Moderne Aspekte des Wissensverarbeitung

Ein interaktiver Lernbehelf für das Web Based Training

Diplomarbeit

an der

Technischen Universität Graz

vorgelegt von

Gerald Reif

Institut für Informationsverarbeitung und Computergestützte neue Medien
(IICM),
Technische Universität Graz
A-8010 Graz

20. Jänner 2000

Diese Arbeit ist in deutscher Sprache verfaßt.

Betreuer:	o.Univ.-Prof. Dr. Dr.h.c. Hermann Maurer
Mitbetreuender Assistent:	Dipl.-Ing. Christian Gütl

Kurzfassung

Die moderne Informationsgesellschaft ist geprägt von den rasch ansteigenden Wissensmengen. Dieses exponentiell ansteigende Wissen führt in der Wissenschaft, aber auch in anderen Lebensbereichen, zu Problemen bei der Informationsbeschaffung. Die Wissensverarbeitung hat die Aufgabe, den Menschen im Umgang mit Wissen zu unterstützen. Um diese Aufgabe zu erfüllen, bedient sich die Wissensverarbeitung der Techniken der Künstlichen Intelligenz.

Im Untersuchungsbereich der vorliegenden Arbeit wird eine Einführung in das Wissenschaftsgebiet der Wissensverarbeitung gegeben. Am Beginn wird auf die Grundbegriffe der Wissensverarbeitung und der Wissensrepräsentation eingegangen. In weiterer Folge werden Techniken der Künstlichen Intelligenz vorgestellt. Zu den Techniken gehören: Problemlösen durch Suchen, Spielbäume, Expertensysteme, Fuzzy Logic, Neuronale Netze, NeuroFuzzy. Weiters wird auf die Möglichkeiten eingegangen, welche Software Agenten in verteilten Systemen bieten. Der Untersuchungsbereich wurde als Kurs für ein Web Based Trainingssystem (WBT-System) gestaltet.

Im Gestaltungsbereich der Arbeit wurde ein interaktiver, dialogbasierter Lernbehelf für das WBT-System GENTLE entwickelt. Der Lernbehelf (Virtual Tutor) stellt ein Expertensystem dar, welches dem Studierenden bei der Auswahl der geeigneten Technik der Künstlichen Intelligenz zur Lösung seines Problems helfen soll. Über einen Frage-Antwort-Dialog bildet sich der Virtual Tutor ein Bild über die Problemstellung und gibt eine Reihung der geeigneten Techniken aus. Im Dialog mit dem Benutzter können vom Studierenden Erklärungstexte abgerufen werden, welche mit Hyperlinks auf das Theoriewissen im Untersuchungsbereich versehen sind.

Abstract

Our modern information society is characterized by the continually increasing knowledge. The exponential growth rate of scientific knowledge leads to problems on the sector of scientific information retrieval and also in other fields. The task of knowledge processing is to support people dealing with knowledge. Therefore knowledge processing is based on the techniques of artificial intelligence.

The theoretical part of this thesis consists of an introduction to the field of knowledge processing. First, the basic concepts of knowledge processing and representation are shown, then follows an introduction to the techniques of artificial intelligence. These techniques comprise Problem Solving by Searching, Gametrees, Expert System, Fuzzy Logic, Neuronal Networks and NeuroFuzzy. Afterwards, the opportunities offered by intelligent software agents in distributed systems are described. This part of the thesis was written as a tutorial for a Web Based Training-System (WBT-System).

In the practical part an interactive tutorial for the WBT-System GENTLE has been developed. The tutorial named „Virtual Tutor" is an Expert System, which helps students to choose the appropriate technique of artificial intelligence for solving a given problem. After a dialogue with the user the Virtual Tutor offers a ranking of the appropriate techniques. The dialogue with the user consists of explanations, including hyperlinks to the basic knowledge treated in the theoretical part.

Ich versichere hiermit, diese Arbeit selbständig verfaßt, andere als die angegebenen Quellen und Hilfsmittel nicht benutzt und mich auch sonst keiner unerlaubten Hilfsmittel bedient zu haben.

Meinen Eltern.

Inhaltsverzeichnis

Kapitel 1

Einleitung

1.1 Motivation

Der Mensch in der modernen Informationsgesellschaft steht zunehmend vor dem Problem der rasch steigenden Wissensmengen. Speziell in der Wissenschaft führt die exponentiell ansteigende Anzahl der Wissenschaftler zu einem exponentiellen Ansteigen der Anzahl der Publikationen[1]. Nach [Tröger1998] liegt die Verdoppelungszeit für den wissenschaftlich ausgebildeten Bevölkerungsanteil weltweit bei unter 15 Jahren. In der selben Geschwindigkeit steigt auch die Anzahl der Wissenschaftlichen Publikationen. Dies erschwert es einem Wissenschaftler umfassend über den aktuellen Wissenstand auf seinem Forschungsgebiet informiert zu sein.

Aber auch in der Wirtschaft, im Arbeits- und Privatleben steht der entscheidungstragende Mensch zunehmend vor dem Problem, aus der Fülle des ihm zur verfügungstehenden Wissens, in jeder Situation die für ihn handlungsrelevanten Informationen zu gewinnen, welche er für ein rationales Handeln benötigt. Zudem besitzt der Entscheidungsträger in vielen Fällen keinen Überblick über das, ihm zur verfügungstehende Wissen.

Ziel der Wissensverarbeitung ist es, Möglichkeiten zu schaffen, dem Menschen beim Umgang mit Wissen zu unterstützen. Dazu gehört sowohl das Exzerpieren von Informationen aus der Umwelt und das Speichern der Informationen als Wissen, das automatisierte Erlernen von Wissen und das Wiedergewinnen von Informationen aus gepeicherten Wissen. Eine genaue Definition von Wissensverarbeitung ist in Abschnitt 2.1.4 zu finden.

[1] Bereits 1980 verdoppelte sich in der Physik die Anzahl der wissenschaftlichen Arbeiten alle 13 bis 15 Jahre. [Dobrov1980]

Im Rahmen der Forschung auf dem Gebiet der Wissensverarbeitung wird an Lösungen zu verschiedensten Wissensproblemen gearbeitet. Dazu zählen:

- Die Unterstützung des Benutzers bei der Suche nach Informationen in Wissensspeichern.
- Die Entlastung des Menschen bei Routieneaufgaben durch den Einsatz von wissensbasierten Systemen.
- Der Umgang mit unsicherem Wissen in computerbasierten Systemen.
- Die Formalisierung von Expertenwissen zur Wissensweitergabe, sei es an Menschen oder an Software-Systeme.
- Das Erstellen von wissensbasierten interaktiven Unterrichtsmaterialien.
- Das automatisierte Erlernen von Wissen anhand von gegebenen Beispieldaten.

Zur Erreichung der obengenannten Ziele baut die Wissensverarbeitung auf den Erkenntnissen der Informationswissenschaften, der Bibliothekslehre und der Logik auf und bedient sich auch der Techniken der Künstlichen Intelligenz, welche in der nachfolgenden Arbeit vorgestellt werden sollen.

1.2 Strukturierung

Die vorliegende Arbeit ist in zwei Teile gegliedert: Im ersten Teil, dem Untersuchungsbereich, soll ein umfassender Überblick über das Gebiet der Wissensverarbeitung gegeben werden. Aufbauend auf dem im Untersuchungsbereich erarbeiteten Wissen, soll im zweiten Teil, dem Gestaltungsbereich, zur Lernunterstützung ein interaktiver dialogbasierter Tutor für das Web Based Trainigssystem GENTLE[2] entwickelt werden. Zudem wurde das im Untersuchungsbereich erarbeitete Wissen als HTML-Seiten aufbereitet und steht dem interaktiven Tutor in GENTLE als Hintergrundbibliothek zur Verfügung.

1.2.1 Der Untersuchungsbereich

In Kapitel 2 werden zunächst die grundlegenden Begriffe der Wissensverarbeitung vorgestellt und definiert. Weiters wird auf die Wissensverbreitung in der Wissenschaft, die Wissensspeicherung und die Wissenswiederauffindung, sowohl in der

[2] `http://wbt.iicm.edu`

traditionellen Form der Bibliotheken als über das Internet, eingegangen. Am Ende des Kapitels werden die Probleme beleuchtet, die sich durch das exponentielle Wissenswachstum ergeben.

In Kapitel 3 werden eingangs Versuche vorgestellt, den Begriff „Künstliche Intelligenz" zu definieren. Weiters werden die Anforderungen besprochen, welche die Künstliche Intelligenz an einen Wissensspeicher stellt. Der Hauptteil dieses Kapitels stellt Wissensrepräsentationsmethoden vor, welche diese Anforderungen erfüllen. Weiters wird auf die Notwendigkeit eingegangen, unsicheres Wissen zu speichern und zu verarbeiten.

Kapitel 4 behandelt das Problemlösen durch Suchen. Es werden unterschiedliche Algorithmen zur blinden Suche, zur heuristisch informierten Suche und zur Suche nach der optimalen Lösung vorgestellt.

In Kapitel 5 werden Algorithmen vorgestellt, welche es dem Computer ermöglichen Strategiespiele zu spielen. Speziell wird dabei auf den MiniMax- und auf den Alpha-Beta-Algorithmus eingegangen.

Zu Beginn des Kapitels 6 werden die Definitionen und Ziele eines Expertensystems beleuchtet. Weiters wird auf die unterschiedlichen Arten von Wissen menschlicher Experten eingegangen und die Motive für die Erstellung, sowie die Vor- und Nachteile eines Expertensystem diskutiert. Am Ende des Abschnitts wird auf den Aufbau und die Arbeitsweise eines Expertensystems eingegangen. Zudem wird mit den Certainty Factors eine Technik vorgestellt, mit deren Hilfe es möglich ist, unsicheres Wissen in Expertensystemen zu verarbeiten.

Eingangs wird in Kapitel 7 die mathematische Theorie der Fuzzy Sets vorgestellt. In weitere Folge wird die Arbeitsweise eines Fuzzy Regel Systems erläutert und dessen Vor- und Nachteile gegenüber einer klassischen Regelung besprochen.

Zu Beginn des Kapitels 8 werden die Grundlagen von Neuronale Netzen vorgestellt. Anschließend wird das Perzeptron und der dazugehörige Perzeptron-Lernalgorithmus erläutert. Am Ende des Kapitels wird der Backpropagation Algorithmus für mehrschichtige Neuronale Netze hergeleitet.

Anhand der Vor- und Nachteile von Neuronalen Netzen und der Fuzzy Logic werden in Kapitel 9 NeuroFuzzy Systeme vorgestellt. Dabei wird im Speziellen auf Kooperative Systeme und auf Hybride Systeme eingegangen.

In Kapitel 10 werden intelligente Software-Agenten als Möglichkeit vorgestellt, komplexe, verteilte Probleme zu bewältigen. Zu diesem Zweck greifen die Agenten auf die Techniken der künstlichen Intelligenz zurück, welche in den vorangegangenen Kapiteln vorgestellt wurden. Nach der Definition von intelligenten Agenten werden die Möglichkeiten besprochen, die sich für die Agenten durch die Kommunikationsfähigkeit ergeben. Desweiteren werden Sicherheitsaspekte besprochen und Beispiel-Implementationen für Agenten vorgestellt.

1.2.2 Der Gestaltungsbereich

Im Gestaltungsbereich der Arbeit wird, aufbauend auf die im Untersuchungsbereich gewonnenen Erkenntnisse, beispielhaft in einer Testimplementierung ein interaktiver dialogbasierter Lernbehelf, der „Virtual Tutor", entwickelt. Ziel des Virtual Tutors ist es, dem Studierenden bei der Auswahl einer Technik der künstlichen Intelligenz, zur Lösung seines Problems, zu unterstützen. Um den Lerneffekt für den Studierenden zu steigen, ist der Virtual Tutor in der Lage, jede an den Benutzer gestellte Frage im Input-Dialog zu begründen und einen Ausblick auf die möglichen Auswirkungen der Antwort zu geben. Zusätzlich können Erklärungen über das Zustandekommen der vom Virtual Tutor ausgegebenen Lösung abgefragt werden.

Die Implementierung des Virtual Tutors erfolgte als Java-Applet. Somit ist die Anwendung jederzeit für den Studierenden über das Netz erreichbar. Zudem bietet die Realisierung als Applet auch den Vorteil, daß sämtliche Erklärungstexte mit Verweisen zu einer Hintergrundbibliothek versehen werden können. Als Hintergrundbibliothek dient das als HTML-Seiten aufbereitete Wissen, welches im Untersuchungsbereich erarbeitet wurde. Im Kapitel 11 wird das Konzept des Virtual Tutors diskutiert, die Implementierung schrittweise vorgestellt und ein Ausblick für zukünftige Erweiterungsmöglichkeiten des Systems gegeben.

Teil I

Der Untersuchungsbereich

Kapitel 2

Der Begriff Wissensverarbeitung

Im einleitenden Kapitel werden zunächst die Grundbegriffe der Wissensverarbeitung eingeführt und erklärt. Anschließend werden historische Methoden der Wissensspeicherung, der Wissensverbreitung und der Wissenswiederauffindung vorgestellt und die zusätzlichen Möglichkeiten und Probleme aufgezeigt, die sich heute durch die Verwendung von modernen elektronischen Medien wie dem Internet ergeben. Im weiteren wird auf die Schwierigkeiten eingegangen, die sich durch die immer rasanter steigenden Informationsmengen in unserer modernen Informationsgesellschaft ergeben. Lösungsansätze für diese Probleme werden mit den Techniken der künstlichen Intelligenz (KI) aufgezeigt, welche in Kapitel 3 vorgestellt werden.

2.1 Begriffsdefinitionen

Der Begriff Information wird in verschiedenen Disziplinen der Wissenschaft und der Technik unterschiedlich — mit verschieden Schwerpunkten — definiert, wie im folgenden gezeigt wird. An Hand verschiedener Überlegungen wird eine Annäherung an die, in der Wissensverarbeitung verwendeten, Definitionen beschritten.

2.1.1 Informationsdefinition in der Nachrichtentechnik

In der Informationstheorie von Shannon und Weaver (1949) wird der Informationsgehalt einer Nachricht aus rein mathematischer Sicht definiert. Der Informationsgehalt I einer einzelnen Nachricht muß folgende Eigenschaften erfüllen [Herter1990]:

1. Der Informationsgehalt I_x einer Nachricht muß umso größer sein, je kleiner die Wahrscheinlichkeit p_x ihres Auftretens ist.

2. Eine Nachricht mit der Wahrscheinlichkeit $p_x = 1$ muß den Informationsgehalt $I_x = 0$ haben.

3. Der Informationsgehalt von voneinander unabhängigen Nachrichten soll sich addieren.

Die drei Kriterien werden durch folgenden formalen Ansatz erfüllt:

$$I_x = ld \frac{1}{p_x} \quad \ldots \text{ in bit/Symbol} \tag{2.1}$$

Das erste Kriterium ist durch die Verwendung der inversen Wahrscheinlichkeit erfüllt. Das zweite Kriterium legt die Verwendung des Logarithmus nahe. Das dritte Kriterium macht die logarithmische Definition zwingend notwendig. Treten zwei unabhängige Nachrichten mit den Wahrscheinlichkeiten p_1, p_2 auf, so muß der Informationsgehalt der kombinierten Nachricht $I = I_1 + I_2$ sein. Der formale Zusammenhang der Wahrscheinlichkeit des Auftretens beider Nachrichten ist $p = p_1 * p_2$. Durch die obige Definition 2.1 wird auch nachfolgende Gleichung erfüllt:

$$I = ld \frac{1}{p_1 * p_2} = ld \frac{1}{p_1} + ld \frac{1}{p_2} = I_1 + I_2 \tag{2.2}$$

Somit sind alle, der in Definition 2.1 gestellten Forderungen erfüllt.

Mit den bisherigen Überlegungen ist es möglich den Informationgehalt von einzelnen Nachrichtenelementen zu bestimmen. Die Entropie H stellt dagegen den mittleren Informationsgehalt aller, mit einer Symbolmenge[1] darstellbaren, Nachrichten dar. Dazu muß die Wahrscheinlichkeit des Auftretens jedes einzelnen Elementes $p_1, \ldots, p_n$ der Symbolmenge bekannt sein. Die Entropie wird über das arithmetische Mittel aller Werte I_x gebildet, die mit der Wahrscheinlichkeit ihres Vorkommens gewichtet sind [Herter1990].

$$H = \sum_{i=1}^{n} p(x_i)\, ld \frac{1}{p(x_i)} \quad \ldots \text{ in bit/Symbol} \tag{2.3}$$

Mit dieser Definition (Gleichung 2.3) kann gemessen werden, wieviel Information über eine gegebene Übertragungsstrecke übertragen werden kann. Es wird jedoch die Bedeutung der Nachricht für den Empfänger, z.B. eine Person, nicht berücksichtigt. Ob die Information inhaltlich wahr oder falsch, ob sie präzise,

[1] Als Symbolmenge kann z.B. die Menge der Elemente eines Zeichensatzes dienen.

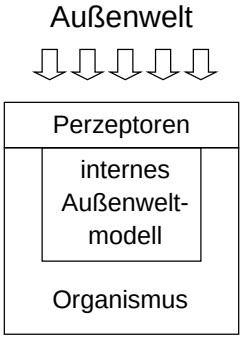

Abbildung 2.1: Nach dem Modell von Wersig nimmt der Organismus die Außenwelt nicht unmittelbar, sondern über den Umweg von Perzeptoren wahr.

nützlich oder wertvoll ist, bleibt außer Betracht; z.B. kann mit Hilfe der Entropie der mittlere Informationsgehalt der Nachricht ,,1+1=2"gemessen werden. Voraussetzung dafür ist, daß die Wahrscheinlichkeit des Auftretens jedes einzelnen Zeichens bekannt ist. Doch enthält diese Nachricht für die meisten Empfänger keine Information, weil dieser Zusammenhang allgemein bekanntes Wissen darstellt. Diese Nachricht teilt uns nichts Neues, Unerwartetes mit. Diese Art der Definition von Information findet daher hauptsächlich in der Nachrichtentechnik ihre Anwendung [Meadow1992].

Eine viel weiterreichende, philosophische Definition kommt aus dem Bereich der Informationswissenschaft und ist im folgenden Abschnitt beschrieben.

2.1.2 Definitionen in der Informationswissenschaft

G. Wersig geht bei seiner Definition von einem Modell des Organismus aus, in dem der Organismus in seinem Inneren ein Arbeitsmodell der Außenwelt aufbaut. Die Außenwelt ist nach dieser Vorstellung für den ,,Organismus" nicht unmittelbar zugänglich, sondern nur über ,,Perzeptoren" (Sinnesorgane) erfaßbar, siehe Abbildung 2.1. Ein ,,internes Außenweltmodell" im Organismus repräsentiert für diesen die ihn umgebende Realität. Wissen wird als die Struktur des internen Außenweltmodells definiert, Denken als Operation an diesem. Der Organismus befindet sich in einem Zustand der Ungewißheit, wenn er in einer bestimmten Lage über kein adäquates Programm zur Lösung der Situation verfügt. In diesem Fall muß er sein internes Außenweltmodell erst verändern, um auf die neue problematische Situation sinnvoll reagieren zu können. Dieser Prozeß wird als Denken bezeichnet. Anhand dieser Modellvorstellung lassen sich die folgenden Definitionen treffen [Wersig1971][Rauch1982]:

- Wissen ist die Struktur des internen Außenweltmodells.

- Ungewißheit besteht, wenn für eine problematische Situation kein Lösungsweg im internen Außenweltmodell vorliegt.
- Denken nennt man Operationen am internen Außenweltmodell.
- Redundanz ist Information, die keine Korrektur des internen Außenweltmodells mit sich bringt.
- Information ist die Reduktion von Ungewißheit.

Ausgehend von den in Abschnitt 2.1.1 und 2.1.2 vorgestellten Überlegungen werden im nachfolgenden Abschnitt für die Wissensverarbeitung notwendige Definitionen getroffen.

2.1.3 Definitionen in der Wissensverarbeitung

In der Wissensverarbeitung ist die Unterscheidung zwischen Begriffen Daten, Information und Wissen sehr wichtig. Diese Begriffe werden in der weiteren Arbeit wie folgt verwendet [Rauch1996][Meadow1992]:

Daten: Eine Kette von elementaren Symbolen (z.B. Zeichen), die einem bestimmen, vereinbarten Syntax folgen.

Information: Daten welche eine Zustandsänderung im Empfängersystem (Computer, Mensch, usw.) hervorrufen. Mit dieser Definition sind auch folgende Eigenschaften verbunden:

- Information muß einen gewissen Neuigkeitswert haben.
- Information ist kontextabhängig; Was für eine Person Information darstellt, muß noch lange nicht für eine andere Person gelten.
- Information kann redundant sein und in bestimmten Fällen doch den Informationscharakter nicht verlieren, nämlich dann, wenn unsicheres vorhandenes Wissen bestätigt wird.
- Information kann zu Verunsicherung führen, insbesondere, wenn neue Informationen im Gegensatz zu bisher erworbenen Wissen stehen.
- Information ist zeitabhängig, das heißt, der Zeitpunkt zu dem ich eine Information erhalte, bestimmte deren Wert. Im Extremfall kann zu spät erworbenes Wissen damit seinen Informationscharakter verlieren, z.B. Börsennachrichten.

- Information ist ein immaterielles Gut und wird gehandelt. Es kann in Beschaffung, Nutzung und Weiterleitung Kosten verursachen.
- Information ist auch ein gesellschaftlicher Faktor. Insbesondere in demokratischen Systemen muß der Bürger die Möglichkeit haben, seine Interessen gut informiert zu vertreten. Nur wenn Information keinen ausschließlichen Warencharakter hat, können gesellschaftliche Entscheidungen für alle möglichst transparent und informationell abgesichert getroffen werden.

Wissen: Wissen ist die in einem System (Computer, Mensch, Bibliothek, usw.) gespeicherte Information. Dabei kann Wissen folgende Eigenschaften besitzen:

- Wissen wird allgemein für wahr gehalten.
- Wissen kann falsch sein, z.B. Wissen, das durch falsche, irreführende Information entstanden ist.
- Allgemein für wahr gehaltenes Wissen kann sich als falsch herausstellen, z.B. das geozentrische Weltbild galt bis in das 16. Jahrhundert als allgemein anerkannt. Durch die Erkenntnisse von Kopernikus fand ein Paradigmenwechsel statt, seitdem wird das heliozentrische Weltbild allgemein anerkannt.
- Wissen kann Widersprüche enthalten, z.B. können in einem System zwei sich widersprechende Fakten gespeichert sein. Dieser Widerspruch tritt erst zu Tage, wenn zur Lösung eines Problems beide Fakten herangezogen werden.

Wissen kann somit als Informations-Ressource aufgefaßt werden. Abhängig von der Qualität der Systeme kann ein qualifizierteres System mehr Informationen aus einem Wissenspeicher extrahieren, als ein weniger Qualifiziertes. Ähnlich verhält es sich auch mit natürlichen Ressourcen, wie z.B. den Bodenschätzen. Unabhängig von der Qualifikation kann jeder Mensch ein Goldnugget als Wurfobjekt verwenden, jedoch nur der qualifizierte Goldschmied kann aus dem Edelmetall ein Schmuckstück erzeugen. Analog dazu muß ein System viele Qualifikationen besitzen, um aus Wissen, für das System wichtige Informationen zu extrahieren. Dieser Extraktionsvorgang stellt einen aktiven Prozeß dar. Ohne diesen Prozeß wird die Informations-Ressource Wissen nicht ausgeschöpft. Ohne diesen Prozeß bleibt ein Goldnugget ein Stein, bzw. ein Buch ein Dekorgegenstand.

Die in diesem Abschnitt getroffenen Definitionen für Daten, Information und Wissen sollen nun abschließend durch ein anschauliches Beispiel zusammengefaßt werden: Bei einer Fernsehübertragung werden die **Daten** entsprechend der

PAL-Norm (Syntax) codiert und übertragen. Nur syntaktisch korrekt übertragene Daten können beim Empfänger ordnungsgemäß auf der Bildröhre dargestellt werden.

Der Zuschauer hat das **Wissen**, daß er die dargestellten Gegenstände und Personen als solche erkennt und die Sprache des Sprechers versteht. Ohne dieses Wissen wäre das Fernsehbild nur eine Anordnung farbiger, sich bewegender Punkte.

Erfährt der Zuseher z.B. in einer Sportsendung, daß sein Lieblingsverein das letzte Spiel gewonnen hat, so wird er sich freuen. Er hat die **Information** aufgenommen und danach gehandelt. Die Informationen über das Spiel sind in sein Wissen übergegangen. Läßt sich der Zuseher jedoch nur vom Fernsehen berauschen oder ist er nicht an Sport interessiert, so wird er sich kaum an die Ergebnisse erinnern und hat somit keine Information aufgenommen.

Bedeutung

Ein verwandter Begriff zu Daten, Wissen, Information und deren Eigenschaften, ist die Bedeutung. Mit Bedeutung ist die Aussage gemeint, für die ein Wort, ein Satz oder mehrere Sätze bis hin zu einem Buch stehen. Wie aus dem allgemeinen Sprachgebrauch bekannt ist, ist die Bedeutung eines Wortes oder eines Satzes nicht eindeutig. Die Richtige Interpretation der Aussage „Ich habe einen Virus." macht Wissen um den Kontext, in dem die Aussage gefallen ist notwendig. Darüberhinaus wird auch Wissen über die Verwendungsmöglichkeiten der einzelnen Worte benötigt, sowohl auf der Senderseite der Aussage, als auch auf der Empfängerseite. Das Wort „Virus" im obigen Beispiel kann in diesem Zusammenhang für einen Computervirus oder für einen Krankheitserreger stehen. Die Wörter „Ich habe" können im Zusammenhang mit dem Wort „Virus" folgende Bedeutungen annehmen: „Ich habe" kann sich auf meinen Computer beziehen, auf dem sich ein Virus befindet, auf meinen Körper, in dem er als Krankheitserreger auftritt oder auf ein Reagenzglas, in dem ich eine Virenkultur halte. [Meadow1992]

Somit ergibt sich für die Bedeutung folgende Definition:

Die **Bedeutung** ist die situationsabhängige Aussage einer Menge von Worten, die von den Worten selbst, dem Wissen des Senders und des Empfängers und der Beziehung der Worte zueinander abhängt.

Die Bedeutung spielt in der Wissensverarbeitung speziell auf dem Gebiet der Spracherkennung, der automatisierten Fremdsprachenübersetzung und in der Informations Extraction eine Rolle.

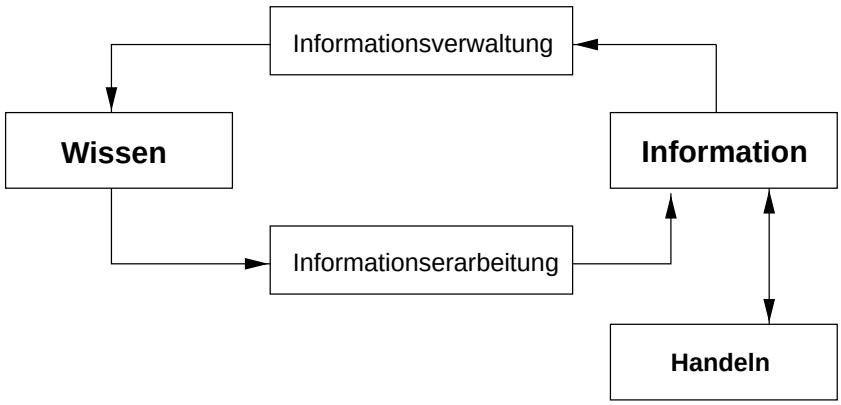

Abbildung 2.2: Kreislauf von Wissen und Information, nach Rainer Kuhlen. [Rauch1996]

2.1.4 Versuch der Definition von Wissensverarbeitung

In der Literatur gibt es keine anerkannte Definition von Wissensverarbeitung. Um eine Vorstellung über den Begriff zu erhalten, soll hier der Versuch unternommen werden, Wissensverarbeitung zu definieren.

Abbildung 2.2 zeigt den Kreislauf der Transformation von Wissen in Information und umgekehrt. Darin steht **Handeln** für denken, logisches schießen und kommunizieren, durch das wir Informationen aus unserer Umgebung aufnehmen. Handeln steht aber auch für das Verhalten in problematischen, unbekannten Situationen, in denen ein System (Mensch, Computer, usw.) Informationen benötigt, um sinnvoll zu handeln.

Im Prozeß der **Informationsverwaltung** werden die Informationen für die dauerhafte Speicherung aufbereitet. Der Speicher kann einerseits ein herkömmlicher Wissensspeicher wie eine Bibliothek sein, aber auch das menschliche Hirn, eine Datenbank oder die Wissensbasis eines Computersystems [Rauch1996].

Im Prozeß der **Informationserarbeitung** wird aus dauerhaft gespeicherten Wissen, die für die Situation nötige handlungsrelevante Information gewonnen. Damit verbunden ist das Wiederauffinden, das Verstehen und das Interpretieren des gespeicherten Wissens [Rauch1996].

Definition Wissensverarbeitung:

Bei der **Wissensverarbeitung** handelt es sich um jene Wissenschaft, die sich mit der Beschleunigung, der Rationalisierung und Automatisierung der Transformation von Wissen in Information und umgekehrt, befaßt. Dabei werden einerseits

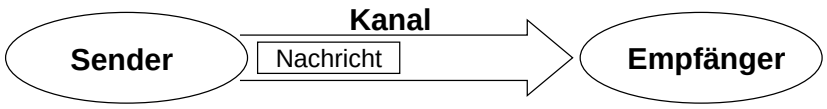

Abbildung 2.3: „Sender - Kanal - Empfänger" Schema [Rauch1993]

Informationen aus der Umgebung exzerpiert und als Wissen gespeichert, andererseits werden aus gespeicherten Wissen Informationen gewonnen, um sinnvolles handeln und entscheiden zu ermöglichen.

Um dies zu erreichen bedient sich die Wissensverarbeitung der Methoden der künstlichen Intelligenz, der Heuristik und des Automatisieren Erkennens. Die künstliche Intelligenz (KI) befaßt sich mit der Simulation menschlicher Intelligenz. Einen genaueren Definitionsversuch und eine Einführung in die wichtigsten Techniken findet sich unter Kapitel 3. Unter Heuristik versteht man den Versuch, Techniken des menschlichen Problemlösens in der Programmierung umzusetzen. In Abschnitt 4.2 werden heuristische Suchmethoden in Graphen besprochen. Das automatisierte Erkennen befaßt sich mit der Bild- und Spracherkennung. [Kupka1997]

2.2 Wissensverbreitung

Die Wissensverbreitung erfolgt durch einen Kommunikationsprozeß zwischen dem Informationsproduzenten, dem Sender, und dem Informationsempfänger. Dieser Prozeß kann entweder direkt oder über den Umweg eines Speichers erfolgen. [Rauch1993]

2.2.1 „Sender - Kanal - Empfänger" Schema

Bei dieser Art der Kommunikation müssen Sender und Empfänger zeitgleich miteinander in Verbindung stehen (siehe Abbildung 2.3). Bei dem Kanal kann es sich um ein direktes Gespräch, ein Telefonat, ein Chatt, eine Videokonferenz oder ähnliches handeln. Diese Art der Kommunikation zwischen dem Informationsproduzenten und dem Empfänger wird man in der Praxis recht selten antreffen. Viel häufiger tritt das „Sender - Speicher - Empfänger" Schema auf. [Rauch1993]

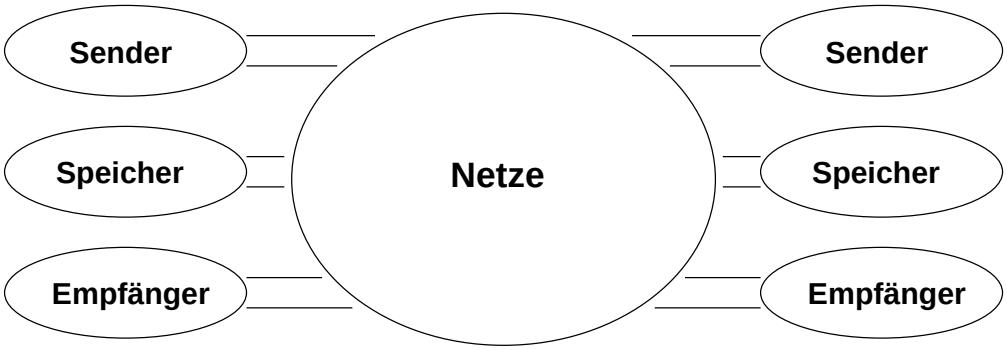

Abbildung 2.4: ,,Sender - Speicher - Empfänger" Schema, [Rauch1993]

2.2.2 ,,Sender - Speicher - Empfänger" Schema

Dieses Schema erlaubt es, daß Sender und Empfänger nicht zeitgleich präsent sein müssen, ja einander nicht mehr zu kennen brauchen (siehe Abbildung 2.4). Der Sender produziert seine Dokumente[2] für einen Speicher. Aus diesem Speicher können nun ein oder mehrere Empfänger, je nach Bedarf, Dokumente abrufen. Die Dokumente stellen somit die kleinsten Einheiten des Wissensspeichers dar. Diese Speicher müssen die einzelnen Dokumente in einer Form aufbewahren, die eine Speicherung über einen größeren Zeitraum gestattet und Möglichkeiten zur Verfügung stellen, gezielt nach bestimmten Dokumenten zu suchen. Beispiele für solche Speicher sind Bibliotheken, das Internet, die Massenmedien, der Buchhandel, usw. Die Kommunikation des Senders bzw. des Empfängers mit dem Speicher erfolgt über Netze. Bei den Netzen kann es sich um ein elektronisches Netz wie das Internet handeln, aber auch um herkömmliche Netze wie das Vertriebsnetz eines Verlages oder die Fernleihe der Universitätsbibliothek. [Rauch1993]

Der Nachteil dieses Schemas ist, daß der Sender den Empfänger im allgemeinen nicht mehr kennt und dadurch nicht auf seine speziellen Interessen, Wünsche und Kommunikationsmöglichkeiten eingehen kann. Der Sender muß sich vielmehr beim Erstellen eines Dokuments für eine Zielgruppe entscheiden und deren Vorwissen, Wortschatz und Bedürfnisse einschätzen. Dabei kann es zu großen Mißverständnissen kommen, z.B. eine Einführung in eine Textverarbeitung, die ständig unbekannte Fachbegriffe verwendet.

Der Empfänger ist in einer ähnlichen Situation. Er kennt im Normalfall den Produzenten eines Dokuments nicht. Es ist somit schwer für ihn zu beurteilen, von welcher Qualität, von welchem Niveau und für welche Zielgruppe ein Dokument

[2] Bei den Dokumenten handelt es sich um die kleinsten aus einem Speicher abrufbaren Wissenseinheiten. Dokumente können z.B. Bücher, Zeitschriften aber auch Akten, Baupläne oder Internetseiten, usw. sein.

geschrieben wurde.

Darum sollen die Speicher auch Informationen beinhalten, die es dem Empfänger ermöglichen, Dokumente hinsichtlich dieser Kriterien zu beurteilen. Ob und in welchem Ausmaß dies bei realisierten Speichern geschieht, wird in Kapitel 2.3 besprochen.

2.2.3 Publikationsformen

Im Sender-Speicher-Empfänger Schema, das in Abschnitt 2.2.2 besprochen wurde, nehmen die in Wissensspeichern gespeicherten Dokumente eine zentrale Rolle ein. Mit ihrer Hilfe wird zusammengehörendes Wissen in Einheiten gebündelt und dem Empfänger zugänglich gemacht. Wissenschaftliche Relevanz haben dabei nur Dokumente von hoher Qualität. Wie die Qualität von Dokumenten beurteilt wird, ist in den nachfolgenden Abschnitten beschreiben. Wissenschaftliche Journale, Tagungsbände (Proceedings) und Fachbücher stellen die Publikationsformen dar, die in der Wissenschaft die größte Bedeutung haben. Sie lassen sich gedruckt, elektronisch, z.B. auf CD-ROM, oder online, z.B. über das Internet oder Online-Datenbanken vertreiben. In den nachfolgenden Abschnitten werden die Schritte bis hin zur Veröffentlichung der einzelnen Publikationen erläutert.

2.2.3.1 Wissenschaftliche Journale und Tagungsbände

Wissenschaftliche Journale und Tagungsbände stellen Publikationsformen mit zwei verschiedenen Aufgabenbereichen dar. Wissenschaftliche Journale haben die Aufgabe die interessantesten und aktuellsten Arbeiten auf ihrem Fachgebiet zu veröffentlichen. In Tagungsbänden werden alle Referate, die im Zuge einer Tagung oder Konferenz vorgetragen wurden, veröffentlicht. Jedoch die typischen Schritte bis eine eingereichte Arbeit in einem Journal abgedruckt wird, bzw. für eine Tagung angenommen wird, sind identisch und sollen im folgenden beschrieben werden. Wie die Schritte im Detail aussehen, ist von Verlag zu Verlag, bzw. von Tagung zu Tagung verschieden. Bei der Beschreibung der Schritte wird hauptsächlich auf wissenschaftliche Journale eingegangen. [Grötschel1995]

Am Beginn steht der Abschluß der wissenschaftlichen Arbeit. Der Autor bzw. eine Autorengruppe bringen anschließend ihre Erkenntnisse zu Papier und senden dieses an einen Verlag. Dabei sind die Vorgaben des Verlages bezüglich der Länge, der Schriftgröße, dem Layout und andere Vorgaben zu beachten. Von einem Redakteur des Verlages wird das Papier (Paper) nun an mehrere Begutachter versandt. Die Begutachter (Referees) sind Wissenschaftler, die im gleichen oder einem ähnlichen Wissenschaftsgebiet arbeiten. Diese haben die Aufgabe, die

vorgelegte Papers nach ihrer Qualität zu beurteilen und auf ihre inhaltliche Richtigkeit zu überprüfen. Sie geben in ihrer Stellungnahme bekannt, ob ein Paper mit leichten Änderungen angenommen wird, noch einmal überarbeitet werden muß oder grundsätzlich abgelehnt wird [Zimmer1998]; z.B. beim Journal of Universal Computer Science (J.UCS) wird ein eingereichtes Paper von drei Referees begutachtet. Damit ein Paper angenommen wird, müssen mindestens zwei Begutachter der Veröffentlichung zustimmen. [JUCS]

Nach der Begutachtung bekommt der Autor das Paper wieder zurück, um die geforderten Änderungen vorzunehmen. Das korrigierte Paper sendet der Autor nun wieder an den Verlag. Dieser prüft die Änderungen, unter Umständen wieder unter Zuhilfenahme der Referees, paßt das Layout noch nach seinen Richtlinien an und gibt das Dokument zum Druck frei. Nach dem Druck geht das Journal an die Bibliotheken, die es abonniert haben. Dieser ganze Prozeß, vom Abschluß der wissenschaftlichen Arbeit bis zu dem Zeitpunkt, zu dem andere Wissenschaftler den Artikel lesen und auf dieses Wissen aufbauen können, dauert zwischen drei Monaten und drei Jahren [Zimmer1998].

Dieser Prozeß ist sehr aufwendig und langwierig. Die Dauer des Prozesses ließe sich durch elektronische Fachjournale (E-Journale) stark reduzieren. Doch warum kommen E-Journale nur langsam auf? In der Arbeit von [Zimmer1998] werden dafür zwei Gründe genannt:

Erstens stellt ein Artikel in einem Fachjournal nicht nur gespeichertes Wissen dar, das an andere Forscher weitergegeben wird, sondern ist gleichzeitig eine Urkunde, die dem Wissenschaftler seine erbrachten Leistungen zuschreibt. Ab dem Zeitpunkt, ab dem eine Arbeit in einer Zeitschrift abgedruckt wurde, ist festgehalten, wann ein Forscher ein bestimmtes Forschungsergebnis erzielt hat. Erst dann ist es „zitierbar" und kann als Grundlage für andere Wissenschaftler dienen. Welchen Arbeiten wirklich zitiert wurden, ist mit Hilfe des Citations Index, der in Abschnitt 2.4.2 besprochen wird, ersichtlich. In ihm ist vermerkt, wer einen Artikel zitiert hat und ihn somit als Grundlage für seine wissenschaftliche Arbeit verwendete.

Zweitens gibt es in jedem Fachgebiet eine ungeschriebene, aber jedem Forscher bewußte, Hierarchie der Zeitschriften. Damit ist es das Ziel eines Forschers möglichst in den meistbedeutendsten Fachjournalen seiner Fachrichtung zu publizieren, um einen internationalen Ruf zu erlangen und ihn zu erhalten. Die von den Begutachtern vorgenommene Auslese stellt einen notwendigen Filter dar, der je nach Prestige der Zeitschrift unterschiedlich streng ist. Erst durch diese Auslese kann sich der Leser auf die Qualität und Zuverlässigkeit des ihm vorliegenden Dokuments verlassen. Da, zum Zeitpunkt des Erstellens der vorliegenden Arbeit, die meistbedeutenden Zeitschriften in geduckter Form erscheinen, ist der Zustrom zu E-Journalen noch gering.

Die Schwerfälligkeit der gedruckten Journale verleiht ihnen einen gewissen Urkundencharakter. E-Journale habe hingegen den Vorteil schneller flexibler und kostengünstiger zu sein. Doch alles was im Internet gespeichert ist, kann verändert oder gelöschte werden. Es kann niemand sicher sein, daß er ein bestimmtes Dokument an einer bestimmen Stelle wiederfindet. Es kann der Host aufgelöst, die Seite verschoben oder gelöscht worden sein. Nie kann der Leser im Netz sicher sein, wer wirklich der Urheber eines Dokuments ist und ob es manipuliert oder gefälscht wurde. Ein elektronisches Dokument befindet sich ständig in einem Zustand der Diskussion. Irgendwann aber muß sich die Diskussion zu zitierbaren Ergebnissen verfestigen, auf die sich ein Forscher beziehen kann und die für Andere nachvollziehbar sind. Ein denkbarer Lösungsansatz für das Problem wäre die Speicherung der einzelnen Versionen eines Dokumentes. Dadurch würde sich die Änderungen und die Entwicklung eines Dokumentes nachvollziehen lassen. Einen anderen Lösungsansatz verfolgt z.B. der Springer Verlag, indem parallel zur gedruckten Auflage auch eine elektronische Online-Ausgabe des Journals herausgegeben wird. Damit wird auch die dauerhafte Archivierung der Dokumente sichergestellt. Der Verlag Elsevier[3] hat auf diese Art über 1000 wissenschaftliche Journale online zugänglich gemacht. Der Springer Verlag[4] plant bis zum Ende des Jahres 1999 sämtliche 400 Fachjournale des Hauses online zur Verfügung zu stellen. Verzeichnisse von Online-Journalen finden sich unter `http://www.edoc.com/ejournal/` oder `http://wuecon.wustl.edu/hyperjrn/director.htm`.

Weitere Probleme die mit E-Journalen verbunden sind, sind rechtlicher Natur. Dürfen E-Journale kopiert werden? Dürfen sie im Universitätsnetzwerk zugängig gemacht werden? Ob und an wieviele Personen dürfen Benutzername und Paßwort weitergegeben werden? Nach welchem Modus werden die Kosten abgerechnet: pro Zugriff oder pro Abonnement? All diese Fragen beantworten die Verlagshäuser mit unterschiedlichen Lösungsansätzen.

Zum Zeitpunkt des Verfassens der vorliegenden Arbeit gibt es über 3000[5] E-Journale, dem gegenüber stehen 900.000 Zeitschriftentiteln, die in der Deutschen Zeitschriftendatenbank erfaßt sind [ZDB1998]. Wie schnell die Anzahl der E-Journale weiter anwächst, ist stark damit verbunden, wie die rechtlichen Probleme und die Probleme mit der Beurkundung und der dauerhaften Archivierung gelöst werden.

