)
- domains
- database
- predicates
- clauses/goal.

Empfehlenswert ist strukturierte (eingerückte) Schreibweise.

```

100 REM EXPPRO1 - Expertensystem-Versuch in BASIC
110 REM -----
120 REM Szene 0: Vorspann
130 REM -----
140 LET D=3 : S=3 : REM Anzahl der Diagnosen und Symptome in "Wissensbasis"
150 DIM D$(D,S+1) : REM Diagnosen in "Wissensbasis"
160 DIM S$(S),W$(S) : REM Symptome mit Werten, Werte des Objekts
170 DIM M(D) : REM Anzahl der Uebereinstimmungen (Matching)
180 LET Z#=STRING$(24,10)
190 LET S$(1)="Kopf : normal, blass, rot .....: "
200     S$(2)="Haut : normal, kuehl, heiss .....: "
210     S$(3)="sonst: hohe T., schneller P., Brechreiz: "
220 FOR I=1 TO D
230   FOR J=1 TO S+1
240     READ D$(I,J)
250   NEXT J
260 NEXT I
270 FOR J=1 TO S : W$(J)="?" : NEXT J
1000 REM Szene 1: Eingabe
1010 REM -----
1020 PRINT Z$ : PRINT "Aktueller Wissensstand" : PRINT
1030 FOR J=1 TO S
1040   PRINT "(";J;") ";S$(J);W$(J)
1050 NEXT J
1060 PRINT
1070 INPUT "Von welchem Merkmal ist etwas bekannt ? (1-3): ",A
1080 PRINT
1090 IF A<1 OR A>3 THEN PRINT "Achtung! Eingabe 1, 2 oder 3!" : GOTO 1060
1100 PRINT "(";A;") ";S$(A)
1110 INPUT "",W$(A)
2000 REM Szene 2: Verarbeitung - Matching
2010 REM -----
2020 PRINT : PRINT
2030 FOR I=1 TO D
2040   M(I)=0
2050   FOR J=1 TO S
2060     IF LEFT$(D$(I,J),3)=LEFT$(W$(J),3) THEN M(I)=M(I)+1
2070   NEXT J
2080 NEXT I
2090 LET ME=0
2100 FOR I=1 TO D
2110   IF M(I)>ME THEN ME=M(I)
2120 NEXT I
3000 REM Szene 3: Ausgabe
3010 REM -----
3020 PRINT "uebereinstimm.Symptome/Merkmaile:";ME
3030 IF ME=0 THEN PRINT "vorlaeufiger Diagnosevorschlag: keiner" : GOTO 10000
3040 FOR I=1 TO D
3050   IF M(I)=ME THEN PRINT "vorlaeufiger Diagnosevorschlag: ";D$(I,4)
3060 NEXT I
10000 REM Szene E: Ende
10010 REM -----
10020 PRINT : PRINT
10030 INPUT "Ist von weiteren Merkmalen etwas bekannt? (j): ",E$
10040 IF E$="j" OR E$="J" THEN 1000
10050 PRINT : PRINT "Ende EXPPRO1"
10060 END
11000 REM "Wissensbasis" (Datenbasis)
11010 REM -----
11020 DATA "blass","kuehl","schneller Puls","Hitzeerschoepfung"
11030 DATA "rot","heiss","hohe Temperatur","Hitzschlag"
11040 DATA "rot","kuehl","Brechreiz","Sonnenstich"

```

Programmnutzung

Der Nutzer

- wählt eine Symptomzeile aus, z.B. 1, also Kopf-Merkmal
- gibt das aktuelle Symptom ein, z.B. rot
- erhält auf dem Bildschirm die beiden Anzeigen, daß
 - . ein Symptom, ein Merkmal übereinstimmt mit einem Symptom der potentiellen Diagnosen
 - . zwei Diagnosevorschläge vorliegen:
 - .. Hitzschlag
 - .. Sonnenstich.

Der Nutzer

- erhält den neuen Wissensstand angezeigt, also Kopf rot
- wählt eine weitere Symptomzeile aus, z.B. 3, sonstiges
- gibt das aktuelle Symptom ein, z.B. Brechreiz
- erhält auf dem Bildschirm die Anzeigen, daß
 - . zwei Symptome übereinstimmen
 - . ein Diagnosevorschlag vorliegt (gestützt durch zwei S.):
 - .. Sonnenstich,

Bild 2.5.

Programmkritik

Das BASIC-Programm EXPPR01 hat einige Vorteile, aber auch viele Nachteile, beispielsweise:

- das Wissen ist direkt im Programm gespeichert
(Behebung möglich: Dateiprogramm, s. Abschnitt 2.1.2)
- das Wissen ist strukturiert gespeichert:
die Daten "blass", "rot", "rot" stehen für die Fakten
"Der Kopf ist rot" usw. jeweils an erster Stelle des Datensatzes
(Behebung möglich, aber aufwendig)
- das Wissen ist gleichartig strukturiert gespeichert:
jeder Satz muß den gleichen Aufbau haben
(Behebung möglich, aber aufwendig)
- neues Wissen (neue Symptome und/oder neue Diagnosen) führt zu Programmänderungen, z.B. zur Veränderung des Menüs
(Behebung möglich, aber aufwendig).

Expertensystemansätze mit BASIC sind in /3.6/,/3.15/ und ausführlich in /3.14/ enthalten.

Aktueller Wissensstand

- (1) Kopf : normal, blass, rot: ?
(2) Haut : normal, kuehl, heiss: ?
(3) sonst: hohe T., schneller P., Brechreiz: ?

Von welchem Merkmal ist etwas bekannt ? (1-3): 1

- (1) Kopf : normal, blass, rot: rot

uebereinstimm.Symptome/Merkmale: 1

vorlaeufiger Diagnosevorschlag: Hitzschlag

vorlaeufiger Diagnosevorschlag: Sonnenstich

Ist von weiteren Merkmalen etwas bekannt? (j): j

Aktueller Wissensstand

- (1) Kopf : normal, blass, rot: rot
(2) Haut : normal, kuehl, heiss: ?
(3) sonst: hohe T., schneller P., Brechreiz: ?

Von welchem Merkmal ist etwas bekannt ? (1-3): 3

- (3) sonst: hohe T., schneller P., Brechreiz: Brechreiz

uebereinstimm.Symptome/Merkmale: 2

vorlaeufiger Diagnosevorschlag: Sonnenstich

Ist von weiteren Merkmalen etwas bekannt? (j): n

Ende EXPPRO1

Bild 2.5

2.1.2. PASCAL

Programmelemente

PASCAL (genannt nach B. PASCAL, dem Erfinder der zweiten Rechenmaschine 1642, WIRTH 1970, hier Turbo Pascal der Fa. BORLAND) ist eine algorithmische (operative, prozedurorientierte) Programmiersprache mit Programmelementen für

- Anfang (PROGRAM) und Ende (END.)
- Deklarationen (VAR, TYPE, CONST)/Zuordnungen (:=)
- Eingabe (READ) und Ausgabe (WRITE)
- Dateiaufbau (ASSIGN, REWRITE, ..., CLOSE),
Bild 2.6
- Suchen in Dateien (SEEK)
- Programmsteuerungen
 - . bedingter Sprung : wenn ... dann ... sonst ...
(IF ... THEN ... ELSE ...)
 - . Schleifen : für ... bis ... tue ...
(FOR ... TO ... DO ...)
 - : solange gilt ... tue ...
(WHILE ... DO ...),
jeweils
von Anfang bis Ende
(BEGIN END;),

anwendbar auf

- Ganzzahlen (INTEGER)
- Dezimalzahlen (REAL)
- Einzelzeichen (CHAR)
- Zeichenketten (STRING)
- Felder (ARRAY)
- Datensätze (RECORD),
Bild 2.7.

Programmstruktur

PASCAL stellt das "Wie", nicht das "Was" in den Vordergrund. PASCAL unterstützt, ja fordert sogar strukturierten Programmierstil: PROGRAM

Deklarationsteil

Anweisungsteil (BEGIN ... BEGIN ... END; ...)
END.

1. Rechenanweisungen

+ - * /	wie in Mathematik: Punkt- vor Strichrechnung (außer Infix- auch Präfix-/Postfix-Schreibung)
= < >	wie in Mathematik
abs	Absolutwert
sin cos usw.	wie in Mathematik
and(,) or(;)	wie in Logik
if(:-)	nicht wie in Logik wenn...dann, sondern (dann)...wenn... (bei Regeln)
random	Erzeugung von Zufallszahlen 0...1

2. Eingabe-Ausgabe-Anweisungen

readchar	Eingabe eines Zeichens
readln	Eingabe einer Zeile (bis 150 Zeichen von Tastatur oder bis 64 KBytes von Diskette)
write	Ausgabe
nl	neue Zeile, Leerzeile
makewindow	Aufbau eines Bildschirmfensters

3. Steueranweisungen

cut(!)	Backtracking verhindern (Lösungsbaum abschneiden, ist immer erfolgreich) Achtung: Kritik von ROBINSON: "CUT entspricht GOTO"!
fail	Backtracking erzwingen (Regel scheitern lassen)
cut fail	Backtracking verhindern und Regel scheitern lassen
repeat	Schleife, Abbruch durch Bedingung und cut

4. Zeichenkettenanweisungen

str_real	Zahl aus Zeichenkette
str_len	Länge einer Zeichenkette
frontchar	linkes Zeichen aus Zeichenkette
fronttoken	linker Teil aus Zeichenkette
concat	Zeichenkette aus zwei Zeichenketten

5. Dateianweisungen

file	Datei deklarieren (s. domains)
openread, -write	Datei zum Lesen/Schreiben öffnen
filepos	Direktzugriff auf Datei
closefile	Datei schließen
Dynamische Datenbanken (im RAM):	
asserta	Fakten einspeichern, vorn
assertz	Fakten einspeichern, hinten
retract	Fakten ausspeichern und somit löschen

6. Rahmenanweisungen

/*...*/	Bemerkung
.	Ende (Fakt, Regel, Programm)

Bild 2.9

Programm

EXPPR01 ist ein kleines Turbo-Prolog-Programm, das

- bei einigen Symptomen (hier maximal 8 als j/n)
- eine oder keine Diagnose (hier aus 3)
- vorschlagen soll, Bild 2.10, s.a. /3.7/.

Domains sind

- symptom=symbol, z.B. schneller_Puls
- antwort=char, z.B. 'j' oder 'n'.

Database speichert

- zutreffende Symptome als ja(symptom), z.B. ja(roter_Kopf)
- unzutreffende Symptome als nein(symptom).

Predicates sind

- start (als "Programmrahmen")
- diagnose(symbol)
- lösche_fakten (in der dynamischen Datenbank)
- symptom(symptom)
- frage(symbol,symptom).

Clauses sind

- Start-Regel für
 - . das "Machen" eines Fensters 1, weiß auf schwarz (?) mit Rahmen (?) und Überschrift "Aktueller Wissensstand", beginnend bei 0,0 über 25 Zeilen und 80 Spalten
 - . die Ausgabe der Eingabeaufforderung und von Leerzeilen
 - . die Ermittlung des Diagnosevorschlags
 - . die Ausgabe des Diagnosevorschlags
 - . das Löschen der Fakten (ja, nein) in der Datenbank
- Löschen-Regeln (für alle, für "anonyme" Variablen (_))
- Diagnose-Regeln mit zugehörigen Symptomen
- Symptom-Regeln, und zwar
 - . für gespeicherte Symptome, die dann selbstverständlich nicht mehr abgefragt werden müssen (cut !)
 - . für zu erfragende Symptome
- Frage-Regeln, also Fragetextausgabe, Antworteingabe, Antwortausgabe und Antwortspeicherung.