[3] Verlag: http://www.elsevier.nl/
Die Online-Journale finden sich unter: http://www.sciencedirect.com/

[4] Verlag: http://www.springer.de/
Die Online-Journale finden sich unter: http://link.springer.de/

[5] Über 1755 Journale sind unter dem Online Computer Library Center (OCLC) einsehbar [OCLC1999], dazu kommen noch die Journale des Springer Verlags, von Elsevier und anderen Verlagshäusern.

Working Paper Archives

Einen anderen Ansatz bestreiten die Working Paper Archives (WPA). Sie bezeichnen sich selbst nicht als elektronische Journale. Diese Archive beruhen auf einer Idee des Hochenergiephysikers Paul Ginsparg[6] (1991). In ihnen kann ein Forscher seine Preprints — noch unbegutachtete Vorversionen von Veröffentlichungen — im Internet zugänglich machen, noch bevor sie ein halbes Jahr oder noch später in einem referierten Journal abgedruckt werden. Dadurch wurde die Kommunikation auf dem Gebiet der Physik stark beschleunigt [Stix1998a]. Basierend auf dieser Idee wurden auch weitere Working Paper Archives eingerichtet, z.B. das Economics Working Paper Archives an der Washington University in St. Louis[7]. Seit August 1998 gibt es das Computing Research Repository[8], ein WPA auf dem Gebiet der Informatik. Es bleibt jedoch die Frage offen, inwieweit diese Preprints als seriöse Quellen herangezogen und zitiert werden können. [Stix1998a]

2.2.3.2 Das Fachbuch

Während in einem wissenschaftlichen Journal die letzten Forschungsergebnisse stark komprimiert zusammengefaßt sind, sodaß sie oft ohne fundiertes Hintergrundwissen nicht verstanden werden können, umfassen Fachbücher ein größeres Teilgebiet eines Wissenschaftszweiges, das sie entsprechend der Zielgruppe inhaltlich aufbereiten. Auch hier sollen nur die typischen Schritte bis zur Veröffentlichung angeführt werden. Welche Schritte im Detail notwendig sind, ist von Verlag zu Verlag verschieden. [Ebel1998]

Am Anfang steht die Idee zu einem Buchthema. Der Autor arbeitet zunächst ein detailiertes Inhaltsverzeichnis aus und vermerkt bei jedem Kapitel, den geplanten Inhalt. Ein Beispielkapitel arbeitet der Autor vollständig aus. Dieses Konzept sendet er nun an mehrere Verlagshäuser. Die Idee zu einem Buch kann aber auch durch eine, schon bestehende Publikation wie z.B. eine Diplomarbeit, eine Dissertation oder ein Vorlesungsskriptum entstehen. In diesem Fall liegt ein fertiger Entwurf vor, den man an die Verlagshäuser schickt.

Der Verlag leitet das Manuskript weiter an mehrere Gutachter. Auf Grund der Stellungnahme der Gutachter, der „Blattlinie" und Überlegungen zur eigenen Produktpalette entscheidet der Verlag, ob ein Manuskript angenommen wird. Wurde das Manuskript angenommen, beginnt ein Kommunikationsprozeß zwischen dem Verlag und dem Autor, der je nach Verlag von unterschiedlicher Intensität ist. Der Autor beginnt die Kapitel auszuarbeiten und sendet die fertigen an den Verlag. Dieser überprüft sie auf inhaltliche und formale Richtigkeit

[6] http://xxx.lanl.gov
[7] http://econwpa.wustl.edu/
[8] http://www.acm.org/repository/

und sendet sie zur erneuten Korrektur an den Autor zurück. Welcher Aufwand in diesen Prozeß gesteckt wird, hängt unter anderem auch von der Intention ab, die mit diesem Buch verfolgt wird. Soll ein Buch z.B. für die nächsten 10 Jahre zum Standardwerk auf diesem Wissensgebiet werden, wird der Aufwand der in die Nachbearbeitung des Manuskripts gesteckt wird, entsprechend größer sein. Bevor ein Buch endgültig in den Druck geht, wird es nochmals von einem Lektor auf seine formale Richtigkeit überprüft. Dabei werden die verwendeten Zeichensätze, die Seitenumbrüche, das Inhaltsverzeichnis, usw. kontrolliert.

Elektronische Bücher sind im Internet heute kaum zu finden. Der Grund dafür ist in den hohen Kosten zu suchen, die der Publikationsprozeß verursacht und dem Problem, diese Kosten durch die Veröffentlichung im Internet zu decken. Ein weiterer Grund sind die Probleme die sich durch das Copyright ergeben. Die meisten Bücher die zur Zeit elektronisch abrufbar sind, wurden als Serviceleistung oder als Werbung für die gedruckte Ausgabe von den Verlagshäusern ins Internet gebracht.

Um die oben angeführten Probleme zu analysieren und Lösungsansätze zu finden, wurde von der Europäischen Kommission das LIBERATION[9] Projekt ins Leben gerufen. Ziel des Projekts ist es, einen Prototypen einer innovativen digitalen Bibliothek zu erstellen, die den Anforderungen der Verlagshäuser und der Benutzer entgegenkommt, den Prototypen zu analysieren und Schlüsse für künftige Arbeiten zu ziehen. Projektkoordinator ist das Institut für Informationsverarbeitung und Computerunterstützte neue Medien Graz (IICM[10]) [Stubenrauch1998].

In dem auf Hyperwave[11] basierenden Prototypen sind zum Zeitpunkt des Verfassens der vorliegenden Arbeit ca. 60 Arbeiten mit insgesamt ca. 200.000 Einzelseiten gespeichert. Trotz der direkten Beteiligung von drei großen Verlagshäusern (Springer, Addison-Wesley, BIFAB) war es schwer, umfassende qualitativ hochwertige Inhalte für das LIBERATION-Projekt zu erhalten. Ein Ergebnis des Projekts ist, daß die Zukunft des online zugänglichen elektronischen Fachbuchs davon abhängt, daß zusätzlich zu den bereits im Projekt implementierten Methoden, von Verlagshäusern und Bibliotheken neue Methoden ausgearbeitet werden, wie die Zugriffe auf die Werke verrechnet werden können. [Stubenrauch1998]

2.3 Wissensspeicher

Wie in Abschnitt 2.2.2 erklärt, stellen Wissensspeicher das zentrale Glied in der Verbindung zwischen dem Informationssender und dem Empfänger dar. Ihnen

[9] http://www.iicm.edu/liberation
[10] http://www.iicm.edu
[11] http://www.hyperwave.com

kommt die Aufgabe zu, jedem Benutzer, egal ob Mensch oder Maschine, das gewünschte Wissen zur Verfügung zu stellen. Dabei haben Menschen und Maschinen unterschiedliche Möglichkeiten und Anforderungen was die Suche und die Darstellung des geforderten Wissens betrifft.

2.3.1 Arten von Wissensspeichern

Eine Art Wissensspeicher zu unterscheiden ist nach der Art ihres Zugangs. Beim Internet oder bei Datenbanken z.B. erfolgt der Zugang elektronisch, bei Bibliotheken durch entlehnen eines Werkes, usw.

Eine andere, für die Wissensverarbeitung viel wichtigere Unterscheidung zwischen Wissensspeichern, ist die Art, in der das Wissen in den Dokumenten gespeichert ist. Da früher die Menschen die einzigen Systeme waren, die Wissen verarbeitet haben, sind bis heute die meisten Dokumente, in einer für den Menschen lesbaren Form, gespeichert. Durch das Aufkommen der Wissensverarbeitung wird es aber notwendig, die Anforderungen und Fähigkeiten des Computers bei der Wissensspeicherung zu berücksichtigen. Es genügt nicht alleine Dokumente in elektronischer Form abzuspeichern, sondern es müssen zusätzliche Informationen gespeichert werden, die das automatisierte verarbeiten und erschließen der Dokumente ermöglichen. Diese Tatsache wird bis jetzt im World Wide Web zu wenig berücksichtigt. Ora Lassila faßt dies in folgender Aussage zusammen:

> *„The Web was built for human consumption, and although everything on the Web is machine-readable, it ist not machine-understandable. This makes it very hard to automate anything on the Web and — because of the sheer volume of information — impossible to manage manually."* [Lassila1998]

Welche Möglichkeiten existieren nun, um Wissen in Dokumenten zu speichern? Die älteste und somit am weitesten verbreitete Möglichkeit ist die analoge Speicherung, wie z.B. auf Papier, Fotos, Tonbandaufnahmen oder Filmen. Diese Dokumente sind für Computer nicht zugänglich. Erst nach der Digitalisierung können sie in einem Computer gespeichert werden. Das Wissen, das in diesen Dokumenten steckt, ist jedoch für einen Computer nur schwer zugänglich, z.B. über eine Schrifertkennung. Erst dann liegt ein Dokument in seinen elementaren Symbolen[12] vor. Heute werden die meisten Dokumente direkt in dieser Form am Computer erstellt. Ein Beispiel dafür ist eine ASCII-Datei die mit einem Texteditor erstellt wurde. Sie liegen daher unmittelbar, ohne Nachbearbeitung, in elementaren Symbolen vor. Bei dieser Art der Speicherung sind alle Symbole gleichwertig.

[12] Elementare Symbole sind z.B. bei einem Text die Buchstaben, bei einer Grafik die geometrischen Grundelemente, wie der Kreis, das Rechteck, usw.

Dies stellt bei der Suche in Dokumenten ein großes Problem dar. So kann z.B. für ein Textdokument nur eine Volltextsuche realisiert werden. Man erhält durch eine Suche nach einem Begriff alle Dokumente, in denen dieser Begriff auftritt. Wie jeder aus persönlicher Erfahrung weiß, kann diese Treffermenge sehr groß sein. Zudem können die gefundene Dokumente aus verschiedenen Wissenschaftsgebieten stammen, in denen der gesuchte Begriff in einem unterschiedlichen Kontext vorkommen kann. [Lassila1998]

Um dieses Problem in den Griff zu bekommen, erweist es sich als sinnvoll, spezielle Attribute des Dokumentes getrennt zu speichern, z.B. den Autor, das Erstellungsdatum, das Thema, den Titel, usw. Die zusätzlich gespeicherten Informationen werden Metadaten genannt. Damit sind nicht mehr alle Symbole gleichwertig und eine Suche nach bestimmten Attributen wird möglich. Ein Beispiel für die Speicherung von Metadaten sind die Bibliothekskataloge, siehe Abschnitt 2.4.1. Durch sie wird es möglich nach bestimmten Attributen im Bibliotheksbestand zu suchen. Auch beim Internet geht der Trend dazu, zusätzlich in den Dokumenten, Metadaten abzuspeichern. Unter HTML kann dies mit Hilfe der Meta-Tags, bei Hyperwave mit Hilfe der Attribute geschehen. Um zu allen Webresourcen Metadaten speichern zu können, hat das W3C[13] das Resource Description Framework (RDF) ausgearbeitet. Es befindet sich zum Zeipunkt des Verfassens dieser Arbeit noch im Diskussionsstadium. Mit Hilfe des RDF ist es möglich zu jeder Webresource Metadaten zu speichern. Es ist sogar möglich Abhängigkeiten von Metadaten untereinander darzustellen [Lassila1997]. Dadurch wird es möglich eine Suche auf bestimmte Attribute der Dokumente einzuschränken.

Eine Anwendung für Metadaten stellt das Projekt der IEEE Learning Object Metadata (LOM) Workinggroup[14] dar. Ziel der LOM-Arbeitsgruppe ist es einen Standard zu schaffen, um elektronisch unterstützte Unterrichtsmaterialien und Software mit Metadaten zu beschreiben. Dabei wird Augenmerk darauf gelegt, mit einer minimalen Anzahl von Attributen die Unterrichtsmaterialien managebar, wiederauffindbar und evaluierbar zu machen. In LOM werden nicht nur die bibliographischen Daten wie Autor, Titel, Sprache usw. gespeichert, sondern auch pädagogische Attribute wie Art der Interaktion, die Lernmethode, das Leistungsniveau, das erforderliche Vorwissen, usw. Erst die ausführliche Beschreibung der Unterrichtsmaterialien mit Metadaten ermöglicht es dem Lehrer als auch dem Lernenden die passenden Unterrichtsmaterialien aufzufinden, zu beurteilen und somit effektiv einzusetzen. Darüber hinaus wird es möglich, mit der Hilfe von Software-Agenten, wie sie in Abschnitt 10 vorgestellt werden, automatisiert und dynamisch individuelle Lerneinheiten erstellen zu lassen. [LOM1998]

Um jedoch das gesamte in einem Dokument gespeicherte Wissen dem Computer zugänglich zu machen, muß man auf spezielle Wissensrepräsentationen der

[13] http://www.w3c.org

[14] http://ltsc.ieee.org/wg12/index.html

künstlichen Intelligenz zurückgreifen. Diese sind für den Menschen meist nur unter Zuhilfenahme von speziellen Hilfsprogrammen lesbar. Diese Repräsentationen werden in Abschnitt 3.2 vorgestellt.

Aufgrund der vorangegangenen Überlegungen kann man folgende Arten von Wissensspeichern unterscheiden:

- Human readable, nicht digital verfügbar: Alle Analogen Speicher, wie Papier, Fotos, Tonbandaufnahmen, Filme, usw.
- Human readable, digital verfügbar: Digitalisierte Fotos und Texte oder Audio-CDs.
- Human readable, aus elementaren Symbolen: Die Dokumente sind in ihren elementaren Symbolen digital gespeichert, z.B. ASCII-Text, MIDI oder Vektorgrafik.
- Human readable, mit Metadaten: Zusätzlich zum Text sind noch weitere Informationen, wie der Autor, das Erstellungsdatum, usw. gespeichert.
- Machine understandable: Wissensspeicher wie sie in der künstlichen Intelligenz verwendet werden.

Die beiden letzten Speicherarten haben in der Wissensverarbeitung die größte Bedeutung. So wird z.B. durch die Speicherung von Metadaten die effektive Suche mit intelligenten Software Agenten ermöglicht. Die Techniken die diese Software Agenten benutzen, sind in Kapitel 10 beschrieben. Die Maschinen verständliche Art der Wissensspeicherung stellt die Grundlage für die künstliche Intelligenz dar. Die dabei verwendeten Wissensrepräsentationen werden in Abschnitt 3.2 vorgestellt.

2.4 Wissenswiederauffindung

Wie beim Sender-Speicher-Empfänger Schema aus Abschnitt 2.2.2 angeführt, müssen Wissensspeicher Möglichkeiten zur Verfügung stellen, gezielt nach bestimmten Informationen zu suchen. Ausgehend von den historisch gewachsenen Techniken der Bibliothekslehre, werden in diesem Abschnitt die Techniken moderner Internet-Suchdienste beleuchtet.

Es gibt zwei meßbare Kriterien für die Qualität von Suchhilfen in Wissensspeichern, den Recall und die Precision. In Abbildung 2.5 ist die Bedeutung der beiden Begriffe veranschaulicht, die folgendermaßen definiert sind [Chowdhury1999]:

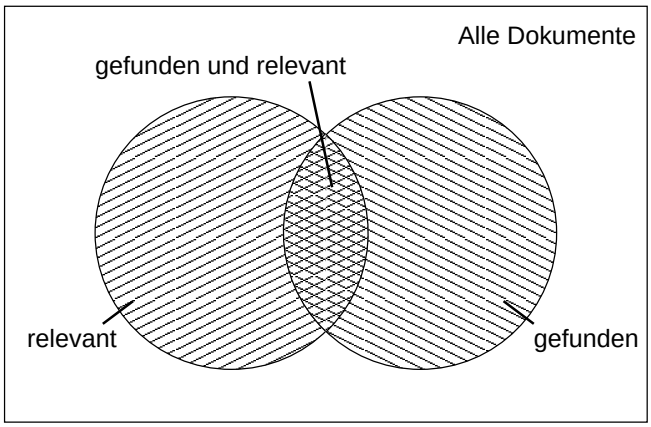

Abbildung 2.5: Es gibt zwei meßbare Kriterien für die Qualität von Suchhilfen in Wissensspeichern, den Recall und die Precision.

$$Recall = \frac{\text{gefundene und relevante Dokumente}}{\text{relevanten Dokumente}} \tag{2.4}$$

$$Precision = \frac{\text{gefundene und relevante Dokumente}}{\text{gefundene Dokumente}} \tag{2.5}$$

Dabei gibt der Recall an, wieviel Prozent aller relevanten Dokumente gefunden wurden. Die Precision gibt an, wieviel Prozent der gefundenen Dokumente relevant waren. Qualitativ hochwertige Suchhilfen zeichnen sich dadurch aus, daß sie für einen bestimmten Recall eine höhere Precision besitzen als schlechte. Dieser Zusammenhang zwischen Recall und Precision ist in Abbildung 2.6 dargestellt.

Abhängig von der gesuchten Information kommt dem Recall und der Precision ein unterschiedlicher Stellenwert zu. Dies soll anhand des nachfolgenden Beispiels erläutert werden. Will man wissen, wie hoch der Mount Everest ist, so reicht es aus ein Dokument zu finden, das die gewünschten Informationen enthält. Um dieses Informationsproblem zu lösen, ist eine hohe Precision gefordert. Der Recall dabei ist vernachlässigbar. Will man aber eine Arbeit über die Geschichte der Höhenmessung des Mount Everest schreiben, wird man an möglichst vielen Dokumenten interessiert sein und in Kauf nehmen müssen, daß einige Dokumente nicht die gewünschten Informationen enthalten. In diesem Fall sollte der Recall möglichst hoch sein, um sicher zu gehen, daß keine wesentlichen Dokumente übersehen werden. [Chowdhury1999]

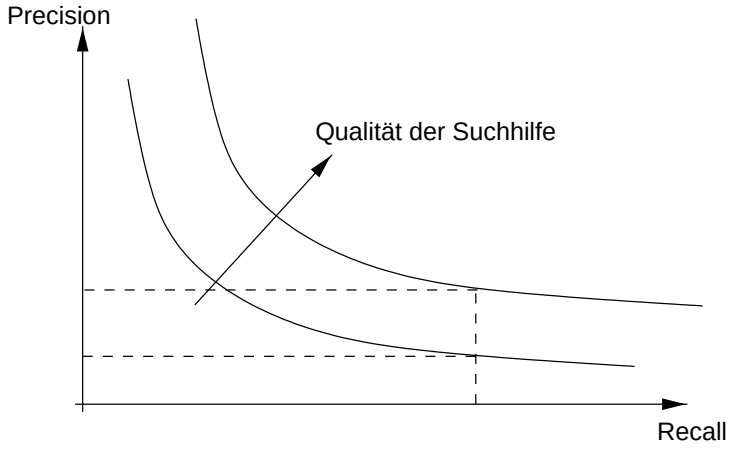

Abbildung 2.6: Qualitative hochwertige Suchhilfen haben bei gleichen Recall eine höhere Precision als schlechte.

2.4.1 Bibliothekskataloge

Bibliotheken sind wohl die Wissensspeicher mit der längsten Tradition. In dieser langen Entwicklungsgeschichte haben sich bestimmte Techniken herausentwickelt, die in weitergeführter Form auch bei Internet-Suchdiensten zu finden sind. Daher ist es sinnvoll, einige Begriffe der Bibliothekswissenschaft zu betrachten [Hacker1992] [Rauch1993]:

Stichwort: Dem Wortlaut eines Dokuments zum Zwecke der Kennzeichnung des Inhalts unverändert entnommene Benennung.

Schlagwort: Zum Zweck der Kennzeichnung des Inhaltes eines Dokuments in die dafür vorgesehene Informationskategorie aufgenommene Benennung. Ein Schlagwort muß nicht im Titel oder im Text des Dokuments vorkommen.

Freies Schlagwort: Schlagwort, das nicht einer vereinbarten und kontrollierten Liste von Schlagworten angehört.

Gebundenes Schlagwort: Schlagwort, das einer Liste vereinbarter und kontrollierter Schlagwörter entnommen worden ist.

Klassifizieren: Ein Klassifizierungs-System entsteht durch den mehrfach nacheinander durchgeführten Prozeß der Untergliederung. Dabei wird gleiches oder ähnliches zusammengefaßt. Um eine Klasse in weitere Unterklassen zu zerteilen, muß pro Teilungsschritt ein zusätzliches unterscheidendes Merkmal herangezogen werden. So enthält jedes Element einer untergeordneten Klasse alle Merkmale der Elemente der übergeordneten Klasse plus

ein weiteres Merkmal. Beim Klassifizieren ordnet man ein Dokument einer bestimmten Klasse mit Dokumenten gleichen Inhalts zu. Ein Beispiel für ein Klassifizierungssystem ist das ACM Computing Classification System[15], welches das Wissensgebiet Informatik in einer Klassenhierarchie ordnet.

Indexieren: Kennzeichnen des Inhalts eines Dokuments durch Zuordnung von Stich- oder Schlagwörter.

Um das Wissen, das in Bibliotheken gespeichert ist, möglichst einfach zugänglich zu machen, wird der Bestand nach unterschiedlichen Gesichtspunkten durch Kataloge erschlossen. Früher gab es eine Unterscheidung zwischen den einzelnen Katalogen, die durch die Einführung des online Katalogs (Online Public Access Catalog, OPAC) verschwunden ist [Hacker1992] [Rauch1993].

- Der Alphabetische Katalog verzeichnet die in der Bibliothek vorhandenen Bücher nach formalen Gesichtspunkten in alphabetischer Reihenfolge. Die formalen Elemente sind vor allem der Verfassername und der Sachtitel. Mit Hilfe dieses Kataloges kann die Frage beantwortet werden, ob die Bibliothek ein bestimmtes Buch, dessen wichtigste Daten bekannt sind, besitzt.
- Der Schlagwortkatalog ist ein Sachkatalog, der den Bibliotheksbestand unter Schlagwörtern verzeichnet, die aus dem Inhalt der Bücher gewonnen werden. Der Schlagwortkatalog eignet sich besonders zur raschen Orientierung über die Literatur zu einem bestimmten, begrenzten Thema.
- Beim Online Katalog sind die wichtigsten Daten zu jedem Werk in einer Datenbank gespeichert, die nach unterschiedlichen, auch verknüpften Kriterien durchsucht werden kann.

Bei der Katalogisierung müssen die Bibliotheken sich strickt an bestimmte Regeln halten, z.B. die alphabetische Behandlung von Umlauten und Sonderzeichen oder das ignorieren führender Artikel im Buchtitel beim Einordnen in den Katalog („Das verschwinden der Kindheit" steht unter „V").

Hinsichtlich der Qualität, des Niveaus oder der Zielgruppe eines Werkes werden in den Katalogen keine Angaben geführt. Es bleibt also dem Benutzer überlassen aufgrund seiner Erfahrung, dem Buchtitel und dem Verlag auf die persönliche Relevanz zu schließen. Im Zweifelsfall wird man sich das Werk ausleihen und selbst beurteilen müssen.

[15] http://www.acm.org/class/1998/

2.4.2 Citations Index

Der Citations Index hilft bei einem Problem, mit dem man bei jeder wissenschaftlichen Recherche zu tun hat. Sucht man zu einem Dokument weiterführende Literatur oder Wissenschaftler und Institutionen die sich auch mit diesem Thema befassen, so erhält man durch die Literaturhinweise nur Quellen die zeitlich vor dem aktuellem Dokument verfaßt wurden. Man hat auf diese Weise keine Möglichkeit weiterführende aktuelle Literatur zu diesem Thema zu finden [Rauch1993].

Ausgangspunkt für den Citations Index ist die Überlegung, daß wissenschaftliche Autoren in der Regel jene Arbeiten zitieren, deren Gedanken und Ergebnisse sie verwenden, weiterentwickeln oder falsifizieren. Im Citations Index findet man also zu einem Werk jene Dokumente, in denen das Werk zitiert wurde. Er stellt hiermit Beziehungen zwischen einer dem Leser bereits bekannten Veröffentlichung und neueren Arbeiten, in denen diese Veröffentlichung zitiert worden ist, her. So ist es möglich, ausgehend von einer bestimmten Veröffentlichung neuere Arbeiten ausfindig zu manchen, die thematisch verwandt sind.

Der Citation Index wird vom Institute for Scientific Information[16], Philadelphia herausgegeben und erscheint in drei Ausgaben, dem Science Citation Index, dem Social Science Citation Index und dem Arts & Humanities Citation Index. Der Aufbau aller drei Kataloge ist annähernd gleich und besteht aus folgenden Teilen [Rauch1993]:

Citation Index: Verzeichnet die zitierten Arbeiten und gibt an, wo und von wem sie zitiert wurden. Eine Altersbeschränkung für die aufgenommenen zitierten Arbeiten gibt es nicht.

Source Index: Enthält die vollständigen bibliographischen Angaben aller im Citation Index aufgenommenen Werke, alphabetisch nach Autoren oder Herausgebern geordnet.

Permuterm Index: Ein alphabetisch nach Stichwörtern geordneter Index. Die Stichwörter werden aus Paaren von jeweils zwei sinntragenden Wörtern des Titels oder des Untertitels gebildet.

Corporate Index: Verzeichnet die Veröffentlichungen nach der Institutsanschrift des erstgenannten Autors. Der Corporate Index ist eine geografischen Auflistung die nach Organisationen unterteilt ist. Mit seiner Hilfe ist es möglich, Veröffentlichungen aus einer bestimmten Universität ausfindig zu machen.

[16] http://www.isinet.com/

Relatet Records: Hilft bei der Suche nach verwandten Arbeiten, auch wenn diese sich nicht gegenseitig zitieren. Im diesem Index sind zu allen Arbeiten Verweise auf jene Arbeiten gespeichert, die mindestens zwei gleiche Quellen zitieren.

Der Science Citation Index (SCI) z.B. ist ein interdisziplinärer Index zur Zeitschriftenliteratur der Naturwissenschaften. Er deckt etwa 3.500 Zeitschriften aus 160 Fachgebieten ab. Im SCI wird jeder Artikel und signifikante Beitrag aus den vollständig abgedeckten Zeitschriften indiziert. Neben den bibliographischen Angaben enthält jeder Datensatz die Bibliographie des Dokuments. Dadurch wird die Suche nach zitierten Autoren, Patenten oder Artikeln ermöglicht. Zudem ist es erlaubt, zu einem Dokument andere Datensätze zu suchen, die auf gleiche Quellen wie das Ursprungsdokument verweisen. Ab dem Jahrgang 1994 wird das Dokument zusätzlich beschlagwortet und, wo im Original vorhanden, der englischsprachige Abstract[17] aufgenommen.

Der Citations Index wird auch dazu verwendet, um Wissenschaftler oder Institutionen zu bewerten. Maßgebend ist hier wie oft deren Publikationen, bzw. die Publikationen einer Institution zitiert wurden [Zimmer1998].

2.4.3 Suchdienste

Das Internet stellt den zur Zeit am schnellsten wachsenden Wissensspeicher dar. Die Dokumente, die auf Web-Servern gespeichert sind, definden sich in ständiger Veränderung. Sie werden veröffentlicht, modifiziert, verschoben und wieder gelöscht. Es existiert keine ,,zentrale Instanz" die diese laufenden Veränderungen registriert. Gerade diese Tatsachen machen es unmöglich, manuell einen Katalog, ähnlich dem Online-Katalog bei Bibliotheken zu erstellen. Da es manuell nicht möglich ist, das gespeicherte Wissen durch eine Suchhilfe zu erschließen, muß diese automatisiert mit der Hilfe von Computern erzeugt werden. Rechner die eine solche Suchhilfe erstellen und diese dem Internet-Benutzer zur Verfügung stellen, werden als Suchdienste bezeichnet [Sander1998] und lassen sich wie folgt einteilen [Neussl1998]:

- Singuläre Suchdienste
  - Indexsuchdienste mit vollautomatischer Auffindung
    - Volltext- und reduzierter Volltextindex

      AltaVista http://www.altavista.com/
      HotBot http://www.hotbot.com/
      Harvest http://harvest.austria.eu.net/

[17] Abstract: Kurzfassung der Arbeit

    * Schlüsselwörter und Metadaten

      Magellan http://www.mckinley.com/
      WWW-Worm http://www.cs.colorado.edu/wwww/

  – Katalogdienste

    Yahoo http://www.yahoo.com/
    Web.de http://www.web.de/
    Dino.online http://www.dino-online.de/
    Henkel http://www.henkel.co.at/henkel/ha_www_1.html

- Metasuchdienst und Kombinationen

  – Metasuchdienst durch Nutzung mehrerer Suchdienste

    MetaCrawler http://metacrawler.cs.washington.edu:8080/
    IBM infoMarket http://infomarket.ibm.com/
    Inference Find http://www.inference.com:8080/

  – Zusammenfassung mehrerer Katalogsuchdienste

    Metaindex European Web http://www.hj.se/hs/bibl/miewww/

  – Kombination von Index- und Katalogsuchdiensten

    Lycos http://www.lycos.com/

Singuläre Suchdienste

Bei singulären Suchdiensten kann man zwei Arten unterscheiden, die indexierenden und die klassifizierenden Suchdienste. Beide unterscheiden sich grundsätzlich in der Art, in der sie den Wissensspeicher Internet erschließen. Die indexierenden Suchmaschinen bedienen sich der Hilfe eines Web-Crawlers, einem Programm, das jeder dem System bekannten Internet-Adresse folgt, die Seite lädt, den darin enthaltenen Text volltextindexiert und nach weiteren Internet-Adressen durchsucht. Auf diese Weise erstellt die Suchmaschine einen Index, der von ihr untersuchten Web-Seiten. Manche Suchdienste berücksichtigen dabei auch Meta-Daten und Strukurinformationen, die in den HTML-Seiten enthalten sind. Bei einer Suchanfrage wird mit Hilfe des Index, eine Liste mit den relevanten Seiten generiert und an den Benutzer übermittelt. Beispiele für solche Suchmaschinen sind AltaVista, Hotbot, usw.

Katalogdienste bauen auf eine Klassenhierarchie auf. Jedes neue Dokument wird durch klassifizieren, wie in Abschnitt 2.4.1 beschrieben, einer Klasse von Dokumenten gleichen Inhalts zugeordnet. Die Suche in diesen Systemen erfolgt durch Navigation in der Klassenhierarchie. Ein Beispiel für eine solche Suchmaschine ist Yahoo.

Suchdienst	Abdeckung
Northern Light:	16%
AltaVista:	15.5%
Inktomi (Snap):	15.5%
Inktomi (HotBot):	11.3%
Inktomi (MSN Search):	8.5%
Infoseek:	8.0%
Google:	7.8%
Inktomi (Yahoo):	7.4%
Excite:	5.6%
Lycos:	2.5%
Euroseek:	2.2%

Tabelle 2.1: Suchdienste decken nur einen geringen Prozentsatz der Internetseiten ab. [SEW1999]

Meta-Suchdienste

Zum Zeitpunkt des Erstellens der vorliegenden Arbeit existieren über 1400 Suchdienste im Internet; eine Auflistung findet man beim Suchdienst Yahoo[18]. Diese Suchdienste haben die Aufgabe, den Inhalt von über 800 Millionen indizierbaren Internetseiten zu erschließen. Doch wie eine Studie von NEC-Forschungsinstitut[19] in Priston zeigt, erschließt jede Suchmaschine nur einen Bruchteil des vorhandenen Informationsangebots. Quantitative Angaben darüber, wieviel Prozent der Internetseiten ein Suchdienst erfaßt, ist aus Tabelle 2.1 ersichtlich. Will man somit einen umfassenden Überblick über ein Themengebiet im Internet erhalten, reicht es nicht aus, nur einen Suchdienst zu verwenden. [SEW1999]

An dieser Stelle haken die Meta-Suchdienste ein. Damit der Benutzer nicht mehrere Suchdienste von Hand durchsuchen muß, erledigt das ein Meta-Suchdienst für ihn. Der Benutzer gibt die Suchbegriffe in die Meta-Maschine ein. Diese gibt die Begriffe parallel an mehrere singuläre Suchdienste weiter. Aus den Ergebnissen werden doppelte Treffer eliminiert und die verbleibenden in einer einheitlichen Form an den Benutzer weitergegeben. Beispiele für solche Suchdienste sind MetaCrawler, Highway61, usw [Sander1998].

[18] `http://dir.yahoo.com/Computers_and_Internet/Internet/World_Wide_Web/Searching_the_Web/`

[19] `http://www.neci.nj.nec.com/`

Multimedia Suche

Die bisher vorgestellten Suchdienste können dem Benutzer beim Auffinden gesuchter Textstellen helfen. Beim Internet handelt es sich aber um ein multimediales Medium. In den Dokumenten können neben Text auch Bilder, Musik, Videos, 3D-Modelle, usw. enthalten sein. Sucht man z.B. nach einem Bild der Französischen Fahne, so wird man diese mit den obigen Suchdiensten unter dem Begriff „Trikolore" nur dann finden, wenn der Begriff im Filenamen der Grafik vorkommt, auf der Seite aufscheint, auf der das Bild abgebildet ist oder im Link-Text enthalten ist, der auf die Grafik verweist. Wünschenwert wäre also ein Suchdienst, bei der man die Suche nicht als Text eingibt, sondern skizziert, wie das gesuchte Bild aussehen soll [Stix1998b].

Einen Ansatz zur Lösung dieses Problems stellt der Suchdienst WebSEEK[20] dar. Es wurde an der New Yorker Columbia-University entwickelt. WebSEEK sendet einen Web-Crawler aus, der die Internetseiten nach Grafiken und Videos durchsucht. Aus deren Namen versucht es Informationen zum dargestellten Inhalt zu extrahieren. Anschließend überprüft WebSEEK welche Farben in einem Bild wo vorkommen. Daraus lassen sich Rückschlüsse auf die Art des Bildes wie Grafik, Foto, Schwarzweiß- oder Graubilder ziehen. Anhand dieser Informationen werden die Bilder in Klassen eingeordnet. Die Suche erfolgt demnach nicht nach einem Suchbegriff, sondern durch Navigation durch die Klassenhierarchie. Informationen aus Videosequenzen extrahiert das Programm aus Einzelbildern. Insgesamt hat WebSEEK mehr als 660 000 Internet-Bilder auf diese Weise indiziert [Stix1998b].

Eine Suche mit WebSEEK kann wie folgt aussehen: In diesem Beispiel soll nach einem Foto gesucht werden, auf dem eine schwarze Katze abgebildet ist. Zuerst wird man die Klasse „Animal", weiters die Klasse „Cat" auswählen. WebSEEK präsentiert dann eine Auswahl an Bildern mit Katzen. Da wir ein Bild einer schwarzen Katze suchen, wählen wir ein Ikon, auf dem eine solche abgebildet ist. Damit engen wir die Suche auf Bilder ein, die eine ähnliches Farbprofil haben. Somit werden fast nur mehr schwarze Katzen angezeigt. Es ist jedoch möglich, daß z.B. ein schwarzer Polster angezeigt wird. Der Benutzer kann die Suche weiter verfeinern, indem er im Suchmuster bestimmte Farben ausschließt oder hinzufügt [Stix1998b].

Ein leistungsfähigeres Programm zum durchsuchen von Bild-Datenbanken stellt das Programm Query by Image Content (QBIC) von IBM dar. QBIC liefert bessere Suchresultate als WebSEEK, da es nicht nur die Farbverteilung beurteilt, sondern beurteilt auch die Struktur nach mehreren Kriterien, wie Kontrast (etwa das Schwarzweiß von Zebrastreifen), Körnigkeit (Kieselsteine/Sand) und Ausrichtung (parallele Zaunlatten/rotationsymetrische Blütenblätter). QBIC kann sogar

[20] http://disney.ctr.columbia.edu/WebSEEK/

einfache Formen erkennen. So ist es möglich, nach einer grünen Fläche mit einem rosa Punkt zu suchen. Als Ergebnis erhält man rote Blüten auf grünem Hintergrund oder Bilder auf denen eine ähnliche Form dargestellt ist, siehe Abbildung 2.7 [Stix1998b]. Unter der Internet-Adresse `http://wwwqbic.almaden.ibm.com/` finden sich drei interessante Demo-Applikationen.

All diese Programme beruhen ausschließlich auf dem Vergleich visueller Merkmale. Sie benötigen immer noch einen menschlichen Begutachter, um entscheiden zu können, ob es sich z.B. bei dem Bild um eine Katze oder um einen Polster handelt.

Ein Programm das selbst entscheidet, ob es sich bei der Abbildung um eine nackte Person handelt oder nicht, wurde von Margaret M. Fleck von der Universität von Iowa in Iowa City und David A. Forsyth von der Universität von Kalifornien in Berkeley entwickelt [Fleck1996]. Ein bearbeitetes Bild wird zunächst von einem Skin-Filter untersucht. Bilder in denen keine hautfarbenen Elemente vorkommen, werden hier ausgeschieden. Im nächsten Schritt wird überprüft, ob die hautfarbenen Stellen der zylindrischen Form von Armen und Beinen entsprechen. Ob es sich dabei wirklich um eine Extremität handelt, kontrolliert der Algorithmus durch die Suche nach weiteren solchen Formen, die in bestimmten Winkeln zu den bereits identifizierten stehen müssen. Bei Tests hat dieses System von 4854 vorgelegten Testbildern 43 Prozent richtig als nackte Personen erkannt. Im Gegentest mit 4289 Abbildungen von bekleideten Personen wurden nur 4 Prozent fälschlicher Weise als nackt bewertet.

2.5 Informationsflut

Der Beginn des Kapitels beschäftigte sich damit, was Wissen ist, später wie es verbreitet und gespeichert werden kann. An dieser Stelle soll anhand der Bibliotheken und des Internets beleuchtet werden, wie schnell das „Wissen der Menschheit" wächst.

In Bibliotheken ist nicht nur das Wissen unserer Generation gespeichert, sondern auch das Wissen aller Generationen vor uns, sofern dieses aufgezeichnet und nicht durch Verfall, Krieg oder Zensur vernichtet wurde. Dieses gespeicherte Wissen wächst wie Kapital, mit Zins und Zinseszinsen exponentiell. So leben z.B. 90 Prozent aller Wissenschaftler, die jemals gelebt haben, heute. Die Aussage trifft nicht nur auf die heutige Situation zu, sondern galt auch schon vor einhundert oder zweihundert Jahren [Rauch1994]. Dieses rasante Wachstum wird auch augenscheinlich, wenn man die „Verdoppelungszeit" des Wissens betrachtet. Die Verdoppelungszeit des Wissens beträgt 10 Jahre, berücksichtigt man alle wissenschaftlichen Publikationen, 15 Jahre, bei einer engeren Auslegung und 20 Jahre,

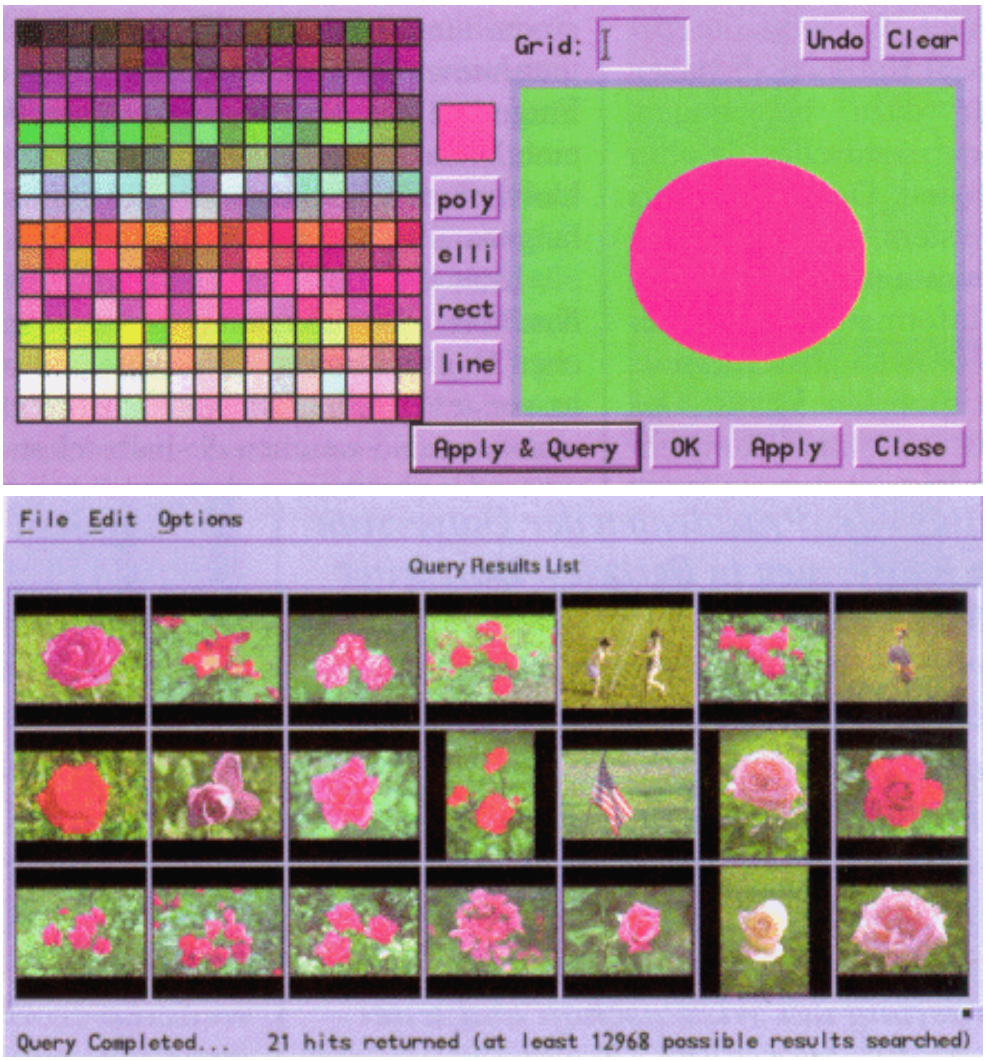

Abbildung 2.7: QBIC ermöglicht die Suche nach vorgegebenen Farbverteilungen in einer Bild-Datenbank. [Stix1998b]

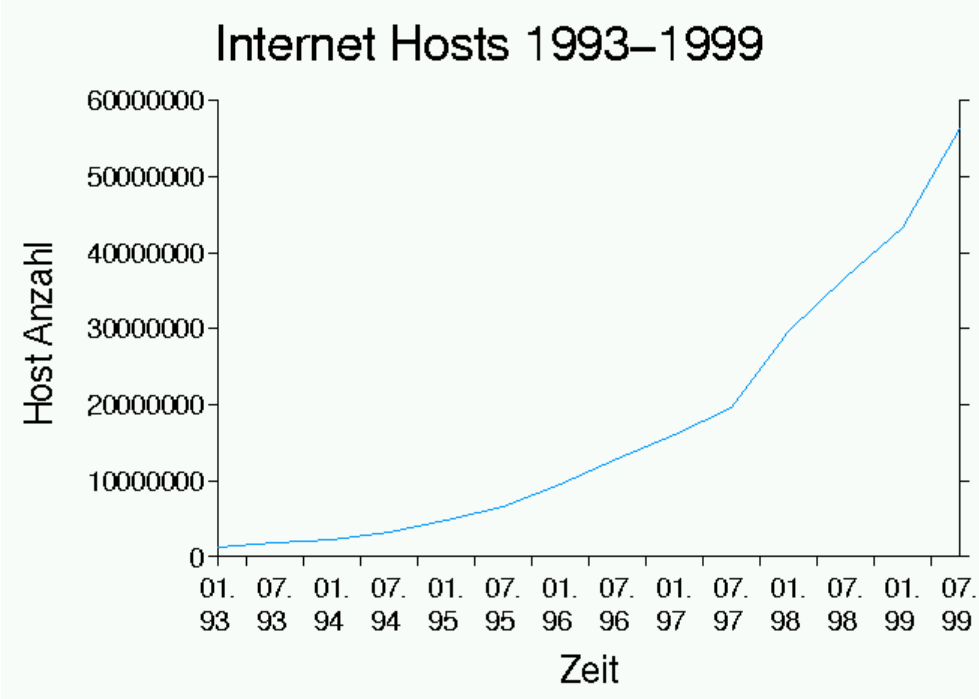

Abbildung 2.8: Wachstumskurver der Internet-Hosts weltweit. [ISC2000]

wenn man nur Publikationen höchster Qualität berücksichtigt [Rauch1994]. Das bedeutet die Verdoppelungszeit des Wissens liegt unter der halben Schaffenszeit einer Wissenschaftler-Generation [Tröger1998].

Auch das Internet als moderne Methode Wissen zu speichern und zu verbreiten, weist enorme Wachstumsraten auf. Dies ist jedoch nicht nur auf die wissenschaftliche Tätigkeit des Menschen zurückzuführen, sondern beruht auch darauf, daß die Wirtschaft das Internet als Werbeträger und Vertriebsmöglichkeit erkannt hat. Laut einer Untersuchung von Alexa Internet[21] wächst das Internet täglich um 1,5 Millionen Seiten [Alexa1998]. Dies spiegelt sich auch in der weltweiten Entwicklung der Anzahl der Hosts wieder. Abbildung 2.8 zeigt dieses exponentielle Wachstum.

Noch vor 300 Jahren, als z.B. Gottfried Wilhelm Leibniz als Philosoph, Bibliothekar und Universalgelehrter tätig war, basierte der wissenschaftliche Gedankenaustausch aus einem regen direkten Briefwechsel unter den Wissenschaftlern, an dem sich fast alle Gelehrten Europas beteiligten. Die geometrische Zunahme der Zahl der Wissenschaftler und Gelehrten und ihr Interesse an einem öffentlichen Gedankenaustausch führte Ende des 17. Jahrhunderts zur Gründung der ersten wissenschaftlichen Zeitschriften [Hauffe1997]. Die Anzahl dieser Zeitschriften ist

[21] http://www.alexa.com

bis heute auf über 900.000 Zeitschriftentiteln, die in der Deutschen Zeitschriftendatenbank erfaßt sind, angewachsen [ZDB1998]. Wenn man bedenkt, daß ein Wissenschaftler im Durchschnitt 100 Quellen durchforsten muß, bevor er eine für ihn relevante findet, so kommt man auf eine Zahl von 10 000 Quellen die ein Wissenschaftler in einem Jahr sichtet, weil sie ihm relevant erscheinen, um die 100 Arbeiten zu finden, die er eingehender studiert [Umstätter1998].

Die obigen Beispiele sollen aufzeigen, welche Probleme durch die steigende Informationsmenge auf die Menschen zukommen. Manuell ist diese gewaltigen Datenmenge nicht mehr zu bewältigen, daher wird zunehmend an Techniken gearbeitet, die es erlauben, den Computer für diese Aufgabe heranzuziehen. Um dies bewältigen zu können müssen sich Computer intelligent verhalten. Wie dieses intelligente Verhalten definiert ist, und mit Hilfe welcher Techniken diese „künstliche Intelligenz" am Computer umgesetzt wird, soll im nächsten Kapitel besprochen werden.

Kapitel 3

Einführung in die künstliche Intelligenz

Zu Beginn dieses Kapitels soll der Versuch unternommen werden, den Begriff künstliche Intelligenz (KI) zu definieren. Weiters werden die Anforderungen besprochen, welche die KI an einen Wissensspeicher stellt. Der Hauptteil dieses Kapitels stellt Speichermethoden vor, die diese Anforderungen erfüllen. Speziell wird dabei auf logikorientierte und prozedurale Methoden eingegangen, Frames, Semantischen Netze und Constraints werden vorgestellt. Am Schluß dieses Kapitels wird auf die Notwendigkeit eingegangen, unsicheres Wissen in Wissensspeichern der KI abzubilden und entsprechende Techniken der Implementierung vorgestellt.

3.1 Definition und Ziele der KI

Wie auch bei der Definition von Wissensverarbeitung gibt es auch für die künstliche Intelligenz keine einheitliche Definition in der Literatur. Um dennoch eine Vorstellung über die Thematik der KI zu erlangen, wird in diesem Abschnitt zunächst der Turing-Test als Einstieg in die Thematik vorgestellt. Anschließend wird augehend von Definitionen durch renommierte Buchautoren eine Deutung des Begriffes künstlicher Intelligenz hergeleitet. Weiters werden die Anwendungsgebiete angeführt, die erst durch die Verwendung von künstlicher Intelligenz für den Computer erschossen wurden.

Der Turing-Test

Eine der ersten Arbeiten, die sich mit maschineller Intelligenz beschäftigt, ist ,,Computing machinery and intelligence" von dem englischen Mathematiker Alan Turing, 1950. In dieser Arbeit beleuchtet Turing die Frage, ob Maschinen je in der Lage sein werden zu denken. Dabei geht er im Speziellen nicht darauf ein, worum es sich bei Intelligenz oder Denken handelt, sondern schlägt einen empirischen Test, den Turing-Test, vor. [Luger1997]

Der Test läuft nach folgendem Konzept ab. Ein menschlicher Richter steht in Verbindung mit einem Computer und einem Menschen. Damit der Test fair und unvoreingenommen gegenüber dem Computer ablaufen kann, werden beide Dialoge ausschließlich über Terminals geführt. Der Richter hat nun die Aufgabe herauszufinden, welcher der beiden Kommunikationspartner der Mensch und welcher der Computer ist. Über die Aufgabe des Richters sind sowohl der Computer als auch der Mensch informiert. Der Computer muß somit versuchen, sich wie ein Mensch zu verhalten. Am Ende des Tests muß der Richter auf Grund der geführten Dialoge entscheiden, welcher der beiden Partner der Computer war. Kann der Richter schlußendlich keine Entscheidung treffen, oder entscheidet er sich falsch, so kann die Maschine nach Turing als intelligent aufgefaßt werden. [Luger1997]

Defintionen von künstlicher Intelligenz

Ausgehend von der Auslegung von künstlicher Intelligenz von Alan Turing sollen an dieser Stelle Definitionen aus der Literatur vorgestellt und interpretiert werden.

> *,,Artificial Intelligence is the study of ideas which enable computers to do things, that make people seem intelligent. [...] The central goals of Artificial Intelligence are to make computers more useful and to understand the principles which make intelligence possible."* [Winston1993]

In dieser Definition geht es einerseits um das Studium der Natur von Wissen und der Mechanismen intelligenten Verhaltens und andererseits um die Erweiterung der Anwendbarkeit von Computern zur Vereinfachung bisheriger Vorgangsweisen sowie zum Einsatz für neue, anspruchsvollere Aufgaben.

> *,,Artificial Intelligence is the study of how to make computers do things at which, at the moment, people are better. [...]"* [Rich1986]

Nach dieser Definition ist künstliche Intelligenz keine Eigenschaft, sondern ein Forschungsgebiet. Ausgehend von den beiden obigen Definitionen und deren Deutungen kann man zwei Grundrichtungen in der KI Forschung unterscheiden [Puppe1991][Gottlob1990]:

1. Die Erforschung und möglichst exakte Simulation von natürlichem intelligenten Verhalten. Dazu ist es notwendig, das Verhalten von Menschen möglichst genau zu beobachten und zu verstehen. In dieser Richtung gibt es starke Überschneidungen mit der Psychologie und der Neurologie.

2. Die Entwicklung von Systemen, deren intelligentes Verhalten auf Methoden beruht, die unterschiedlich zu den von Menschen benutzten Methoden sind. Diese Vorgehenswiese kommt in der Ingenieurswissenschaft häufig vor, z.B. fliegen Flugzeuge nach den selben aerodynamischen Prinzipien wie Vögel, ohne jedoch mit den Flügeln zu schlagen.

Bei den Anwendungsgebieten der KI handelt es sich naturgemäß um besonders komplexe Themengebiete, die mit den Methoden der „herkömmlichen" Informatik nicht bewältigt werden konnten [Gottlob1990].

Game Playing: Brettspiele wie Dame oder Schach gehörten zu den ersten Anwendungen der künstlichen Intelligenz.[1]

Sprachverstehen: Sprachverstehende Systeme sollen es dem Benutzer ermöglichen, in seiner natürlichen Sprache mit dem Computer zu kommunizieren.

Wahrnehmung: Die Forschung auf dem Gebiet der Wahrnehmung beschäftigt sich mit dem Problem menschliche Sinne am Computer nachzubilden, dazu gehören sehen (Bilderkennung) und hören (Spracherkennung).

Theorembeweisen: Die Aufgabe der Theorembeweiser ist, die automatisierte Herleitung und Verifikation von mathematischen und logischen Formeln und Sätzen.

Robotik: In der Robotik geht es um die Entwicklung von Steuerungen für Roboter, dazu müssen diese z.B. in der Lage sein, ihre Umgebung wahrzunehmen.

Expertensysteme: In Expertensystemen ist das Spezialwissen und die Schlußfolgerungsfähigkeit qualifizierter Fachleute auf einem eng begrenzten Anwendungsgebiet im Computer nachgebildet. Die so entstandenen Systeme sollen Fachleute bei ihren Entscheidungen unterstützen. [Puppe1991]

[1] Am 11. Mai 1997 gewann der Computer Deep Blue von IBM gegen den amtierenden Schachweltmeister Garri Kasparow. Dies war das Erste mal in der Menschheitsgeschichte, daß eine Maschine einen menschlichen Schachmeister besiegte. [Neumann1997]

Automatisches Programmieren: Darunter versteht man das automatisierte Erstellen von Programmen, ausgehend von einer formalen Spezifikation.

Maschinelles Lernen: Die Forschung zu maschinellen Lernen beschäftigt sich mit der Entwicklung von Computersystemen, die in der Lage sind, durch die Benutzung von Eingabeinformationen neues Wissen zu konstruieren und vorhandenes Wissen zu verbessern. [Herrmann1997]

3.2 Wissensrepräsentation

Grundvoraussetzung für intelligentes Verhalten von Computer-Systemen ist, daß sie Wissen über ihre Umwelt besitzen. Dieses Wissen muß in einer für den Computer lesbaren Form gespeichert sein, wie dies bereits in Abschnitt 2.3.1 erläutert wurde. *„Intelligente Systeme verhalten sich als würden sie Wissen über ihre Umgebung besitzen und die von ihnen verursachten Veränderungen vorhersehen."*[2] Dabei ist zu beachten, daß das Computersystem selbst Teil der Umwelt ist und dadurch auch Wissen über sich besitzen muß, um bestimmte Aufgaben zu erfüllen.

Praktisch kann man eine Wissensrepräsentation als die Abbildung eines Ausschnitts der realen Welt bezeichnen. Diese Abbildung muß in einer Art und Weise geschehen, daß die darin enthaltenen Informationen automatisiert verarbeiten werden können. Das Wissen kann implizit im Programmcode, somit in der Abfolge der Befehle gespeichert sein, oder explizit an bestimmten Stellen des Systems. [Gottlob1990]

3.2.1 Implizites Wissen

Implizites Wissen steht im Code eines Programmes; z.B. bei der Berechnung einer bestimmten mathematischen Funktion, wie der Multiplikation zweier Matrizen. Der Algorithmus wird nicht abgearbeitet, weil das System eine Methode gesucht hat um zwei Matrizen zu multiplizieren, sondern weil der Programmierer wußte, daß in weiterer Folge das Produkt benötigt wird und hat deshalb den entsprechenden Algorithmus und den Aufruf der entsprechenden Funktion implementiert. Das Wissen, daß dieser Algorithmus an einer bestimmten Stelle aufgerufen

[2] *„Intelligent entities seem to anticipate their environments and the consequences of their actions. They act as if they know, in some sense, what the result would be." Nilsson-Genesereth (1987)*[Gottlob1990]

werden muß, ist also dirlekt im Programmcode enthalten. [Gottlob1990]

```
/* Anweisungen vor der Matrizenmultiplikation */
...
for i := 1 to n do     /* äußere Schleife */
  for j := 1 to m do   /* innere Schleife */
    ... /* Multiplikationsanweisungen */
  endfor;
endfor;
...
/* Anweisungen nach der Matrizenmultiplikation */
```

Auch ein Funktionsaufruf ändert nichts daran. Das Wissen ist in einem Unterprogramm versteckt, aber die Information, wann ein bestimmtes Unterprogramm aufgerufen werden soll, ist fix im Programm enthalten.

```
/* Anweisungen vor der Matrizenmultiplikation */
...
Produkt := Matrix_Mult(A, B);
...
/* Anweisungen nach der Matrizenmultiplikation */
```

3.2.2 Explizites Wissen

Wird Wissen explizit gespeichert, so sind dem System seine Problemlösungsmöglichkeiten für bestimmte Aufgaben bekannt, z.B. die verschiedenen Möglichkeiten die Inverse einer Matrix zu berechnen. Sie sind an einer bestimmten Stelle im System explizit vermerkt, und werden je nach Bedarf entsprechend angewandt. Es wird zwischen deklarativem und prozeduralem Wissen unterschieden [Gottlob1990]:

Deklarativ: Es wird beschrieben, in welchem Verhältnis die betrachteten Einheiten in der realen Welt zueinander stehen. In obigen Beispiel kann die ursprüngliche Matrix (M) und die invertierte Matrix (M^{-1}) in Beziehung mit der Einheitsmatrix (E) gesetzt werden:

$$M * M^{-1} = E$$

Eine getrennte, aktive Komponente muß dann dieses Wissen verwenden, um die Lösung zu berechnen. Der Vorteil liegt in der besseren Verständlichkeit und

Lesbarkeit diese Methode. Der Aufwand, um eine konkrete Lösung zu berechnen, ist aber entscheidend größer.

Prozedural: Es wird Schritt für Schritt angegeben, wie die Aufgabe gelöst werden muß. Der Unterschied zu implizitem Wissen ist jedoch, daß der Aufruf nicht implizit im Code vermerkt ist, sondern das System selbst erkennt, daß es zur Lösung des Problems dieses Wissen benötigt. Die explizite prozedurale Art Wissen zu speichern ist nicht so leicht lesbar, aber effizienter in der Abarbeitung.

3.2.3 Wissensbasierte Systeme

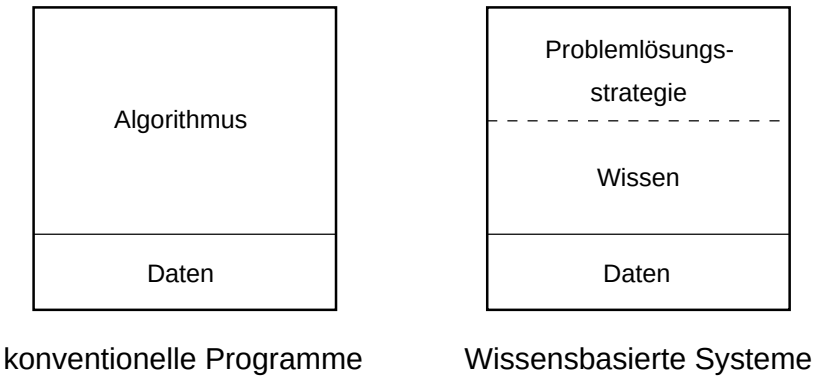

Abbildung 3.1: Grundsätzlicher Aufbau von konventionellen Programmen und Wissensbasierten Systemen.

Systeme, die Wissen explizit speichern, werden auch wissensbasierte Systeme genannt. Der Unterschied zu herkömmlichen, konventionellen Programmen ist in Abbildung 3.1 veranschaulicht. Konventionelle Programme, die Wissen implizit speichern, erschweren es, Wissen nachträglich zu verändern. Dazu wäre ein Eingriff in den Algorithmus notwendig. Bei wissensbasierten Systemen existiert eine klare Schnittstelle zwischen anwendungsspezifischen Wissen und der allgemeinen Problemlösungsstrategie. Die Komponente der allgemeinen Problemlösungsstrategie wird auch Inferenzmaschine genannt. Der Vorteil dieses Konzepts ist, daß man eine Problemlösungsstrategie für mehrere Anwendungsgebiete verwenden kann. Es muß nur die entsprechende Wissensbasis ausgetauscht werden.

Somit ist es Softwareherstellern möglich, „leere" Wissensbasierte Systeme anzubieten, die durch eine anwendungsspezifische Wissensbasis an die eigenen Anforderungen angepaßt werden können. Ein Beispiel für ein solches System ist die Expertensystemshell D3[3], welche aus einer grafischen Benutzeroberfläche und

[3] http://d3.informatik.uni-wuerzburg.de/

einer Problemlösungskomponente besteht. Durch Verwendung von unterschiedlichen Wissensbasen kann das System einfach an verschiedene Anwendungsgebiete angepaßt werden. Mit D3 wurden z.B. ein Expertensystem zur Pflanzenbestimmung, aber auch ein Service -System für Druckmaschinen entwickelt [Ernst1996].

3.2.4 Anforderungen an die Wissensrepräsentationen

In der künstlichen Intelligenz muß eine Wissensrepräsentation hohen Ansprüchen gerecht werden. Sie muß es ermöglichen, Wissen in möglichst einfacher, übersichtlicher Form schnell und effektiv zu verarbeiten. Dabei sind nicht nur Kriterien zu berücksichtigen, die für das automatisierte Verarbeiten von Wissen notwendig sind, es müssen auch die Interessen des Knowledge Engineers[4], das ist die Person, die die Wissensrepräsentation erstellt und wartet, beachtet werden. Daraus lassen sich fünf Anforderungen ableiten, die für eine Wissensrepräsentation wesentlich sind [Gottlob1990]:

Verarbeitbarkeit: Es muß dem Softwaresystem möglich sein, aus bestehendem Wissen durch logisches Verknüpfen, auf neues Wissen zu schließen. Die Wissensrepräsentation sollte das auf effiziente Weise unterstützen.

Expertensysteme versuchen z.B. aus bereits bekannten Fakten und aus Verknüpfungen von Fakten, den Regeln, auf neue Fakten zu schließen, die dann wieder als bekanntes Wissen betrachtet werden. Auf Expertensysteme wird später, im Abschnitt 6, genauer eingegangen.

Flexibilität: Ein bestimmter Repräsentationsansatz soll einerseits geeignet sein, Wissen aus möglichst vielen Anwendungsbereichen (Domains) wie z.B. der Medizin oder der Geologie, darzustellen. Anderseits sollen unabhängig vom Anwendungsgebiet unterschiedliche Aufgabenstellungen, wie z.B. Diagnostik, Konstruktion und bzw. oder Simulation, effizient verwirklicht werden können.

So ist es das Ziel der Diagnose, ein bekanntes Muster, wie z.B. ein Krankheitsbild oder einen Fehler, wieder zuerkennen. Dabei wird die Lösung aus einer vorgegebenen Menge von Alternativen ausgewählt. Im Gegensatz dazu wird bei der Konstruktion die Lösung aus kleinen Bausteinen zusammengesetzt. Es gibt jedoch zuviele mögliche Kombinationen, um aus einer Menge vorgegebener Alternativen auszuwählen. So kann z.B. bei der automatisierten Programmierung nicht aus der Menge aller möglichen Programme ausgewählt werden. Die Simulation unterscheidet sich von der Diagnose und der Konstruktion dadurch, daß kein vorgegebenes Ziel erreicht werden

[4] Auf die Aufgaben des Knowledge Engineers wird im Kapitel Expertensysteme, im Abschnitt 6 ausführlich eingegangen.

soll, sondern nur die Auswirkungen von Handlungen und Entscheidungen simuliert werden sollen.[Puppe1991]

Modularität: Die Wissensbasis eines intelligenten Systems soll möglichst einfach veränderbar und erweiterbar sein. Am einfachsten ist das zu erreichen, wenn die Wissensbasis modular aufgebaut ist. Das bedeutet, spezielle Bereiche des Wissens, wie etwa Wissen über die Lösung eines bestimmten Teilproblems, sollen möglichst unabhängig vom Rest der Wissensbasis hinzugefügt oder verändert werden können.

Da bei deklarativem Wissen Informationen über die Zusammenhänge zwischen einzelnen Wissenselementen, wann und wo sie eingesetzt werden, nicht Bestandteil des Wissens selbst ist, kommt die deklarative Art Wissen zu speichern der Forderung nach Modularität entgegen.

Eine der Modularität verwandte Forderung ist die Trennung zwischen dem gespeicherten Wissen (der Wissensbasis) und dem für die Verarbeitung und Schlußfolgerung zuständigen Teil des Systems (der Inferenzmaschine).

Verständlichkeit: Der Inhalt der Wissensbasis muß gegenüber dem Knowledge Engineer und dem Benutzer gleichermaßen leicht verständlich darstellbar sein. Das heißt allerdings nicht, daß die Darstellung für beide zwangsläufig in derselben Form erfolgen muß, da sie im Umgang mit der Wissensbasis unterschiedliche Ziele verfolgen; z.B. muß es für einen Benutzer nachvollziehbar sein, wie ein System zu einer bestimmten Entscheidung gekommen ist. Dem Knowledge Engineer dagegen ist es wichtig, einen Überblick über das gesamte gespeicherte Wissen zu besitzen, um Lücken oder Widersprüche in der Wissensbasis des Systems aufzudecken.

Darstellbarkeit unsicheren Wissens: In vielen Fällen kann die Wirklichkeit nicht durch Fakten beschrieben werden, die hundertprozentig korrekt sind, sondern nur mit Fakten, die mit einer bestimmten Wahrscheinlichkeit zutreffen. Intelligentes Verhalten besteht nun darin, aus einer Situation das Beste zu machen, in der das richtige Verhalten nicht eindeutig vorgeschrieben ist. Daher müssen intelligente Systeme in der Lage sein, mehrdeutige Aussagen zu verarbeiten. Diese Unsicherheiten können auf mehrere Arten entstehen [Gottlob1990]:

- Inhärente Unsicherheit der Information
- Unvollständigkeit der Information
- Unsicherheit von Schlußfolgerungen
- Zusammenfassung von Informationen, die aus mehreren, eventuell einander widersprechenden Quellen stammen

Bei vielen Anwendungen kann jedoch auf einer Repräsentation unsicheren Wissens verzichtet werden, sodaß dadurch die Wissensrepräsentation und die Verarbeitung vereinfacht wird. Auf die Quellen von Unsicherheit und deren Repräsentation in Wissensspeichern wird in Abschnitt 3.8 näher eingegangen.

3.3 Logikorientierte Methode der Wissensrepräsentation

Aufbauend auf den Theorien der formalen Logik stellt die Prädikatenlogik erster Stufe mit Identität[5] und Funktionssymbolen, kurz PIF, das bekannteste Kalkül zur Darstellung von logischen Sachverhalten dar. Ein Beispiel für eine Formel dieser Repräsentation ist [Gottlob1990]:

$$\forall X((person(X) \wedge \neg(X = eva) \wedge \neg(X = adam)) \Rightarrow \\ \Rightarrow \exists Y(person(Y) \wedge mutter\_von(X) = Y). \qquad (3.1)$$

Diese PIF-Formel sagt aus, daß jede Person ausgenommen Adam und Eva (mindestens) eine Mutter hat.

Eine Formel in der PIF-Logik setzt sich aus folgenden Grundzeichen zusammen: [Gottlob1990]

- Konstantensymbole, die hier mit Kleinbuchstaben beginnende Zeichenfolgen oder Ziffern dargestellt werden; z.B.: $adam, eva, 10$.
- Variablensymbole, hier durch mit Großbuchstaben beginnende Zeichenfolgen dargestellt; z.B.: X, Y.
- Funktionssymbole, zur Darstellung n-ärer Funktionen[6], hier durch Kleinbuchstaben beginnenden Zeichenfolgen dargestellt; z.B. das einstellige Funktionssymbol $mutter\_von$.
- Prädikatensymbole, zur Darstellung n-ärer Prädikate[7], hier durch mit Kleinbuchstaben beginnende Zeichenfolgen dargestellt; z.B.: $person$.
- Junktoren: $\neg, \wedge, \vee, \Rightarrow, \Leftrightarrow$
- Quantoren: $\forall, \exists$

[5] Gleichheit

[6] Funktionen mit n Konstanten oder Variablensymbolen

[7] Prädikate mit n Konstanten oder Variablensymbolen

- Das Gleichheitszeichen =
- Klammern und Kommas als Hilfssymbole; für die Darstellung von Argumenten von Funktionen und Prädikaten und zur Festlegung der Bindungsbereiche von Junktoren und Quantoren.

Die PIF ist zwar eine sehr ausdrucksstarke Logik, doch leider eignet sie sich nicht zur Implementierung. Erstens ist die PIF unentscheidbar, d.h. für eine beliebige Aussage kann nicht entschieden werden, ob sie aus den Fakten folgt oder nicht; z.B. kann aus der Formel $(raum(dunkel) \Rightarrow tageszeit(nacht) \vee jalousie(geschlossen))$ kein weiteres Faktum gewonnen werden, wenn das Faktum $raum(dunkel)$ bekannt ist. Zweitens nehmen Formelmanipulationen in der PIF sehr viel Rechenzeit in Anspruch. Aus diesem Grund werden meist nur stark reduzierte Teilsysteme der PIF in der Praxis verwendet. Diese Teilsysteme werden aber oft um Möglichkeiten erweitert, die in der ursprünglichen PIF nicht vorhanden waren, z.B. um die Möglichkeit eine Implikation mit einer Wahrscheinlichkeitsaussage zu versehen, wie in Kapitel 3.8 besprochen wird. [Gottlob1990]

3.3.1 Einfache Fakten Regel Systeme

Bei Einfachen Fakten Regel Systemen, kurz EFRS, handelt es sich um ein PIF Teilsystem, das sich besonders gut zur Implementierung eignet. Die folgende Definition ist entnommen aus [Gottlob1990].

„Ein EFRS S ist eine endliche Menge von Regeln und Fakten. Die Menge aller Regeln von S wird mit $R(S)$ bezeichnet und die Menge aller Fakten von S mit $F(S)$. Es gilt somit $S = R(S) \cup F(S)$." [Gottlob1990]

Fakten haben die Form $\psi(c_1, \ldots, c_n)$ mit einem Prädikatensymbol ψ und Konstantensymbolen $c_1, \ldots, c_n$. Beispiele für Fakten: *mensch(hans), vater (hans, maria), gerade(2), teiler(2,4).* Die Prädikatensymbole *mensch*, *gerade* sind einstellig; die Prädikatensymbole *vater*, *teiler* sind zweistellig.

Regeln sind Implikationen der folgenden Form:

$$\begin{aligned} Q(\psi_1(t_1^1, \ldots, t_{\alpha_1}^1) \quad &\wedge \quad \psi_2(t_1^2, \ldots, t_{\alpha_2}^2) \\ &\wedge \quad \ldots \psi_n(t_1^n, \ldots, t_{\alpha_n}^n)) \Rightarrow \psi_0(t_1^0, \ldots, t_{\alpha_0}^0) \end{aligned} \tag{3.2}$$

wobei die ψ_i Prädikatensymbole darstellen, die t_j^i entweder Variablensymbole oder Konstantensymbole sind und Q ein Quantorpräfix ist. Der Quantorpräfix enthält für genau jedes in der Formel vorkommende Variablensymbol X einen Allquantor der Form „$\forall X$". Zusätzlich muß jede Variable die rechts vom Implikationszeichen vorkommt auch auf der linken Seite vorkommen. Zur Vermeidung von überflüssigen Klammerzeichen bindet der Junktor „$\wedge$" stärker als „$\Rightarrow$".

Wenn ein Fakt F aus einem EFRS logisch folgt, dann schreiben wir $S \models F$. Die Menge aller Fakten die aus einem EFRS S logisch folgen, bezeichnen wir als $Cons(S)$. [Gottlob1990]

Ein Beispiel für ein Einfaches Fakten Regel System ist:

$$\begin{aligned} S_1 \;=\; & \{\forall X(mensch(X) \Rightarrow sterblich(X)), \\ & \forall X\,\forall Y(kind(X,Y) \wedge mensch(X) \Rightarrow mensch(Y)), \\ & mensch(hans), \\ & kind(hans, ernst) \\ & kind(ernst, olga)\}. \end{aligned} \tag{3.3}$$

Aus dem Faktum $mensch\,(hans)$ und der Regel $kind\,(hans, enst) \wedge mensch\,(hans)$ wird gefolgert, daß $mensch(ernst)$ ein neues Faktum ist. Der analoge Schluß führt zu $mensch(olga)$. Aus der Regel $\forall X(mensch(X) \Rightarrow sterblich(X))$ folgt somit, daß $hans$, $ernst$ und $olga$ sterblich sind, da alle Menschen sind. Alle Fakten, die aus dem EFRS S_1 logisch folgern, erhält man mit $Cons(S_1)$:

$$\begin{aligned} Cons(S_1) \;=\; & \{mensch(hans), kind(hans, ernst), kind(ernst, olga), \\ & mensch(ernst), mensch(olga), sterblich(hans), \\ & sterblich(ernst), sterblich(olga)\}. \end{aligned} \tag{3.4}$$

In einem weiteren Beispiel soll die Mitarbeiterstruktur eines Betriebes mit Hilfe eines EFRS beschrieben werden. Abbildung 3.2 zeigt die Mitarbeiterhierarchie im Betrieb. [Gottlob1990]

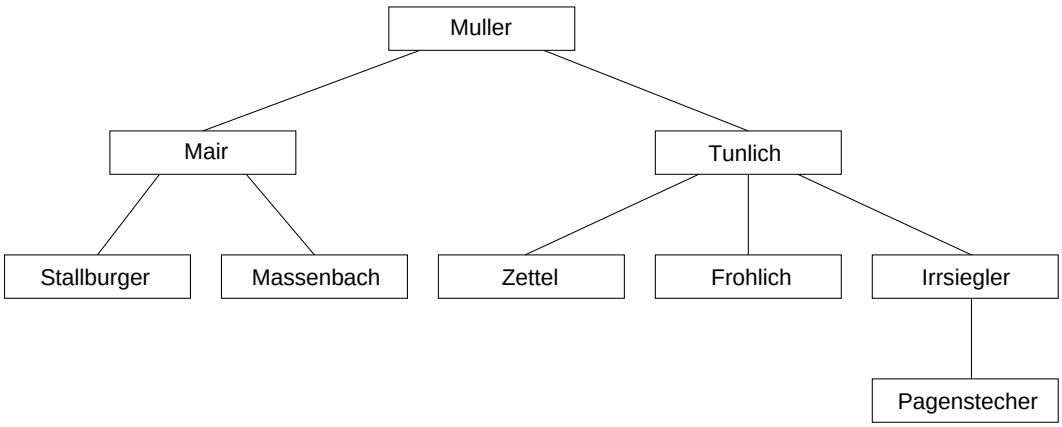

Abbildung 3.2: Die Mitarbeiterhirarchie soll durch ein einfaches Fakten Regelsystem repräsentiert werden.

$angestellter(X)$:	X ist ein Angestellter der Firma
$vorgesetzter(X,Y)$:	X ist der unmittelbare Vorgesetzte von Y
$gleiche\_ebene(X,Y)$:	X und Y sind Kollegen auf der gleichen Ebene in der Mitarbeiterhierarchie

Bei der Umsetzung der in der Abbildung beschriebenen Struktur soll das Prädikat *gleiche_ebene* nicht durch eine Faktenmenge, sondern durch Regeln definiert werden.

$$
\begin{aligned}
S_2 \;=\; \{&angestellter(muller),\\
&angestellter(maier),\\
&angestellter(stallburger),\\
&angestellter(massenbach),\\
&angestellter(tunlich),\\
&angestellter(zettel),\\
&angestellter(frohlich),\\
&angestellter(irrsiegler),\\
&angestellter(pagenstecher),\\
\\
&vorgesetzter(muller, maier),\\
&vorgesetzter(muller, tunlich),\\
&vorgesetzter(maier, stallberger),\\
&vorgesetzter(maier, massenbach),\\
&vorgesetzter(tunlich, zettel),\\
&vorgesetzter(tunlich, frohlich),\\
&vorgesetzter(tunlich, irrsigler),\\
&vorgesetzter(irrsiegler, pagenstecher),\\
\\
&\forall X(angestellter(X) \Rightarrow gleiche\_ebene(X,X)),\\
&\forall X\,\forall Y\,\forall X_1\,\forall Y_1(gleiche\_ebene(X,Y) \wedge vorgesetzter(X,X_1)\\
&\qquad\qquad \wedge vorgesetzter(Y,Y_1) \Rightarrow gleiche\_ebene(X_1,Y_1))\}
\end{aligned}
\tag{3.5}
$$

Für $Cons(S_2)$ erhält man nun alle Fakten, die in S_2 vorkommen und zusätzlich Fakten der Form $gleiche\_ebene(\alpha,\beta)$, wobei α und β Angestellte der gleichen Hierarchieebene sind, z.B. $gleiche\_ebene(stallburger, zettel)$ oder *gleiche_ebene (zettel, irrsiegler).*

Um zum Faktum $gleiche\_ebene(stallburger, zettel)$ zu gelangen, beginnt man bei der Schlußfolgerung der letzten Regel: $gleiche\_ebene(X_1,Y_1)$. Somit lautet die Voraussetzung, daß das Faktum aus den bereits bekannten Fakten folgt:

gleiche_ebene(X,Y) $\wedge$ *vorgesetzter(X, stallburger)* $\wedge$ *vorgesetzter(Y, zettel).* In den bekannten Fakten wird nun nach einem X gesucht, welches das Prädikat $vorgesetzter(X, stallburger)$ erfüllt. Das ist für $X = mair$ der Fall. Äquivalent wird mit Y verfahren, sodaß $Y = tunlich$ folgt. Die einzige offene Voraussetzung ist nun $gleiche\_ebene(mair, tunlich)$. Um diese Forderung zu prüfen wird rekursiv wieder die letzte Regel betrachtet. Nach der Abarbeitung kommt man zu der Vorderung $gleiche\_ebene(muller, muller)$, welche durch die vorletzte Regel $angestellter(muller) \Rightarrow gleiche\_ebene(muller, muller)$ erfüllt ist. Somit kann man schreiben: $S_2 \models gleiche\_ebene(stallburger, zettel)$.

Die Regeln in einem EFRS entsprechen Hornklauseln, das sind Klauseln bei denen nur auf ein Faktum geschlossen wird, d.h. es steht nur ein Prädikatensymbol nach dem Implikationszeichen. Somit stellt z.B. die PIF-Formel (3.1) keine Hornklausel dar. Sie besitzt nach dem Implikationszeichen die zwei Prädikate $person$ und $mutter\_von$. Hornklauseln werden auch in logischen Programmiersprachen z.B. PROLOG verwendet. [Böhringer1988]

3.3.2 Logische Programmiersprachen

Klassische Programmiersprachen sind hauptsächlich auf zwei Schwerpunkte ausgerichtet, um wissenschaftlich-technischen Aufgaben zu lösen und um Daten und Textbestände zu manipulieren. Mit ihrer Hilfe ist es schwer komplexe Wissensbasierte Systeme zu entwickeln. Aus diesem Grund werden solche Systeme vorwiegend in logischen Programmiersprachen, wie Lisp oder Prolog, implementiert. [Gottlob1990]

LISP

LISP wurde in den 50er Jahren von John McCarthy entwickelt und ist die zweitälteste noch in Verwendung stehende Programmiersprache. Nur Fortran ist um ein Jahr älter. LISP steht für *List Programming.* Zielsetzung bei der Entwicklung der Sprache war es, ein System zur Symbolmanipulation zur Verfügung zu haben. Symbolmanipulation wurde als Schlüssel für die Entwicklung intelligenter Programme aufgefaßt.

LISP ist eine funktionale Programmiersprache und verwendet Listen als Datenstruktur. Die Verarbeitung eines Programmes erfolgt durch Verarbeitung von Listen. Für LISP sind Daten und Programme äquivalent, d.h. sie werden in der gleichen Datenstruktur, in Listen, kodiert. Dieser Ansatz verhalf LISP zu einer dominierenden Stellung überall dort, wo anspruchsvolle Symbolverarbeitungsaufgaben gelöst werden mußten, für welche die Sprachmittel bisher bekannter Sprachen nicht ausreichten.

LISP selbst stellt jedoch direkt keine Methoden zur Verfügung, um Fakten logisch zu verknüpfen und so auf neue Fakten zu schließen. Die für ein Wissensbasiertes System notwendigen Komponenten, die Wissensbasis und die Inferenzmaschine, sind somit erst aus LISP-Konstrukten zu erstellen. [Schnupp1986]

Prolog

Prolog ist eine deklarative Programmiersprache und steht für „Programmieren in Logik". Das bedeutet, daß in Prolog, im Gegensatz zu den herkömmlichen prozeduralen Programmiersprachen wie Fortran, Pascal oder C nicht mehr algorithmisch der Lösungsweg angegeben wird, sondern nur mehr die Bedingungen, die die Lösung des Problems erfüllen sollen. Funktionale Sprachen wie LISP sind ebenfalls deklarativ. Während aber funktionale Programmiersprachen auf Funktionen und Funktionsapplikationen beruhen, basieren logische Programmiersprachen auf Relationen und Regelanwendung. [Gottlob1990]

Prolog ist eine deskriptive Programmiersprache, d.h. die Lösung eines Problems wird durch Regeln und Fakten beschrieben. Prozedurale Programmiersprachen sind hingegen imperativ. Die Lösung eines Problems muß in Form von Befehlsfolgen ausprogrammiert werden. Die in der nachfolgenden Gleichung formulierte Aussage

$$Algorithmus = Logik + Kontrolle$$

betont den Unterschied zwischen dem Was (der Logik) und dem Wie (der Kontrolle) in der Programmierung, d.h. in der Implementierung eines Algorithmus. Ein Algorithmus besteht also aus einer logischen Komponente, die das Wissen zur Problemlösung enthält und einer Steuerungskomponente, die bestimmt, wie das Wissen verwendet wird, um damit Probleme zu lösen. Mit dieser Grundidee stellt Prolog eine Sprache dar, in der sich Wissensbasierte Systeme, wie sie in Abschnitt 3.2.3 beschrieben wurden, einfach entwickeln lassen. [Gottlob1990]

3.4 Prozedurale Methode als Wissensrepräsentation

Beim prozeduralem Ansatz wird Wissen in Form einer Abfolge von Anweisungen, den Prozeduren, gespeichert. Im Unterschied zu nicht wissensbasierten Programmen muß der Aufruf einer Prozedur aber explizit der Kontrolle des Programmes unterliegen. Das bedeutet, das Programm weiß über seine Problemlösungsmöglichkeiten bescheid und wendet diese entsprechende der vorliegenden Problemstellungen an. [Gottlob1990]

Beispiel: Es soll der Sachverhalt dargestellt werden, daß alle Personen sterblich sind, daß alle Hunde sterblich sind, daß alle Katzen sterblich sind, und daß Helena und Sokrates Personen sind.

Die logische Repräsentation sieht folgendermaßen aus:

$$\forall X(person(X) \Rightarrow sterblich(X)).$$
$$\forall X(hund(X) \Rightarrow sterblich(X)).$$
$$\forall X(katze(X) \Rightarrow sterblich(X)).$$
$$person(sokrates).$$
$$person(helena).$$

Eine prozedurale Repräsentation könnte so aussehen:

```
boolean function person(x)
  if x=sokrates then return true
  else
    if x=helena then return true
    else return false

boolean function sterblich(x)
  if person(x) then true
  else
    if hund(x) then true
    else
      if katze(x) then true
      else return false
```

Versucht man mit dem Aufruf $sterblich(Sokrates)$ zu überprüfen ob Sokrates sterblich ist, so wird zuerst überprüft ob Sokrates eine Person ist. Wäre Sokrates keine Person, würde überprüft ob Sokrates ein Hund ist. In dieser Kette wird solange fortgeschritten, bis auf die Spezies getroffen wird, der Sokrates angehört oder keine weiteren Abfragen mehr folgen. Geht man davon aus, daß die meisten Objekte Personen sind, die auf Sterblichkeit überprüft werden, ist diese Vorgehenswiese effizient. Schon beim ersten Schritt wird überprüft, ob das Objekt der Spezies Person angehört. Somit ist es bei prozeduraler Wissensspeicherung möglich, durch geschickte Wahl der Abfolge der Einzelschritte, die Abarbeitung für häufig auftretende Vorgänge effizient zu gestalten. Da jedoch alle Lösungsmöglichkeiten explizit berücksichtigt werden müssen, ist es schwer, komplexe Wissensbasen zu erstellen.

Ein weiterer Nachteil ist die geringe Flexibilität. So kann das System die Frage „Sind alle Personen sterblich?" nicht entscheiden. Eine solche Entscheidung

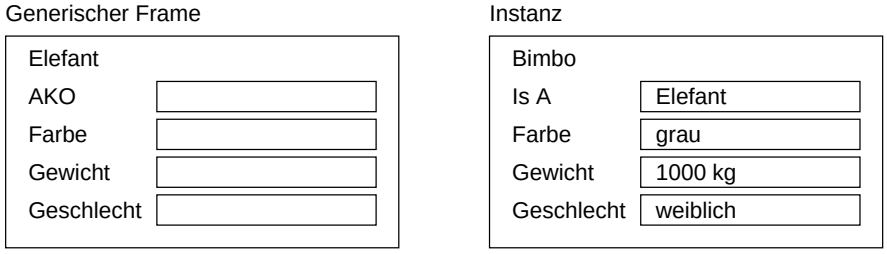

Abbildung 3.3: Der Generische Frame des Objekttyps Elefent und eine Beispielinstanz Bimbo. Im ,,Is A" Slot der Instanz ist gespeichert, daß es sich bei Bimbo um einen Elefanten handelt.

wäre bei einer logikbasierten Repräsentation mit geeignetem Inferenzverfahren möglich. Zudem macht eine Erweiterung der bestehenden Wissensbasis um den Satz

$$\forall X(tier(X) \Rightarrow sterblich(X)).$$

einen Eingriff in eine bereits bestehende Prozedur nötig. Dies schränkt die Modularität sehr stark ein. [Gottlob1990]

3.5 Frames

Die Wissensrepräsentation mit Hilfe von Frames stellt eine objektorientierte Wissensrepräsentation dar. Dabei werden Objekte der realen Welt durch Frames dargestellt. Die Eigenschaften des Objekts werden in den Frames in sogenannten Slots gespeichert. Der Tatsache, daß es in der realen Welt mehrere unterschiedliche Objekte eines Objekttyps gibt, wird mit Hilfe von generischen Frames und deren Instanzen Rechnung getragen. Ein generischer Frame hält für jedes Attribut mit dem ein Objekt beschrieben wird, einen Slot bereit. In einer Instanz des generischen Frames wird nun jedem Slot, entsprechend für das Attribut für das er steht, ein Wert zugeordnet. Abbildung 3.3 zeigt den generischen Frame des Objekttyps Elefant und eine Instanz, die den speziellen Elefanten Bimbo beschreibt. Die Bedeutung des Slots AKO wird später beschrieben, wenn auf die Technik der Vererbung eingegangen wird. [Winston1993]

Die Beziehung zwischen einem generischen Frame und einer Instanz wird mit Hilfe des ,,Is A"-Slot hergestellt. Im Beispiel ist im ,,Is A"-Slot gespeichert, daß es sich bei Bimbo um einen Elefanten handelt. In den übrigen Slots sind die Werte zu den Attributen gespeichert.

In einem generischen Frame wird zu einem Slot nicht nur das Attribut gespeichert, für das der Slot steht, sondern auch zusätzliche Informationen und

Programme. Dazu gehören: [Puppe1991]

- Defaultwerte, die beim Lesezugriff zurückgegeben werden, wenn kein Wert explizit zugeordnet wurde. Im Beispiel mit dem Elefanten wäre das für den Slot Farbe der Wert grau.
- Bedingungen, die der Wert erfüllen muß, z.B. männlich oder weiblich im Slot Geschlecht.
- Prozeduren die beim Schreib-, Lese- oder Löschzugriff auf den Slotwert ausgeführt werden. Dadurch wird erreicht, daß Wissen nicht nur passiv gespeichert wird. Es wird möglich bei jedem Zugriff auf Slots Schlüsse zu ziehen und andere Frames zu modifizieren. Somit ist es möglich, die Wissensbasis auf einfache Art konsistent zu halten.