Goal ist hier die Start-Regel (damit EXPPR01 allein startet).

```
/* EXPPRO1 - Expertensystem-Versuch in Turbo-Prolog */

domains
    symptom=symbol
    antwort=char

database
    ja(symptom)
    nein(symptom)

predicates
    start
    loesche_fakten
    diagnose(symbol)
    symptom(symptom)
    frage(symbol,symptom)

clauses
    start if
        makewindow(1,7,7,"Aktueller Wissensstand",0,0,25,80),nl,
        write("Beantworten Sie jede Frage mit j oder n."),nl,nl,
        diagnose(D),nl,nl,
        write("vorlaeufiger Diagnosevorschlag: ",D),nl,
        loesche_fakten.

    loesche_fakten if retract(ja(_)),fail.
    loesche_fakten if retract(nein(_)),fail.
    loesche_fakten.

    diagnose("Hitzeerschoepfung") if
        symptom(blasser_Kopf),symptom(kuehle_Haut),
        symptom(schneller_Puls),symptom(norm_Temperatur).

    diagnose("Hitzschlag") if
        symptom(roter_Kopf),symptom(heisse_Haut),symptom(hohe_Temperatur).

    diagnose("Sonnenstich") if
        symptom(roter_Kopf),symptom(kuehle_Haut),symptom(brechreiz).

    diagnose("keiner"). 

    symptom(S) if ja(S),!.
    symptom(S) if nein(S),!,fail.

    symptom(blasser_Kopf) if
        frage("Hat der Patient einen blassen Kopf? ",blasser_Kopf).

    symptom(kuehle_Haut) if
        frage("Hat der Patient kuehle Haut, kalten Schweiß? ",kuehle_Haut).

    symptom(schneller_Puls) if
        frage("Hat der Patient schnellen Puls? ",schneller_Puls).

    symptom(norm_Temperatur) if
        frage("Hat der Patient normale Temperatur? ",norm_Temperatur).

    symptom(roter_Kopf) if
        frage("Hat der Patient einen roten Kopf? ",roter_Kopf).

    symptom(heisse_Haut) if
        frage("Hat der Patient eine heisse Haut? ",heisse_Haut).

    symptom(hohe_Temperatur) if
        frage("Hat der Patient eine hohe Temperatur? ",hohe_Temperatur).

    symptom(brechreiz) if
        frage("Hat der Patient Brechreiz? ",brechreiz).

    frage(Text,Stichwort) if
        write(Text),readchar(Antwort),write(Antwort),nl,
        Antwort='j',assert(ja(Stichwort)).

    frage(Stichwort) if
        assert(nein(Stichwort)),fail.

goal
    start.
```

Bild 2.10

Programmnutzung

Das Programm

- beginnt mit der Prüfung der Diagnose "Hitzschlag", weil diese Diagnose-Regel an erster Stelle steht
(Rückwärtsverkettung)
- prüft, ob das erste Symptom (blasser_Kopf) als ja/nein in der Datenbank steht - das ist nicht der Fall
- stellt dem Nutzer also die Frage nach dem Symptom (blasser_Kopf), die der Nutzer verneint; das Zwischenziel ist gescheitert, die Regel ist gescheitert, Backtracking setzt ein (Tiefensuche).

Das Programm

- prüft nun die zweite Diagnose, also "Hitzeerschöpfung"
- prüft das erste Symptom (roter_Kopf) in Datenbank (nein)
- fragt den Nutzer (j)
- prüft das zweite Symptom (heisse_Haut) (nein, n)
- prüft nun die dritte Diagnose, also "Sonnenstich", fragt aber nun nicht mehr nach dem Symptom (roter_Kopf), weil die Antwort (ja(roter_Kopf) bereits in der Datenbank steht.

Der Nutzer

- beantwortet die Fragen mit j oder n
- erhält dann den Diagnosevorschlag (gestützt durch drei S.):
Bild 2.11.

Programmkritik

Das PROLOG-Programm EXPPR01 hat noch viele Nachteile, z.B.:

- das Wissen steht direkt im Programm
(Behebung möglich, s. /3.7, Kapitel 4 und 5/)
- die Schlüsse werden nicht erklärt
(Behebung möglich, s. /3.7, Kapitel 4 und 5/)
- unsicheres Wissen wird nicht verarbeitet
(Behebung möglich, s. /3.7, Kapitel 10/)
- das Programm lernt nicht, stellt beispielsweise nicht die Diagnose-Regeln nach den Diagnose-Häufigkeiten um
(Behebung möglich, s. /3.7, Kapitel 8/).

Expertensystemansätze mit PROLOG sind in /3.3/ und ausführlich in /3.7/ und /3.12/ enthalten.

Beantworten Sie jede Frage mit j oder n.

Hat der Patient einen blassen Kopf?	n
Hat der Patient einen roten Kopf?	j
Hat der Patient eine heisse Haut?	n
Hat der Patient kuehle Haut, kalten Schweiß?	j
Hat der Patient Brechreiz?	j

vorlaeufiger Diagnosevorschlag: Sonnenstich

Bild 2.11

2.2. Programmiersysteme

2.2.1. XPRO

Programmerkmale

XPRO (Expert System Program, Fa. LANGNER Expertensysteme 1988) ist ein kleines Rahmensystem, eine kleine Shell zum Aufbau von Expertensystemen, s. z.B. /3.2/:

- Wissensrepräsentation : regelorientiert
- Wissenssicherheit : nur sicheres oder symbol. Wissen
- Regelgewinnung : deduktiv, also von außen
- Regelkettungsstrategie : Rückwärtsverkettung
- Regelsuchstrategie : Tiefensuche
- Regelanzahl, maximal : 1000 (!)
- Regelaufbau, -syntax : ... WENN ... UND ...
- Regelklärung : ja (Eingabe: warum)
- Programmiersprachen : C, PASCAL, PROLOG
- Betriebssysteme : MS-DOS, DCP
- Hauptspeicherbedarf : 64 KB, möglichst 128 KB.

Programmbeispiel

Mit einem Textprogramm (WS oder TP) wird eine Datei, beispielsweise diadat.wb, aufgebaut,

Bild 2.12.