Vererbung

Die Stärke des Framekonzepts ist die Vererbung. Ein generischer Frame kann Slots an einen anderen generischen Frame vererben. Diese Beziehung wird mit Hilfe des AKO-Slots dargestellt. AKO steht für ,,A Kind Of". Im erbenden Frame ist im AKO-Slot vermerkt, von welchem Frame er Slots erbt. Abbildung 3.4 zeigt eine Framestruktur, in der Informationen zu einem Ereignis gespeichert werden. Es gibt zwei mögliche Ereignisse, Unglücke und Feiern. Die Ereignisse teilen sich wieder in Erdbeben und Überschwemmungen, bzw. in Hochzeiten und Geburtstage. Jedes Ereignis findet an einem bestimmten Ort zu einer bestimmten Zeit statt. Darum werden diese Slots weitervererbt an Unglücke und Feiern. Weiters hat jede Feier einen Gastgeber und Gäste. Diese und die Slots die vom Ereignis geerbt wurden, werden weiter an die verschiedenen Feiern vererbt. Eine Instanz einer Geburtstagsfeier könnte also wie in Abbildung 3.5 aussehen. [Winston1993]

3.6 Semantische Netze

Semantische Netze sind nicht als eigenständige Wissensrepräsentation aufzufassen, sie dienen vielmehr dazu, die Zusammenhänge in den bisher besprochenen Repräsentationen übersichtlich darzustellen [Puppe1991]. So wurde z.B. in Abbildung 3.2 die Aufgabenstellung der Implementierung einer Mitarbeiterhierarchie graphisch als semantisches Netz dargestellt, bevor die logische Repräsentation erarbeitet wurde, oder in Abbildung 3.4 wurde die Framehierarchie zur besseren Verständlichkeit als semantisches Netz dargestellt, ohne dies explizit zu erwähnen.

Semantische Netze stellen gerichtete Graphen dar, die aus Nodes, Links und Link Labels bestehen. Diese Begriffe sind folgendermaßen definiert [Winston1993]:

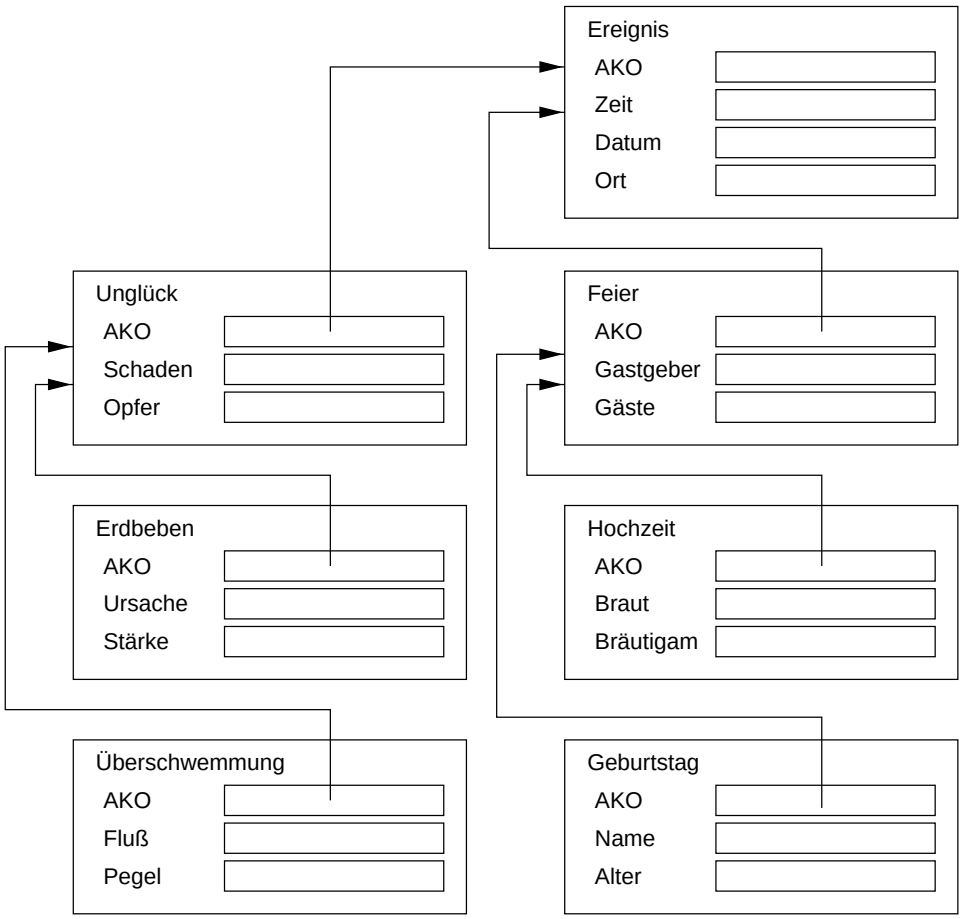

Abbildung 3.4: Beispiel für eine Frame Struktur für zwei unterschiedliche Arten von Ereignissen. Die Vererbung erfolgt über zwei Ebenen: Unglück und Feier erben vom Frame Ereignis; Erdbeben und Überschwemmung erben vom Frame Unglück und somit auch vom Frame Ereignis.

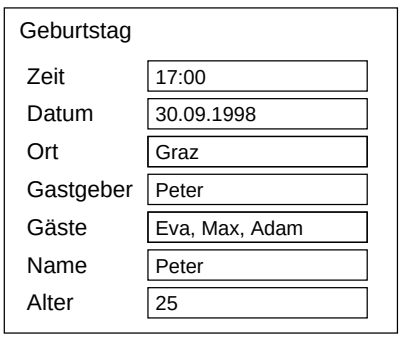

Abbildung 3.5: Beispiel für eine Instanz Geburtstag, die duch Vererbung erzeugt wurde.

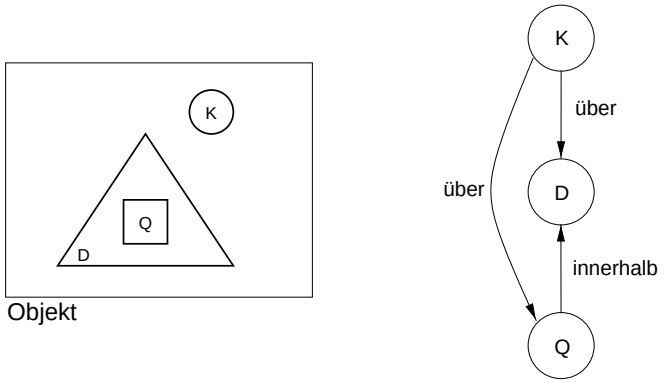

Abbildung 3.6: Ein Beispiel für eine Repräsentation durch ein semantisches Netz; Mit Hilfe des semantischen Netzes wird die räumliche Beziehung der geometrischen Objekte untereinander ausgedrückt.

Node, beschreibt ein Objekt der realen Welt.

Link, stellt die Beziehung zwischen zwei Nodes dar.

Link Labels, beschreibt die Art der Beziehung, die mit einem Link ausgedrückt wird.

Abbildung 3.6 stellt ein Beispiel für ein semantisches Netz dar. Es drückt die räumlichen Beziehungen der geometrischen Objekte untereinander aus.

Anhand semantischer Netze werden in Kapitel 4 und 5 Algorithmen erklärt, die in der Wissensverarbeitung häufig angewandt werden. Dazu gehören vor allem Algorithmen zur Suche in Graphen und zur Behandlung von Spielbäumen.

3.7 Constraints

Mit der Hilfe von Constraints können Relationen, d.h. Beziehungen zwischen Variablen, dargestellt werden. Dabei eignen sich Constraints besonders zur Darstellung von Randbedingungen, die die Lösung des Problems auf jeden Fall erfüllen muß. Ein Beispiel dafür ist der Wunsch eines Lehrers bei der Stundenplanerstellung, daß er einen bestimmten Tag in der Woche frei haben möchte. Durch jedes neue Constraint wird der Lösungsraum zusätzlich eingeschränkt [Puppe1991].

Während Constraints ungerichtete Zusammenhänge zwischen Variablen ausdrücken, die nach jeder Variable hin aufgelöst werden können, z.B. $U = I * R$, repräsentieren Regeln gerichtete Zusammenhänge, z.B. $A \Rightarrow B$. Ein Constraint

kann häufig als mathematische Gleichung betrachtet werden, ein Constraint-System als Gleichungssystem. Unter Constraint-Propagierung versteht man dann die Berechnung der Lösung des Gleichungssystems. Es lassen sich aber nicht nur mathematische Gleichungen beschreiben, sondern auch Ungleichungen und nicht-numerische Zusammenhänge. Constraints eignen sich deshalb zur quantitativen und qualitativen Modellierung von mathematischen und physikalischen Systemen [Puppe1991].

Es existieren die unterschiedlichsten Möglichkeiten Constraints zu realisieren, z.B. läßt sich die Beziehung zwischen Körpergröße und Idealgewicht folgendermaßen darstellen:

- Tabelle, z.B. mit Hilfe einer Gewichtstabelle
- Funktion, z.B. Normalgewicht = Körpergröße - 100
- Prädikate: Übergewicht = $\frac{\text{tatsächliches Gewicht}}{\text{Normalgewicht}} < 0,8$

Als Eingabe für den Propagierungs-Algorithmus dient ein Constraint-System, sowie eine Variablenbelegung, soweit diese bekannt ist. Als Ausgabe erhält man eine Belegung der bisher unbekannten Variablen. Wenn alle Variablen einen eindeutigen Wert annehmen, liegt eine eindeutige Lösung vor. Es ist aber auch möglich, daß für eine Variable eine Wertemenge von möglichen Lösungen zurückgegeben wird, oder eine Variable unter den gegebenen Bedingungen keinen gültigen Wert annehmen kann. Die Aufgabe des Propagierungs-Algorithmus ist es also, den Wertebereich einer Variablen solange einzuschränken, bis keine weiteren Einschränkungen mehr möglich sind. Es hängt von der Komplexität dieses Algorithmus ab, welche Ausdrücke in Constraints verarbeitet werden können.

- Nur Feste Werte: $X = 5$
- Wertemenge: $X \in \{3, 5, 6, 8, 10\}$
- Symbolische Ausdrücke: $X = 2 * Y$

Veranschaulichung der Arbeitsweise anhand eines Beispiels

Mit Hilfe des nachfolgenden Beispiels, ,,Der Hilferuf eines Studenten an seinen Vater" [Puppe1991], soll die Arbeitsweise des Propagierungs-Algorithmus erläutert werden.

Es soll jedem Buchstaben eine Ziffer zugewiesen werden, so daß obige Addition korrekt ist. Es dürfen keine Ziffern doppelt zugeordnet werden. Die höchstwertige

```
  SEND
+ MORE
------
 MONEY
```

Abbildung 3.7: Aufgabenstellung des Problems „Send More Money".

Ziffer jeder Zahl muß ungleich Null sein. Wieviel Geld benötigt der Student also von seinem Vater?

Der erste und vielleicht schwerste Schritt bei der Lösung des Problems ist die geeignete Formulierung. Die gegebene Darstellung stellt eine Gleichung in acht Variablen dar. Statt dessen teilt man nun die Gleichung entsprechend der Spalten und erhält somit fünf Gleichungen. Eine mögliche Formulierung der Constraints wäre eine Mischung aus Funktionen und Prädikaten, z.B. für die letzte Spalte „wenn $(D + E < 10)$ dann $(D + E = Y)$, sonst $(D + E = 10 + Y)$". Diese Formulierung ist aber sehr umständlich, da für die folgenden Spalten immer mehr Fallunterscheidungen nötig werden. Eine weitaus bessere Beschreibung zeigen die Gleichungen (3.6) bis (3.10) , bei der die Überträge der Gleichung als neue Variablen $U_1, \ldots, U_4$ dargestellt werden. Da die führenden Stellen (M, S) ungleich Null sein müssen, ergeben sich die unten gezeigten Wertebereiche.

$$C_1: \quad D + E = Y + 10 * U_1 \tag{3.6}$$
$$C_2: \quad N + R + U_1 = E + 10 * U_2 \tag{3.7}$$
$$C_3: \quad E + O + U_2 = N + 10 * U_3 \tag{3.8}$$
$$C_4: \quad S + M + U_3 = O + 10 * U_4 \tag{3.9}$$
$$C_5: \quad M = U_4 \tag{3.10}$$

$$D, E, N, O, R, Y \in \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$$
$$M, S, \in \{1, 2, 3, 4, 5, 6, 7, 8, 9\}$$
$$U_1, U_2, U_3, U_4 \in \{0, 1\}$$

Aus C_5 folgt, daß M und U_4 die selbe Zahl darstellen, und ist deshalb durch die Schnittmenge der beiden Wertebereiche mit $M = 1$ eindeutig bestimmt. Dadurch kann die Ziffer 1 aus der Wertebereichen der einzelnen Variablen gestrichen werden, nicht aber aus dem Wertebereichen der Überträge. In der weiteren Folge wird nicht mehr angegeben, daß eine eindeutig zugeordnete Zahl aus den Wertebereichen gestrichen wird. Constraint C_4 reduziert sich nun auf $S + U_3 = O + 9$. Damit können keine weiteren Werte eindeutig bestimmt werden. Es wird also eine Annahme nötig, die auf ihre Richtigkeit überprüft wird. Da U_3 die kleinste Wertemenge besitzt, wird eine Annahme für diese Variable getroffen.

Annahme: $U_3 = 1$. C_4 vereinfacht sich zu $S = O + 8$. Daraus ergibt sich, daß $S \in \{8, 9\}$ ist und $O = 0$, da wegen $M = 1$ die Ziffer aus dem Wertebereich gestrichen wurde. Damit vereinfacht sich C_3 zu $E + U_2 = N + 10$, was für keine gültige Wertebelegung erfüllte sein kann, da $E < 10$ sein muß. Die Annahme $U_3 = 1$ war also falsch, und muß ebenso wie alle daraus abgeleiteten Variablen zurückgesetzt werden.

Damit folgt die neue Annahme $U_3 = 0$. C_4 vereinfacht sich zu $S = O + 9$ und ist nur mit $S = 9$ und $O = 0$ erfüllbar. Für C_3 ergibt sich nun $E + U_2 = N$. Die Annahme $U_2 = 0$ führt zum Widerspruch $E = N$, also muß $U_2 = 1$ sein. Daraus folgt $E + 1 = N$. Direkt läßt sich mit C_3 keine weitere Variable bestimmen. Daher ersetzt man in C_2 N durch $E + 1$. Diesen Schritt macht das symbolische Propagieren möglich. Für C_2 ergibt sich $R + U_1 = 9$. Da die Ziffer 9 nicht mehr im Wertebereich der Variablen R ist, folgt $R = 8$ und $U_1 = 1$. Für C_1 gilt nun $D + E = Y + 10$. Da jede der Variablen D, E, N, Y noch sechs Werte annehmen kann, scheint eine Fallunterscheidung wenig sinnvoll. Auch hilft uns das symbolische propagieren nicht weiter. Dagegen hilft die Propagierung der Wertemengen. Die aktuellen Wertemengen sind D, E, N, $Y \in \{2, 3, 4, 5, 6, 7\}$. Durch die Gleichung $E + 1 = N$ folgt, daß für N der Wert 2 und für E der Wert 7 herausfällt. Aus $D + E = Y + 10$ folgt die Ungleichung $D + E > 11$. Dadurch werden die Wertemengen weiter eingeschränkt: $E \in \{5, 6\}$ und $D \in \{6, 7\}$. Da für E nur mehr zwei Werte zur Verfügung stehen, kann wieder eine Annahme getroffen werden. Aus $E = 6$ folgt $D = 7$ und $N = 7$ und stellt einen Widerspruch dar. Daher muß $E = 5$ sein. Damit ergeben sich für die restlichen Variablen die Werte: $N = 6$, $D = 7$ und $Y = 2$. Die eindeutige Lösung des Problems ist in Abbildung 3.8 dargestellt.

```
  9567
+ 1085
------
 10652
```

Abbildung 3.8: Eindeutige Lösung des Problems „Send More Money".

Komplexität von Constraintsystemen

Die im Beispiel angegebene Lösung kann nur von einem sehr leistungsfähigen Constraintsystem gefunden werden. Entsprechend der Fähigkeiten des Constraint-Propagierungs-Algorithmus lassen sich die Constraint-Systeme wie folgt einteilen [Puppe1991]:

- Einfache Constraintsysteme, die nur feste Werte propagieren können; diese

würden für obiges Beispiel keine Lösung finden, da nach dem ersten Schritt für jede Variable mehr als ein Wert in Frage kommt.

- Ein weitgehend einfaches Constraintsystem, das zusätzlich auch Fallunterscheidungen beherrscht, würde erst nach sehr viel willkürlichem raten eine Lösung finden. Wäre die Wertemenge der Variablen unendlich, z.B. die reellen Zahlen, wäre die Fallunterscheidung prinzipiell unmöglich.

- Ein Constraintsystem, das auch in der Lage ist Wertemengen zu propagieren, würde den Suchraum weitgehend einschränken, müßte jedoch auch noch viel raten.

- Bei komplexen Problemen erweist sich die symbolische Propagierung als leistungsstarke Technik. Sie ist jedoch sehr aufwendig zu implementieren, da es vom System algebraische Umformungen verlangt.

- Wenn ein System über alle oben angeführten Techniken verfügt, benötigt es noch gute Heuristiken, um zu entscheiden, wann welche Technik angewandt werden soll. So eignet sich z.B. eine Fallunterscheidung nur, wenn der Wertebereich der Variable möglichst kein ist und sich die einzelnen Fälle nicht weiter verzweigen.

Aus den oben angeführten Punkten ist ersichtlich, daß die Komplexität des Propagierungs-Algorithmus das Aufgabengebiet bestimmt, in dem das Constraint-System sinnvoll eingesetzt werden kann.

3.8 Unsicheres Wissen

In den bisher besprochenen Repräsentationen wurde Wissen immer in Form von eindeutigen Fakten, Regeln, Bedingungen oder Abbildungen gespeichert. Oft ist es jedoch nicht möglich, eindeutige Aussagen über die reale Welt zu treffen oder einen strikten Zusammenhang zwischen Bedingung und Schlußfolgerung herzustellen. Vielmehr können Aussagen und Schlußfolgerungen nur mit einer bestimmten Unsicherheit getroffen werden. Eine Möglichkeit diese Unsicherheit darzustellen ist, anzugeben mit welcher Wahrscheinlichkeit die Aussagen und Schlußfolgerungen zutreffen. Quellen für die Unsicherheit von Information können sein [Gottlob1990]:

Inhärente Unsicherheit der Information, z.B. die Information eines Temperatursensors mit der Toleranz von $\pm 1°C$.

Unvollständigkeit der Information , es werden z.B. fehlende Informationen durch Annahmen ersetzt, sind alle Schlußfolgerungen, die auf dieser Annahme basieren, unsicher.

Unsicherheit von Schlußfolgerungen: Es kann kein strikter Zusammenhang zwischen Bedingung und Schlußfolgerung hergestellt werden, z.B. ,,Wenn der Patient mehr als 38°C Fieber hat, dann ist er mit der Wahrscheinlichkeit von 0,7 an Grippe erkrankt."

Zusammenfassung von Informationen aus mehreren Quellen, widersprechen sich z.B. die Aussagen von zwei Experten, so erhöht dies die Unsicherheit dieser Aussagen.

Unsicherheiten können nummerisch oder symbolisch dargestellt werden. Bei der symbolischen Repräsentation wird der Grad der Unsicherheit durch Elemente aus einer vorgegebenen Menge von Symbolen ausgedrückt, z.B. durch Begriffe wie ,,meistens", ,,beinahe", ,,immer", usw. Bei dieser Methode ist es aber schwierig die Fortpflanzung der Unsicherheit beim Schlußfolgern zu berücksichtigen.

Bei der nummerischen Repräsentation wird die Unsicherheit durch die Zuordnung von einem oder mehreren Zahlenwerten, z.B. der Wahrscheinlichkeit, ausgedrückt. Für die Darstellung der Fortpflanzung der Unsicherheit gibt es mathematische Formalismen. Jedoch ist es oft nicht möglich, einer Aussage oder Schlußfolgerung einen exakten Wahrscheinlichkeitswert zuzuordnen. Im nachfolgenden werden drei Verfahren zur nummerischen Repräsentation von Unsicherheit vorgestellt.

Theorem von Bayes

In der klassischen Wahrscheinlichkeitsrechnung ist die Wahrscheinlichkeit als

$$p(X) = \frac{\text{Anzahl der günstigen Ereignisse}}{\text{Anzahl der möglichen Ereignisse}} \tag{3.11}$$

definiert. Diese Definition erweist sich für die Darstellung von Unsicherheiten als nicht geeignet. Daher wird die Wahrscheinlichkeit als der Grad des Vertrauens einer Person in eine Hypothese definiert. Eine Hypothese bezeichnet eine Annahme über einen Sachverhalt. Die Wahrscheinlichkeiten sind Werte im Intervall [0,1]. Folgende Schreibweisen werden im weiteren verwendet [Gottlob1990]:

$p(X)$	...	Wahrscheinlichkeit, daß X wahr ist
$p(X_1, X_2, ..., X_n)$	...	Wahrscheinlichkeit, daß $X_1, X_2, ..., X_n$ alle wahr sind
$p(X_1, X_2, ..., X_n \| Y_1, Y_2, ...Y_n)$	...	Wahrscheinlichkeit, daß $X_1, X_2, ..., X_n$ alle wahr sind, unter der Voraussetzung, daß $Y_1, Y_2, ..., Y_n$ wahr sind (bedingte Wahrscheinlichkeit)

Allgemeines Theorem von Bayes

Gegeben sei eine Menge von Hypothesen $H = \{h_1, h_2, ..., h_n\}$ und eine Menge von Ereignissen $E = \{e_1, e_2, ..., e_m\}$. Das allgemeine Bayes'sche Theorem baut auf folgenden Voraussetzungen auf [Gottlob1990]:

Die Hypothesen in der Menge H schließen sich gegenseitig aus:

$$p(h_i, h_j) = 0 \text{ für } i \neq j$$

Die Menge H ist erschöpfend:

$$\sum_{i=1}^{n} p(h_i) = 1 \tag{3.12}$$

Jedes Teilergebnis e_i ist bedingt unabhängig von jeder Hypothese:

$$p(e_1, e_2, ..., e_m | h_i) = \prod_{j=1}^{m} p(e_j | h_i) \tag{3.13}$$

Das allgemeine Bayes'sche Theorem besagt, daß die a-posteriori Wahrscheinlichkeit $p(h_i | e_1, e_2, ..., e_m)$ einer Hypothese h_i als Funktion der bedingten Wahrscheinlichkeiten $p(e_1, e_2, ..., e_m | h_i)$ sowie der a-priori Wahrscheinlichkeiten $p(h_i)$ berechnet werden kann [Gottlob1990]:

$$p(h_i | e_1, e_2, ..., e_m) = \frac{p(e_1, e_2, ..., e_m | h_i) p(h_i)}{\sum_{k=1}^{n} p(e_1, e_2, ..., e_m | h_k) p(h_k)} \tag{3.14}$$

Der Spezialfall für ein Ereignis E und eine Hypothese H sieht folgendermaßen aus:

$$p(H|E) = \frac{p(E|H)p(H)}{p(E)} \tag{3.15}$$

Die Anwendung des Satzes soll nun durch ein Beispiel veranschaulicht werden. Im Beispiel tritt das Ereignis E „Die Reifen eines Autos quietschen." mit der Wahrscheinlichkeit $p(E) = 0,05$ auf; die Hypothese H „Die Bremsen des Autos sind schlecht eingestellt." mit der Wahrscheinlichkeit $p(H) = 0.02$.

Nehmen wir weiters an, daß schlecht eingestellte Bremsen oft, aber nicht immer, ein Quietschen der Räder verursachen. Die bedingte Wahrscheinlichkeit dafür ist $p(E|H) = 0,7$. Wenn man nun ein Quietschen der Reifen beobachtet, so kann man mit Hilfe des Bayes'schen Theorem die Wahrscheinlichkeit berechnen, daß die Bremsen schlecht eingestellt sind.

$$p(H|E) = \frac{0,7 * 0,02}{0,05} = 0,28$$

Durch die Beobachtung des Ereignisses E hat sich die Wahrscheinlichkeit der Hypothese H von 0,02 auf 0,28 erhöht. Die Berechnung von $p(H|E)$ ausgehend von $p(H)$ kann als Neubewertung der Hypothese H beim Eintreten des Ereignisses E aufgefaßt werden. Darin liegt die Stärke dieses Theorems. Mit ihm läßt sich die Fortpflanzung der Unsicherheit berechnen. Der Nachteil ist jedoch, daß zu jedem Ereignis und zu jeder Hypothese die Wahrscheinlichkeit und die entsprechenden bedingten Wahrscheinlichkeiten gespeichert werden müssen. Dies erfordert eine große Datenmenge, die schwer zu beschaffen ist und auch oft nicht mit mathematischer Exaktheit beschafft werden kann. [Gottlob1990]

Certainty Factors

In der Praxis werden daher aus oben genannten Gründen Unsicherheiten anstelle von Wahrscheinlichkeiten durch sogenannte Certainty Factors ausgedrückt. Jeder Regel wird ein fester Certainty Factor zugeordnet. Den Fakten, die eine Regel erfüllen muß, sind dynamische Certainty Faktoren zugeordnet. Bei der Abarbeitung einer Regel wird aus den festen und dynamischen Faktoren der Certainty Factor der Schlußfolgerung berechnet. Somit erhält man neue, bewertete Fakten. Der Vorteil der Certainty Factors liegt darin, daß für jede Regel und für jedes Faktum nur ein Wert gespeichert werden muß. Es werden keine bedingten Wahrscheinlichkeiten gespeichert. Daher sind sie einfacher zu handhaben und leichter zu implementieren. Die Implementierung von Certainty Factors wird im Abschnitt 6.8 anhand von Expertensystemen genauer ausgeführt. [Gottlob1990]

Fuzzy Logik

Der Grundgedanke der Fuzzy Logik ist es, eine Theorie der unscharfen Mengen zu entwickeln. Durch die unmittelbare Verknüpfung zwischen Mengenlehre und

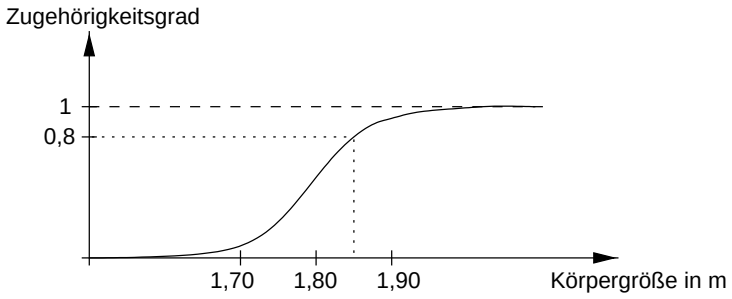

Abbildung 3.9: Beispiel für die Definition einer Fuzzy Menge. Gezeigt wird die Zuordnung zur Menge der großen Personen mit Hilfe der Mitgliedsgradfunktion.

Logik ist damit auch die Theorie der unscharfen Logik verknüpft. An dieser Stelle soll nur erwähnt werden, wie in der Fuzzy Logik unsicheres Wissen gespeichert wird. Näher wird auf dieses Thema in Kapitel 7 eingegengen. [Rojas1993]

In der klassischen Mengenlehre gilt für ein Objekt, daß es entweder Element oder kein Element der Menge ist. In der Fuzzy Logik besitzt eine Menge keine so scharfen Grenzen. Damit ist es möglich, daß ein Element nur „zu einem bestimmten Grad" einer Menge angehört. Der Grad der Zugehörigkeit wird durch einen Wert aus dem Intervall [0,1] angegeben. Dieses Konzept erweist sich als vorteilhaft, wenn man Zuordnungen aus dem natürlichen Sprachgebrauch darstellen will.

Ein Beispiel dafür ist die Aussage: „Die Person X ist groß.". Befragt man mehrere Personen wo für sie die Grenze liegt, ab wann eine Person als groß gilt, so wird man keinen eindeutigen Grenzwert erhalten. Für manche beginnt groß schon bei 1,75m, für andere erst bei 1,85m. In der Fuzzy Logik kann diese unscharfe Grenze mit der Mitgleidsgradfunktion ausgedrückt werden. Abbildung 3.9 zeigt einen solchen Verlauf für die Zugehörigkeit zur Menge der großen Personen. Die Mitgliedsgradfunktion ordnet z.B. eine Person die 1,85m groß ist, mit dem Grad 0,8 zur Menge der großen Personen zu.

Kapitel 4

Problemlösen durch Suchen

Voraussetzung dafür, daß ein Problem durch Suchen gelöst werden kann, ist, daß der Ausgangspunkt für die Suche, der Start, und das Ziel der Suche bekannt ist. Der Start- und Zielpunkt sind Zustände der ,,realen" Welt. Zusätzlich müssen Aktionen bekannt sein, die ausgeführt werden können, um vom Start zum Ziel zu gelangen. Aktionen sind somit Handlungen die zu erlaubten Zwischenzuständen führen. Ziel der Suche ist es somit, eine Aktionsfolge zu finden, die vom Start zum Ziel führt. [Russell1995]

Bei der Formulierung des Starts, des Ziels und der Aktionen muß die ,,reale" Welt bis zu einem bestimmten Grad abstrahiert werden. Ohne diese Abstraktion würde die Komplexität des Suchproblems zu groß werden. Dies soll anhand eines Beispiels erklärt werden: Ein Vertreter will von einer Stadt zu einer anderen Stadt reisen. Denkbare Aktionen, um dieses Problem zu lösen, wären, bewege den Fuß $20cm$ nach vorne oder schlage das Lenkrad 6 Grad nach links ein, usw. Diese detailierten Einzelaktionen würden jedoch zu einem zu komplexen Suchproblem führen. Viel sinnvoller und zielführender ist es das Problem zu abstrahieren und die Aktionen als Fahrt von einer größeren Stadt zur einer anderen Stadt, mit der eine direkte Straßenverbindung besteht, zu definieren. Die reale Welt wird, zur Lösung dieses Problems, somit zu einer Straßenkarte abstrahiert.

Zusammenfassend kann man sagen, daß das Problemlösen durch Suchen aus folgenden drei Schritten besteht: [Russell1995]

Formulierung des Problems: Es werden der Start, das Ziel und die erlaubten Aktionen festgelegt. Dabei wird eine sinnvolle Abstraktion der ,,realen" Welt vorgenommen.

Suche: Es wird eine Folge von Aktionen gesucht, die vom Start zum Ziel führen.

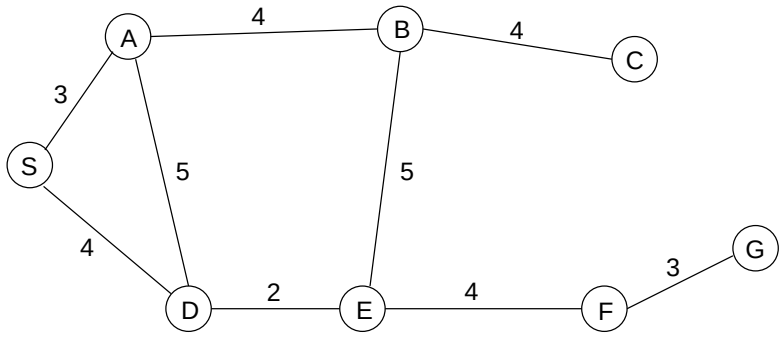

Abbildung 4.1: Straßenkarte als Semantisches Netz dargestellt. Die Städte entsprechen den Nodes, die Links den Straßen und Link Labels den Kosten, die eine Benützung der Straße verursacht.

Ausführung: Es wird die Aktionsfolge, die man als Ergebnis der Suche erhalten hat, Schritt für Schritt abgearbeitet.

In den nachfolgenden Abschnitten werden nun Algorithmen zur Suche näher beleuchtet. Als Grundlage für die Erklärung dient das in Abbildung 4.1 dargestellte semantische Netz, das man auch als Graphen bezeichnen kann. Darin stellt S den Start-Node und G den Ziel-Node dar.

Vorstellen kann man sich den Graphen z.B. als eine Straßenkarte. Die Nodes stellen Städte dar, die Links Straßen und die Link Labels die Kosten, die eine Benützung dieser Verbindung verursacht. Erlaubte Aktionen sind somit die Fahrt auf einer Straße von einer Stadt zur nächsten. Um einen Weg vom Start-Node S zum Ziel-Node G auf dieser Straßenkarte zu finden, sind zwei Arten von Kosten zu berücksichtigen [Winston1993]:

1. Die Kosten, die für das Auffinden eines Lösungspfades aufgewendet werden müssen.

2. Die Kosten, die durch das Benutzen eines gefundenen Pfades entstehen.

Muß man häufig von der Stadt S zur Stadt G reisen, wird man an einem möglichst kurzen und kostensparenden (optimalen) Lösungspfad interessiert sein. Um diesen zu finden, wird man auch hohe Suchkosten in Kauf nehmen. Will man jedoch nur einmal von S nach G reisen und ist es schwer den optimalen Lösungspfad mit den geringsten Kosten zu finden, wird man sich mit dem ersten gefundenen Lösungspfad zufriedengeben, auch wenn man weiß, daß es vielleicht bessere gibt. Die in Abschnitt 4.1 Blinde Suche und 4.2 Heuristische Suche vorgestellten

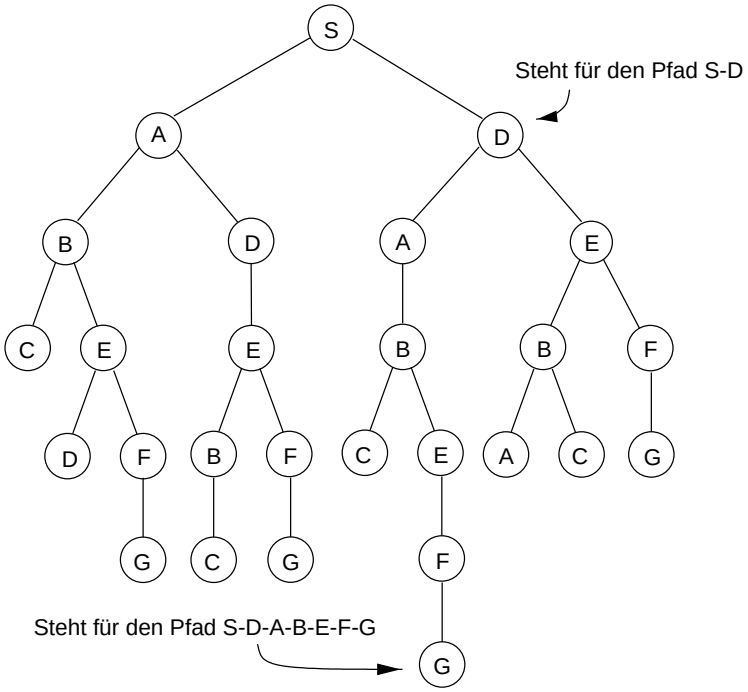

Abbildung 4.2: Der Suchbaum für das Straßenkartenproblem von Abbildung 4.1. Jeder Node bezeichnet einen Pfad, der vom Start-Node ausgeht. [Winston1993]

Methoden sind in der Lage einen möglichen Lösungsweg zu finden. Suchverfahren, die den optimalen Lösungsweg finden, werden in Abschnitt 4.3 vorgestellt.

Die offensichtlichste Methode einen Weg vom Start zum Ziel zu finden, ist das Betrachten aller möglichen Wege. Wege die zu Schleifen führen, werden natürlich nicht mit berücksichtigt. Ausgehend vom Start-Node lassen sich nun alle möglichen Pfade als Baum darstellen, wie in Abbildung 4.2 gezeigt wird. Dabei ist zu beachten, daß jeder Node im Baum einen Pfad darstellt. Alle Nodes sind jedoch nur mit dem Buchstaben des Endpunktes des Pfades gekennzeichnet. [Winston1993]

Eine Eigenschaft des so entstandenen Suchbaums ist der Branching-Faktor. Darunter versteht man die Anzahl der Child-Nodes b die von einem Node ausgehen. Damit entspricht der Branching-Faktor der Anzahl der Nachbar-Nodes im Graphen. Ist diese Anzahl für alle Nodes im Baum gleich, so spricht man von einem Baum mit dem Branching-Faktor b. Der Branching-Faktor gibt somit eine Auskunft über das Wachstum des Baumes. Ein Baum mit dem Branching-Faktor b besitzt in der Ebene d die Anzahl von b^d Child-Nodes. Die Ebenen werden ausgehend vom Root-Node gezählt, der sich in der Ebene 0 befindet. [Winston1993]

4.1 Blinde Suche

Unter blinder Suche versteht man, daß zur Auffindung eines Lösungspfades keine weiteren Informationen über das Suchproblem vorliegen, die für einen effektivere Suche herangezogen werden könnten. In unserem Beispiel könnten das die Luftlinien-Entfernungen zwischen den einzelnen Städten sein. Zu den Blinden Suchen gehören die Tiefensuche, die Breitensuche und die Randomisierte Suche. [Winston1993]

4.1.1 Tiefensuche (depth-first search)

Bei der Tiefensuche wird ausgehend vom Start-Node ein Nachbar-Node besucht. Den Nachbarn des Start-Nodes im Graphen entsprechen die Children des Start-Nodes im Suchbaum. Ist dieser Nachbar noch nicht der Ziel-Node, so wird ein Nachbar dieses Nodes besucht. Dies geschieht solange, bis sich entweder eine Schleife bildet, oder der Algorithmus in eine Sackgasse gerät. In diesem Fall geht der Algorithmus soviele Schritte zurück, bis er einen Node findet, dessen Nachbarn er noch nicht alle besucht hat und setzt mit einem solchen Nachbarn fort. Abbildung 4.3 zeigt diese Vorgangsweise. Der Algorithmus ist in Abbildung 4.4 dargestellt [Winston1993]. Die Tiefensuche eignet sich damit besonders für Suchbäume, deren Äste eine vertretbare Länge nicht überschreiten.

4.1.2 Breitensuche (breadth-first search)

Die Breitensuche untersucht ausgehend von Start-Node alle Nachbar-Nodes. Ist der Ziel-Node noch nicht erreicht, werden alle Nachbar-Nodes der bisher untersuchten Nachbarn betrachtet, usw. Aus der Sicht des Suchbaums arbeitet man sich somit Ebene für Ebene nach unten, wie dies in Abbildung 4.5 dargestellt wird. Der Algorithmus für die Breitensuche ist in Abbildung 4.6 dargestellt. [Winston1993]

Welcher der beiden Algorithmen verwendet werden soll, die Tiefen- oder die Beitensuche, hängt von der Aufgabenstellung ab. Die Tiefensuche eignet sich besonders für Suchbäume, deren Äste nach einer vertretbaren Anzahl von Schritten in eine Sackgasse oder zum Ziel führen. Die Tiefe des Baumes darf nicht zu groß sein. Besonders schlecht verhält sich die Tiefensuche, wenn der Suchbaum lange Äste besitzt, die nicht zum Ziel führen.

Die Breitensuche eignet sich auch für Bäume mit großer Tiefe. Jedoch darf der Suchbaum nicht stark verzweigen. Ein großer Branching-Faktor führt zur exponentiellen Explosion der Nodes-Anzahl. [Winston1993]

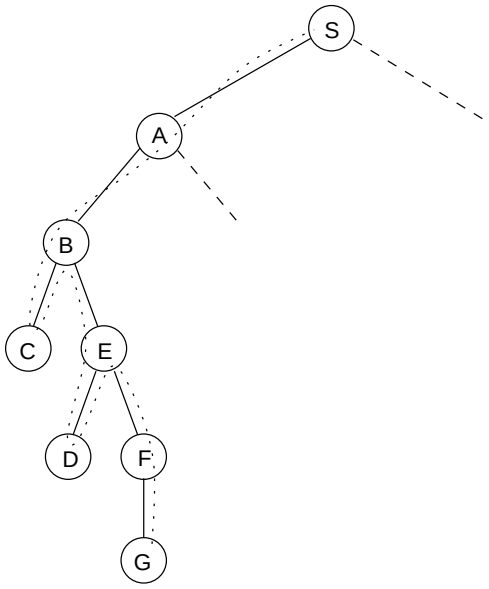

Abbildung 4.3: Ein Beispiel für eine Tiefensuche. Mit jedem Schritt wird eine Ebene tiefer untersucht, bis das Ziel gefunden ist oder man an ein Blatt gelangt ist. In diesem Fall geht man den Baum soweit zurück, bis man einen Node gefunden hat, dessen Children noch nicht fertig betrachtet sind.

- Erzeuge einen Pfad, der nur aus dem Start-Node besteht und schreibe ihn in die Queue
- Solange, bis der erste Pfad in der Queue zum Ziel-Node führt oder die Queue leer ist
  - Lösche den ersten Pfad aus der Queue. Ermittle den Endpunkt des Pfades und erzeuge je einen neuen Pfad zu allen Nachbarn dieses Endpunktes.
  - Entferne alle neuen Pfade, die Schleifen enthalten.
  - Füge die verbleibenden Pfade am **Beginn** der Queue ein.
- Wenn der Zielknoten erreicht wurde, gib den Pfad aus.

Abbildung 4.4: Algorithmus für die Tiefensuche

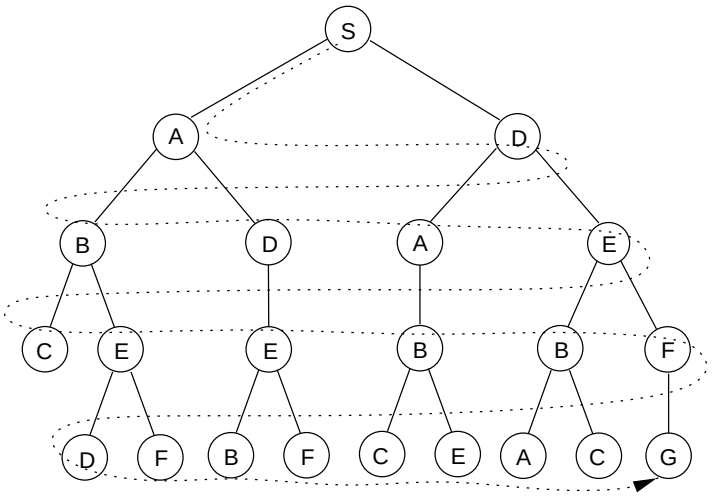

Abbildung 4.5: Ein Beispiel für eine Breitensuche. Der Suchbaum wird Ebene für Ebene durchsucht.

- Erzeuge einen Pfad, der nur aus dem Start-Node besteht und schreibe ihn in die Queue
- Solange, bis der erste Pfad in der Queue zum Ziel-Node führt oder dieQueue leer ist
  - Lösche den ersten Pfad aus der Queue. Ermittle den Endpunkt des Pfades und erzeuge je einen neuen Pfad zu allen Nachbarn dieses Endpunktes.
  - Entferne alle neuen Pfade, die Schleifen enthalten.
  - Füge die verbleibenden Pfade am **Ende** der Queue ein.
- Wenn der Zielknoten erreicht wurde, gib den Pfad aus.

Abbildung 4.6: Algorithmus für die Breitensuche

4.1.3 Nichtdeterministische Suche

Besitzt man keine Informationen über den Branching Faktor oder über die voraussichtliche Tiefe des Suchbaumes, ist es schwer, eine Entscheidung zwischen den beiden oben vorgestellten Algorithmen zu treffen. Einen Mittelweg für diesen Fall stellt die Nichtdeterministische Suche dar. Bei diesem Algorithmus wird durch Zufall bestimmt, von welchem noch nicht fertig abgearbeiteten Node, die Nachbarn untersucht werden. Der Algorithmus gleicht dem der Breiten- und Tiefensuche, jedoch werden die untersuchten Pfade an zufälliger Stelle in die Queue geschrieben. Der Algorithmus ist in Abbildung 4.7 dargestellt [Winston1993].

- Erzeuge einen Pfad, der nur aus dem Start-Node besteht und schreibe ihn in die Queue
- Solange, bis der erste Pfad in der Queue zum Ziel-Node führt oder die Queue leer ist
  - Lösche den ersten Pfad aus der Queue. Ermittle den Endpunkt des Pfades und erzeuge je einen neuen Pfad zu allen Nachbarn dieses Endpunktes.
  - Entferne alle neuen Pfade, die Schleifen enthalten.
  - Füge die verbleibenden Pfade an **zufälliger Stelle** in die Queue ein.
- Wenn der Zielknoten erreicht wurde, gib den Pfad aus.

Abbildung 4.7: Algorithmus für die Nichtdeterministische Suche.

4.2 Heuristisch informierte Suche

Bei vielen Problemen lassen sich die Suchkosten dramatisch verringern, wenn man zusätzliche Informationen heranzieht, die dabei helfen, den Nachbar-Node auszuwählen, der als nächstes betrachtet werden soll. Suchverfahren die solche Informationen in ihre Suchentscheidungen einfließen lassen, heißen Heuristische[1] Suchverfahren. In unserem Beispiel sollen als zusätzlichen Informationen die Luftlinie-Entfernung von den Nodes zum Ziel-Node herangezogen werden. Für das Straßenkarenproblem sind die Entfernungen in Abbildung 4.8 angegeben. Anhand dieser

[1] Heuristik: Kunst (Methode) der Wahrheitsfindung. Heuristische Prinzipien sind Regeln, Hypothesen, versuchsweise Annahmen, die nur vorläufig, in Hinblick auf das zu findende aufgestellt, nicht tatsächlich bzw. endgültig betrachtet werden. [BE1978]

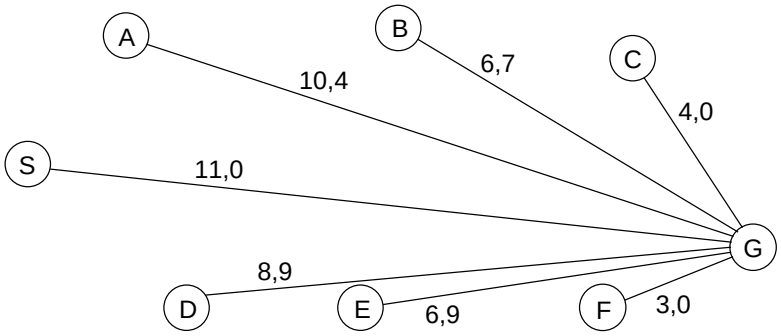

Abbildung 4.8: Zusätzlich zu den Informationen aus Abbildung 4.1 sind auch die Luftlinie-Entfernungen der Nodes zum Ziel-Node bekannt. Heuristische Suchverfahren verwenden diese zusätzliche Information, um die Suche effektiver zu gestalten.

Abbildung soll das Hill-Climbing, das Beam Search und das Best-First Search Verfahren vorgestellt werden. [Winston1993]

4.2.1 Hill-Climbing

Hill-Climbing-Search verhält sich wie die Tiefensuche, mit dem Unterschied, daß zuerst der Node betrachtet wird, der die kürzeste Luftlinie-Entfernung zum Ziel hat. Abbildung 4.9 zeigt dieses Vorgehen. Bei der ersten Entscheidung liegt D näher am Ziel als A. Daher werden zuerst die Children von D untersucht. Der Algorithmus ist in Abbildung 4.10 dargestellt und gleicht dem der Tiefensuche. Es werden jedoch die Pfade sortiert, bevor sie in die Queue geschrieben werden. [Winston1993]

Die grundsätzliche Vorgehensweise des Hill-Climbe Algorithmus und die Probleme die bei seiner Anwendung entstehen können, sollen anhand eines anderen Beispiels erklärt werden. Diese Probleme können auch bei den später vorgestellten heuristischen Suchverfahren auftreten.

Ein Bergsteiger will einen Gipfel erklimmen. Es ist neblig. Der Bergsteiger besitzt weder eine Karte, noch befindet er sich auf einem Weg. Die einzigen Hilfsmittel die er besitzt, sind ein Kompaß und ein Höhenmesser. Um den Gipfel zu erreichen wird der Bergsteiger von seiner Position aus einen Schritt nach Norden machen, die Höhe feststellen und wieder einen Schritt zurück auf seine ursprüngliche Position gehen. Die selbe Prozedur wiederholt er für die anderen drei Himmelsrichtungen. Dies entspricht dem Untersuchen der Child-Nodes im Suchbaum. Der Bergsteiger hat nun für jede Himmelrichtung notiert, in welche Höhe er gelangen kann, wenn er einen Schritt in dieser Richtung geht und wird sich für die

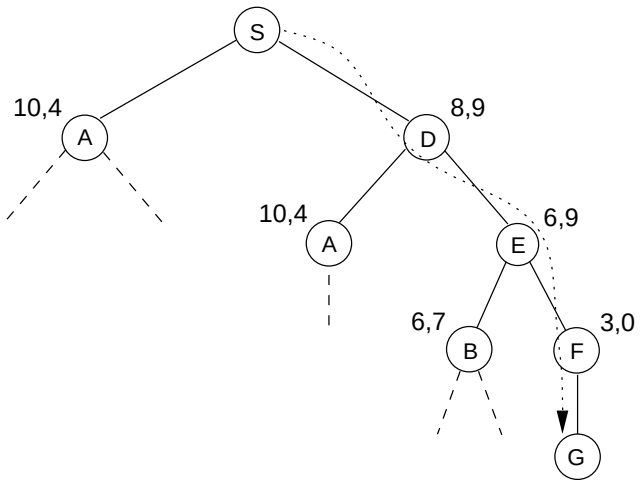

Abbildung 4.9: Der Hill-Climbing Algorithmus funktioniert wie die Tiefensuche, jedoch wird immer zuerst der scheinbar beste Pfad weiterverfolgt. Es gibt aber keine Garantie, daß der scheinbar beste Pfad auch tatsächlich der Beste ist oder überhaupt zum Ziel führt.

- Erzeuge einen Pfad, der nur aus dem Start-Node besteht und schreibe ihn in die Queue
- Solange, bis der erste Pfad in der Queue zum Ziel-Node führt oder die Queue leer ist
  - Lösche den ersten Pfad aus der Queue. Ermittle den Endpunkt des Pfades und erzeuge je einen neuen Pfad zu allen Nachbarn dieses Endpunktes.
  - Entferne alle neuen Pfade, die Schleifen enthalten.
  - Sortiert die verbleibenden neuen Pfade aufsteigend, nach der Distanz zum Ziel.
  - Füge die verbleibenden Pfade am **Beginn** der Queue ein.
- Wenn der Zielknoten erreicht wurde, gib den Pfad aus.

Abbildung 4.10: Algorithmus für Hill Climbing

Richtung entscheiden, in die er die größte Höhe erreicht. Diese Vorgehensweise ist sehr effektiv. Der Bergsteiger kann jedoch auf folgenden drei Probleme stoßen. Die Bezeichnungen sind zwar an das Bergsteigen angelehnt, können aber bei allen heuristischen Suchenverfahren auftreten [Winston1993]:

- **Vorberge:** Dieses Problem tritt dann auf, wenn sich auf der Ebene, auf der sich der Bergsteiger befindet, mehrere lokale Maxima existierten, wie dies in Abbildung 4.11a dargestellt ist. Gerät der Hill-Climbe Algorithmus in die Nähe eines lokalen Maximums, so wird er von diesem angezogen. Er findet damit ein lokales Maximum, nicht aber das Ziel, ein globales Maximum.
- **Platos:** Der Bergsteiger befindet sich auf einer Ebene gleich schlechter Lösungen, weitweg von einem Maximum, das ihn anzieht. Im Extremfall sind die Maxima ganz scharfkantig ausgeprägt, wie die in Abbildung 4.11b Dargestellt ist. Der Hill-Climbe Algorithmus wird somit von keinem Maximum angezogen.
- **Grade:** Der Bergsteiger befindet sich auf dem Rücken eines Grades, der nicht in eine Richtung verläuft, in der sich der Bergsteiger bewegen kann. Da es in alle Richtungen nur bergab geht, erscheint die aktuelle Position wie ein Maximum. Eine solche Situation ist in 4.11c abgebildet.

Die drei oben angeführten Probleme tragen zwar Bezeichnungen, die aus der Sprache des Bersteigens kommen, sie treten aber auch bei anderen Anwendungsgebieten auf, die mit heuristischen Suchverfahren gelöst werden sollen. Beim Beispiel mit der Straßenkarte aus Abbildung 4.1 wäre der Node C z.B. ein Vorberg. Von diesem lokalen Maximum führt keine Straße zum Ziel-Node G.

Allgemein formuliert arbeitet der Hill-Climbe Algorithmus nach dem folgenden Prinzip: Der Algorithmus unternimmt von der aktuellen Position aus alle machbaren Schritte und mißt und bewertet wie weit ihn die einzelnen Schritte dem Ziel näher gebracht haben. Anschließend entscheidet er sich für den Schritt, der ihn scheinbar dem Ziel am nächsten bringt. Verfängt sich der Algorithmus an einem lokalen Maximum, bei dem man sich durch alle Schritte vom Ziel weg bewegt, so kann man nichtdeterministisch fortfahren. Man bewegt sich eine zufällige Anzahl von Schritten in zufällige Richtungen von der aktuellen Position fort, in der Hoffnung, dem Einzugsgebiet des lokalen Maximums zu entkommen und setzt den Hill-Climbe Algorithmus fort. Ähnlich verfährt man, wenn man sich auf einer Ebene befindet, in der Hoffnung in das Einzugsgebiet eines Maximums zu gelangen [Winston1993].

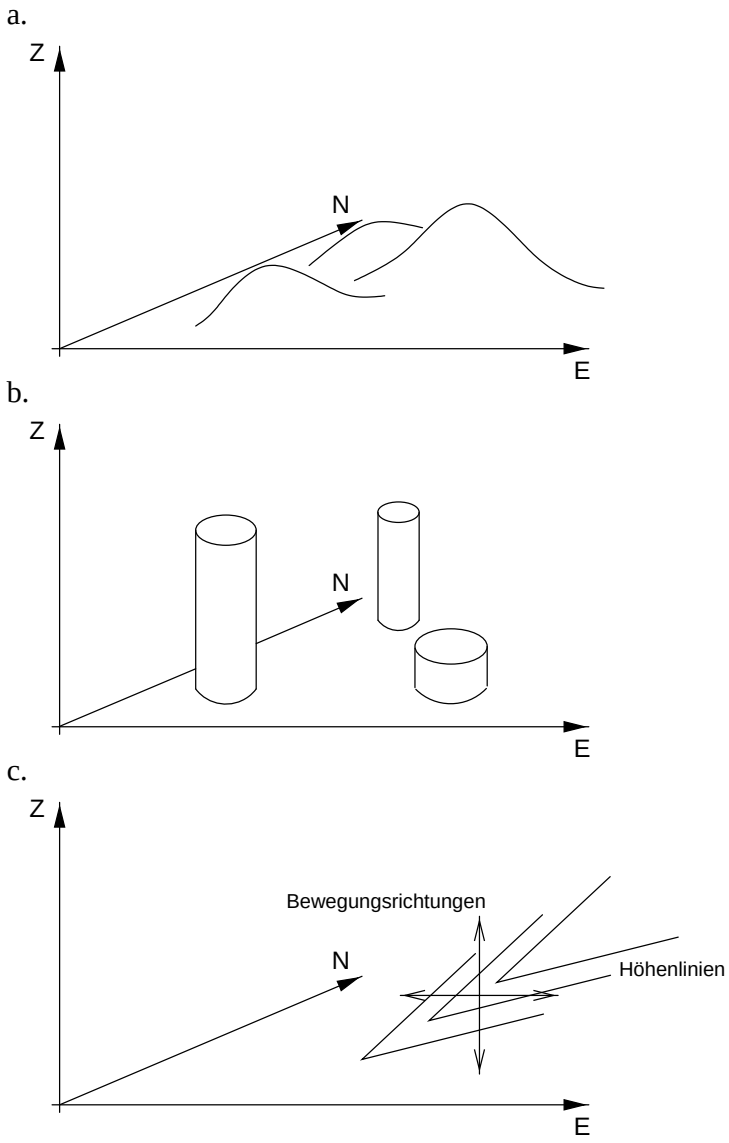

Abbildung 4.11: Probleme, auf die man beim Hill-Climbing Algorithmus stoßen kann: a. Vorberge, b. Ebene, c. Grade.

4.2.2 Beam Search

Beam Search verhält sich ähnlich wie die Breitensuche. Der Baum wird Ebene für Ebene abgearbeitet. Jedoch werden pro Ebene nur die w besten Pfade weiter betrachtet. Der Baum bleibt somit unabhängig vom Branching Faktor nur w Pfade breit und explodiert nicht exponentiell. Abbildung 4.12 zeigt die Vorgangsweise des Beam Search Algorithmus mit $w = 2$ anhand des Straßenkartenproblems. Es ist möglich, daß der Beam-Search Algorithmus keine Lösung eines Problems findet, obwohl eine Lösung existiert. Dies ist dann der Fall, wenn die w scheinbar besten Pfade einer Ebene in weiterer Folge nicht zum Ziel führen. [Winston1993]

4.2.3 Best-First Search

Das Best-First Search Verfahren verhält sich ähnlich dem Hill-Climbing. Der Unterschied liegt im Verhalten, wenn sich ein Pfad als Sackgasse herausstellt. Das Hill-Climbing geht soviel Nodes zurück, bis es zu einen Node gelangt, von dem aus es noch nicht alle Möglichkeiten untersucht hat. Gerät das Best-First Verfahren in eine Sackgasse, so betrachtet es die Bewertungen alle Pfade, die es bereits gegangen ist und setzt den besten gefundenen, noch nicht besuchten Pfad, fort. Für das Beispiel des Straßenkartenproblems verhält sich Best-First Search und Hill-Climbing identisch, da die Suchen in keine Sackgasse gerät, wie Abbildung 4.9 zeigt. [Winston1993]

Zusammenfassung

Die oben vorgestellten heuristischen Suchverfahren können nur dann eingesetzt werden, wenn sich für jeden Node eine Maßzahl bestimmen läßt, die die Entfernung zum gewünschten Suchziel widerspiegelt. Das Hill-Climbing und das Beam Search Verfahren arbeiten effektiv, wenn ein Zielpfad unter jenen Pfaden ist die bei jedem Entscheidungspunkt als die bestmöglichen erscheinen. Best-First Search arbeitet auch dann noch effektiv, wenn gute Pfade auf den ersten Blick nicht als solche erscheinen.

4.3 Optimale Netzsuche

In dem Graphen, der in Abbildung 4.1 dargestellt ist, sind bei den Links zwischen den Nodes Zahlen angegeben, die Link-Labels. Diese Link-Labels entsprechen den Kosten, die durch eine Benützung dieses Links entstehen. Ein optimales Suchverfahren hat nun die Aufgabe, einen Pfad von Start-Node S zum Ziel-Node G

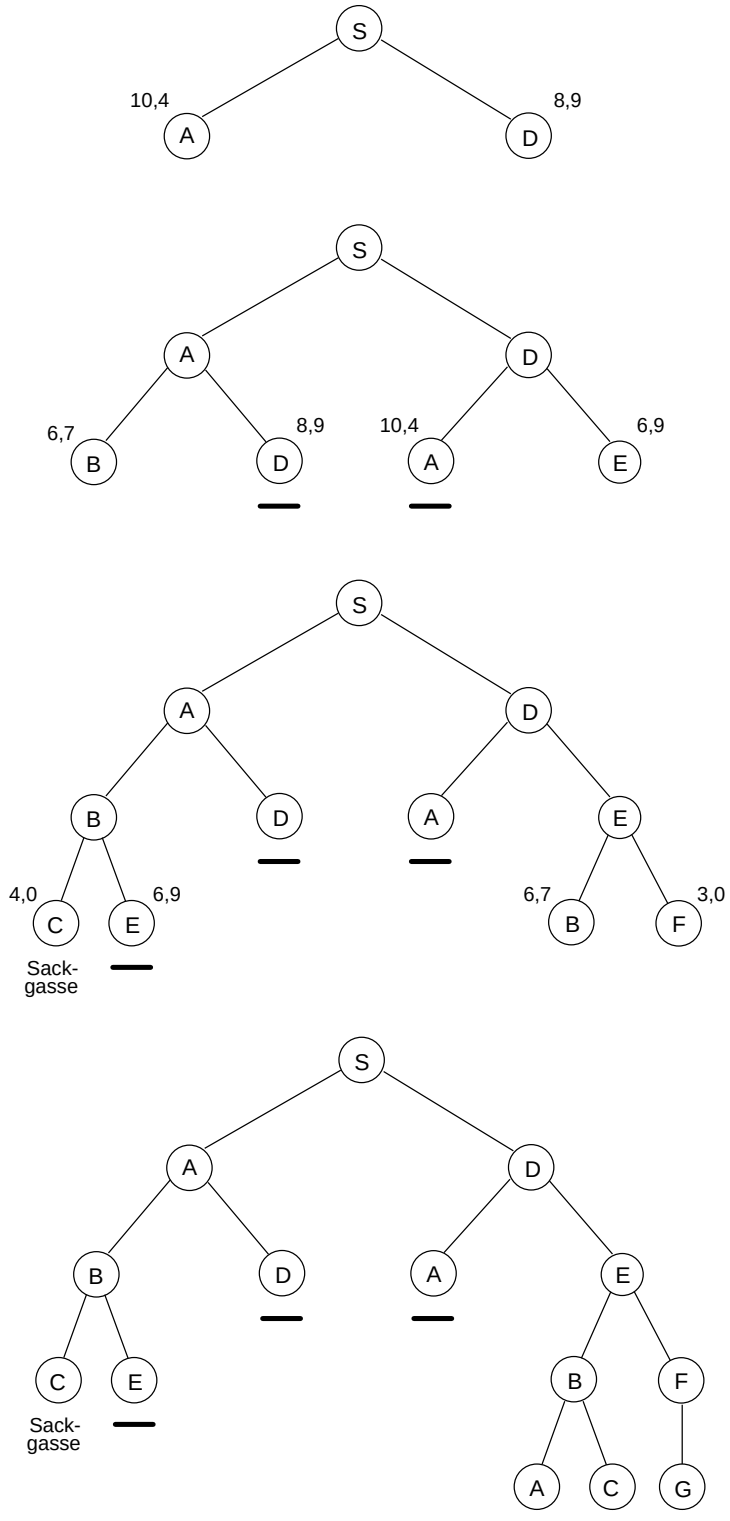

Abbildung 4.12: Beam-Search mit $w = 2$. Es werden pro Ebene im Suchbaum nur die scheinbar w besten Pfade weiterverfolgt.

zu finden, bei dem die Summe der Kosten der benutzten Links am geringsten ist. Sieht man den Graphen aus Abbildung 4.1 als Straßenkarte an, so können die Link-Labels z.B. den Entfernungen zwischen den einzelne Städten entsprechen. Der gesuchte optimale Pfad wäre somit die kürzestmögliche Verbindung vom Start zum Ziel. In diesem Kapitel sollen nun Algorithmen vorgestellt werden, mit deren Hilfe man den optimalen Lösungspfad suchen kann. Dazu gehören die British Museum Procedure, Branch-and-Bound Search, Branch-and-Bound Search mit Abschätzung der Restkosten, Branch-and-Bound Search mit Eliminierung redundanter Doppelwege und den A* Algorithmus.

4.3.1 British Museum Procedure

Ein mögliches Verfahren dazu ist die British Museum Procedure. Bei diesem Algorithmus werden zuerst alle möglichen Pfade von Start zum Ziel gesucht und anschließend der kürzeste daraus ausgewählt. Die Suche nach allen möglichen Lösungspfaden kann mit Hilfe der Tiefen- oder Breitensuche erfolgen, indem man den Algorithmus geringfügig ändert. Die Suchschleife soll nicht abgebrochen werden, sobald eine Lösung gefunden wurde, sondern erst nach dem für alle möglichen Pfade überprüft wurde, ob sie zum Ziel führen. Diese Methode eignet sich jedoch nur für sehr kleine Graphen. Für größere Graphen ist der Aufwand zu groß alle Pfade zu betrachten.

4.3.2 Branch-and-Bound Search

Die Idee, die hinter der Branch-and-Bound Search steht, ist folgende: Angenommen man weiß von einem Pfad, daß er vom Start zum Ziel führt und möchte überprüfen, ob es sich um den optimalen Lösungspfad handelt, so reicht es, alle anderen Pfade nur soweit zu betrachten, bis deren Kosten größer werden, als die Kosten der Benützung des optimalen Pfades.

Die Umsetzung dieser Idee in einem Algorithmus sieht dabei wie folgt aus. Der Algorithmus erzeugt ausgehend vom Start-Node Pfade zu allen Nachbarn und berechnet die Kosten die eine Benützung des Pfades verursacht. Anschließend wird der Pfad ausgewählt, der die geringsten Kosten verursacht. Vom Endpunkt dieses Pfades aus werden nun alle Pfade erzeugt, die zu den Nachbarn dieses Nodes führen. Im nächsten Schritt werden die Kosten jedes neuen Pfades bestimmt. Aus der Menge aller Pfade wird nun der Pfad mit den geringsten Kosten ausgewählt und Pfade zu den Nachbarn des Endpunktes erzeugt. Diese Vorgehensweise wiederholt sich solange, bis ein Pfad zum Ziel-Node führt. Somit ist ein scheinbar optimaler Pfad gefunden. Um zu überprüfen, ob es sich dabei tatsächlich um den optimalen Pfad handelt, muß man alle anderen Pfade noch solange betrachten,

bis deren Kosten die des scheinbar optimalen Pfades überschreiten oder mit niedereren Kosten zum Ziel führen. Überschreiten alle Pfade die Kosten des scheinbar optimalen Pfades, weiß man, daß es sich bei dem scheinbar optimalen Pfad tatsächlich um den optimalen Pfad handelt. Findet man jedoch einen Pfad, der niedrigere Kosten verursacht, so entspricht dieser dem optimalen Pfad. Der in Abbildung 4.13 dargestellte Algorithmus arbeitet besonders effektiv, wenn sich falsche Pfade schnell hohe Kosten verursachen und sie somit nicht mehr weiter berücksichtigt werden müssen. Abbildung 4.14 zeigt die Schritte beim Aufbau des Suchbaums des Straßenkartenproblems von Abbildung 4.1. [Winston1993]

- Erzeuge einen Pfad, der nur aus dem Start-Node besteht und schreibe ihn in die Queue
- Solange bis der erste Pfad in der Queue zum Ziel-Node führt oder die Queue leer ist
  - Lösche den ersten Pfad aus der Queue. Ermittle den Endpunkt des Pfades und erzeuge je einen neuen Pfad zu allen Nachbarn dieses Endpunktes.
  - Entferne alle neuen Pfade, die Schleifen enthalten.
  - Füge die verbleibenden Pfade in Queue ein.
  - Sortiere die Queue aufsteigend, nach den Pfadkosten.
- Wenn der Zielknoten erreicht wurde, gib den Pfad aus.

Abbildung 4.13: Branch-and-Bound Search Algorithmus.

4.3.3 Branch-and-Bound Search mit Abschätzung der Restkosten

Die Effizienz des Branch-and-Bound Search Algorithmus läßt sich steigern, wenn man eine Abschätzung der Restkosten als Grundlage verwendet, um zu entscheiden, welcher Pfad als nächstes um einen Schritt zu allen möglichen Nachbarn erweitert werden soll. Man sagt auch, der Pfad wird expandiert. Existiert eine gute Schätzung der Restkosten, so stellt die Summe aus den Kosten des bisher gegangenen Pfades und die geschätzten Kosten bis zum Ziel, eine gute Schätzung für die Länge des Pfades vom Start zum Ziel dar. Zum leichteren Verständnis beziehen sich die Bezeichnungen in den weiteren Erklärungen auf das Straßenkartenproblem von Abbildung 4.1. Darum wird anstelle von Kosten, von Weglängen

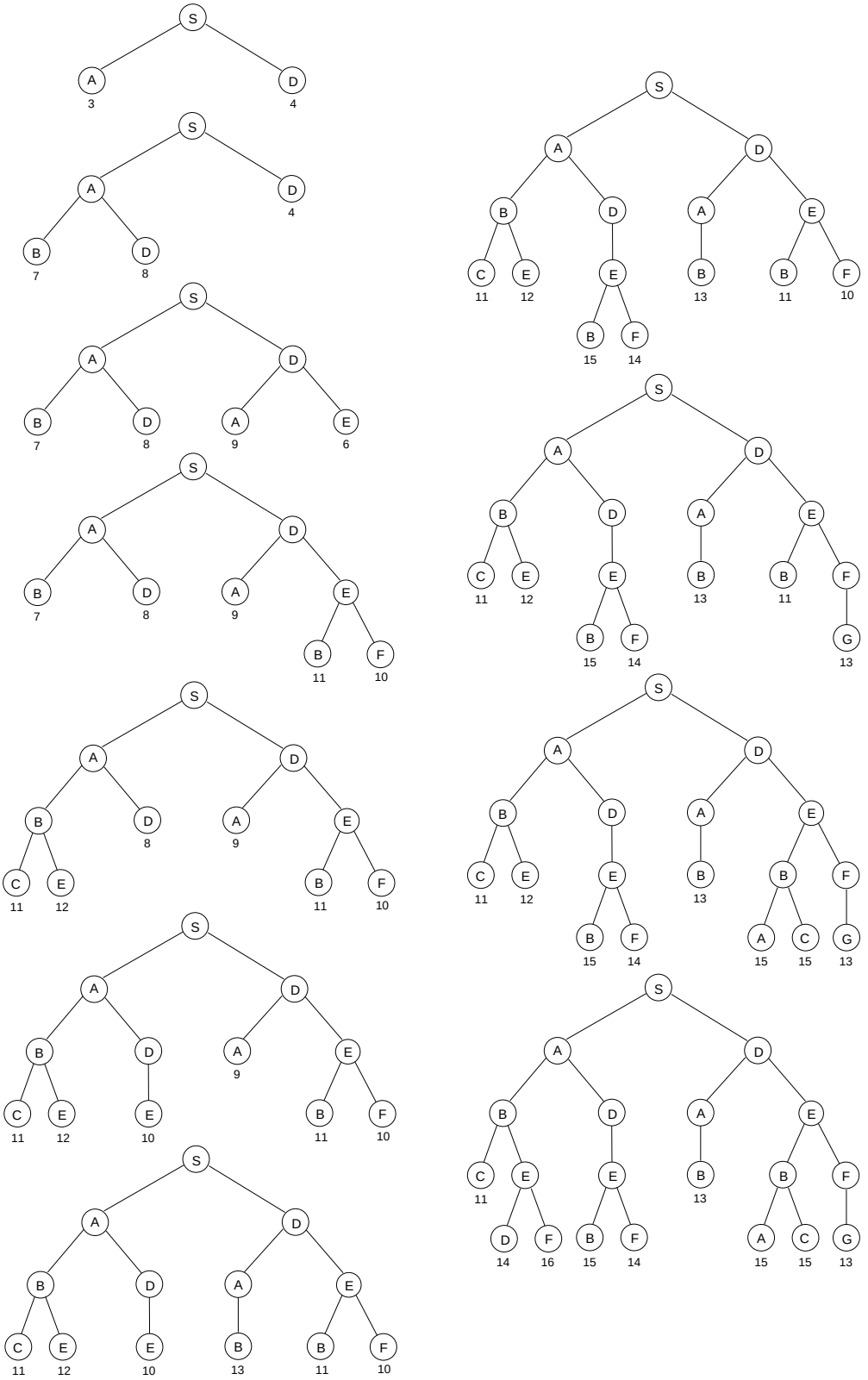

Abbildung 4.14: Branch-and-Bound Search Algorithmus angewandt auf das Straßenkartenprobelem.

- Erzeuge einen Pfad, der nur aus dem Start-Node besteht und schreibe ihn in die Queue
- Solange bis der erste Pfad in der Queue zum Ziel-Node führt oder die Queue leer ist
  - Lösche den ersten Pfad aus der Queue. Ermittle den Endpunkt des Pfades und erzeuge je einen neuen Pfad zu allen Nachbarn dieses Endpunktes.
  - Entferne alle neuen Pfade, die Schleifen enthalten.
  - Füge die verbleibenden Pfade in Queue ein.
  - Sortiere die Queue aufsteigend, nach der Summe der Pfadkosten und der geschätzten Kosten zum Ziel.
- Wenn der Zielknoten erreicht wurde, gib den Pfad aus.

Abbildung 4.15: Branch-and-Bound Search Algorithmus mit Abschätzung der Restkosten.

zwischen den Städten gesprochen. Dabei darf jedoch nicht vergessen werden, daß es sich bei den Kosten auch um Zeitaufwand, um Geldeinheiten, usw. handeln kann.

Der Algorithmus berechnet nun zu jedem Pfad die Länge des geschätzten Pfades zum Ziel und expandiert im nächsten Schritt jenen Pfad mit den geringsten geschätzten Länge. Dies geschieht solange, bis der Algorithmus beim Expandieren auf den Ziel-Node stößt. Dieser Pfad ist dann der optimale Lösungspfad, wenn die Schätzung der nachfolgenden Bedingung genügt. Die Schätzung des Restweges darf nie größer sein, als die tatsächliche Länge des Pfades zum Ziel. Wäre dies der Fall, so könnte der Suchalgorithmus dauerhaft vom optimalen Lösungspfad abgebracht werden. Dadurch kann man sich nicht mehr sicher sein, daß die Suche den optimalen Pfad als Ergebnis liefert. Unterschätzt die Schätzung den Restweg, im Extremfall mit der Länge Null, so wird dadurch lediglich die Auswahl der zunächst expandierten Pfade beeinflußt und die Effizienz sinkt, das Suchergebnis wird jedoch immer dem optimalen Pfad entsprechen. Der Algorithmus für die Branch-and-Bound Search mit Restkostenabschätzung ist in Abbildung 4.15 dargestellt.

Für das Straßenkartenproblem stellt die Luftlinie-Entfernung zum Ziel eine gute Schätzung für den Restwert dar. Die Entfernungen sind in Abbildung 4.8 dargestellt. Es ist auch sichergestellt, daß die Schätzung nie größer ist, als die

tatsächliche Entfernung zum Ziel, denn eine Straße kann nie kürzer sein, als die Luftlinie-Verbindung zwischen zwei Städten. Der schrittweise Aufbau des Suchbaumes ist in Abbildung 4.16 dargestellt. Der Branch-and-Bound Search Algorithmus mit Abschätzung der Restkosten arbeitet besonders effektiv, wenn eine gute Schätzung der Restkosten zur Verfügung steht.

4.3.4 Branch-and-Bound Search mit Eliminierung längerer Doppelwege

Der Branch-and-Bound Search Algorithmus mit Eliminierung längerer Doppelwege soll anhand von Abbildung 4.17 erläutert werden. Im ersten Schritt wurde der Start-Node expandiert und die beiden Pfade A-S und S-D sind entstanden. Da der Pfad S-A kürzer ist, wird dieser zuerst weiterbetrachtet. Es entstehen die Pfade S-A-B und S-A-D. Es stellt sich nun die Frage, ob es überhaupt sinnvoll ist, den Pfad S-A-D mit der Länge 8 weiter zu betrachten, da bereits ein Pfad bekannt ist, der mit der Länge 4 zum Node D führt (S-D). Dieser Teilpfad kann somit sicher nicht zum optimalen Lösungspfad beitragen.

Der Branch-and-Bound Search Algorithmus mit Eliminierung längerer Doppelwege basiert somit auf der Grundlage, daß er alle Pfade ignoriert, die zu einem Node führen, der auch durch einen anderen, kürzeren Pfad erreicht werden kann. Der Algorithmus dazu ist in Abbildung 4.18 dargestellt. Den schrittweisen Aufbau des Suchbaumes findet man in Abbildung 4.19.

A^* Algorithmus

Der A^* Algorithmus stellt einen Branch-and-Bound Search Algorithmus mit Abschätzung der Restkosten und Eliminierung längerer Doppelwege dar. Die Auswahl des nächsten expandierten Pfades erfolgt über die Restwertabschätzung. Zunächst werden längere redundante Doppelwege nicht mehr weiter berücksichtigt. Bezüglich der Restwertabschätzung gilt auch die Voraussetzung, daß der Schätzwert nie größer sein darf, als der tatsächlich verbleibende Weg zum Ziel. Der Algorithmus ist in Abbildung 4.20 dargestellt.

Zusammenfassung

Zusammenfassend kann man sagen, daß den British Museum Algorithmus nur für sehr kleine Suchprobleme anwendbar ist. Der Branch-and-Bound Search Algorithmus hat seine Stärken, wenn schlechte Pfade sehr schnell hohe Kosten verursachen. Ergänzt man diesen Algorithmus um die Restwertabschätzung, so steigt die Effizienz, je besser die Schätzung ist. Der Schätzwert darf jedoch nicht die

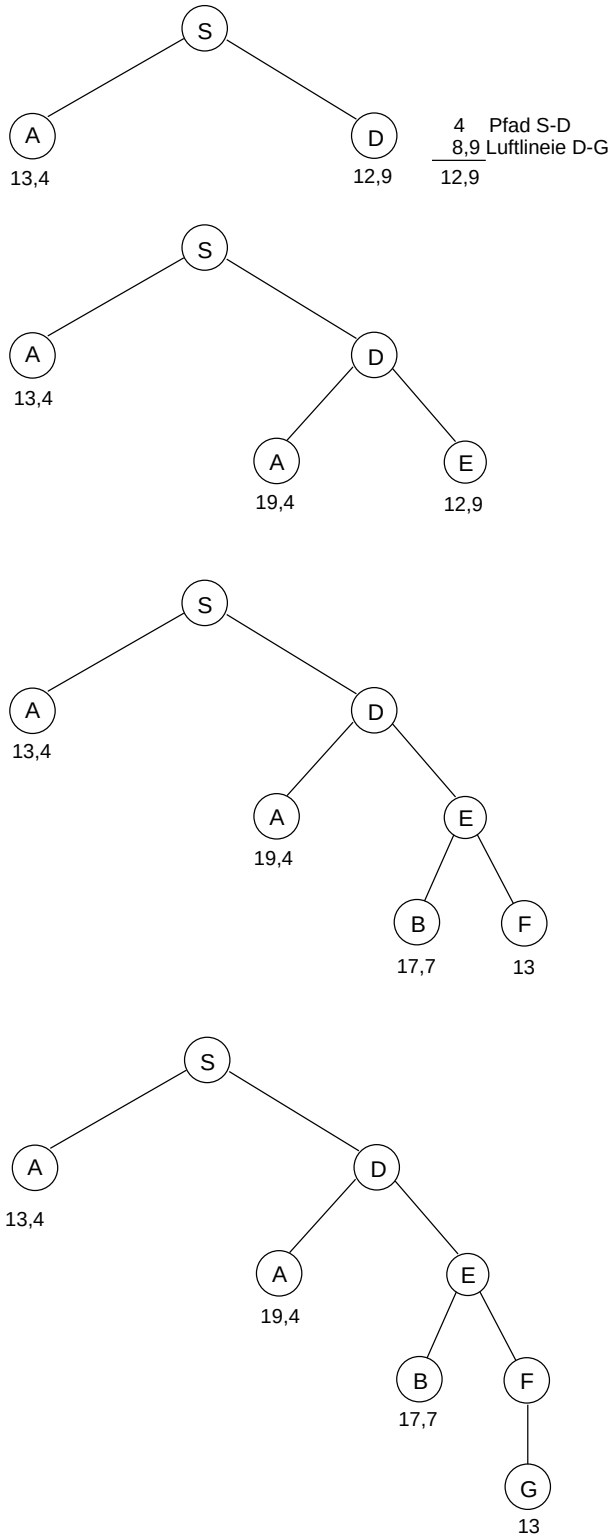

Abbildung 4.16: Branch-and-Bound Search mit Abschätzung der Restkosten angewandt auf das Straßenkartenprobelem.

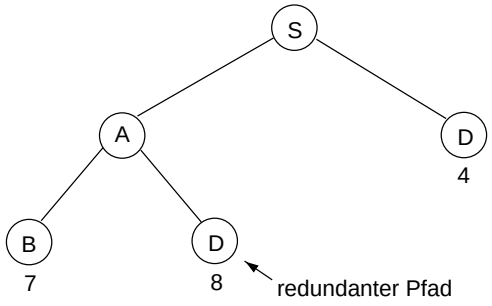

Abbildung 4.17: Branch-and-Bound Search Algorithmus mit Eliminierung längerer Doppelwege verfolgt nur den kürzesen Pfad zu einem Node weiter.

tatsächlich verbleibenden Kosten zum Ziel überschreiten. Der Branch-and-Bound Search Algorithmus mit Eliminierung längerer Doppelwege arbeitet besonders effektiv, wenn viele Pfade zu einem gleichen Node führen. Der A* Algorithmus sollte dann eingesetzt werden, wenn sowohl Branch-and-Bound Search Algorithmus mit Abschätzung der Restkosten und der Branch-and-Bound Search Algorithmus mit Eliminierung längerer Doppelwege ihre Stärken aufweisen.

- Erzeuge einen Pfad, der nur aus dem Start-Node besteht und schreibe ihn in die Queue
- Solange bis der erste Pfad in der Queue zum Ziel-Node führt oder die Queue leer ist
  - Lösche den ersten Pfad aus der Queue. Ermittle den Endpunkt des Pfades und erzeuge je einen neuen Pfad zu allen Nachbarn dieses Endpunktes.
  - Entferne alle neuen Pfade, die Schleifen enthalten.
  - Füge die verbleibenden Pfade in Queue ein.
  - Wenn zwei oder mehrere Pfade zu einem gleichen Node führen, lösche diese Pfade mit Ausnahme des Pfades, der den Node mit den minimalen Kosten erreicht.
  - Sortiere die Queue aufsteigend, nach den Pfadkosten
- Wenn der Zielknoten erreicht wurde, gib den Pfad aus.

Abbildung 4.18: Branch-and-Bound Search Algorithmus mit Eliminierung längerer Doppelwege.

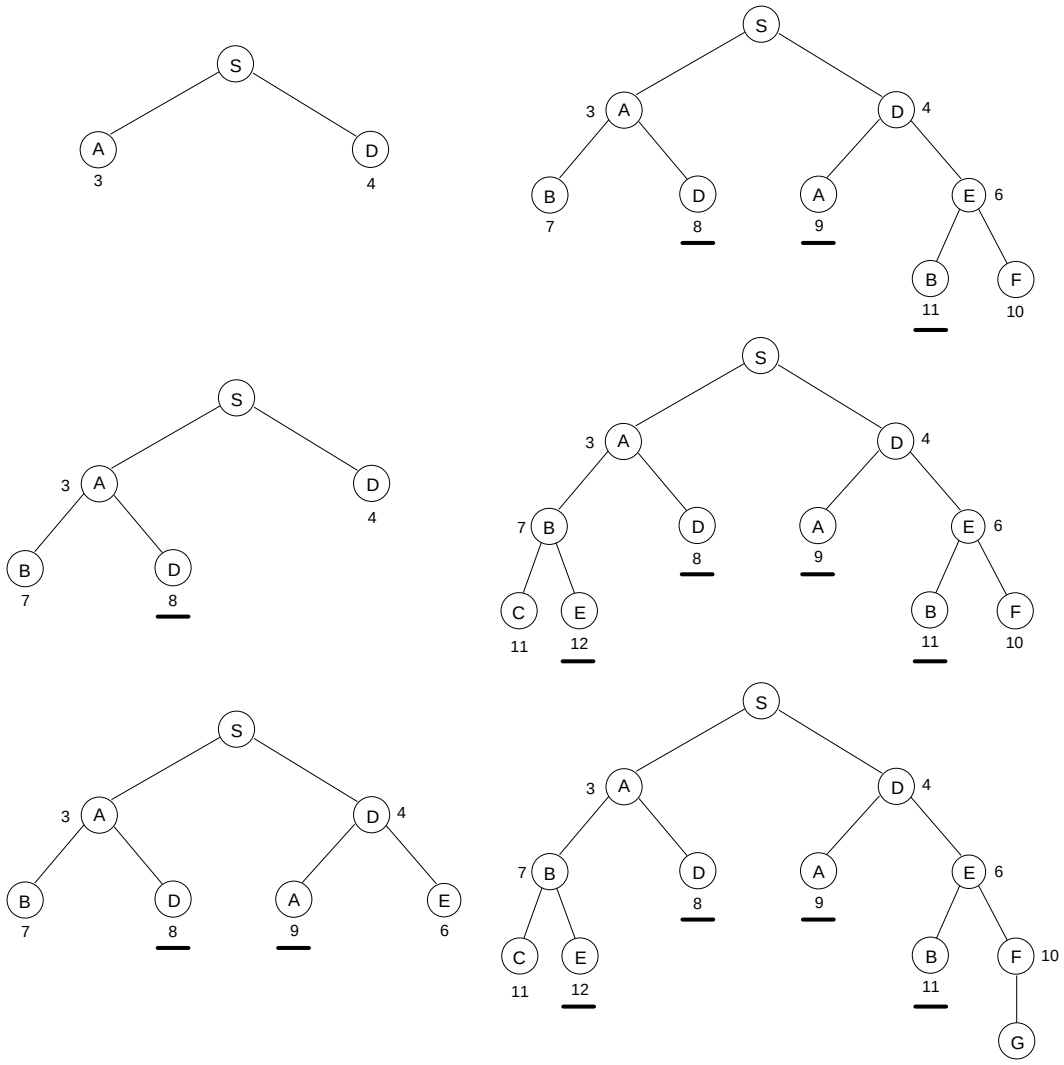

Abbildung 4.19: Branch-and-Bound Search Algorithmus mit Eliminierung längerer Doppelwege angewandt auf das Straßenkartenprobelem.

- Erzeuge einen Pfad, der nur aus dem Start-Node besteht und schreibe ihn in die Queue
- Solange bis der erste Pfad in der Queue zum Ziel-Node führt oder die Queue leer ist
  - Lösche den ersten Pfad aus der Queue. Ermittle den Endpunkt des Pfades und erzeuge je einen neuen Pfad zu allen Nachbarn dieses Endpunktes.
  - Entferne alle neuen Pfade, die Schleifen enthalten.
  - Füge die verbleibenden Pfade in Queue ein.
  - Wenn zwei oder mehrere Pfade zu einem gleichen Node führen, lösche diese Pfade mit Ausnahme des Pfades, der den Node mit den minimalen Kosten erreicht.
  - Sortiere die Queue aufsteigend, nach der Summe der Pfadkosten und der geschätzten Kosten zum Ziel.
- Wenn der Zielknoten erreicht wurde, gib den Pfad aus.

Abbildung 4.20: Der A* Algorithmus.

Kapitel 5

Spielbäume

Spielbäume und die dazugehörenden Algorithmen ermöglichen es, Computern eine bestimmte Art von Strategiespielen zu spielen. Bei den Spielen handelt es sich um Zweipersonenspiele, bei denen beide Spieler vollständig wissen, welche Züge gemacht wurden und welche noch möglich sind. Beispiele dafür sind Spiele wie Dame und Schach. Kartenspiele oder Backgammon bei denen der Zufall eine Rolle spielt, gehören nicht dazu [Nilsson1982].