Programmnutzung

Der Nutzer

- lädt XPRO mit "xpro"
- lädt die Wissensbasis mit "lade diadat.wb"
- startet XPRO mit "start".

Der Nutzer

- gibt "?" ein, wenn er die potentiellen Symptome nicht kennt
- erhält die potentiellen Symptomwerte (einschließlich der Werte "unbekannt" und "warum" sowie "ende")
- erhält nach Eingabe von "warum" diejenige Regel, die gerade geprüft wird
- erhält nach Eingabe der aktuellen Symptome den Diagnosevorschlag,

Bild 2.13.

```
ziel (diagnose_vorschlag).
diagnose_vorschlag (hitzeerschoepfung) WENN
    kopf (blass) UND
    haut (kuehl) UND
    sonst (schneller_puls).
diagnose_vorschlag (hitzschlag) WENN
    kopf (rot) UND
    haut (heiss) UND
    sonst (hohe_temperatur).
diagnose_vorschlag (sonnenstich) WENN
    kopf (rot) UND
    haut (kuehl) UND
    sonst (brechreiz).
? (kopf) : Wie sieht der Kopf aus?
! (kopf) : blass, rot.
? (haut) : Wie fuehlt sich die Haut an?
! (haut) : heiss, kuehl.
? (sonst) : Welche Symptome gibt es sonst?
! (sonst) : schneller_puls, hohe_temperatur, brechreiz..
```

Bild 2.12

```
XPRO> start
Wie sieht der Kopf aus?
>> ?
Bitte antworten Sie mit:
blass, rot, unbekannt, warum, ende

Wie sieht der Kopf aus?
>> warum
Uberpruft wird gerade die Regel:
diagnose_vorschlag(hitzeerschoepfung) WENN
    kopf(blass) UND
    haut(kuehl) UND
    sonst(schneller_puls).

Wie sieht der Kopf aus?
>> rot

Wie fuehlt sich die Haut an?
>> kuehl

Welche Symptome gibt es sonst?
>> brechreiz

diagnose_vorschlag = sonnenstich
Konsultation beendet.
```

Bild 2.13

XPRO>

2.2.2. GURU

Programmerkmale

GURU (Ehrwürdiger, Lehrer, Fa. MDBS Inc.) ist ein großes Rahmensystem, eine Shell zum Aufbau von Expertensystemen, s. z.B. /3.2/:

- Wissensrepräsentation : regelorientiert
- Wissenssicherheit : unsicheres Wissen (CF und Fuzzy)
(mehrere Algebren wählbar)
- Regelgewinnung : deduktiv
- Regelkettungsstrategie : Rückwärts-/Vorwärtsverkettung
(mehrere Auswahlstrategien wählb.)
- Regelsuchstrategie : Tiefen-/Breitensuche
(mehrere Suchintensitäten wählb.)
- Regelanzahl, maximal : unbegrenzt
- Regelaufbau, -syntax : RULE: name
IF ... AND/OR ... THEN ... AND/OR
- Regelerklärung : ja
- Programmiersprache : C
- Betriebssysteme : MS-DOS, UNIX, VMS
- Hauptspeicherbedarf : 640 KB.

GURU hat weitere bemerkenswerte Merkmale:

- Integration üblicher Standard-Computerfunktionen, nämlich Schreiben, Rechnen, Zeichnen (auch Freihandzeichnen), Suchen in Datenbanken
- Dialog in fast natürlicher Sprache
- Verwaltung einer Konsultationsbibliothek
(für jede Konsultation eine Datei)
- Aufzeichnung aller Abhängigkeiten in einem Regelbaum
für jede Wissensbasis
- Wählen, Abfragen und Übernehmen von Informationen aus internationalen Datenbanken und anderen Diensten (automatisch, wenn eine GURU-Regel so formuliert ist)
- Einstellbarkeit der Bandbreite von Merkmalwerten
(z.B. Körpertemperatur zwischen 35 und 42°C)
- Erweiterbarkeit von Regeln um Kommentare für die Erklärungskomponente,

Bild 2.14.

RULE: R1

IF: symptom = roter_kopf
AND symptom = kuehle_haut
THEN: diagnose_vorschlag +=
("hitzeerschoepfung" CF 10;
"hitzschlag" CF 25;
"sonnenstich" CF 65)

REASON:

Ein roter Kopf ist fuer
- Hitzschlag
- Sonnenstich
charakteristisch;
eine kuehle Haut ist fuer
- Hitzeerschoepfung
- Sonnenstich
charakteristisch.
Die beiden Symptome sind
fuer einen Diagnosevorschlag
mit CF ueber 65 nicht ausreichend.

Bild 2.14

2.2.3. RULEMASTER

Programmerkmale

RULEMASTER (Regelmeister, Regelmacher, Fa. RADIANT Corp.) ist ein mittelgroßes Rahmensystem, eine Shell zum Aufbau von Expertensystemen, s. z.B. /3.2/:

- Wissensrepräsentation : regelorientiert
- Wissenssicherheit : unsicheres Wissen (CF und Fuzzy)
- Regelgewinnung : induktiv und/oder deduktiv
- Regelkettungsstrategie : Rückwärts-/Vorwärtsverkettung
- Regelsuchstrategie : Tiefen-/Breitensuche
- Regelaufbau, -syntax : IF ... IS ... ELSE ...
- Regelerklärung : ja
- Programmiersprachen : C, portierbare Ergebn. (C, FORTRAN)
- Betriebssysteme : MS-DOS, DCP, UNIX
- Hauptspeicherbedarf : 512 KB, möglichst 640 KB.

Programmbeispiel

RULEMASTER akzeptiert deduktive Regeln als Regelbäume
(in der Sprache RADIAL für operative Deduktionsmoduln),
beispielsweise:

MODULE: diagded

INTENT: "Ermittlung von Diagnosen"

CHILD: kopf

CHILD: haut

STATE: ermittlung

IF (kopf) IS

"rot": IF (haut) IS

"heiss": (advise "Diagnose: Hitzschlag", GOAL)

ELSE (advise "Diagnose: Sonnenstich", GOAL)
ELSE (advise "Diagnose: keine", GOAL)

GOAL OF diagded.

RULEMASTER generiert auch Regeln induktiv
(aus den EXAMPLES in Induktionsmoduln),
Bild 2.15

und erzeugt daraus RADIAL- bzw. C- und FORTRAN-artige Moduln.

MODULE: diagint

DECLARATIONS:

| INTENT: "Ermittlung von Diagnosen"

 CHILD: kopf

 CHILD: haut

STATE: ermittlung

ACTIONS:

 keine | advise "Diagnose: keine" |

 hitze | advise "Diagnose: Hitzschlag" |

 sonne | advise "Diagnose: Sonnenstich" |

CONDITIONS:

 kopf | kopf |(blass, rot)

 haut | haut |(heiss, kuehl)

EXAMPLES:

 blass heiss =: keine

 blass - =: keine

 rot heiss =: hitze

 rot kuehl =: sonne

Bild 2.15

2.3. Beispiele

2.3.1. MYCIN

MYCIN (auf Pilzkulturen hinweisende Endung von Antibiotika, z.B. Streptomycin, SHORTLIFFE 1976) war eines der ersten Expertensysteme, s. z.B. /3.4/,/2.5/.

Die Grundlagen sind:

- Meningitis und Blutinfektionen sind zwar makroskopisch, also leit-symptomatisch mit Regeln der Art

 WENN Nackensteife

 UND hohes Fieber

 UND Bewußtseinstrübung

 DANN Meningitis (CF 90 % bzw. 0.9)

diagnostizierbar (Therapie: oft nur symptom-bezogen);

Meningitis und Blutinfektionen sind aber mikroskopisch, ursächlich erst mit mikrobiologischen Verfahren (Züchtung von Kulturen des Erregers mit Nährösungen) therapierelevant diagnostizierbar (Therapie: ursachen-, erreger-bezogen).

- Erregerkulturen sind meist erst nach Tagen auswertbar, Erregertypen sind aber oft schon vorher an Farbe, Form oder Verhalten erkennbar.
- Spezielle Erregertypen erfordern spezielle Antibiotika.

Das Wissen besteht aus

- etwa 500 Regeln,

 Bild 2.16

- Faktoren für die

 . Regelsicherheit (Diagnosesicherheit bei Regelanwendung)

 . Faktenunsicherheit (Symptomerhebungunsicherheit),

woraus dann Schlüsse mit einer Gesamtsicherheit entstehen,

 Bild 2.17, s.a. Abschnitt 1.1.3.

MYCIN hatte anfangs eine - von Experten eingeschätzte - Diagnose-Trefferrate und Therapie-Trefferrate von 50 bis 70 Prozent, später von etwa 90 Prozent, s.a. Abschnitt 0.

Aus MYCIN entstand ein Rahmensystem, die Shell EMYCIN (Empty MYCIN).

Regel 156:

WENN Kultur: Blut
UND Morphologie: Stäbchen
UND Gramfärbung: gramnegativ
UND Erregerort: obere Harnwege
UND Eingriffe: nicht Harnwegeeingriffe
UND Behandlung: nicht wegen Harnweginfektion
DANN Organismus (Erreger): E.coli (0.6, also 60 %)

Bild 2.16

Faktenunsicherheiten (Symptomerhebungunsicherheiten):

Kultur	Morph.	Gramf.	Erregerort	Eingriffe	Behandlung
1.0	0.9	0.8	0.9	1.0	1.0

Minimum: 0.8

Regelsicherheit: 0.6

Gesamtsicherheit der Regel 156:

$$0.8 * 0.6 = \underline{0.48}$$

Gesamtsicherheit einer Regel ..., die ebenfalls E.coli
stützt: = 0.50

Gesamtsicherheit, daß der Erreger E.coli ist:

$$0.48 + 0.50 - 0.48 * 0.50 = 0.98 - 0.24 = \underline{0.74}$$

Bild 2.17

2.3.2. SPES

SPES (strukturiertes psychopathologisches Erfassungssystem nach KÜHNE, GRÜNES und KOSELOWSKI 1983) ist Voraussetzung für ein Expertensystem /2.2/.

Die Grundlagen:

- Bisher stützt sich die psychiatrische Diagnose
 - . vorwiegend auf anamnestische Daten, explorative Befunde und verlaufsspezifische Kriterien - die psychopathologischen Merkmale
 - . weniger auf neurophysiologische und biochemische Merkmale.
- Bisher entsteht die psychiatrische Diagnose
 - . nach nicht-standardisierter Erfassung psychopathologischer Merkmale
 - . auch von Lehrmeinung und Qualifikation abhängig
 - . vielleicht zu stark subjektiv;
- künftig entsteht die psychiatrische Diagnose auch
 - . nach standardisierter Erfassung
 - . weniger von Lehrmeinungsvielfalt und Qualifikation abhängig
 - . somit auch objektivierter
 - . rechentechnisch synthetisch vorbereitbar und auch analysierbar.

Das Wissen entsteht in einer vorbereitenden Lernphase an einer Lernstichprobe (hier: 4880 Patienten):

- von Symptomen,
Bild 2.18
- Über multivariate statistische Verfahren, beispielsweise
 - . Clusteranalysen
 - . Konfigurationsfrequenzanalysen
 - . Diskriminanzanalysen,s. z.B. /4.3/
- zu Syndromen,
zu statistisch und psychiatrisch sinnvollen Typen (hier: 44),
Bild 2.19, s.a. /3.5, Programm 2.2.1/.

Ein Expertensystem für die Psychiatrie könnte in der Kannphase künftig psychopathologische Typen /2.2/, neurophysiologische Klassen /2.5/ und biochemische Klassen zur Diagnose-Unterstützung nutzen.