Die aktuelle Brettsituation, die möglichen Züge und die daraus folgenden neuen Brettsituationen werden mit Hilfe eines Spielbaumes repräsentiert. Abbildung 5.1 zeigt einen Spielbaum, bei dem die Spieler bei jedem Zug zwischen zwei möglichen Zügen wählen können. Die Nodes in einem Spielbaum repräsentieren die Brettsituationen, die Links stehen für die erlaubten Züge, die zu neuen Brettsituationen führen [Winston1993].

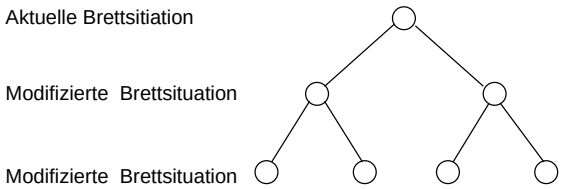

Abbildung 5.1: Beispiel eines Spielbaumes. Nodes repräsentieren Brettsituationen, Links gültige Züge.

Für ein komplexes Spiel ist es jedoch nicht möglich, den vollständigen Spielbaum aufzubauen. Für Schach wären dazu 10^{120} Nodes notwendig [Nilsson1982]. Darum baut man den Baum nur bis zu einer vertretbaren Ebene auf und bewertet anschießend die entstandenen Brettsituationen.

Um bewerten zu können, ob eine Brettsituation gut oder schlecht für einen Spieler ist, wird ein Situation-Analyzer benötigt. Der Situation-Analyzer gibt

abhängig von der Brettsituation einen Zahlenwert zurück. Vom Absolutbetrag des Zahlenwerts hängt es ab, wie gut eine bestimmte Situation ist. Hohe Zahlenwerte stehen dabei für eine günstige Brettsituation. Das Vorzeichen des Zahlenwerts drückt aus für welchen der beiden Spieler die Situation günstig ist. Ein Spieler hat somit das Ziel den Zahlenwert zu maximieren, der Maximierer, der andere ihn zu minimieren, der Minimierer. Daraus ergibt sich auch der Name des MiniMax-Algorithmus, der nun vorgestellt werden soll.

5.1 MiniMax Algorithmus

Der MiniMax Algorithmus baut einen Spielbaum bis zu einer bestimmten Ebene auf und ordnet jedem Blatt[1] mit Hilfe des Situation-Analyzer einen Zahlenwert zu (siehe Abbildung 5.2 oben). Bei der abgebildeten Situation ist der Maximierer am Zug. Beim nächsten Zug ist der Minimierer an der Reihe, usw. Welche Person in welcher Ebene am Zug ist, ist rechts vom Baum angegeben.

Der Maximierer hat das Ziel zu der Brettsituation mit der höchsten erreichbaren Bewertung zu gelangen. Der höchste Wert, der einem Blatt zugeordnet wurde, ist 8 im rechten Teilbaum. Entscheidet sich der Maximierer für diesen Teilbaum, so gibt er den Minimierer im nächsten Schritt die Möglichkeit sich zwischen den Werten -1 und 8 zu entscheiden. Dieser wird sich für die Brettsituation mit der Bewertung -1 entscheiden. Der Maximierer hat durch seinen Zug erreicht, daß er bei einer für ihn ungünstigen Brettsituation landet. Hätte er sich für den linken Teilbaum entschieden, hätte der Minimierer nur eine Auswahl zwischen Situationen, die mit 2 oder 7 bewertet wurden. Die beste Bewertung die der Maximierer somit nach zwei Zügen erreichen kann, ist die 2.

Der MiniMax Algorithmus beginnt seine Analyse eine Ebene über den Blättern. Befindet er sich in einer Minimierer Ebene, ordnet er dem Knoten das Minimum der Bewertungen seiner Kinder zu, bzw. in einer Maximierer Ebene das Maximum. Nachdem er dies für allen Knoten dieser Ebene erledigt hat, nimmt er diese Bewertung eine Ebene höher vor, wie in Abbildung 5.2 mitte und unten gezeigt wird. Der dem Root-Knoten zugeordnete Wert entspricht der Bewertung der Brettsituation, die der Spieler schlechtesten falls erreichen kann. Der schlechteste Fall tritt dann ein, wenn der Gegner die Brettsituationen gleich bewertet und bei der Auswahl seiner Zügen keinen Fehler macht. Der Algorithmus ist in Abbildung 5.1 dargestellt.

Da der Situations Analyzer eine sehr rechenaufwendige Aufgabe darstellt, sollte man sich überlegen, ob er überhaupt für jedes Blatt aufgerufen werden

[1] Blatt: Node in der untersten Baumebene, welcher keine Children besitzt.

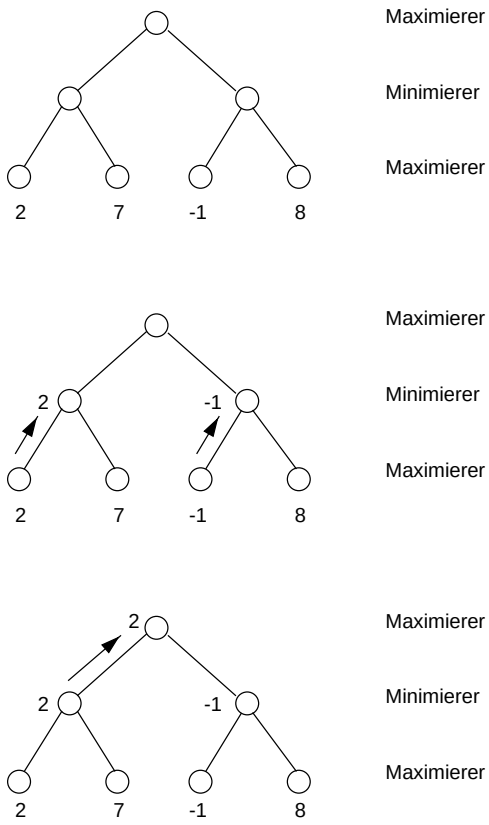

Abbildung 5.2: Bestimmung des bestmöglichen Zuges mit Hilfe des MiniMax-Algorithmus. [Winston1993]

```
function minimax(N) is
begin
  if N is a leaf then
    return the estimated score of this leaf
  else
    let N1, N2, .., Nm be the successors of N
    if N is a Min node then
      return min{minimax(N1), ..,minimax (Nm)}
    else
      return max{minimax(N1), .., minimax(Nm)}
  end
```

Abbildung 5.3: Der MiniMax Algorithmus.

muß. Der Alpha-Beta Algorithmus, der im nächsten Abschnitt behandelt wird, zeigt, das dies nicht nötig ist. [Winston1993]

5.2 Alpha-Beta-Algorithmus

Der Alpha-Beta-Algorithmus ist eine Verbesserung von MiniMax. Durch ihn ist es nicht mehr notwendig, für jedes Blatt eine Bewertung durchzuführen, es muß nicht einmal der vollständige Suchbaum aufgebaut werden. Welche Äste im Baum nicht aufgebaut werden müssen, soll mit Hilfe von Abbildung 5.4 erklärt werden.

In Abbildung 5.4 oben wurde der linke Teilbaum bereits aufgebaut und die Blätter bewertet. In der Minimierer-Ebene kann man schon erkennen, daß die beste Brettsituation, die der Minimierer erreichen kann, die Situation mit der Bewertung 2 ist. Diese Beobachtung ist beim Knoten mit „=2" vermerkt. Eine Ebene darüber weiß der Maximierer nun, daß die niedrigste Bewertung, die er erreichen kann, die 2 ist. Dies ist beim Knoten mit $\geq$2 vermerkt, siehe Abbildung 5.4 mitte.

Im nächsten Schritt wird begonnen den rechten Teilbaum aufzubauen. Sobald das erste Blatt bewertet ist, weiß der Minimierer, daß er zumindest zu einer Brettsituation kommen kann die mit -1 bewertet wurde. Beim Knoten wird diese Information mit $\leq$-1 vermerkt. Siehe Abbildung 5.4 unten. Der Maximierer eine Ebene höher kann daraus schließen, daß die höchste Bewertung die er mit die-

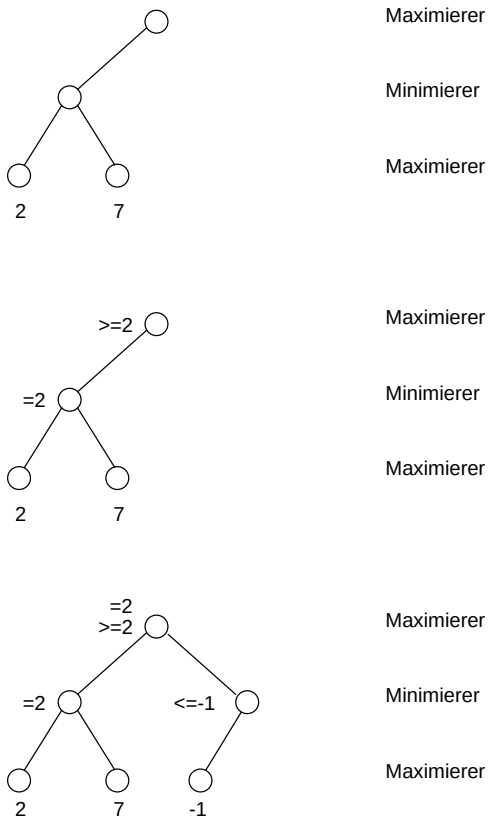

Abbildung 5.4: Beim Alpha-Beta Algorithmus werden nur jene Teile des Baumes aufgebaut, in denen noch die Möglichkeit besteht, einen besseren Zug, als die bisher bekannten, zu finden. [Winston1993]

sem Teilbaum erreichen kann, die -1 ist. Nun ist bei diesem Knoten aber schon vermerkt, daß es einen Teilbaum gibt, mit dem der Maximierer zur Bewertung 2 gelangt. Es macht also keinen Sinn diesen Teilbaum weiter aufzubauen.

Durch den Alpha-Beta Algorithmus werden nur jene Teile des Baumes aufgebaut, in denen die Möglichkeit besteht einen besseren Zug zu finden, als den Besten der zur Zeit bekannt ist. Das Prinzip das dahinter steckt lautet: Wenn man feststellt, daß ein Zug schlecht ist, schaue nicht nach, wie schlecht er wirklich ist. [Winston1993]

Bei der Implementierung des Algorithmus werden zwei Hilfsvariablen, alpha und beta, eingeführt. Diese Variablen haben folgende Eigenschaften:

- Die Bewertung eines Knoten kann nicht niederer als *alpha* und nicht höher als *beta* werden.
- Während der Abarbeitung des Algorithmus können sich die Werte für *alpha* und *beta* ändern, jedoch wird der Wert für *alpha* nie sinken, der für *beta* nie steigen.
- Der *alpha*-Wert eines Knoten ist nie niederer, als der *alpha* Wert seiner Vorgänger.
- Der *beta*-Wert eines Knoten ist nie höher, als der *beta* Wert seiner Vorgänger.
- Wenn ein Knoten auf einer Minimierer Ebene das letzte mal besucht wurde, entspricht seine Bewertung dem *alpha* Wert des Knotens.
- Wenn ein Knoten auf einer Maximierer Ebene das letzte mal besucht wurde, entspricht seine Bewertung dem *beta* Wert des Knotens.

Der Algorithmus ist in Abbildung 5.2 dargestellt. Der Aufruf der Funktion erfolgt mit $minimax\text{-}ab(Startknoten, -\infty, +\infty)$.

Bei den meisten Brettspielen haben die Spieler nicht beliebig lange Zeit, sich für einen bestimmten Zug zu entscheiden. Sie müssen innerhalb einer vorgegebenen Zeit einen Zug durchführen. Dies stellt die beiden bisher vorgestellten Algorithmen vor ein Problem. Es muß beim Start des Algorithmus festgelegt werden, bis zu welcher Tiefe der Baum aufgebaut werden soll. Unterschätzt man die Zeit für die Analyse und baut zuviele Ebenen des Spielbaumes auf, so ist der Algorithmus in der zur verfügungstehenden Zeit noch nicht fertig abgearbeitet. Der Computer verliert das Spiel, da er die Zeit überschreitet. Überschätzt man die Zeit für die Analyse, so geht Rechenzeit ,,verloren", in der man vielleicht einen besseren Zug hätte finden können. Um dieses Problem zu lösen und zu jeder Zeit über den bestmöglichen, bekannten Zug bescheid zu wissen, verwendet man daher die Technik des Progressiv Deepening. [Winston1993]

```
function minimax-ab(N, a, a)
begin
  set alpha value of N to a
  set beta value of N to b;
  if N is a leaf then
    return the estimated score of this leaf
  else
    if N is a Min node then
      for each successor Ni of N do
        let Val be minimax-ab(Ni, alpha of N, beta of N);
        set beta value of N to min{beta value of N, Val};
        if beta value of N <= alpha value of N then exit;
      od
      Return beta value of N;
    else
      for each successor Ni of N do
        let Val be minimax-ab(Ni, alpha of N, beta of N);
        set alpha value of N to max{alpha value of N, Val};
        if alpha value of N >= beta value of N then exit;
      od
      return alpha value of N;
end;
```

Abbildung 5.5: Der Alpha-Beta-Algorithmus.

5.3 Progressiv Deepening

Um über den bestmöglichen Zug zu jeder Zeit bescheid zu wissen, analysiert man beim Progressiv Deepening den Baum nach dem Aufbau jeder Ebene mit dem MiniMax-Algorithmus; zuerst Ebene 1, dann Ebene 2, usw. Nach dem Ablauf der zur Verfügung stehenden Zeit, wird die Berechnung in der untersten Ebene abgebrochen. Der bestmögliche bekannte Zug ist somit der, der sich aus der Analyse der vorletzten Ebene ergeben hat. Dies ist die letzte Ebene die fertig analysiert wurde. [Winston1993]

Auf den ersten Blick erscheint diese Vorgangsweise sehr zeitaufwendig. Doch die nachfolgende Analyse wird zeigen, daß sich der Mehraufwand für Progressiv Deepening in Grenzen hält. Bei der Analyse wird davon ausgegangen, daß der Situation-Analyzer für jeden Knoten die größten Kosten bei der Bewertung des Baumes verursacht. Daher kann man die Analyse auf die Betrachtung der Anzahl der Knoten beschränken. Gegeben ist ein Spielbaum mit dem Branching-Faktor b und der Tiefe d. In der untersten Ebene besitzt der Baum somit b^d Knoten. Die Anzahl der Knoten im gesamten Baum bis zur vorletzten Ebene $d-1$ kann mit folgender Formel berechnet werden:

$$b^0 + b^1 + \ldots + b^{d-1} = \frac{b^d - 1}{b - 1} \tag{5.1}$$

Bildet man das Verhältnis der Knoten in der letzten Baumebene zu allen Knoten im Baum darüber so erhält man:

$$\frac{b^d(b-1)}{b^d - 1} \approx b - 1 \tag{5.2}$$

Betrachtet man einen Baum mit dem Branching-Faktor $b = 16$, so sagt Gleichung (5.2) aus, daß der Aufwand den Baum bis zur vorletzten Ebene zu bewerten um $\frac{1}{15}$ geringer ist, als die letzte Ebene zu bewerten. Der für das Progressiv Deepening zusätzlich notwendige Aufwand hält sich somit in Grenzen. [Winston1993]

Kapitel 6

Expertensysteme

Zu Beginn dieses Abschnittes wird auf die Definition des Begriffes Expertensystem eingegangen. Es wird beleuchtet, aus welchen unterschiedlichen Arten das Wissen eines menschlichen Experten besteht und wie dieses in einem Computersystem umgesetzt werden kann. Weiters wird die Motivation diskutiert, Expertensysteme zu erstellen und deren Vor- und Nachteile beleuchtet. Die Anwendungsgebiete werden anhand von kurzen Beispielen vorgestellt. In weiterer Folge wird der Entstehungsprozeß eines Expertensystems und die daran beteiligten Personen sowie deren Funktionen erklärt. Am Ende dieses Abschnitts wird auf die Arbeitsweise und den Aufbau eines Expertensystems eingegangen. Dabei wird die Wissensrepräsentation, die zur Anwendung kommt, vorgestellt und auf die Techniken eingegangen, in welcher Weise das gespeicherte Wissen verarbeitet wird.

6.1 Definition von Expertensystemen

Die Erforschung von Expertensystemen stellt eine erfolgreiche Teildisziplin der künstlichen Intelligenz dar. Die Grundidee, die Expertensystemen zugrunde liegt, ist, daß Wissen hochspezialisierter Fachleute in einem Computersystem nachgebildet wird. Auf diese Art sollen menschliche Experten bei Routieneaufgaben unterstützt und entlastet werden. Ein weiteres Ziel ist es, das Wissen, das menschliche Experten durch lernen und jahrelange Erfahrung gesammelt haben, automatisiert verarbeitbar gespeichert wird und somit jederzeit reproduziert werden kann. Zusammengefaßt wird dies in der Definition nach [Puppe1991]:

> *„Expertensysteme sind Programme, mit denen das Spezialwissen und die Schlußflogerungsfähigkeit qualifizierter Fachleute auf eng begrenzten Aufgabengebieten nachgebildet werden soll."* [Puppe1991]

Dabei geht man von der Vorstellung aus, daß menschliche Experten ihre Problemlösungen aus Einzelkenntnissen zusammensetzten, die sie auswählen und in passender Anordnung anwenden. Ein Expertensystem benötigt daher detailierte Einzelkenntnisse über das Aufgabengebiet und Strategien, wie dieses Wissen zur Lösung eines Problems angewendet werden kann. Um ein Expertensystem zu erstellen, muß das Wissen von einem oder mehreren Experten formalisiert und im Computer repräsentiert werden. Die Form der Wissensrepräsentation, die sich dafür besonders eignet, ist die der logischen Wissensrepräsentation durch Regeln, die in Abschnitt 3.3.1 unter Einfache Fakten Regelsysteme vorgestellt wurde. Die Regeln haben die Form "Wenn X, dann Y"; z.B. Wenn die Temperatur zu niedrig ist, dann die Heizleistung erhöhen. [Puppe1991]

Zusätzlich zu dieser Wissensrepräsentation muß es in einem Expertensystem auch Möglichkeiten geben, das Wissen (Fakten und Regeln) zu interpretieren bzw. Schlußfolgerungen zu ziehen. Diese Aufgabe übernimmt die Problemlösungskomponente (Inferenzmaschine), die unabhängig vom zu bearbeitenden Wissen ist. Das heißt, das in den Regeln gespeicherte Expertenwissen legt nur fest, was in einer bestimmten Situation getan werden soll, in welcher Reihenfolge die Regeln zu Problemlösung verwendet werden, entscheidet die Inferenzmaschine. [Gottlob1990]

6.2 Zielsetzung und Motivation

Im vorigen Abschnitt wurde der Versuch unternommen, Expertensysteme zu definieren. In diesem Abschnitt sollen nun die Gründe betrachtet werden, Expertensysteme zu entwickeln und deren Zielsetzungen erläutert werden. Die Zielsetzungen für ein Expertensystemprojekt sind: [Gottlob1990]

- die Bereitstellung neuer Serviceleistungen, besonders im Dienstleistungsbereich;
- die Entwicklung eines neuen Produktes, entweder als eigenständiges Softwaresystem oder durch Integration eines Expertensystems in ein Analyse- oder Diagnosegerät;
- die Verbesserung von Qualität, Sicherheit, Produktivität und Arbeitsbedingungen – Hauptziele im Rahmen der industriellen Produktion; und
- die Verringerung von Fehleranzahl, Ausschuß und Ressourcenbedarf, d.h. der Versuch, den Produktionsprozeß besser in den Griff zu bekommen.

Die Motivation und die Gründe ein Anwendungsprojekt mit Hilfe der Unterstützung eines Expertensystems zu lösen, kann sowohl expertenbezogen als

auch produktbezogen gesehen werden. Aus der expertenbezogenen Sichtweise ergeben sich folgende Gründe: [Gottlob1990]

- der Experte ist mit Aufgaben überlastet, die für ihn Routine sind. Diese Routineaufgaben sollten ihm von einem Expertensystem abgenommen werden, damit er sich den schwierigen Problemen widmen kann;
- der Experte kann nicht vor Ort sein, etwa bei mangelndem Servicepersonal, oder bei Weltraum- und militärischen Projekten;
- es gibt nur einen Experten, der in der Zentrale sitzt. Man möchte jedoch sein Wissen auch in den Filialen verfügbar machen;
- die Anzahl und/oder die Komplexität der Probleme hat so zugenommen, daß der Experte überfordert ist;
- der Experte geht bald in Pension oder wechselt die Firma, man möchte aber sein Wissen nicht mit seinem Ausscheiden verlieren.

Aus der Sicht des zu entwickelnden Produktes ergeben sich folgende Gründe, bei der Realisierung ein Expertensystem zu implementieren: [Gottlob1990]

- um die Qualität eines Produktes zu erhöhen, liefert man das zugehörige Expertenwissen mit;
- die Problemstellung hat eine Komplexität, die intelligente Unterstützung bei der Problemlösung erfordert;
- die Sicherheit in kritischen Situationen wird erhöht;
- es werden Leistungen an bisher nicht erreichten Orten und/oder Tageszeiten (z.B. Nachts oder am Wochenende) ermöglicht.

6.3 Aufbau von Expertensystemen

Den oben angeführten Anforderungen an ein Expertensystem kommt das Konzept von wissensbasierten Systemen, das in Abschnitt 3.2.3 behandelt wurde, entgegen. Bei diesem Konzept existiert eine klare Schnittstelle zwischen dem gespeicherten Wissen und der Problemlösungskomponente. Es ist jedoch nicht zwingend vorgeschrieben, daß Expertensysteme nach diesem Konzept aufgebaut sind. Es wäre auch möglich, daß die Regeln wie bei konventionellen Programmen fix in den Programmzeilen zu codieren. Dies würde aber die Reihenfolge

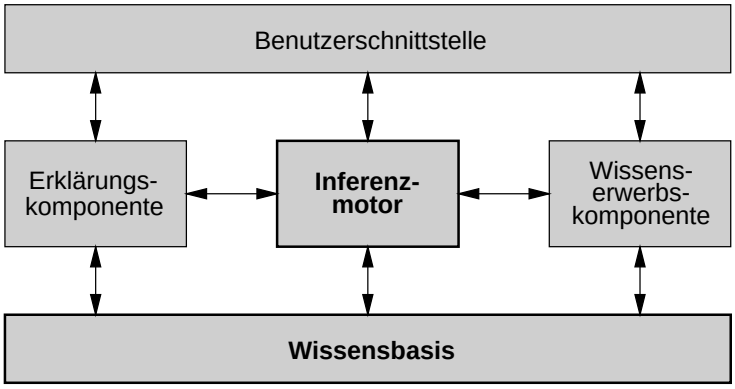

Abbildung 6.1: Aufbau eines Expertensystems. Die Wissensbasis und die Inferenzmaschine nehmen eine zentrale Rolle ein. [Gottlob1990]

der Abarbeitung der Regeln vorschreiben und die Vorteile der Modularität gehen verloren.

Die Hauptaufgabe eines Expertensystems ist es, Wissen zu verarbeiten. Aus diesem Grund nehmen die Wissensbasis und der Inferenzmotor im Aufbau eines Expertensystems eine zentrale Rolle ein, wie dies in Abbildung 6.1 dargestellt ist. Zusätzlich muß das System in der Lage sein, über eine Benutzerschnittstelle mit dem Anwender zu kommunizieren und die Ergebnisse zu präsentieren. Das System sollte auch die Fähigkeit besitzen, den Lösungsweg zu analysieren und Entscheidungen zu begründen und zu erklären. Weiters sollten Expertensysteme Möglichkeiten bereitstellen, die Wissensbasis zu erweitern. Nachfolgend sollen die Aufgaben der Komponenten eines Expertensystems erklärt werden, die in Abbildung 6.1 dargestellt sind. [Gottlob1990]

Wissensbasis: In der Wissensbasis ist das problembezogene Wissen eines Expertensystems gespeichert. Um die Vorteile der expliziten Wissenspeicherung auszunutzen, sollte in keiner anderen Komponente problembezogenes Wissen gespeichert sein. Der Inhalt der Wissensbasis kann grob unterschieden werden in *generisches Wissen*, das unabhängig vom aktuellen Anwendungsfall gespeichert ist, und dem *fallspezifischen Wissen*, das zur Lösung des aktuellen Anwendungsfalls notwendig ist. Lernfähige Systeme können fallspezifisches Wissen, nach der Lösung eines Problems, in den generischen Teil der Wissensbasis aufnehmen. Das generische Wissen umfaßt Fakten zum Problembereich, Wissen über deren Zusammenhänge und deren Schlußfolgerungsmechanismen, und Wissen über Strategien, wie das vorhandene Wissen eingesetzt werden kann.

Inferenzmotor: Der Inferenzmotor stellt die zentrale Problemlösungskompo-

nente dar. Hier werden die, aus der Problemstellung extrahierten Fakten entsprechend der Regeln verknüpft und auf neue Fakten geschlossen. Die Reihenfolge in der dies geschieht, wird von der Inferenzmaschine selbst festgelegt. Nach Beendigung des Schlußflogerungsprozesses werden die neugewonnenen Fakten von der Benutzerschnittstelle aufbereitet und als Lösung des Problems dem Anwender präsentiert.

Erklärungskomponente: Die Erklärungskomponente eines Expertensystems liefert Informationen über das Zustandekommen der Lösung. Typische Fragestellungen dabei sind:

- Wie wurde die Lösung eines Problems gefunden?
- Warum wird eine bestimmte Information vom System nachgefragt?
- Warum wurde eine bestimmte Lösung nicht gefunden?

Wissenserwerbskomponente: Die Wissenserwerbskomponente hat die Aufgabe, den Aufbau und die Erweiterung der Wissensbasis zu unterstützen. Sie stellt Funktionen zur Verfügung, die die Konsistenz und Vollständigkeit des gespeicherten Wissens zu überprüfen.

Benutzerschnittstelle: Die Benutzerschnittstelle unterscheidet zwei Arten der Interaktionen mit dem Expertensystem: Die Kommunikation mit dem Anwender, der nach der Lösung eines bestimmen Anwendungsproblems sucht, und der Kommunikation mit dem Knowledge Engineers, das ist die Person, die die Wissensbasis erstellt und wartet. Zu diesem Zweck werden in beiden Fällen die Fakten über die Problemlösung annähernd natürlichsprachlich aufbereitet oder Zusammenhänge graphisch dargestellt.

6.4 Arten von Wissen

Wie bei der Beschreibung der Wissensbasis angedeutet, benötigt ein Expertensystem, wie auch menschliche Experten Wissen, um eine gegebene Problemstellung zu lösen. Das Wissen läßt sich dabei in vier Kategorien einteilen: [Gottlob1990]

Fakten zu einem Problembereich sind zumeist in Lehrbüchern enthalten. Sie stellen die Basis dar, auf der das gesamte Teilgebiet aufbaut. Der menschliche Experte eignet sich dieses Wissen in seiner Ausbildung (Studium, Fachausbildung) an, bzw. sucht es bei Bedarf in Wissensspeichern, die meist in gedruckter Form (z.B. Bücher, Manuals, Datenblätter) oder auch elektronisch zur Verfügung stehen. Beispiele für Faktenwissen sind Naturgesetze oder mathematische Gesetze.

Wissen über die Zusammenhänge der Fakten beschreibt festgelegte Verfahrensweisen und deren Auswirkungen. Auch dieses Wissen ist großteils in Lehrbüchern zu finden, z.B. der physikalische Zusammenhang zwischen Luftdruck und Wetter.

Wissen über Schlußfolgerungsmechanismen und Heuristiken für die Lösung von Problemen, eignet sich der Experte durch langjähriges Arbeiten in einem bestimmten Problembereich an. Dieses Wissen stellt das eigentliche Erfahrungswissen menschlicher Experten dar und beinhaltet auch den Umgang mit unvollständigen Informationen und mit Informationen aus den Randbereichen des Aufgabengebietes. Dieses Wissen ist nicht in Lehrbüchern zu finden, wäre auch schwer darin darzustellen, und macht einen Experten deshalb so wertvoll für ein Unternehmen. Wie im nachfolgenden Abschnitt erarbeitet wird, kann Wissen dieser Art nur sehr bedingt in Expertensystemen umgesetzt werden.

Strategisches Wissen ist, wie das Wissen über Schlußfolgerungsmechanismen und Heuristiken, Erfahrungswissen, das sich der Experte im Laufe der Zeit aneignet. Es geht dabei darum, wie man ein Problem angeht, um effizient zu einer Lösung zu gelangen.

6.4.1 Menschlicher Experte versus Expertensystem

Der praktische Nutzen eines Expertensystems für die Wirtschaft besteht in der Möglichkeit, Experten bei Routinetätigkeiten zu entlasten bzw. einfache Probleme auch ohne den Einsatz von Experten zu lösen. Durch Expertensysteme wird es möglich, Expertenwissen automatisiert zu verarbeiten und zu duplizieren, z.B. an andere Firmen weiter zu verkaufen. Weiters besteht die Möglichkeit, Wissen über die Dienstzeit des Experten hinaus zu archivieren, wenn ein Experte in Pension geht oder die Firma wechselt. [Gottlob1990]

Im weiteren sollen die Vor- und Nachteile von Expertensystemen erarbeitet werden, in dem die Fähigkeiten von menschlichen Experten denen von Expertensystemen gegenübergestellt werden. Um ein Problem zu lösen, muß ein menschlicher Experte folgende Fähigkeiten besitzen: [Puppe1991]

- Das Problem verstehen
- Das Problem lösen
- Die Lösung erklären
- Randgebiete überblicken

- Seine Kompetenzen bei der Problemlösung einschätzen
- Neues Wissen erwerben und strukturieren

Dabei sind menschliche Experten auch dazu in der Lage, intuitiv zu handeln ohne ihre Entscheidungen begründen zu können und Probleme auch unter Verwendung von unvollständigem und unsicherem Wissen zu lösen [Gottlob1990].

Derzeitige Expertensysteme sind aber nur in der Lage Probleme zu lösen und die Lösung in begrenzten Umfang zu erklären. Die anderen oben erwähnten Fähigkeiten sind für die Ausübung der Tätigkeit eines Experten im allgemeinen genauso wichtig, können aber von den Expertensystemen, die zum Zeitpunkt des Erstellens der vorliegenden Arbeit existieren, nicht erfüllt werden. Würde z.B. ein menschlicher Experte nicht über die Grenzen und Randgebiete seines Spezialgebietes bescheid wissen und nicht in der Lage sein, seine Wissenslücken zu beurteilen, so würde man diesem Experten, nach der ersten darauf beruhenden Fehlentscheidung, nicht mehr vertrauen. [Puppe1991]

Die größte Schwierigkeit eines Expertensystems liegt im Verstehen des Problems. Während menschliche Experten über umfassende sensorische und verbale Fähigkeiten verfügen, um aus der Fülle der Daten die problemrelevanten Umstände herauszufiltern, vorzuinterpretieren und auf ihre Glaubwürdigkeit zu testen, erfordern Expertensysteme eine streng formalisierte Eingabe, deren Korrektheit kaum überprüft werden kann, die aber die Qualität der Problemlösung entscheidend mitbestimmen. Daher eignen sich Expertensysteme vor allem zum Einsatz in solchen Gebieten, bei denen die Datenerfassung wenig fehleranfällig ist, das Expertensystem selbst keine endgültigen Entscheidungen trifft, sondern in einen redundanten Entscheidungsprozeß eingebunden ist, und das Wissen des zu implementierenden Fachbereichs sehr schmal ist. In diesem schmalen Bereich kann das Wissen jedoch sehr tief strukturiert sein. Um Expertensystemen nicht die letzte Entscheidung zu überlassen und ihnen somit nicht eine zu große Verantwortung zu übertragen, werden Expertensysteme häufig dazu verwendet, Lösungen von menschlichen Experten auf ihre Richtigkeit zu überprüfen [Gottlob1990].

Aus den oben angeführten Gründen lassen sich Expertensysteme nach der Art der Interaktion mit der Umwelt einteilen. Die Einteilung ist in Abbildung 6.2 graphisch dargestellt. Dabei gliedert man Expertensysteme danach, ob der Austausch von Input bzw. Output direkt mit der Umwelt erfolgt oder ob die Interaktion mit der Umwelt über den Umweg eines Menschen stattfindet.

Grundsätzlich gilt für Expertensysteme, je autonomer das System, desto sicherer müssen die Entscheidungen sein. Daher sind *Steuernde Systeme* dazu gezwungen, bei Entscheidungen ihre Systemgrenzen zu überprüfen. Die Erklärungskomponente kann bei solchen Systemen unter Umständen entfallen. *Sensorgeführte Systeme* müssen in der Lage sein, Datenfehler zu erkennen und zu korrigieren. Bei

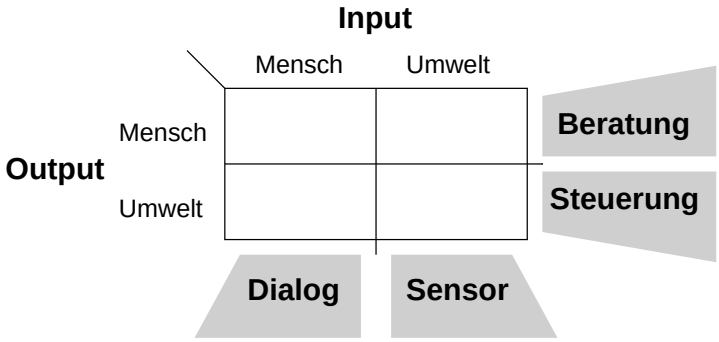

Abbildung 6.2: Möglichkeiten der Interaktion eines Expertensystems mit seiner Umgebung. [Gottlob1990]

der Erstellung eines *Dialoggeführten Systems* muß ein Modell der Benutzergruppe berücksichtigt werden, um damit festzulegen, welche Daten sinnvollerweise vom Benutzer erhoben werden können, bzw. welche Art der Erklärung für die Beratung notwendig ist. *Dialoggeführte Beratungssysteme* stellen den überwiegenden Teil der Expertensystemanwendungen dar. Nicht zuletzt ist dies darauf zurückzuführen, daß auf der Inputseite der Mensch für die Vorabverarbeitung der Daten sorgen muß, und auf der Outputseite der Expertensystementwickler die letztenscheidung und somit auch die Verantwortung gerne dem Menschen überläßt. [Gottlob1990]

Den oben angeführten Nachteilen von Expertensystemen im Vergleich zu menschlichen Experten stehen folgende Vorteile gegenüber, die den Expertensystemen zu wirtschaftlicher Bedeutung verholfen haben: [Schnupp1986]

Urteilsfähigkeit: Die Urteilsfähigkeit eines Expertensystems ist immer gleich, das bedeutet, Expertensysteme kennen keinen Einfluß von Streß oder anderen emotionalen Faktoren. Ermüdung und Langeweile, wie sie bei Menschen bei Überwachungsaufgaben vorkommen, sind für Expertensysteme kein Problem. Ein Expertensystem kann daher auch unter extremen Bedingungen eingesetzt werden.

Wissenstransfer: Bei der Erstellung eines Expertensystems ist es notwendig, daß das Wissen von Experten explizit dargestellt und dokumentiert wird. Dieses Wissen kann somit auch anderen Leuten (Laien) zum Lernen zu Verfügung gestellt werden. Dies kann z.B. durch die Erklärungskomponente des Systems geschehen.

Verbreitung: Expertenwissen kann mit Hilfe eines Expertensystems viel leichter verbreitet werden. Dazu ist nur das Kopieren des Systems auf eine anderen Rechner nötig.

Kosten: Da nur die Entwicklung eines Expertensystems größere Kosten verursacht, ein fertiges Expertensystem praktisch kostenlos dupliziert werden kann, ist der Einsatz von Expertensystemen billiger als die Ausbildung eines menschlichen Experten.

6.5 Anwendungsgebiete von Expertensystemen

In der Geschichte der künstlichen Intelligenz wurde der Weg von der Suche nach allgemeinen Problemlösern hin zu stark problemspezifischen Ansätzen beschritten. Es wurde erkannt, daß für unterschiedliche Problemtypen unterschiedliche Lösungsansätze notwendig sind [Russell1995]. Expertensysteme sind die Konsequenz aus dieser Entwicklung. Ihr Ziel ist es, Probleme auf einem stark eingeschränkten Anwendungsgebiet zu lösen. Dabei können folgende Problemtypen unterschieden werden [Puppe1991]:

Interpretation: Ableitung einer Situationsbeschreibung aus Sensordaten. (z.B. Geologie - Aus den Daten von Probebohrungen wird die Struktur der geologischen Formation ermittelt)

Diagnostik: Ableitung von Systemfehlern aus Beobachtungen; Erkennen von Ursachen. (z.B. Erstellen von medizinischen Diagnosen.)

Überwachung: Vergleich von Beobachtungen mit Sollwerten. (z.B. Patient Monitoring in der Medizin - Das System erhält laufend Daten des Patienten und vergleicht diese mit Sollwerten. Bei einer Abweichung schlägt das System Alarm.)

Steuerung: Vergleich von Beobachtungen mit Sollwerten und Veranlassen von Aktionen. (z.B. Patient Monitoring in der Medizin - Ähnlich der Vorgehensweise bei der Überwachung, jedoch setzt das System selbst Handlungen, um den kritischen Zustand zu beseitigen.)

Design: Konfiguration von Produkten unter bestimmten Voraussetzungen. (z.B. Chemie - Herstellung komplexer organischer Moleküle, integrierte Schaltungen, usw.)

Planung: Entwurf einer Folge von Aktionen zum Erreichen eines Ziels. (z.B. Technik - Zusammenstellung eines Reperaturplanes bei einem Maschinenschaden) In der Produktionsplanung lassen sich dadurch die vorhandenen Ressourcen optimal einsetzen.

Vorhersage: Ableitung von möglichen Konsequenzen gegebener Situationen. (z.B. Wirtschaft - Ausgehend von der derzeitigen geopolitischen Lage wird der zukünftige Bedarf an Rohöl geschätzt)

Es zeigt sich, daß mehrere der obengenannten Problemtypen mit den selben Problemlösungsstrategien gelöst werden können. Somit ist es sinnvoll Expertensysteme nach ihren Strategien einzuteilen und die Problemtypen den Strategien zuzuordnen. Durch diese Einteilung erhält man einen Ansatzpunkt, mit welcher Strategie man an die Lösung eines konkreten Problems herangehen soll. [Puppe1991] teilt die Problemlösungsstrategien nach folgenden drei Typen ein:

Diagnostik: Das Ziel der Diagnose ist es, ein bekanntes Muster, wie z.B. ein Krankheitsbild oder einen Fehler, wieder zuerkennen. Dabei wird die Lösung aus einer vorgegebenen Menge von Alternativen ausgewählt.

Konstruktion: Bei der Konstruktion wird die Lösung aus kleinen Bausteinen zusammengesetzt. Es gibt jedoch zuviele mögliche Kombinationen, um aus der Menge aller vorgegebener Alternativen auszuwählen. So kann z.B. bei der automatisierten Programmierung nicht aus der Menge aller möglichen Programme ausgewählt werden. Stattdessen sind bestimmte Fakten im System Modulen zugeordnet, die entsprechend kombiniert werden. Das Expertensystem hat somit die Aufgabe gezielt nach Fakten der Aufgabenstellung zu fragen, um die notwendigen Module so zu kombinieren, daß die Aufgabenstellung erfüllt ist.

Simulation: Die Simulation unterscheidet sich von der Diagnose und der Konstruktion dadurch, daß kein vorgegebenes Ziel erreicht werden soll, sondern nur die Auswirkungen von Handlungen und Entscheidungen simuliert werden sollen. Es werden somit von einem Ausgangszustand die möglichen Folgezustände hergeleitet.

6.5.1 Praktische Anwendungen von Expertensystemen

Die Entwicklung von Expertensystemen begann im Jahre 1965. Nach über 10 Jahren der Forschung begannen Expertensysteme ab den 80er Jahren an wirtschaftlicher Bedeutung zu gewinnen. Im folgenden sind einige Meilensteine in der Geschichte von Expertensystemen angeführt [Puppe1991][Gottlob1990]:

- **DENTRAL** wurde als eines der ersten Systeme an der Stanford University, Lindsay entwickelt und ist ein System zur Strukturanalyse organischer Substanzen. Die zu interpretierenden Daten werden aus dem Massenspektrogramm gewonnen.
- **MYCIN** wurde ebenfalls an der Stanford University entwickelt und ist ein System zur Diagnose und Therapie von bakteriellen Infektionskrankheiten

des Blutes. In MYCIN ist das Wissen in über 450 Regeln gespeichert. MYCIN zeichnet sich zusätzlich durch seine klare Trennung zwischen Wissensrepräsentation und Inferenzmaschine aus. Die Entwicklung dieser Technik machte es möglich, daß in weiterer Folge das medizinische Wissen aus MYCIN entfernt wurde und die Inferenzmaschine unter dem Namen **EMYCIN** als Grundlage für andere Expertensysteme diente. EMYCIN war somit die erste Expertensystem-Shell.

- **PROSPECTOR** ist ein geologisches System zur Interpretation von Meßdaten aus Probebohrungen. Ziel ist es herauszufinden, ob ein bestimmtes Mineral in einer Region vorhanden ist oder nicht.
- **XCON** war bei Digital im Einsatz und diente zur Konfiguration von VAX Rechneranlagen. Ausgehend von der Bestellung einer Rechneranlage stellte XCON sicher, daß alle erforderlichen Komponenten vorhanden waren und sorgte für das korrekte Zusammenspiel der Komponenten inklusive der Verkabelung. XCON baute auf mehr als 2000 Regeln auf und war das erste erfolgreiche System dieser Größenordnung.

6.6 Entwicklungsstufen eines Expertensystems

Der Wissenserwerb umfaßt die Identifikation, Formalisierung und Wartung des Wissens, das ein Expertensystem benötigt, um Probleme lösen zu können und stellt zur Zeit den Falschenhals bei der Entwicklung von Expertensystemen dar. Der Erfolg eines Expertensystem hängt entscheidend vom Umfang und der Qualität der Mitarbeit der beteiligten Experten ab, an die hohe Anforderungen gestellt werden. Aufbau und Wartung einer Wissensbasis erfordern, daß sowohl objektive Fakten aus Lehrbüchern als auch subjektives Expertenwissen eingebracht wird. Es sollen möglichst vielfältige Wissensquellen zur guten Abdeckung des Fachbereiches verwendet werden. [Gottlob1990]

6.6.1 Beteiligte Personen am Wissenserwerb

Wie bereits oben erwähnt, sind ein oder mehrere Personen notwendig, um Wissen für den Aufbau eines Expertensystems zur Verfügung zu stellen. Dieses Wissen muß formalisiert und in maschinengerechter Form dargestellt werden. Da die meisten Fachexperten aber keine Informatiker sind, fällt diese Aufgabe dem sogenannten Knowledge Engineer zu. Der Knowledge Engineer stellt somit das Bindeglied zwischen den menschlichen Experten und dem Computer dar.