1.3 DENKEN

(Symptombewertung: nicht vorhanden = 0, vorhanden = 1)

obligat		akzessorisch	
Konzentrationsschwäche	201	Merkschwäche	211
Einfallsarmut	202	Gedankendrägen	212
Déjà Phänomene	203	Gedankenabreißen	213
Bizarre Einfälle	204		214
Überwertige Ideen	205	Wahneinfall	215
Wahnähnliche Gedanken	206	Wahn systematisierung	216
Wahrwahrnehmung	207		217
Wahn	208		218

Bild 2.18

Patient: Leerbefund (symptomfrei) Einrichtung:

Pat.-Nr.: 1111111111 PKZ : 111111111111

Datum : 12.05.88 Uhrzeit: 13

IKK(9.) : 295.0 Behandlung: stationär

U.-Code : kos

Explorativer Befund: nicht erhoben (evtl. pruefen)

Affektivitaet: Typ 1= relativ ungestoert

Wahrn./Ich. : Typ 1= ungestoert

Denkstoerung : Typ 8= relativ ungestoert

veg. Stoerung.: Typ 1= vegetativ ungestoert

Nichtexplor.Befund: nicht erhoben (evtl. pruefen)

Nichtexplor. : Typ 11= objektiv relativ unauffällig

Gesamtyp 40

= E: / O: psychopathologisch relativ unauffällig

Gesamtypgruppe DN2

= affektiv /veget. Restsymptome, nichtexplor.unauffällig (subakut)

IKK-Spezifität des Gesamtyps

IKK(9.)	295.	:	296.	:	300.	:											
.0	.1	.2	.3	.4	.6	:.0	.2	.1	.3	:.4	.5	:297	306	303	org übr.		
-	-	-	-	-	-	-	-	-	-	-	-	T	TTT	+++	.	---	---

Bild 2.19

2.3.3. PPSA

PPSA (psychophysiologische Systemanalyse nach MICHEL) wird der Kern eines Expertensystems /2.3/,/2.4/.

Die Grundlagen sind:

- Psychophysiologische Regulationsprozesse sind wichtige Merkmale des bio-psycho-sozialen Status, beispielsweise Ausdruck des bio-psycho-sozialen Alters und Alterns;
also sind die Regulationsmechanismen zu testen.
- Physische Belastung ist im allgemeinen regulationsstabilisierend,
psychische Belastung ist im allgemeinen regulationsdestabilisierend;
also muß unter psychischer Belastung getestet werden.
- Einzelmerkmale kennzeichnen die Regulationsmechanismen schlecht (Herzfrequenz trennt kaum Herzkranke/Gesunde), Komplexmerkmale kennzeichnen sie dagegen gut;
also müssen mehrere Merkmale erfaßt und multivariat statistisch vorverarbeitet werden.

Das Wissen entsteht

- aus Meßdaten
 - (Blutdruck, Atmung, Hautwiderstand, EKG, EEG, EMG usw. sowie Reaktionszeiten und Fehlerraten), abgeleitet bei
 - . 3 Minuten Ruhephase 1 nach der Instruktion
 - . 5 Minuten Übungsphase (Lernphase) zur Mustererkennung
 - . 3 Minuten Ruhephase 2
 - . 5 Minuten Arbeitsphase (Kannphase) zur Mustererkennung
 - . 3 Minuten Ruhephase 3,
- Bild 2.20
- aus Fragedaten
 - (Alter, Beruf, Belastungen, Störungen usw.).

Das Expertensystem entsteht in zwei Phasen:

- Lernphase zum Finden von Schwellwerten, Regeln und Modellen
 - Kannphase zum Finden von Diagnosen und Prognosen,
- Bild 2.21.

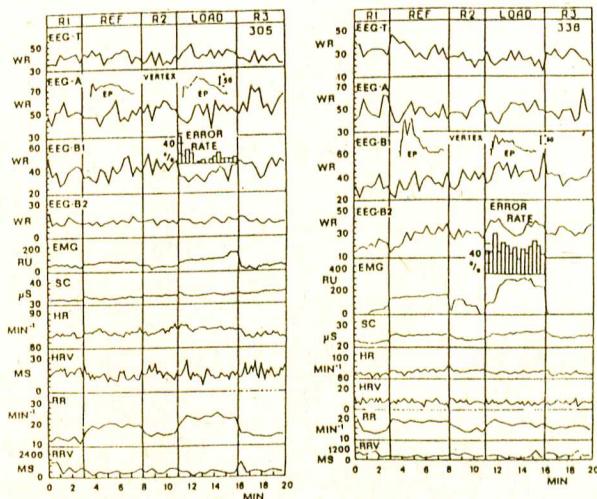


Bild 2.20 (Proband 305 und Proband 338)



Bild 2.21

3. Ausblick

3.0. Übersicht

In der Künstlichen Intelligenz (KI, auch AI), einem jungen Zweig der jungen Wissenschaft Informatik (s. /1.1//1.2//1.3//1.4/), gibt es zwei Richtungen:

- die mehr funktionelle Modellierung von Denkverfahren, die derzeitige Expertensysteme hervorbrachte (bisher gefördert, Grenzen sind jedoch bald absehbar)
- die mehr strukturelle Modellierung vom Denkgebilde Gehirn, die neurale Netze als künftiges Computerkonzept hervorbrachte (nunmehr gefördert, Grenzen noch nicht absehbar).

3.1. Software

Wünschenswert sind Expertensysteme, die lernen können, und zwar

- nicht nur auswendig lernen (s. Datenverarbeitung)
- nicht nur aus Beispielen lernen (s. RULEMASTER)
- auch durch Belehrung lernen (s. neurale Netze)
- auch durch Analogie lernen
- auch durch Beobachtung lernen.

Lernen ist teilweise bereits jetzt einfach realisierbar, und zwar

- durch Umspeichern von Regeln
- durch Sortieren von Regeln
- durch Modifizieren der Regelbasis
- durch Bildung von Metaregeln,

Bild 3.1, s.a. /3.7/.

Wünschenswert sind Expertensysteme, die diskutieren können, und zwar

- in natürlicher Schrift- oder gar Lautsprache
- über ein begrenztes, aber nicht zu enges Thema
- durch Beantworten von Fragen ("warum", "warum nicht", "wie", "was nun")
- durch Stellen von Fragen
- durch Antworten der Art "ja, aber ...", "nein, aber ...",

Bild 3.2, s.a. /3.17//3.10//3.1/.

Beispiele: Metaregeln (Regeln über Regeln, Überregeln)

WENN roter_Kopf

DANN wende Regeln 2 und 3 an.

WENN wenige Symptome

DANN füge "vielleicht" vor Diagnosevorschlag ein.

WENN Sicherheit kleiner als CF 25

DANN gib Diagnosevorschlag nicht aus.

Bild 3.1

Nutzer: "Patient N.N. hat einen blassen Kopf.

Hat er Sonnenstich ?"

Rechner: "Nein, aber ...

er leidet vielleicht unter Hitzeerschöpfung,

für die

- blasser Kopf

- kühle Haut

- schneller Puls

charakteristisch sind.

Empfehlenswert sind:

- Patient in den Schatten legen

- Patient Salzwasser geben

- Patient zudecken, wenn er fröstelt.

Oder ist der Patient bewußtlos ?"

Nutzer: "Ja."

Rechner: "Sofort:

- Seitenlage

- Notruf !"

Bild 3.2

3.2. Hardware

Klassische Computer arbeiten seriell, Schritt für Schritt (z.B. Nutzung einer Regel nach der anderen); sie arbeiten dort gut, wo der Mensch schlecht arbeitet: jeder Personalcomputer löst in einigen Sekunden mehr Rechen-aufgaben als ein Mensch in seinem gesamten Leben!

Künftige Computer arbeiten parallel, Ebene für Ebene (z.B. Nutzung vieler Regeln gleichzeitig); sie arbeiten auch dort gut, wo der Mensch gut arbeitet: jeder Mensch löst in einigen Sekunden eine Situationserkennungs-aufgabe - und das können auch die besten Computer (noch) nicht.

An einem kleinen Beispiel mit nur

- 16 Neuronen statt etwa 20 000 000 000 je Gehirn
- 4 Verbindungen statt etwa 10 000 je Neuron
- 4 Sensorbildpunkten statt etwa 400 000 bei Videokameras zeigt,
- wie neurale Netze aufgebaut sind (Struktur)
- wie neurale Netze genutzt werden (Funktion),

Bild 3.3 nach /3.9/:

Neurale Netze bestehen aus

- Eingangsebene, Mittelebenen (hier 2), Ausgangsebene
- Synapsenfeld, Verarbeitungsfeldern (1 bzw. 2), Schwellwert-feld je Neuron.

Neurale Netze erkennen so:

- Der Sensor bildet einen rechtsgeneigten Balken (/) ab.
- Die Eingangsebene gibt das Signal 0101 weiter.
- Die Mittelebene 1 gibt das Signal 0001 weiter
(das ist jedoch noch nicht eindeutig).
- Die Mittelebene 2 gibt das Signal 0001 weiter.
- Die Ausgangsebene gibt das Signal 0001 aus, also:
- Das neurale Netz erkennt den rechtsgeneigten Balken (/).

Neurale Netze werden bereits entwickelt:

- Simulation auf klassischen Computern, auch Personalcomputern (z.B. Programm BrainMaker von CALIFORNIA SCIENTIFIC SOFTWARE)
- Entwicklung von Mikroprozessoren für jeweils etwa 100 Neuronen (z.B. BELL LABORATORIES), s. /3.9/.

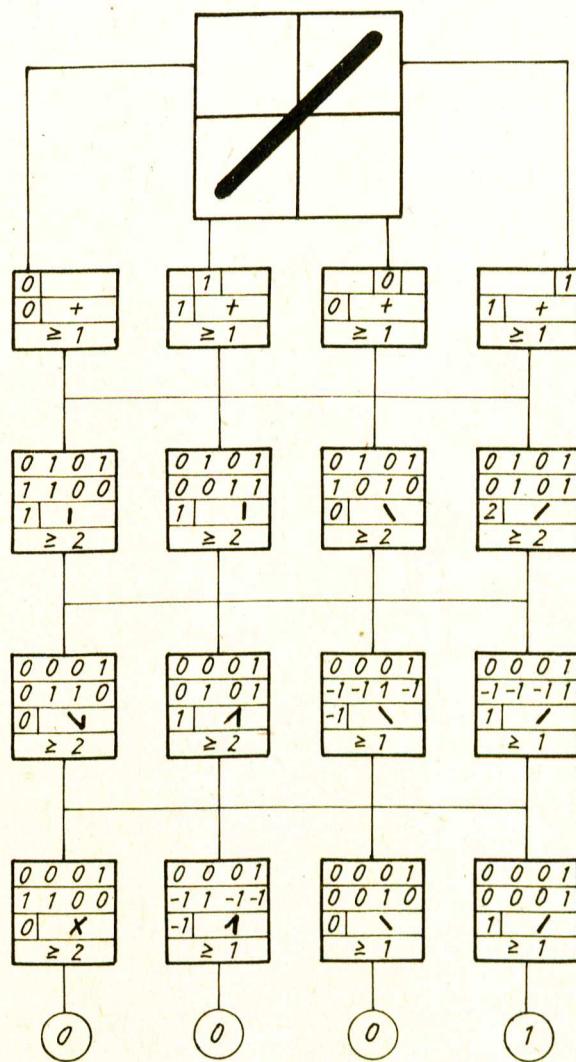


Bild 3.3

4. Quellen

Philosophie

- /1.1/ Dreyfus, H.L.; Dreyfus, S.E.
Künstliche Intelligenz
Von den Grenzen der Denkmaschine und dem Wert der Intuition
(a.d.Amerikanischen)
Reinbek bei Hamburg: Rowohlt 1987
(rororo computer)
- /1.2/ Jobst, E.; Nier, M.
Mikroelektronik und künstliche Intelligenz
Berlin: Akademie-Verlag 1987
- /1.3/ Liebscher, H.
Geist aus der Maschine ?
Philosophische Überlegungen zur künstlichen Intelligenz
Berlin: Dietz 1989
(Philosophische Positionen)
- /1.4/ Roth, M.
Die intelligente Maschine
Der Computer als Experte
Leipzig, Jena, Berlin: Urania-Verlag 1988
- /1.5/ Weizenbaum, J.
Mut ist ansteckend
Wochenpost (1989) 19, S. 16-17
(Interview: Versuchung und Verweigerung der Informatiker 2)