Die Aufgabe des Knowledge Engineer ist es nun, dem Experten sein Wissen möglichst vollständig zu entlocken. Dieses Wissen muß danach strukturiert und

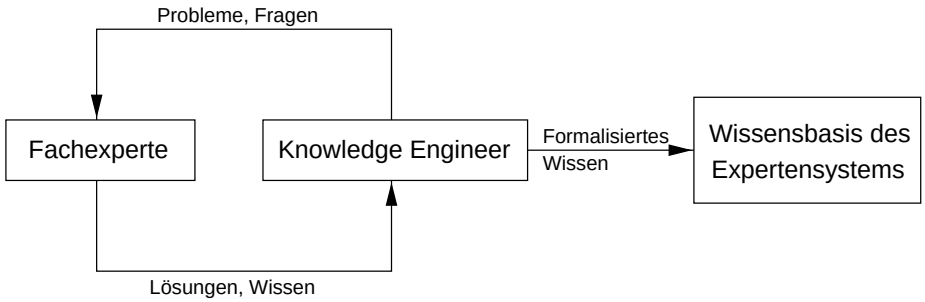

Abbildung 6.3: Der Knowledge Engineer hat die Aufgabe, das Wissen, das er vom Fachexperten erhält, zu formalisieren, damit es in der Wissensbasis des Expertensystems gespeichert werden kann.

formal dargestellt werden. Abbildung 6.3 veranschaulicht diesen Vorgang. Das Hauptproblem an dieser Vorgangsweise ist es, möglichst viel Wissen zu extrahieren. Für die Befragung der Experten durch den Knowledge Engineer haben sich bestimmte Interviewtechniken herausentwickelt. [Puppe1991]

Interviewtechniken dienen dem Wissenstransfer vom Experten zum Knowledge Engineer. Das Ergebnis des Interviews sind immer verbale Aufzeichnungen, die anschließend vom Knowledge Engineer interpretiert werden müssen. In weiterer Folge formalisiert der Knowledge Engineer das Wissen und stellt es in maschinengerechter Form dar. Interviewtechniken können nach zwei Kriterien aufgeteilt werden: Nach der Art des Interviews und nach der Art der Testfälle, deren Lösung während des Interviews beobachtet und kommentiert wird. Die wichtigsten Interviewtypen sind: [Puppe1991]

- Unstrukturiertes (traditionelles) Interview: Während der Experte über das Wissen spricht oder Testfälle löst, stellt der Wissensingenieur mehr oder weniger spontane Fragen.
- Laut Denken Modell: Der Experte denkt laut, während er ein Problem löst und erläutert die einzelnen Schritte.
- Introspektion: Der Experte beschreibt von sich aus wie er einen Fall löst oder welche Problemlösungsstrategie er benutzt. Im Gegensatz zum Laut Denken Modell beschreibt er bei der Introspektion eine Zusammenfassung seines Problemlösungsverhalten, nachdem er einen Fall gelöst hat.
- Strukturiertes Interview: Protokolle, die mit einer der anderen Interviewmethoden bereits erstellt wurden, werden von demselben oder von einem anderen Experten kommentiert und eingeschätzt.

Während das unstrukturierte Interview und die Introspektion dem Wissensingenieur helfen, mit dem Problembereich vertraut zu werden, bekommt man mit dem Laut-Denken-Protokoll die zuverlässigsten Aussagen über die Problemlösungsstrategien, die der Experte tatsächlich benutzt. Ein strukturiertes Interview ist zur Validierung gewonnener Daten und zum Auffüllen von Lücken sinnvoll. [Puppe1991]

Ein wichtiges Kriterium von Testfällen, vor allem für Laut-Denken-Protokolle, ist die Ähnlichkeit des Falles mit der realen Problemsituation des Experten. Ein anderes Kriterium ist der Schwierigkeitsgrad des Falles. Daraus ergeben sich folgende Arten von Testfällen: [Puppe1991]

- Typische Fälle, die der Experte routinemäßig löst.
- Fälle mit begrenzter Informationsangabe: Dabei werden dem Experten bestimmte, normalerweise vorhandene Informationen vorenthalten, um die Bedeutung von Einzelinformationen herauszufinden.
- Fälle mit Beschränkung der Verarbeitungskapazität: Mögliche Beschränkungen sind die Zeit, die dem Experten zur Verfügung gestellt wird, oder bestimmte Fragestellungen, auf die sich der Experte konzentrieren soll.
- Schwierige Fälle, die der Experte nicht routinemäßig lösen kann: Zu schwierigen Fällen wird man übergehen, wenn die Basisproblemlösungsstrategien bereits identifiziert sind.

6.6.2 Schwierigkeiten beim Wissenserwerb

Obwohl der Begriff Knowledge Engineer, auch Wissensingenieur, nahelegt, daß die Extraktion des Wissens von Experten eine im Prinzip gut verstandene Tätigkeit sei, trifft eher das Gegenteil zu, sodaß man eher von „Wissenskünstlern" sprechen sollte. [Puppe1991]

Eine der Hauptursachen ist die unvermeidliche Unvollständigkeit der verbalen Daten von Experten. Gründe dafür sind: [Puppe1991]

- Experten vergessen meist viele wichtige Faktoren zu nennen, da sie diese für selbstverständlich halten oder das Wissen nicht durch entsprechende Reize aktiviert wird.
- Komplexere Wissensbereiche, vor allem bildhaftes Wissen, können verbal kaum adäquat beschrieben werden.
- Teile des Wissens können unbewußt sein.

- In der Sprache wird viel Wissen durch Referenz auf als bekannt vorausgesetztes Wissen kommuniziert. Dieses Wissen muß der Knowledge Engineer auf Grund seines Verständnisses des Problembereiches ergänzen. Dieser Vorgang ist sehr problematisch.
- Experten können aus vielen Gründen nicht dazu bereit sein, ihr Wissen preiszugeben.
- Viele Experten haben Schwierigkeiten, ihre Vorgehensweise zu erklären. Das ist auch für Laut-Denken-Protokolle kritisch, die nur dann einen Wert haben, wenn durch das laute Denken das Problemlösungsverhalten des Experten nicht wesentlich beeinflußt wird.

6.6.3 Entwicklungsphasen eines Expertesystems

Als beste Vorgangsweise zur Erstellung eines Expertensystems erweist sich eine evolutorische, inkrementelle Entwicklungsstrategie, die von einfachen zu komplexen Problemstellungen führt, wobei die Organisation und Repräsentation des Systems ständig verbessert und erweitert werden. Es wird möglichst rasch ein einfaches, lauffähiges System hergestellt, um durch Experten-Feedback den weiteren Entwicklungsfortgang in geeignete Bahnen zu lenken. Der Entwicklungskreislauf eines Expertensystems ist in Abbildung 6.4 dargestellt. Diese Vorgehensweise nennt man Rapid Prototyping oder Incremental Development. [Gottlob1990]

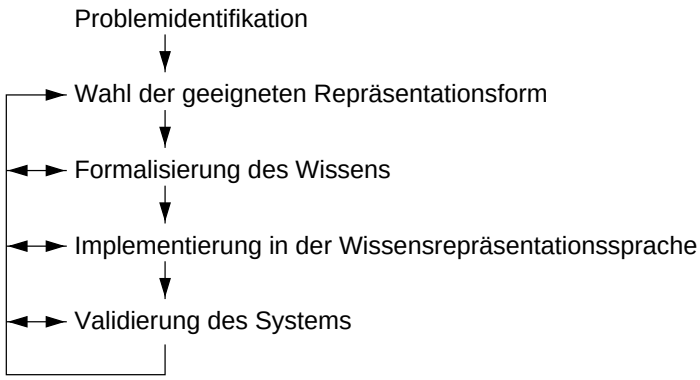

Abbildung 6.4: Der Entwicklungskreislauf eines Expertensystems. Es wird möglichst schnell ein Prototyp erstellt, der anschließend immer weiter verfeinert und erweitert wird. [Gottlob1990]

Beim Rapid Prototyping ist darauf zu achten, daß am Ende jedes Schritts der Prototyp mit den in der Planung festgelegten Vorgaben übereinstimmt. Falls dies nicht der Fall ist, muß der vorhergegangene Schritt wiederholt werden. Entfällt

diese Kontrolle oder wird sie nur ungenügend genau durchgeführt, pflanzen sich Fehler aus vorhergegangenen Schritten eventuell bis hin zum fertigen System fort und es muß eine entsprechend große Anzahl von Schritten im Entwicklungskreislauf zurückgegangen werden. Eine genaue Kontrolle des in Entwicklung befindlichen Systems mit den Vorgaben aus der Planung läßt die Zyklen im Entwicklungskreislauf klein bleiben und hilf somit, die Entwicklung effizient zu halten.

In der nachfolgenden Auflistung werden die Entwicklungsphasen bei der Erstellung eines Expertensystems detailierter betrachtet: [Schnupp1986]

1. Erstes Design der Wissensbasis

   - Problemidentifikation: Definition von Zielen, Einschränkungen, Ressourcen, der teilnehmenden Personen und ihrer Rollen.
   - Konzeption: Detailierte Beschreibung der Problemstellung und wie sie in Unterprobleme zerlegt werden kann. Beschreibung der Elemente zur Problemlösung in Form von Hypothesen, Fakten und Lösungsstrategien.
   - Formalisierung: Wahl einer Wissensrepräsentation im Computer für die in der Konzeption identifizierten Elemente. Hier kommen zum ersten Mal Implementierungsüberlegungen zum Tragen.

2. Implementation und Test eines Prototyps

   Ist einmal eine geeignete Wissensrepräsentation gewählt, kann mit der Implementierung eines Prototyps begonnen werden, der einen Teilbereich des endgültigen Systems umfaßt. Die richtige Auswahl des Bereiches ist von großer Bedeutung. Dieser muß Repräsentativ für das Gesamtsystem, aber einfach zu testen sein.

   Sobald der Prototyp akzeptable Ergebnisse produziert, kann er auf detailliertere Varianten des Zentralproblems erweitert und mit komplexen Fällen getestet werden.

3. Verfeinerung und Generalisierung der Wissensbasis

   Diese Phase kann sehr viel Zeit beanspruchen, bis das gewünschte Expertenniveau erreicht ist. Die Grundstrukturen liegen hier bereits fest, es wird eigentlich ,,nur" mehr das System mit detailliertem, verfeinertem Wissen erweitert.

4. Implementierung der Mensch-Maschine Schnittstelle

   In dieser Phase wird eine dem Problem entsprechende komfortable Benutzerschnittstelle entwickelt. Die bisher verwendete Benutzerschnittstelle diente dem Knowledge Engineer zu Testzwecken und ist deshalb in viele

Fällen nur sehr sporadisch ausgeführt. Weiters erfolgt in dieser Phase, die Erstellung der Dokumentation, der Wartungs- und Schulungsunterlagen.

5. Einführung des Systems in das Unternehmen

   Das Expertensystem wird in dieser Phase in die vorhandene Informations-Technologie (IT) des Unternehmen eingebunden. Dazu zählt nicht nur die Installation auf den Arbeitsplatzrechnern, sondern auch die Schulung der zukünftigen Benutzer. Erfahrungen und Hinweise der Benutzer sollen aufgezeichnet werden und bei der laufenden Verbesserung und Verfeinerung des Expertensystems berücksichtigt werden.

Wie generell bei der Einführung von neuer Informations Technologie in einem Unternehmen muß auch bei der Einführung eines Expertensystems der zukünftige Benutzer frühzeitig in den Entwicklungsprozeß miteingebunden werden. Dies gilt speziell für die Einführung eines Expertensystems, da viele der zukünftigen Benutzer keine Vorstellung von dem Begriff ,,Expertensystem" haben und somit eine große Hemmschwelle vor dem System besitzen, mit dem sie in Zukunft arbeiten sollen. Eine frühe Integration des Anwenders in den Entwicklungsprozeß hilft dabei diese Hemmschwellen abzubauen. Zudem kann der Knowledge Engineer dadurch wertvolle Hinweise über die Problemsichtweise des Anwenders gewinnen.

6.7 Wissensverarbeitung in Expertensystemen

In diesem Abschnitt soll erläutert werden, in welcher Form Wissen in Expertensystemen gespeichert wird und wie das gespeicherte Wissen verknüpft wird, um auf neues Wissen zu schließen. Anhand von Beispielen werden dabei zwei grundlegende Methoden der Wissensverarbeitung in Expertensystemen, die Vorwärtsverkettung und die Rückwärtsverkettung, vorgestellt. [Puppe1991]

6.7.1 Wissensspeicherung

Da Experten Wissen oft in Form von Regeln formulieren, wird in Expertensystemen meist eine logische Wissensrepräsentation durch Fakten und Regeln verwendet. Regeln bestehen aus einer Vorbedingung und einer Aktion. Vorbedingungen bestehen wiederum aus der Verknüpfung von ein oder mehreren Fakten. Nachfolgend sind zwei Beispiele für Regeln angegeben:

Regel 1	Wenn	*Nackensteife* und
		hohes Fieber und
		Bewußtseinstrübung zutreffen
	Dann	*besteht der Verdacht auf Meningitis*
Regel 2	Wenn	*Verdacht aud Meningitis* besteht
	Dann	*Nimm sofort Antibiotika*

In Regel 1 sind ,,Nackensteife", ,,hohes Fieber" und ,,Bewußtseinstrübung" die verknüpften Fakten und ,,besteht der Verdacht auf Meningitis" die Aktion.

Wenn man die Aktionen der beiden Beispielregeln betrachtet, stellt man fest, daß es sich dabei um zwei unterschiedliche Arten von Aktionen handelt. Bei ,,besteht der Verdacht auf Meningitis" handelt es sich um eine Hypothese, die dann wahr ist, wenn alle Fakten zutreffen. Man spricht von Implikation[1] oder Deduktion[2]. In Regel 2 entspricht die Aktion ,,Nimm sofort Antibiotika" einer Handlung. [Puppe1991]

Zusammenfassend kann man sagen, die Regeln eines Expertensystems bestehen aus: [Winston1993]

- **Vorbedingungen** (engl. antecedents), die aus der Verknüpfung ein oder mehrerer Fakten zusammengesetzt sind,
- und aus ein oder mehreren **Aktionen** (engl. conclusions). Bei den Aktionen unterscheidet man zwei Arten: [Winston1993]
  - Implikationen oder Deduktionen, mit denen der Wahrheitsgehalt einer Hypothese hergeleitet wird; z.B. Regel 1. Expertensysteme deren Aktionen Implikationen oder Deduktionen sind nennt man *Deduction Systems*.
  - Handlungen, mit denen ein Zustand verändert wird; z.B. Regel 2. Expertensysteme deren Aktionen Handlungen sind, nennt man *Reaction Systems*.

Zur graphischen Veranschaulichung kann eine Regel ähnlich eines Schaltsymbols gezeichnet werden. Dies ist in Abbildung 6.5 dargestellt.

Die Aufteilung des Wissens in möglichst kleine ,,Wissensstücke", den Regeln, macht eine Wissensbasis modular und damit relativ einfach veränderbar. Es ist

[1] Implikation: Logische Folgebeziehung [LF]

[2] Deduktion: Ableitung von Schlüssen aus bereits bewiesenen Schlüssen oder Aussagen mithilfe logischer Schlussregeln [LF]

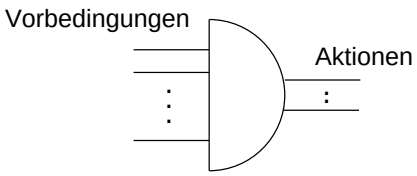

Abbildung 6.5: Veranschaulichung einer Regel als Schaltsymbol. Die Eingänge des Schaltsymbols entsprechen den verknüpften Fakten in der Vorbedingung, die Ausgänge entsprechen den Aktionen.

auch relativ leicht möglich, diesen Grundaufbau der Regeln für anwendungsspezifische Notwendigkeiten zu erweitern. So ist es z.B. möglich, Regeln zur Darstellung von unsicherem oder unvollständigem Wissen um Unsicherheitsangaben oder Ausnahmen zu erweitern.

Um die Strukturierung und die Überschaubarkeit der Wissensbasis zu erhöhen, ist es zumeist möglich zusammengehörende Regeln, das sind Regeln, die einen bestimmten Teil des Problems behandeln, zu sogenannten Wissensinseln zusammenzufassen, das heißt, ein Teilproblem wird von einer Wissensinsel gelöst und kann unter Umständen andere Wissensinseln dabei aktivieren oder von anderen Wissensinseln aus aktiviert werden. Oft ist es auch möglich, die Wissensinseln in eigene Teil–Wissensbasen zu legen, die nur dann in den Computer geladen werden, wenn das zugehörige Teilproblem gerade behandelt wird. Diese Strukturierung reduziert die im Computer aktuell zu untersuchenden Regeln und steigert so die Performance eines Expertensystems. [Gottlob1990]

Nach der Betrachtung der Wissensspeicherung soll nun betrachtet werden, wie diese Regeln intern abgearbeitet werden, um mit den vorhandenen Fakten zu neuen Schlußfolgerungen zu kommen. Es gibt zwei prinzipielle Arten der Regelverarbeitung: Die Vorwärtsverkettung und die Rückwärtsverkettung.

6.7.2 Wissensverarbeitung: Vorwärtsverkettung

Bei der Vorwärtsverkettung (forward chaining) leitet der Regelinterpreter alle Schlußfolgerungen her, die aus den, dem Expertensystem bekannten Fakten, herleitbar sind, das heißt, das System prüft alle Regeln, deren Vorbedingungen erfüllt sind und arbeitet diese Regeln ab. Zu einer Regel die abgearbeitet wird und deren Vorbedingung erfüllt ist, sagt man auch die Regel „feuert". Danach prüft das System wieder, welche Regeln abgearbeitet werden können. Dies soll anhand eines Beispiels, das in Abbildung 6.6 dargestellt ist, veranschaulicht werden. [Winston1993]

In Abbildung 6.6 ist ein Regelbaum dargestellt, der aus fünf Regeln besteht.

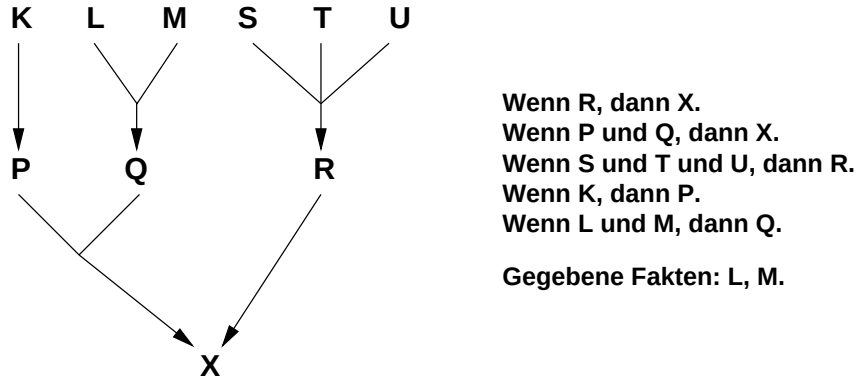

Abbildung 6.6: Beispiel eines Regelsystems.

Mittels der gegebenen Fakten L und M wird nun im Forward Chaining Verfahren versucht, etwas zu beweisen. Das System hat kein vorgegebenes Ziel, sondern sieht sich alle Regeln durch und schaut nach, welche abgearbeitet werden können. In dem Beispiel kann nur eine einzige Regel feuern, nämlich, wenn L und M erfüllt sind, dann schließe Q. Das heißt, Q wird als neues Faktum erschlossen und zu den gegebenen Fakten L und M hinzugefügt. Danach kann keine weitere Regel ausgeführt werden, was dazu führt, daß die Vorwärtsverkettung aufhört zu arbeiten.

6.7.3 Wissensverarbeitung: Rückwärtsverkettung

Während man mit der Vorwärtsverkettung nur Schlußfolgerungen aus einer vorgegebenen Faktenmenge beziehen kann, eignet sich ein rückwärtsverkettender Regelinterpreter (Backward Chaining) auch zum gezielten Erfragen noch unbekannter Fakten. Ein Backward Chaining Regelinterpreter startet mit einem vorgegebenen Ziel. Wenn das Ziel nicht in der Menge der bekannten Fakten vorkommt, entscheidet der Regelinterpreter zunächst, ob es abgeleitet werden kann oder ob es erfragt werden muß. Ein Faktum kann dann abgeleitet werden, wenn zumindest eine Regel existiert, in der das Faktum auf der rechten Seite, der Seite der Aktion, vorkommt. Existiert keine solche Regel, bleibt dem System nur die Möglichkeit, dieses Faktum vom Anwender zu erfragen. [Winston1993]

Im Falle der Ableitung werden alle Regeln abgearbeitet, in deren Aktionsteil das Ziel enthalten ist. Wenn bei der Überprüfung der Vorbedingungen einer Regel ein Parameter unbekannt ist, wird ein Unterziel zur Bestimmung dieses Parameters generiert und der Backward Chaining Mechanismus zur Bestimmung dieses Unterziels rekursiv herangezogen. Das Endergebnis ist die Bestimmung eines Wertes für das vorgegebene Ziel und für alle Unterziele, die Evaluierung der

relevanten Regeln und das Stellen der notwendigen Fragen. Die Rückwärtsverkettung enthält implizit eine Dialogsteuerung, wobei die Reihenfolge der gestellten Fragen von der Reigenfolge der Regeln zur Herleitung eines Parameters und von der Reihenfolge der Aussagen in der Vorbedingung einer Regel abhängt. Je präziser das Ziel, desto kleiner der Suchbaum von zu überprüfenden Regeln und zu stellenden Fragen. Als Beispiel für die Veranschaulichung dient wieder das Regelsystem aus Abbildung 6.6.

Beim Backward Chaining wird ein vorgegebenes Ziel aktiv bewiesen. In unserem Fall soll es das Faktum X sein. Es existieren zwei Wege zum Bewiesen von X. Der eine führt von X über R nach S, T und U. Der zweite führt von X nach P und Q und dann nach K bzw. L und M.

Nehmen wir an, daß der Regelinterpreter zuerst den ersten Weg zum Beweisen von X durchsucht. Es gibt eine Regel, die X beweisen kann, nämlich „Wenn R dann X". R ist nicht in der Menge der bekannten Fakten, deshalb wird als neues Ziel R etabliert und der Backward Chaining Mechanismus auf R angewendet. Auch für R existiert eine Regel, nämlich „Wenn S und T und U, dann R". Weder S,T noch U sind in der Menge der bekannten Fakten, deshalb muß der Regelinterpreter S,T und U ermitteln. Da aber für diese Fakten keine Regeln zu deren Ermittlung zur Verfügung stehen, muß der Regelinterpreter diese Fakten vom Benutzer erfragen. Werden alle Fakten S ,T und U vom Benutzer als gegeben eingegeben, so wird R zu wahr und in letzter Konsequenz auch X zu wahr. Damit ist X bewiesen und der Regelinterpreter kann aufhören zu arbeiten, da er das vorgegebene Ziel beweisen konnte.

Wird nun eine der Fakten S, T oder U vom Benutzter als nicht bekannt eingegeben, kann R nicht bewiesen werden und aus dem ersten Suchweg auch X nicht als wahr bewiesen. Somit muß der Regelinterpreter auch noch den zweiten Weg über P, Q und K, L und M zum Beweisen des Faktums X durchlaufen. Dies geschieht in derselben Weise wie dies beim ersten Weg gezeigt wurde. Das heißt, zum Beweisen von X müssen P und Q bekannt sein. Q kann aus den gegebenen Fakten L und M (sind schon in der Menge der gegebenen Fakten) zu wahr ermittelt werden. P wird durch Erfragen des Faktums K vom Benutzer ermittelt. Wird das Faktum K als gegebenes Faktum vom Benutzer angegeben, so wird P zu wahr. Aus den Fakten P und Q kann somit X zu wahr bewiesen und in die Menge der bewiesenen Fakten aufgenommen werden.

6.7.4 Vorwärtsverkettung versus Rückwärtsverkettung

Während in Reaction Systems immer die Vorwärtsverkettung verwendet wird, kann ein Deduction System entweder mit Vorwärtsverkettung oder Rückwärtsverkettung arbeiten. Welche Art der Verkettung nun tatsächlich verwendet wird,

hängt von der Art der Anwendung ab. [Winston1993]

Für die Vorwärtsverkettung wird man sich entscheiden, wenn viele Fakten sehr wenigen Regeln gegenüber stehen, oder alle möglichen Fakten vorgegeben sind und man alle daraus ableitbaren Schlußfolgerungen wissen möchte. [Winston1993]

Wenn die gegebenen oder eruierbaren Fakten zu einer großen Anzahl von Schlüssen führen, aber die Anzahl der Wege zum gewünschten Ziel klein ist, sollte man auf Rückwärtsverkettung zurückgreifen. Würde man Vorwärtsverkettung anwenden, würden viel Regeln abgearbeitet, deren Ergebnis schlußendlich nicht weiter benötigt wird. Ein Anwendungsfall für Rückwärtsverkettung ist, wenn keine oder wenige Fakten bekannt sind und man ein oder mehrere Hypothesen beweisen möchte. [Winston1993]

6.7.5 Konfliktlösungstrategien

Wie oben gezeigt wurde, kann es vorkommen, daß mehrere Regeln gleichzeitig feuerbereit sind. Da aber immer nur eine Regel feuern kann, muß eine Regel aus dieser Menge der feuerbereiten Regeln ausgewählt und bearbeitet werden. Die Menge der feuerbereiten Regeln bezeichnet man als Konfliktmenge und die Auswahl einer bestimmten Regel aus dieser Menge als Konfliktlösungsstrategie. Bei der Konfliktlösungsstrategie unterscheidet man folgende Methoden: [Winston1993]

- Auswahl nach der Reihenfolge
  - Die erste anwendbare Regel feuert (Trivialstrategie)
  - Die aktuellste Regel feuert, d.h. die Regel, deren Vorbedingungen sich auf möglichst neue Einträge in der Faktenmenge bezieht.
- Auswahl nach der syntaktischen Struktur der Regel
  - Die spezifischste Regel feuert, d.h. die Regel, deren Vorbedingung die einer anderen Regel als Vorbedingung vorkommt und noch noch mit zusätzlichen Aussagen verknüpft ist.
  - Die syntaktisch größte Regel feuert, d.h. die, die meisten Aussagen in der Vorbebingung enthält
- Auswahl mittels Zusatzwissen
  - Die Regel mit der höchsten Priorität feuert. Dazu muß jeder Regel eine Priorität, die z.B, als Zahl repräsentiert sein kann, zugeordnet werden.
  - Zusätzliche Regeln, sogenannte Meta-Regeln steuern den Auswahlprozeß.

Der Konfliktlösungsstrategie kommt speziell bei Reaction Systemen eine besonders große Rolle zu. So können in vielen Fällen bestimmte Handlungen erst dann ausgeführt werden, wenn andere Handlungen bereits abgeschlossen sind. So kann man z.B. ein Auto erst starten, wenn man den Zündschlüssel in das Zündschloß gesteckt hat. In Deductioin Systemen entsteht durch das Feuern einer Regel nur ein neues Faktum, darum spielt die Reihenfolge, in der die Regeln feuern, in Bezug auf das Ergebnis keine Rolle. Die Reihenfolge der Abarbeitung der Regeln hat jedoch einen direkten Einfluß auf die Geschwindigkeit, mit der ein bestimmtes Ziel bewiesen werden kann. Anschaulich wird dies, wenn man sich vorstellt, es feuern alle jene Regeln zuerst, die keine Fakten liefern, die notwendig sind, um das geforderte Ziel zu beweisen. Die Konfliktlösungsstrategie hat somit einen direkten Einfluß auf die Performance eines Expertensystems. [Winston1993]

6.8 Certainty Factor

Abschließend soll in diesem Kapitel die Möglichkeit betrachtet werden, unsicheres Wissen mit der Hilfe von Regeln zu repräsentieren und zu verarbeiten. Auf die Quellen von Unsicherheit wurde bereits in Abschnitt 3.8 eingegangen. In diesem Abschnitt sollen nun die Certainty Factors, als Möglichkeit unsicheres Wissen in Fakten Regelsystemem zu berücksichtigen, vorgestellte werden.

Beim Prinzip der Certainty Factors handelt es sich um ein intuitives Konzept, welches in engen Zusammenhang mit dem Expertensystem MYCIN entwickelt wurde. Dabei unterscheidet man zwei grundsätzlich verschiedene Arten von Certainty Factors: [Gottlob1990][Heinsohn1999]

Dynamischer Certainty Factor: Der dynamische Certainty Factor wird dazu verwendet, die Sicherheit von Fakten, die nicht als definitiv wahr oder falsch bekannt sind, quantitativ zu bewerten. Dem Faktum wird dabei ein Zahlenwert aus dem Intervall $[-1, 1]$ zugeordnet. Der Zahlenwert -1 entspricht dabei definitiv falsch, der Wert 1 definitiv richtig.

Meist basieren diese quantitativen Bewertungen von Fakten auf einem bestimmten Hintergrundwissen, welches in diesem Zusammenhang als Evidenz E bezeichnet wird. Im Laufe der Bearbeitung des Problems fließt neues Wissen in Form von Fakten in die Wissensbasis des Systems ein. Im Rahmen dieses Prozesses ändert sich die Evidenz E und somit unter Umständen auch die Certainty Factors, die den Fakten zugeordnet sind. Der Certainty Factor $CF(F \mid E)$ entspricht somit der Sicherheit des Faktums F auf Basis der Evidenz E. Da sich, wie oben erläutert, die Certainty Factors im Zuge der Wissensverarbeitung laufend verändern, spricht man von dynamischen Certainty Factors.

Fester Certainty Factor: Der feste Certainty Factor tritt in Zusammenhang mit Regeln auf. Mit seiner Hilfe wird die unsichere Abhängigkeit zwischen der Vorbedingung (Prämisse) und der Aktion (Konklusion) ausgedrückt. Die Vorbedingung kann dabei auch aus der komplexen Verknüpfung von unsicheren Fakten bestehen. Ein Beispiel für eine solche Regel ist:

Wenn *Nackensteife* und
hohes Fieber und
Bewußtseinstrübung zutreffen
Dann *besteht der Verdacht (0.7) auf Meningitis*

Diese Regel drückt die bedingte Sicherheit aus, daß es sich bei der Krankheit um Meningitis handelt. Der Certainty Factor $CF(H \mid F_1 \wedge F_2 \wedge F_3) = 0.7$ repräsentiert die bedingte Sicherheit der Hypothese H unter der Bedingung, daß die Fakten F_1, F_2 und F_3 wahr sind. Die festen Certainty Factors werden bei der Erstellung der Regelbasis von den Fachexperten den Regeln zugeordnet.

Die wesentlichen Arbeitsschritte eines Expertensystems, welches mit unsicheres Wissen mit der Hilfe von Certainty Factors berücksichtigt, können folgendermaßen zusammengefaßt werden: [Heinsohn1999]

- Die Abarbeitung der Regen erfolgt nach dem in Abschnitt 6.7.2 vorgestellten Schema. Die dynamischen Certainty Factors der bekannten Fakten werden dabei vom Benutzer in das System eingegeben.
- Im Fall einer komplexen zusammengesetzten Prämisse wird aus den Certainty Factors der einzelnen Fakten der Certainty Factor der zusammengesetzten Prämisse berechnet.
- Ist der dynamische Certainty Factor der Prämisse bekannt, kann unter Einbeziehung des festen Certainty Factors der Regel die Regelanwendung erfolgen. Der Sicherheitsfaktor der Hypothese wird berechnet und stellt das zum jetzigen Zeitpunkt über die Hypothese bekannte Wissen dar.
- Ein und die selbe Hypothese wird unter Umständen durch mehrere Regeln bewertet. Das bedeutet die Hypothese kommt in mehreren Regeln als Aktion vor. Die Anwendung dieser Regeln führt somit auch zu mehreren Certainty Factors, die die Sicherheit derselben Hypothese bewerten. Den Vorgang, der diese Einzelbewertungen zusammenfaßt, bezeichnet man als *parallele Kombination.*

Anhand eines einfachen Beispiels sollen in weiterer Folge die oben angeführten Schritte erläutert werden. Zudem werden die dazu notwendigen mathematischen Formeln vorgestellt. Als Beispiel soll die nachfolgende Regelbasis dienen: [Heinsohn1999]

$$\begin{array}{lrcl} \text{Regel 1:} & A_1 \wedge A_2 \wedge A_3 & \xrightarrow{0.7} & H \\ \text{Regel 2:} & B_1 \vee B_2 & \xrightarrow{0.4} & H \\ \text{Regel 3:} & C & \xrightarrow{-0.8} & H \end{array}$$

Als erste anwendbare Regel wird die Regel 1 angenommen. Regel 1 kann angewendet werden, wenn die Certainty Factors der Prämissenelemente A_1, A_2 und A_3 bekannt sind. Auf der Basis der einzelnen Certainty Factors $CF(A_i \mid E)$ kann der Certainty Factor $CF(A_1 \wedge A_2 \wedge A_3 \mid E)$ der zusammengesetzten Prämisse berechnet werden. E ist die Evidenz zu den Bewertungen der Fakten A_i. Es erfolgt die Regelanwendung, die zu dem Wert $CF(H^1 \mid E)$ führt. H^1 ist hier die Bewertung der Hypothese H nach der erfolgten Anwendung der ersten Regel. Analog erfolgt die Anwendung von Regel 2 auf der Basis der Bewertungen $CF(B_i \mid F)$ und die Berechnung von $CF(H^2 \mid F)$. F ist die Evidenz zu den Bewertungen der Fakten B_i.

Als Ergebnis der Anwendung von Regel 1 und 2 können nun die Certainty Factors $CF(H^1 \mid E)$ und $CF(H^2 \mid F)$ der Hypothese H parallel kombiniert werden und erhält somit das zusammengefaßte Ergebnis $CF(H^3 \mid E, F)$.

Auf analoge Weise kann Regel 3 angewendet werden und in die bereits berechneten Werte einbezogen werden. Man erhält abschließend den Certainty Factor $CF(H \mid E, F, G)$ wobei G die der Bewertung von C zugrundeliegende Evidenz ist.

Der im Rahmen der Abarbeitung der Regeln erstellte Und-Oder-Kombinationsgraph ist in Abbildung 6.7 dargestellt.

6.8.1 Das mathematische Modell

Der Certainty Factor gibt ein Maß für das Vertrauen in den Wahrheitsgehalt eines Faktums an. Da es jedoch in vielen Fällen einfacher ist ein Maß für das Vertrauen bzw. das Mißtrauen in eine Aussage anzugeben, als direkt den Certainty Factor zu bestimmen, basiert die in MYCIN getroffene Definition auf zwei weiteren Größen: dem Maß des Mißtrauens (measure of desbelive) MD und dem Maß des Vertrauens (measure of belive) MB. Wobei beide Maße wie folgt verstanden werden können: [Heinsohn1999]

$$\begin{array}{lcl} MB(h \mid E) & = x: & \text{„Im Fall der Evidenz E wächst das Vertrauen,} \\ & & \text{daß } h \text{ wahr ist um } x\text{.“} \\ MD(h \mid E) & = x: & \text{„Im Fall der Evidenz E wächst das Mißtrauen,} \\ & & \text{daß } h \text{ wahr ist um } x\text{.“} \end{array}$$

Der Certainty Factor CF wird dann über die normierte Differenz der Maße MB und MD berechnet:

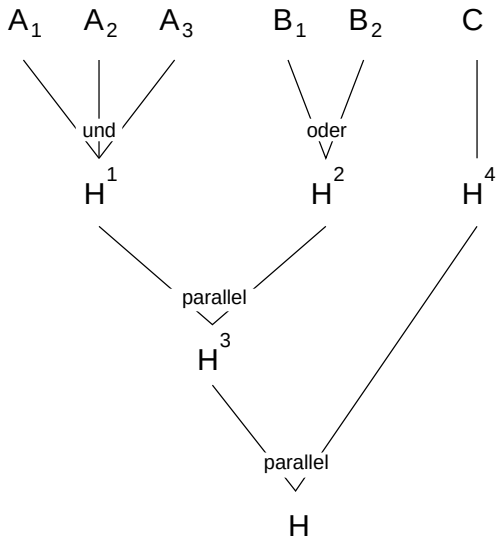

Abbildung 6.7: Der sich aus der Regelbasis ergebende Und-Oder-Kombinationsgraph. H^i bezeichnet im Graph den berechneten Certainty Factor der Hypothese H zum Zeitpunkt i. [Heinsohn1999]

$$CF(h \mid E) = \frac{MB(h \mid E) - MD(h \mid E)}{1 - \min\{MB(h \mid E), MD(h \mid E)\}} \tag{6.1}$$

Zusätzlich zu den Werten $MB(h \mid E)$, $MD(h \mid E)$ und $CF(h \mid E)$, die auf Basis einer Beobachtung E Vertrauen in die Fakten quantifizieren, können Certainty Factors auch einer Regel zugeordnet sein. Dabei handelt es sich um die festen Certainty Factors. In diesem Zusammenhang quantifiziert $CF(h \mid a)$ den Grad des Vertrauens in die Gültigkeit der Hypothese h unter der Bedingung, daß die Prämisse a als definitiv gültig ($\hat{=}$ Certainty Factor $= 1$) bekannt ist. Der Certainty Factor einer Regel wird folgendermaßen angeschrieben:

$$a \overset{CF(h \mid a)}{\longrightarrow} h$$

Die Sicherheit zusammengesetzter Aussagen

Für den Fall, daß die Prämisse einer Regel aus der booleanschen Verknüpfung von Fakten besteht, wird der Certainty Factor der Prämisse durch die Maximums- bzw. Minimumsbildung bestimmt: [Heinsohn1999]

$$CF(a_1 \wedge a_2 \mid E) = \min\{CF(a_1 \mid E), CF(a_2 \mid E)\} \tag{6.2}$$
$$CF(a_1 \vee a_2 \mid E) = \max\{CF(a_1 \mid E), CF(a_2 \mid E)\} \tag{6.3}$$

Anwendung von Regeln

Die Anwendung einer Regel, welche mit einem festen Certainty Factor versehen ist, und deren Prämisse mit einem dynamischen Certainty Factor bewertet ist, führt zu einer Bewertung der Hypothese, auf die geschlossen wird. Dabei müssen der Certainty Factor der Prämisse $CF(a \mid E)$ und der Regel $CF(h \mid a)$ in geeigneter Weise berücksichtigt werden. Für den Fall, daß die Prämisse mit einem nicht negativen Certainty Factor bewertet ist, erfolgt die Regelabarbeitung nach folgender Formel: [Heinsohn1999]

$$CF(h \mid E) \;=\; CF(a \mid E) \cdot CF(h \mid a) \qquad \text{falls } CF(a \mid E) \geq 0 \tag{6.4}$$

Für den Fall, daß die Prämisse einer Regel mit einem negativen Certainty Factor bewertet ist, darf diese Formel nicht angewendet werden. Diese Vorgehensweise wird einsichtig, wenn man voraussetzt, daß jede angewendete Regel eine, zumindest zu einem bestimmten Grad, erfüllte Prämisse besitzen muß. Für eine negative Prämisse ist diese Voraussetzung nicht erfüllt. Somit darf diese Regel nicht angewendet werden. [Heinsohn1999]

Parallel Kombination

Wird eine Hypothese von mehreren Regel bewertet, so müssen diese Bewertungen zu einer Gesamtbewertung zusammengeführt werden. Dies geschieht mit Hilfe der parallelen Kombination. Wird eine Hypothese h durch zwei Regeln

$$a \overset{CF(h \mid a)}{\longrightarrow} h$$

$$b \overset{CF(h \mid b)}{\longrightarrow} h$$

gestützt, so sind nach der erfolgten Regelanwendung die Certainty Factors $CF(h \mid E_1)$ und $CF(h \mid E_2)$ zur Berechnung des endgültigen Ergebnisses zu kombinieren. Die Kombination basiert auf der nachfolgenden Formel: [Heinsohn1999]

Seien $s = CF(h \mid E_1)$ und $t = CF(h \mid E_2)$, dann gilt

$$CF(h \mid E_1, E_2) \;=\; \begin{cases} s + t - s \cdot t & \text{falls } s, t \geq 0 \\ \dfrac{s + t}{1 - \min\{|s|, |t|\}} & \text{falls } s \cdot t \in (-1, 0] \\ \text{undefiniert} & \text{falls } s \cdot t = -1 \\ s + t + s \cdot t & \text{falls } s, t < 0 \end{cases} \tag{6.5}$$

Bei der Anwendung der Formel 6.5 müssen die beiden Spezialfälle

$$CF(h \mid E_1) = 1, \quad CF(h \mid E_2) = -1 \quad \text{und}$$
$$CF(h \mid E_1) = -1, \quad CF(h \mid E_2) = 1$$

berücksichtigt werden. In beiden Fällen schließen sich die beiden Bewertungen vollständig aus und sollten zu einer Fehlermeldung führen. [Heinsohn1999]

Die oben definierte Funktion für die parallele Kombination ist sowohl kommutativ, als auch assoziativ. Das Ergebnis der Kombination ist somit unabhängig von der Reihenfolge, mit der die entsprechenden Certainty Factors kombiniert werden. Auf diese Weise können durch hintereinander Anwendung der Formel 6.5 auch die Certainty Factors mehrerer Hypothesen kombiniert werden.

Eine wichtige Eigenschaft der Formel 6.5 ist, daß die Kombination mit einer ,,leeren" Informationsquelle (mit $CF(h \mid E_i) = 0$) keine Änderung am Gesamt-Certainty Factor hervorruft. [Heinsohn1999]

Rechenbeispiel

Die Anwendung der oben angeführten Rechenvorschriften für die Propagierung von Certainty Factors sollen nun anhand eines konkreten Zahlenbeispiel veranschaulicht werden. Als Regelbasis sollen folgende Regeln dienen: [Heinsohn1999]

Regel 1: $A_1 \wedge A_2 \wedge A_3 \xrightarrow{0.7} H$
Regel 2: $B_1 \vee B_2 \xrightarrow{0.4} H$
Regel 3: $C \xrightarrow{-0.8} H$

Der Und-Oder-Kombinationsgraph der Regelbasis ist Eingangs in Abbildung 6.8 abgebildet ist. Die für die Berechnung notwendigen Zahlenwerte sind in Tabelle unten angeführt. Die Certainty Factors wurden mit Hilfe der Formel 6.1 aus dem ,,measure of belive" (MB) und dem ,,measure of disbelive" (MD) berechnet.

	A_1	A_2	A_3	B_1	B_2	C
MB	1	1	0.6	0.5	1	0.5
MD	0.2	0	0	0.5	0	0
CF	1	1	0.6	0	1	0.5

Durch Anwendung der Formeln 6.2 und 6.3 kann der Certainty Factor der Prämissen der Regeln berechnet werden.