Medizin

- /2.1/ Benes^V, Heike
Abhängigkeit der Neuroleptika-Dosierung schizophrener
Patienten von nicht-medikamentösen Faktoren (Arbeitstitel)
Schwerin: Bezirksnervenklinik Schwerin 1989
(Dissertationsentwurf. Mentor: OMR Prof.Dr.sc.med.Giercke)
- /2.2/ Kuhne, G.-E.; Hempel, H.-D.; Koselowski, G.
Grundlagen eines rechnergestützten Klassifikators in der
Psychiatrie
Psychiat. Neurol. med. Psychol. 39 (1987) 3, S. 129-133
- /2.3/ Michel, J.; Cammann, H.; Koch, B.; Fleischer, B.;
Freude, M.; Uhlmann, G.; Vasadze, G.S.; Kubaneishvili, E.S.;
Dumbadze, G.G.; Mineev, I.F.; Chatiashvili, M.N.;
Zibadze, A.D.
Computer-supported Psychophysiological System Analysis -
New Ways of Function Analysis of Human Organism
in:
Willems, J.L.; van Bemmel, J.H.; Michel, J. (eds)
Progress in Computer-Assisted Function Analysis
Amsterdam, New York, Oxford: North-Holland 1988
(pp. 133-142)
- /2.4/ Michel, J.
Persönliche Mitteilung 1989

- /2.5/ Poppe, W.; Poppe, G.; Läuter, E.
Die Klassifikation endogener Psychosen durch ihren neuro-
physiologischen Befund
Psychiat. Neurol. med. Psychol. 33 (1981) 11, S. 681-688
- /2.6/ Ohmann, C.
Computergestützte Diagnose und Expertensysteme
Dtsch. med. Wschr. 114 (1989) 7, S. 268-275

Informatik

- /3.1/ Appelrath, H.-J.
Von Datenbanken zu Expertensystemen
Berlin(W), Heidelberg, New York: Springer 1985
(Informatik-Fachberichte 102)
- /3.2/ Fischer, R.
PC-Expertensysteme
Haar bei München: Markt & Technik 1989
(mit Diskette: Shell XPRO)
- /3.3/ Geske, U.
Programmieren mit PROLOG
Berlin: Akademie-Verlag 1988
(Informatik, Kybernetik, Rechentechnik 21)
- /3.4/ Hennings, R.-D.; Munter, H.
Artificial Intelligence. 1. Expertensysteme
(Nachdruck der 1. Auflage von 1985)
Berlin: Lehmann und Mathware-Verlag 1988
- /3.5/ Herrig, D.
Programme für die Nervenklinik
in:
Giercke, K. (Hrsg.)
Schweriner BNK-Schriften 3/1988
Schwerin: Bezirksnervenklinik Schwerin 1988
- /3.6/ James, M.
Künstliche Intelligenz in BASIC
(a.d. Englischen)
Landsberg am Lech: mvg (moderne verlagsgesellschaft) 1985
(Computer lernen)
- /3.7/ Janson, A.
Expertensysteme und Turbo-Prolog
München: Franzis 1989
(Franzis-Arbeitsbuch)
- /3.8/ Knauss, W.
Turbo-Prolog
Grundlagen, Programmertechniken, Anwendungen
München, Wien: Hanser 1987
- /3.9/ Leckebusch, J.
Neurale Netze
Das Gehirn steht Pate für den Computer der Zukunft
P.M.Computerheft (1989) 1, S. 40-44

- /3.10/ Puppe, F.
Diagnostisches Problemlösen mit Expertensystemen
Berlin(W), Heidelberg, New York...: Springer 1987
(Informatik-Fachberichte 148. Subreihe KI)
- /3.11/ Rolle, G.
Expertensysteme für Personalcomputer
Würzburg: Vogel 1988
(Chip-Wissen)
- /3.12/ Savory, S.E.
Grundlagen von Expertensystemen
München, Wien: Oldenbourg 1988
(Lehrbuch der Nixdorf Computer AG)
- /3.13/ Schnupp, P.; Leibrandt, U.
Expertensysteme
Nicht nur für Informatiker
Berlin(W), Heidelberg, New York, Tokyo: Springer 1986
- /3.14/ Soll, H.-J.
KI - Expertensysteme programmieren
Einführung in die Funktionsweise mit Beispielen in BASIC
München: Franzis 1989
(Franzis-Fachbuch)
- /3.15/ Stede, M.
Einführung in die Künstliche Intelligenz
(2 Bde.)
Spredlingen: Luther 1983/1984
- /3.16/ Stoyan, H.
LISP - Anwendungsgebiete, Grundbegriffe, Geschichte
Berlin: Akademie-Verlag 1980
- /3.17/ Wernecke, W.; Horlacher, E.; Stauss, M.; Erben, A.;
Fehrle, T.
KEYSTONE. Ein wissensbasiertes System mit natürlichsprachlicher Dialogkomponente
Informatik. Forschung und Entwicklung 3 (1988) 4, S. 153-163

Statistik

- /4.1/ Bocklisch, St.F.
Prozeßanalyse mit unscharfen Verfahren
Berlin: Verlag Technik 1987
- /4.2/ Faulbaum, F.; Uehlinger, H.-M. (Hrsg.)
Fortschritte der Statistik-Software 1
Stuttgart, New York: Fischer 1988
- /4.3/ Herrig, D.
Statistik für die Nervenklinik
in:
Giercke, K. (Hrsg.)
Schweriner BNK-Schriften 4/1989
Schwerin: Bezirksnervenklinik Schwerin 1989