Regel 1: $CF(A_1 \wedge A_2 \wedge A_3 \mid E) = \min\{1, 1, 0.6\} = 0.6$
Regel 2: $CF(B_1 \vee B_2 \mid F) = \max\{0, 1\} = 1$
Regel 3: $CF(C \mid G) = 0.5$

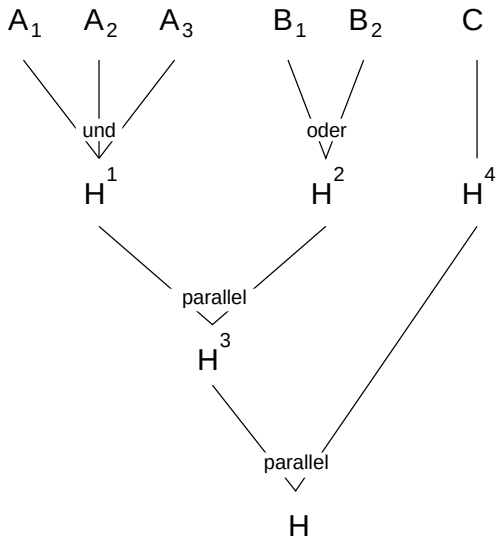

Abbildung 6.8: Der sich aus der Regelbasis ergebende Und-Oder-Kombinationsgraph. H^i bezeichnet im Graph den berechneten Certainty Factor der Hypothese H zum Zeitpunkt i. [Heinsohn1999]

Im nächsten Schritt fließen in die Anwendung von Regel 1 die Regelgewichtung $CF(H \mid A_1 \wedge A_2 \wedge A_3)$ und die Prämissensicherheit $CF(A_1 \wedge A_2 \wedge A_3 \mid E)$ ein. Unter Anwendung von Formel 6.4 erhält man:

$$\begin{aligned} CF(H^1 \mid E) &= CF(A_1 \wedge A_2 \wedge A_3 \mid E) * CF(H \mid A_1 \wedge A_2 \wedge A_3) \\ &= 0.6 * 0.7 \\ &= 0.42 \end{aligned}$$

Analog erhält man für die Anwendung von Regel 2

$$CF(H^2 \mid F) = CF(B_1 \vee B_2 \mid F) * CF(H \mid B_1 \vee B_2) = 0.4$$

und für Regel 3

$$CF(H^3 \mid G) = CF(C \mid G) \cdot CF(H \mid C) = -0.4.$$

Man beachte, daß der negative Certainty Factor von Regel 3 zu einer negativen Gewichtung des Wertes $CF(H^3 \mid G)$ geführt hat. Durch die Regelabarbeitung haben wir nun drei unabhängige Bewertungen für H erhalten.

$$\begin{aligned} CF(H^1 \mid E) &= 0.42 \\ CF(H^2 \mid F) &= 0.4 \\ CF(H^4 \mid G) &= -0.4 \end{aligned}$$

Diese müssen jetzt durch parallel Kombination zu einem Wert zusammengeführt werden. Dies erfolgt mit Hilfe der Gleichung 6.5. Im ersten Schritt werden die ersten beiden Informationen kombiniert.

$$CF\,(H^3 \mid E, F) \;=\; 0.42 + 0.4 - 0.42 * 0.4 = 0.65$$

Dieses Ergebnis wird nun mit $CF\,(H^4 \mid G)$ kombiniert.

$$CF\,(H \mid E, F, G) \;=\; \frac{0.65 - 0.4}{1 - 0.4} = 0.42$$

Nach Abarbeitung aller Regeln erhält man somit für die Bewertung der Hypothese H den Certainty Factor $CF\,(H \mid E, F, G) \;=\; 0.42$.

In diesem Abschnitt wurden die Certainty Factors als Möglichkeit vorgestellt, unsicheres Wissen in einem wissensverarbeitenden System zu speichern und zu verarbeiten. Im nachfolgenden Abschnitt wird einen weitere Möglichkeit beleuchtet, unsicheres Wissen zu verarbeiten. Die vorgestellte Methode der Fuzzy Logic beruht dabei auf der Verwendung von unscharfen Mengen.

Kapitel 7

Fuzzy Logik

In Expertensystemen, die im vorigen Abschnitt vorgestellt wurden, wird Expertenwissen zumeist in Form von Regeln gespeichert. Diese Regeln besten aus Vorbedingungen, die entweder wahr oder falsch sein können, und Aktionen, die, abhängig von den entsprechenden Vorbedingungen, ausgeführt werden oder nicht. Menschliche Experten formulieren ihre Regeln aber oft in einer Art, in der nicht genau festgelegt ist, wann eine Vorbedingung erfüllt ist; z.B. „Wenn die Temperatur ungefähr 19°C beträgt, dann muß das Heizungsventil ein wenig geöffnet werden." Menschen sind in der Lage diese Regel zu verstehen und können danach handeln. Für Computer stellen jedoch die unscharfen Begriffe „ungefähr 19°C" oder „einwenig öffnen" ein Problem dar. Ab wann soll ein Expertensystem eine Temperatur als „ungefähr 19°C" interpretieren? Doch zugleich zeigt sich, daß viele komplexe Probleme durch die Verwendung von unscharfen Regeln einfach beschrieben werden können. Daher ist es das Ziel, dem Computer das Umgehen mit unscharfen Begriffen zu ermöglichen. Dies geschieht mit Hilfe der 1965 von Lotfi A. Zadeh an der Universität von Kalifornien in Berkeley entwickelten Theorie der unscharfen Mengen (fuzzy set theory) und deren Anwendung, die in diesem Abschnitt vorgestellt werden soll.

7.1 Klassische Mengen versus Fuzzy Sets

In der klassischen Mengenlehre kann ein Element nur entweder einer Menge angehören oder nicht. Will man nun den Begriff „angenehme Raumtemperatur" beschreiben, so definiert man die Menge der angenehmen Raumtemperaturen. Dies kann in der klassischen Mengenlehre nur durch die Definition eines scharf abgegrenzten Intervalls geschehen, z.B. die Temperaturen zwischen 19°C und 24°C, wie dies in Abbildung 7.1a. dargestellt ist. Somit würde eine Temperatur

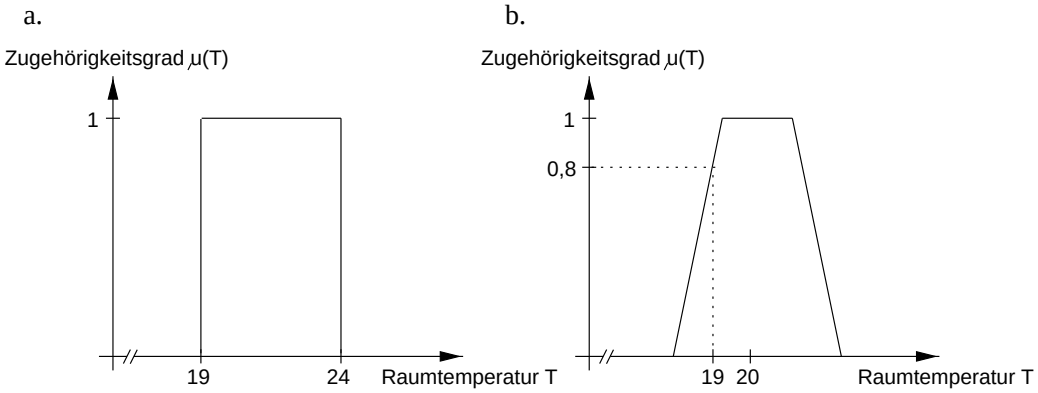

Abbildung 7.1: Definition des Begriffs: Angenehme Raumtenperatur.
a. nach der klassischen Mengenlehre: Es können nur scharfe Grenzen zwischen Element und nicht Element gezogen werden.
b. nach der Theorie der unscharfen Mengen: Ein Element kann auch nur zu einem bestimmten Grad einer Menge angehören; z.B. gehört die Temperatur 19°C mit dem Grad $0,8$ der Menge der angenehmen Raumtemperaturen an.

von $18,9$°C als nicht angenehm eingestuft. Dieser scharfe Übergang von angenehm zu nicht angenehm entspricht jedoch nicht dem menschlichen Empfinden. [Zimmermann1993]

Im Gegensatz zu klassischen Mengen können Elemente bei Fuzzy Sets auch nur bis zu einem bestimmten Grad einer Menge angehören. Dieser Zugehörigkeitsgrad wird üblicherweise durch eine Zahl aus dem Intervall $[0, 1]$ beschrieben. Dabei bedeutet der Grad 1 volle Zugehörigkeit zur Menge und der Grad 0 keine Zugehörigkeit. In Abbildung 7.1b. ist die Zugehörigkeitsfunktion zur unscharfen Menge der angenehmen Raumtemperaturen angegeben. Bei dem Beispiel gehört die Temperatur 19°C mit dem Zugehörigkeitsgrad $0,8$ der Menge der angenehmen Raumtemperaturen an. [Zimmermann1993]

7.2 Definitionen

An dieser Stelle sollen einige, für das Verständnis von Fuzzy Regel Systemen notwendige Definitionen der Fuzzy Set Theory angeführt werden: [Rojas1993] [Nauck1994]

Definition **Fuzzy-Menge** (fuzzy set, unscharfe Menge): Gegeben sei

1. Eine Grundmenge U (auch Universum) mit den Elementen u.

2. Eine totale Abbildung $\mu_A : U \longrightarrow [0,1]$ vom Universum in das abgeschlossene Intervall $[0,1]$, dann ist

$$A := \{(u, \mu_A(u)) \mid u \in U, \mu_A(u) \in [0,1]\} \tag{7.1}$$

eine Fuzzy-Menge (fuzzy set). $\mu_A(u)$ gibt den Zugehörigkeitsgrad, bzw. den Grad der Mitgliedschaft eines Elements $u \in U$ zur Menge A an. Sie heißt Mitgliedsgradfunktion (membership function), Zugehörigkeitsfunktion bzw. charakteristische Funktion. [Rojas1993]

Besteht das Bild der Funktion $\mu_A(u)$ aus den diskreten Werten $\{0,1\}$, dann ist A eine Menge im klassischen Sinn. Eine solche Menge heißt dann gewöhnliche Menge (ordinary set).

Das Intervall $[0,1]$ heißt „membership space" M. M kann auch eine beliebige teilweise geordnete Menge sein.

Unscharfe Begriffe wie „angenehm", „hoch", „nieder", „kalt", usw. werden als **linguistische Werte** bezeichnet und durch unscharfe Mengen beschrieben.

Als Zugehörigkeitsfunktion sollte eine möglichst einfache Funktion gewählt werden. Fuzzy-Mengen in der Form von Dreiecks-, Trapez- und Gauß-Funktionen, die durch wenige Parameter beschrieben werden können, eignen sich besonders gut für die Speicherung in Computern und die Durchführung von Berechnungen. Beispiele für solche unscharfen Mengen sind in Abbildung 7.2 dargestellt. [Kruse1997]

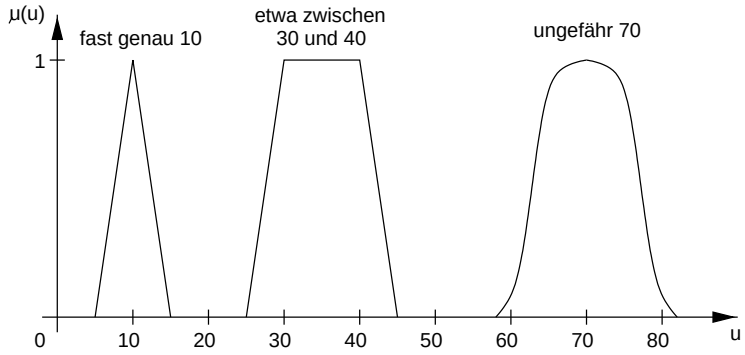

Abbildung 7.2: Beispiele für Fuzzy-Mengen. Dreiecks-, Trapez- und Gauß-Funktionen lassen sich durch wenige Parameter beschreiben und sich einfach verarbeiten. [Kruse1997]

Damit auf der Basis von Fuzzy Sets ein Regelsystem arbeiten kann, muß jeder mögliche Wert einer Variable zumindest einer unscharfen Menge angehören.

Der Wertebereich einer Variable wird daher in linguistische Werte gegliedert, sodaß die zugehörigen unscharfen Mengen den ganzen Bereich überdecken und sich überlappen. Diese Aufgliederung nennt man unscharfe Fuzzy Zerlegung oder Fuzzy Partitioning. Ein Beispiel dafür ist in Abbildung 7.3 dargestellt. [Kruse1997]

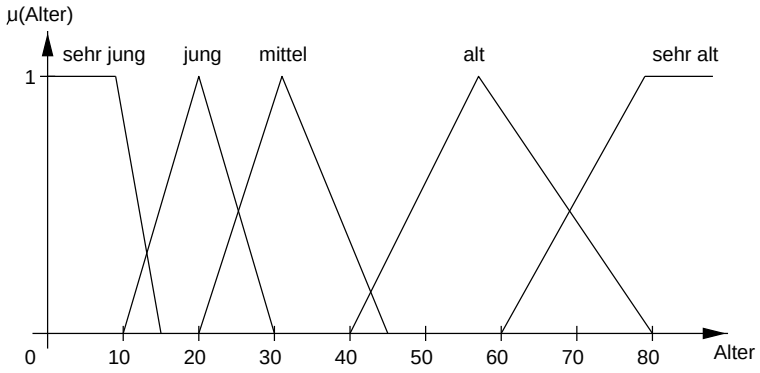

Abbildung 7.3: Beispiel für eine Fuzzy Zerlegung. Die Fuzzy-Mengen überlappen sich und decken den vollständigen Wertebereich ab.

Ausgehend von den oben getroffenen Definition einer Fuzzy Menge können weiters folgende Definitionen getroffen werden:

Definition (**Gleichheit**): Zwei Fuzzy-Mengen A und B heißen gleich und werden $A = B$ geschrieben, genau dann wenn

$$\mu_A(u) = \mu_B(u) \qquad \forall u \in U \tag{7.2}$$

gilt.

Definition (**Leere Menge**): Eine Fuzzy-Menge A heißt leer und wird $A = \emptyset$ bezeichnet, genau dann wenn

$$\mu_A(u) = 0 \qquad \forall u \in U \tag{7.3}$$

gilt.

Definition (**Teilmenge**): Eine Fuzzy-Menge A heißt Teilmenge einer Fuzzy-Menge B, bezeichnet durch $A \subseteq B$, genau dann wenn

$$\mu_A(u) \leq \mu_B(u) \qquad \forall u \in U \tag{7.4}$$

gilt.

Mengenoperationen

Definition (**Komplement**): Ist A eine Fuzzy-Menge über einem Universum U, dann ist das Komplement A' (bzw. $\neg A$) gegeben durch

$$A' = \{(u, (1 - \mu_A(u))) \mid u \in U\}. \tag{7.5}$$

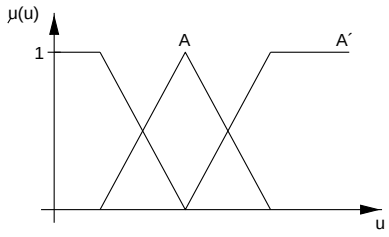

Abbildung 7.4: Komplementbildung bei Fuzzy-Mengen.

Definition (**Vereinigung**): Sind A und B Fuzzy-Mengen über U, dann ist die Vereinigung $A \cup B$ ebenfalls eine Fuzzy-Menge C und die Zugehörigkeitsfunktion ist gegeben durch

$$\mu_C(u) = max\{\, \mu_A(u), \mu_B(u) \,\} \qquad \forall u \in U. \tag{7.6}$$

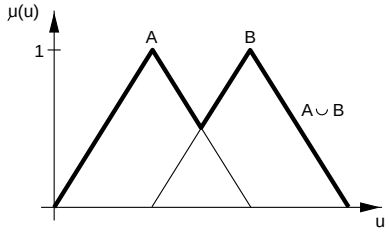

Abbildung 7.5: Vereinigung zweier Fuzzy-Mengen.

Definition (**Durchschnitt**): Sind A und B Fuzzy-Mengen über U, dann ist die Durchschnitt $A \cap B$ ebenfalls eine Fuzzy-Menge C und die Zugehörigkeitsfunktion ist gegeben durch

$$\mu_C(u) = min\{\, \mu_A(u), \mu_B(u) \,\} \qquad \forall u \in U. \tag{7.7}$$

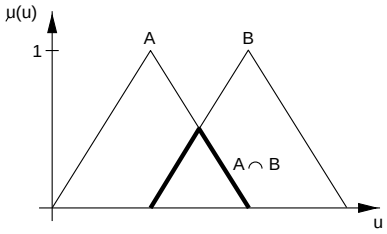

Abbildung 7.6: Durchschnitt zweier Fuzzy-Mengen.

Logische Verknüpfung linguistischer Werte

In den Vorbedingungen von Regeln werden linguistische Werte durch logische Operationen miteinander verknüpft. Bevor dies jedoch geschehen kann, müssen die aktuellen Eingangswerte erst fuzzyfiziert werden. Unter **Fuzzyfizierung** versteht man die Ermittlung der Zugehörigkeitsgrade zu den unscharfen Mengen, ausgehend von aktuellen Werten der Eingangsgrößen. Der Zugehörigkeitsgrad wird mit Hilfe der Zugehörigkeitsfunktion $\mu(u)$ ermittelt, d.h. der Eingangswert führt über die Zugehörigkeitsfunktion zu einem Zugehörigkeitsgrad der entsprechenden unscharfen Menge.

Bei der **UND-Verknüpfung** ist der Zugehörigkeitsgrad der Vorbedingung durch das Minimum der Zugehörigkeitsgrade der aktuellen Eingangsgrößen bestimmt. Es wird somit das Minimum der fuzzyfizierten Eingangsgrößen gebildet. Das Ergebnis der UND-Verknüpfung entspricht somit einer Zahl aus dem Intervall $[0, 1]$. Dieser Vorgang ist in Abbildung 7.7 dargestellt.

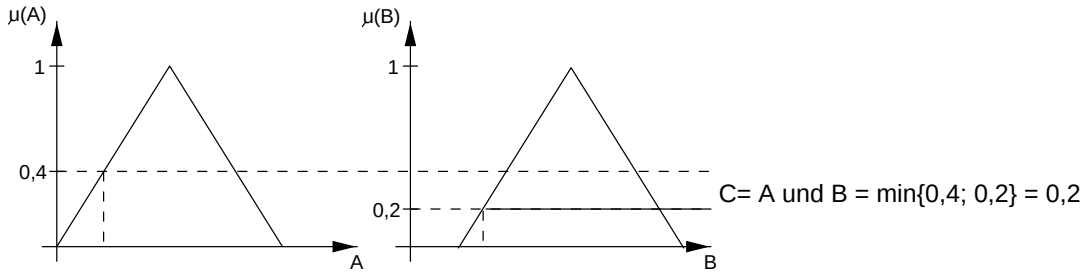

Abbildung 7.7: UND-Verknüpfung von zwei linguistischen Werten.

Bei der **ODER-Verknüpfung** ist der Zugehörigkeitsgrad der Vorbedingung durch das Maximum der Zugehörigkeitsgrade der aktuellen Eingangsgrößen bestimmt. Es wird somit das Maximum der fuzzyfizierten Eingangsgrößen gebildet. Das Ergebis der ODER-Verknüpfung entspricht somit einer Zahl aus dem Intervall $[0, 1]$. Dieser Vorgang ist in Abbildung 7.8 dargestellt.

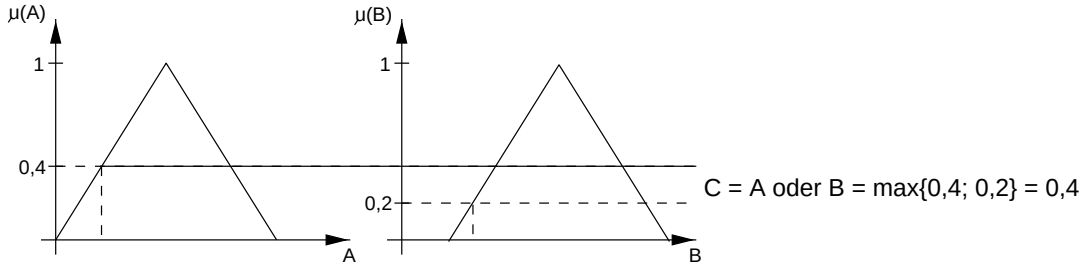

Abbildung 7.8: ODER-Verknüpfung von zwei linguistischen Werten.

Die hier vorgestellten Definitionen für Fuzzy Sets und Fuzzy Logic sind voll kompatibel zur klassischen Mengenlehre und zur klassischen Logik. Dies kann leicht veranschaulicht werden, indem man nur Zugehörigkeitsfunktionen verwendet, die auf die diskreten Werte $\{0, 1\}$ abbilden. Diese Definitionen stellen jedoch nicht die einzige Möglichkeit dar, logische Operationen auf unscharfe Ausdrücke zu definieren. In der Literatur findet man dafür auch andere Möglichkeiten. [Hofer1995] Als Beispiel dafür soll an dieser Stelle kurz die Definition über die *beschränkten Summen* angeführt werden. Die Und-Verknüpfung erfolgt über $C = A \wedge B = max\{0,\ \mu_A(a) + \mu_B(b) - 1\}$, die Oder-Verknüpfung über $C = A \vee B = min\{1,\ \mu_A(a) + \mu_B(b)\}$ mit $a \in A$, $b \in B$. [Rojas1993] Für die weiteren Betrachtungen wird jedoch auf die ersteren Definitionen von logischen Operationen auf Fuzzy Mengen zurückgegriffen.

7.3 Arbeitsweise eines Fuzzy Regel Systems

Die Arbeitsweise eines Fuzzy Regel Systems soll anhand eines einfachen Regelbeispiels erklärt werden. Das Beispiel ist gegeben, durch folgende Regeln: [Kruse1997]

1. Wenn x positiv klein ist und y positiv klein ist, dann ist z positiv klein.
2. Wenn x positiv groß ist und y positiv klein ist, dann ist z positiv groß.

Die Zugehörigkeitsfunktionen zu den in den obigen Regeln verwendeten linguistischen Werte „x positiv klein", „x positiv groß", „y positiv klein", „z positiv klein" und „z positiv groß" sind nachfolgend in der Abbildung 7.9 dargestellt. [Kruse1997]

Die Arbeitsweise eines Fuzzy Regel Systems besteht im wesentlichen aus drei Schritten: [Hofer1995]

- Fuzzyfizierung der konkreten Eingangsgrößen.
- Inferenz und Komposition der Regeln.
- Defuzzyfizierung der konkreten Ausgangsgrößen.

Fuzzyfizierung der konkreten Eingangsgrößen

Für jede Eingangsgröße wird mit Hilfe der Zugehörigkeitsfunktion der Fuzzy-Menge der Zugehörigkeitsgrad zum entsprechenden linguistischen Wert bestimmt. [Hofer1995]

Inferenz und Komposition der Regeln

Im nächsten Schritt werden die linguistischen Werte logisch miteinander verknüpft. In diesem Beispiel sind die linguistischen Werte UND verknüpft. Daher ist der Grad der Vorbedingung durch das Minimum der Zugehörigkeitsgrade der Eingangsgrößen bestimmt. Der Grad der Vorbedingung muß nun umgelegt werden, auf den linguistischen Wert der Aktion der Regel. Diesen Schritt nennt man Inferenz. Die Inferenz erfolgt, indem man das Minimum zwischen dem Grad der Vorbedingung und der Zugehörigkeitsfunktion der Aktion bildet. Graphisch kann dies als ,,abschneiden" der Zugehörigkeitsfunktion auf Höhe des Grades der Vorbedingung angesehen werden, wie dies in Abbildung 7.9 in der letzten Spalte dargestellt ist. Diese Methode der Inferenzbildung stellt jedoch nur eine Möglichkeit dar. Eine andere Möglichkeit ist die Produktbildung zwischen dem Grad der Vorbedingung und der Zugehörigkeitsfunktion der Aktion. Für welche Art der Inferenzbildung man sich entscheidet, hängt von der Art der Anwendung ab. [Hofer1995]

Da in einem Regelsystem meist mehrere Regeln eine Ausgangsgröße betreffen, müssen die Zugehörigkeitsfunktionen der Aktion nach der Inferenzbildung zu einer Gesamtzugehörigkeitsfunktion zusammengefaßt werden. Diesen Schritt nennt man Komposition. Die gebräuchlichste Methode die Komposition durchzuführen, ist die Maximum Methode. Dabei erhält man die Gesamtzugehörigkeitsfunktion durch die Maximum-Bildung über alle Zugehörigkeitsfunktionen der Aktionen, die eine Ausgangsgröße betreffen. Dieser Schritt ist in der letzten Zeile von Abbildung 7.9 dargestellt. [Hofer1995]

Defuzzyfizierung der konkreten Ausgangsgrößen

Als letzten Schritt in einem Fuzzy Regel System muß noch eine konkrete Ausgangsgröße aus der Gesamtzugehörigkeitsfunktion ermittelt werden. Diesen Vor-

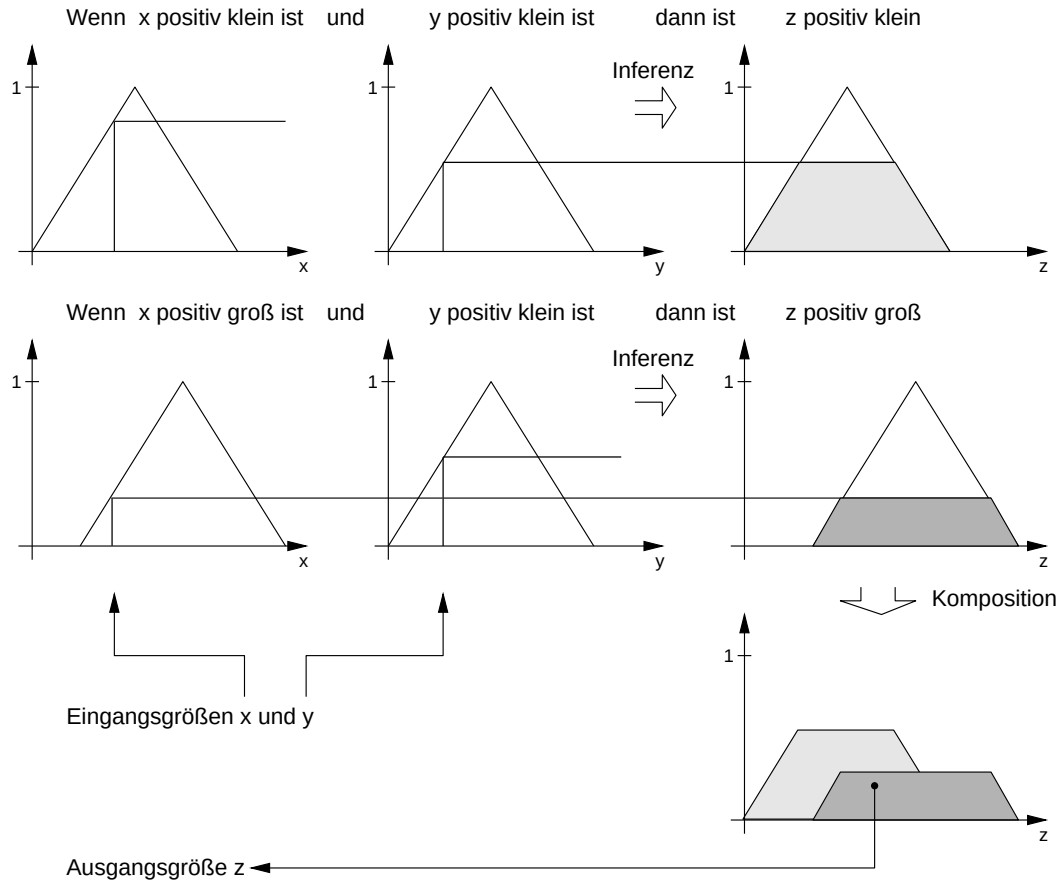

Abbildung 7.9: Anschauungsbeispiel eines Fuzzy Regel Systems. Die drei Arbeitsschritte sind: Fuzzyfizierung, Inferenz und Komposition, Defuzzyfizierung. [Kruse1997]

gang nennt man Defuzzyfizierung. Auch dieser Schritt kann durch mehrere unterschiedliche Methoden erfolgen. Die gebräuchlichste ist die Schwerpunkt-Methode. Dabei wird als Ausgangsgröße der Abszissenwert des Schwerpunktes der unter der Gesamtzugehörigkeitsfunktion gelegenen Fläche verwendet. [Hofer1995]

7.4 Anwendungsbeispiel: Container-Kran

Anhand eines konkreten Beispiels soll gezeigt werden, wie einfach der Grobentwurf eines Fuzzy Regel Systems erfolgen kann: Eine Ladung soll mit Hilfe eines Container-Krans von einer Ausgangsposition (z.B. ein Schiff) zum Ziel (z.B. ein Eisenbahnwaggon) transportiert werden. Die Last ist mit dem Krankopf durch ein Seil verbunden und pendelt, sobald sich der Kran in Bewegung setzt. Das Pendeln stört nicht während der Fahrt, aber beim Absetzen des Containers. Es soll nun ein Regelsystem entworfen werden, welches den Container möglichst schnell von der Ausgangsposition zum Ziel transportiert. [Altrock1993]

Dieses Problem ist auch mit Hilfe der klassischen Regelungstechnik lösbar. Allerdings reicht ein einfacher PID-Regler nicht aus, da die Stabilisierungsstrategie stark nichtlinear von der Position des Krans zum Ziel abhängt. Beim Anfahren ist der Pendelausgleich nicht wichtig, da die Last so schnell wie möglich an das Ziel gebracht werden soll, je näher man jedoch zum Ziel kommt, desto wichtiger ist die Stabilisierung. [Altrock1993]

Ein Mensch ist jedoch in der Lage, mit etwas Erfahrung einen solchen Kran auch bei Windeiflüssen und unterschiedlichen Lasten recht gut zu steuern, ohne erst Differentialgleichungen lösen zu müssen. Will man dieses Erfahrungswissen mit Hilfe der Fuzzy Logic automatisieren, muß man zunächst durch Beobachtung des Prozesses gewisse Zustände identifizieren, z.B: schnelle Fahrt, Last pendelt stark nach hinten, Ziel weit weg, usw. Für diese Zustände werden in der Folge Wenn-Dann Regeln aufgestellt. Diese sind in Tabelle 7.1 dargestellt. Interpretiert kann die Tabelle wie folgt werden:

Solange der Krankopf noch weit vom Ziel entfernt ist, wird, unabhängig wie stark die Last pendelt, der Motor auf volle Leistung gestellt. Dies entspricht der Strategie: „Wenn man noch weit vom Ziel entfernt ist, fahre so schnell wie möglich." Kommt der Krankopf jedoch näher, so verändert sich die Strategie. Pendelt die Last nur gering, so wird die Geschwindigkeit leicht verringert, um ein zu abruptes abbremsen beim Ziel und das damit verbundene starke Pendeln zu verhindern. Pendelt die Last jedoch stark, so wird schon jetzt versucht, die Pendelbewegung in den Griff zu bekommen, indem man den Krankopf in die entgegengesetzte Richtung der Pendelschwingung bewegt. Befindet man sich in der unmittelbaren Umgebung des Ziels, so liegt das Hauptaugenmerk auf dem

	Abstand		
Lastwinkel	groß	gering	null
positiv, groß	schnell voraus	schnell voraus	langsam zurück
positiv, klein	schnell voraus	langsam voraus	langsam zurück
null	schnell voraus	langsam voraus	stop
negativ, klein	schnell voraus	langsam voraus	langsam voraus
negativ, groß	schnell voraus	langsam zurück	schnell voraus

Tabelle 7.1: Regeln zur Steuerung eines Containerkrans. Die Eingangsgrößen sind der Abstand des Krankopfes zum Ziel und der Lastwinkel. Eilt die Ladung dem Krankopf voraus, so ist der Lastwinkel positiv. Die Ausgangsgröße ist die Motorleistung der Kranantriebes. Abstand, Lastwinkel und Motorleistung sind linguistische Variablen und werden durch Fuzzy-Mengen repräsentiert. [Altrock1993]

Ausgleich der Pendelbewegung. Dies erreicht man durch langsame Bewegung in die entgegengesetzte Richtung der Pendelbewegung. [Altrock1993]

Gegenüber der Lösung mit Hilfe der Klassischen Regelungstechnik wurden folgende Vorteile erzielt: [Altrock1993]

- Die gewünschte Regelstrategie kann ohne aufwendige mathematische Modellbildung auf der Basis des vorhandenen technischen Wissens erstellt werden.
- Auch Nichtspezialisten der Regelunstechnik können ein solches System aufbauen.
- Da jedem Systemzustand leicht verständliche Regeln zugeordnet sind, beschleunigen sich die Inbetriebnahme und die spätere Modifikation.
- Im Gegensatz zu einer konventionellen Lösung, deren komplexe mathematischen Berechnungen einen leistungsfähigen Rechner benötigen, lassen sich Fuzzy Regel Systeme auch auf einfachen Rechnern einsetzen.

Der Nachteil der Fuzzy Regel Systeme liegt in der Feinabstimmung. Der erste Entwurf der Regelbasis und der entsprechenden Fuzzy Mengen liefert zwar häufig sehr schnell eine Regelung, die das System im groben unter Kontrolle bringt, aber für eine genaue Regelung muß die Zugehörigkeitsfunktion der unscharfen Mengen oder das Verfahren, mit denen man die Fuzzy Mengen verknüpft, in aufwendiger Feinarbeit nachgebessert werden. Für diese Nachbesserungen stehen

kaum methodische Hilfen zur Verfügung, daher erfolgen diese in den meisten Fällen über Versuchen. Dabei wäre ein lernfähiges System von Vorteil, wie dies im nächsten Kapitel vorgestellt wird. [Kruse1997] In Kapitel 9 wird darüber hinaus mit NeuroFuzzy eine Technik vorgestellt, welches die Vorteile eines lernfähigen Systems mit den Vorteilen der Fuzzy Logic verbindet.

Kapitel 8

Neuronale Netze

Am Ende des vorigen Abschnitts wurde angeführt, wie schwer die Feinabstimmung eines Fuzzy Regel Systems fallen kann. In diesem Abschnitt soll nun anhand der Neuronalen Netze ein Modell eines lernfähiges Systems vorgestellt werden, welches mit Hilfe von Trainingsdaten auf die Durchführung seiner Aufgabe trainiert wird. Das Modell der Neuronalen Netze lehnt sich dabei stark an die Funktionsweise der Neuronen im menschlichen Gehirn an.

8.1 Grundlagen der Neuronalen Netze

Für das Studium der Neuronalen Netze gibt es ganz unterschiedliche Motive, je nachdem welche Forschergruppe daran beteiligt ist. Biologen sind an Modellen interessiert, mit denen natürliche Neuronale Netze verstanden werden können. Mathematiker untersuchen welche Klassen von Funktionen mit Hilfe des Modells der Neuronalen Netze berechnet werden können. Informatiker sind daran interessiert, die Palette der Möglichkeiten zu erforschen, die sich durch den Einsatz eines Neuronalen Netzes als lernfähiges und fehlertolerantes System eröffnen. Die Forschung an Neuronalen Netzen stellt sich somit als stark interdisziplinär heraus. [Rojas1993] Dabei können die einzelnen Disziplinen viel von einander profitieren. So basiert z.B. das Modell des künstlichen Neurons auf den Untersuchungen von natürlichen Nervenzellen. Umgekehrt konnten durch die Simulation von Neuronalen Netzen, Hypothesen über die Informationsverarbeitung im menschlichen Hirn bestätigt oder widerlegt und somit verworfen werden [Hinton1992].

Mit künstlichen Neuronalen Netzen werden natürliche biologische Neuronale Netze als informationsverarbeitende Systeme nachgeahmt. Dazu müssen zunächst die wesentlichen Eigenschaften der biologischen Nervenzellen unter dem Gesichtspunkt der Informationsverarbeitung studiert werden, um dann abstrakte Neuro-

nale Netze zu entwerfen und sie anschließend durch Simulation zu untersuchen. Auch wenn sich die unterschiedlichen Modelle des Gehirns in vielen Aspekten unterscheiden, herrscht Übereinstimmung darüber, daß das Wesen der Funktion des Nervensystems die „Kontrolle durch Kommunikation" ist. Während jedoch elektronische Schaltelemente in der Lage sind, im Nanosekunden-Bereich zu schalten, sind natürliche Neuronen im Gehirn mit einer Schaltzeit im Millisekunden-Bereich vergleichsweise langsam. Trotzdem ist das menschliche Gehirn aber in der Lage, Probleme zu lösen, die für jeden konventionellen Rechner in weiter Ferne liegen. Dies liegt vorallen an der großen Anzahl der Neuronen und der starken Vernetzung. [Rojas1993] Der menschliche Kortex[1] z.B. besteht aus ca. 100 Milliarden Neuronen die mit ca. 100 Billionen Synapsen[2] miteinander verbunden sind [Kurzweil1999].

In der nachfolgenden Übersicht soll betrachtet werden, in welchen Anwendungsgebieten künstliche Neuronale Netze in der Praxis verwendet werden. Dabei handelt es sich in den meisten Fällen um Anwendungsgebiete, die mit den Methoden der „konventionellen Informatik" nur mit großem Aufwand in den Griff bekommen werden. Neuronale Netze werden zur Bewältigung dieser Aufgaben nicht im klassischen Sinn programmiert, sie werden vielmehr auf ihre Aufgabe trainiert. [Luger1997]

- **Klassifizierung:** Bei der Klassifizierung wird für jedes Eingangsmuster durch das Neuronale Netz entschieden, zu welcher Klasse von Eingangssignalen es gehört. Ein klassisches Beispiel für die Klassifizierung ist die Mustererkennung (Pattern Recognition). Dabei wird versucht, in verrauschen Eingangssignalen bestimmte Strukturen zu identifizieren. Bei der Schrifterkennung wird z.B. versucht eine Pixel-Matrix (Eingangssignal) einem Buchstaben des Alphabets (Klasse) zuzuordnen. [Luger1997]

- **Assoziativspeicher:** Bei dieser Art des Speicherzugriffs wird nicht eine bestimmte Speicheradresse (Inputvektor) auf den Inhalt einer Speicherzelle abgebildet, vielmehr wird die „Umgebung" der Speicheradresse auf die Speicherzelle abgebildet. Für den Fall, daß der Eingangsvektor x mit der Speicherzelle y assoziiert wird, so soll auch die Eingabe $\tilde{x}$ zu der Speicherzelle y führen, wenn $\tilde{x}$ in der Umgebung von x liegt. Auf diese Weise können z.B. verrauschte Eingaben dem richtigen Ausgangswert zugeordnet werden. [Rojas1993]

[1] Großhirnrinde: Jene Schicht des menschlichen Gehirns, die für Sinneswahrnehmungen und das Denken zuständig ist. [EB1999]

[2] Kontaktstelle zwischen den Neuronen. Wissen wird in Neuronalen Netzen in der Stärke dieser Kontaktstellen, den synaptischen Stärken, gespeichert. [Rojas1993]

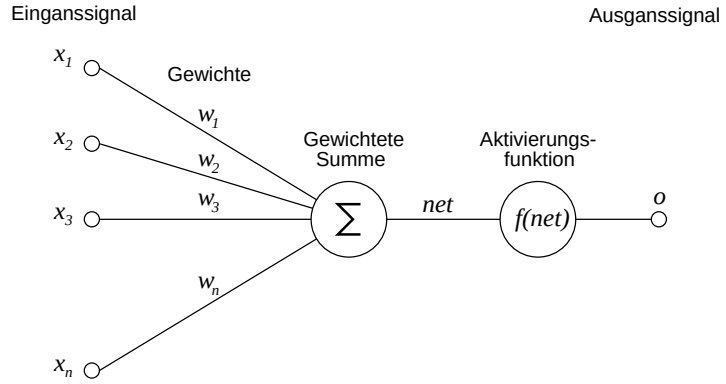

Abbildung 8.1: Schematische Darstellung eines künstlichen Neurons: Aus den Eingangssignalen x_i wird die gewichtete Summe net gebildet. Aus dieser Summe wird dann mit der Aktivierungsfunktion $f(net)$ der Ausgangswert o berechnet.

8.2 Das künstliche Neuron

An dieser Stelle soll auf die Basis eines Neuronalen Netzes, das künstliche Neuron, eingegangen werden. Ein künstliches Neuron besteht aus folgenden Bestandteilen, die in Abbildung 8.1 graphisch veranschaulicht sind: [Luger1997]

Eingangssignale x_i **:** Diese Daten können von der Umgebung oder vom Ausgang eines anderen künstlichen Neuron stammen. Unterschiedliche Netzwerkmodelle erlauben dabei unterschiedliche Wertebereiche. Typische Wertebereiche sind die reellen Zahlen, das Intervall $[0, 1]$ oder die diskreten Werte $\{0, 1\}$.

Gewichte w_i **:** Jeder Verbindung in einem Neuronalen Netz ist eine reelle Zahl als Gewicht zugeordnet. Dieses Gewicht beschreibt die Stärke der Verbindung. In diesen Verbindungsgewichten ist das Wissen des Neuronalen Netzes gespeichert.

Netto-Input net **:** Der Netto-Input entspricht der Summe der gewichteten Eingangssignale.

$$net = \sum_i w_i x_i \tag{8.1}$$

Aktivierungsfunktion $f(net)$ **:** Die Aktivierungsfunktion bestimmt, abhängig vom Netto-Input net, den Output o des Neurons. Dieser Output o kann das Eingangssignal für ein anderes Neuron sein oder das Ausgangssignal des Neuronalen Netzes. Als Aktivierungsfunktion wird meist eine der nachfolgenden Grundtypen verwendet:

Schwellwertfunktion:

$$f(net) = \begin{cases} 0 & : \quad net < 0 \\ 1 & : \quad net \geq 0 \end{cases} \tag{8.2}$$

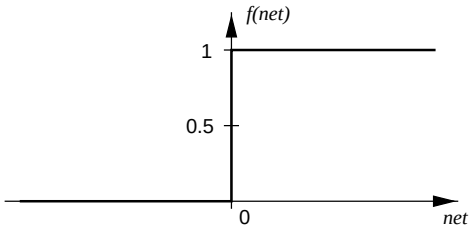

Abbildung 8.2: Schwellwertfunktion als Aktivierungsfunktion.

Der Funktionsverlauf ist in Abbildung 8.2 graphisch dargestellt. Neuronen (Gatter), die diese Aktivierungsfunktion verwenden, nennt man Schwellwertgatter. Das in Abschnitt 8.3 vorgestellte Perzeptron, stellt ein solches Schwellwertgatter dar.

Sigmoide Funktion:

$$f(net) = \frac{1}{1 + e^{-net}} \tag{8.3}$$

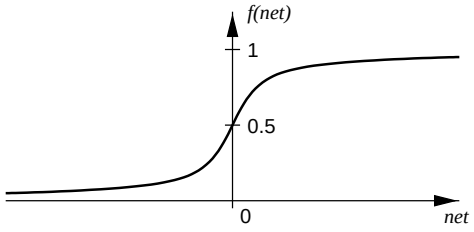

Abbildung 8.3: Sigmoide Funktion als Aktivierungsfunktion.

Der Funktionsverlauf ist in Abbildung 8.3 graphisch dargestellt. Der Unterschied der Sigmoiden Funktion zur Schwellwertfunktion ist, daß diese ein kontinuierliches Ausgangssignal liefert und differenzierbar ist. Erst dadurch werden Lernverfahren wie das Backpropagation-Learning, das in Abschnitt 8.4 vorgestellt wird, möglich. Gatter, die diese Aktivierungsfunktion verwenden, nennt man Sigmoide-Gatter.

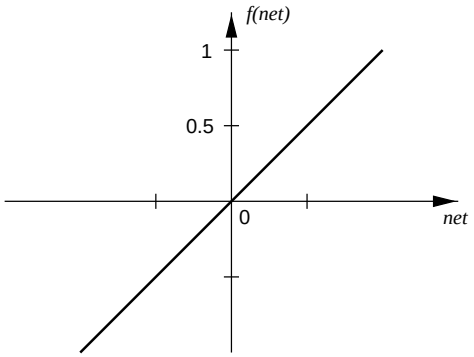

Abbildung 8.4: Lineare Funktion als Aktivierungsfunktion.

Lineare Funktion:

$$f(net) = net \tag{8.4}$$

Der Funktionsverlauf ist in Abbildung 8.4 graphisch dargestellt. Die lineare Funktion stellt die einfachste und am wenigsten mächtige Aktivierungsfunktion dar. Gatter die diese Aktivierungsfunktion verwenden, nennt man Linearegatter. Bei der linearen Aktivierungsfunktion ist zu beachten, daß eine Hintereinanderschaltung mehrerer linearer Gatter nur die selbe Funktion berechnen kann, wie ein einzelnes Gatter. Der Grund dafür ist, daß eine Hintereinanderschaltung von linearen Funktionen wieder linear ist[3].

Zusätzlich zu den Eigenschaften der individuellen Neuronen, werden die Eigenschaften eines Neuronalen Netzes auch durch folgende Charakteristika bestimmt: [Luger1997]

Netzwerktopologie: Unter der Topologie des Netzwerkes wird das Muster der Verbindungen zwischen den einzelnen künstlichen Neuronen verstanden. Meistens werden die Neuronen in Schichten (Layern) angeordnet. Auf die Einschränkungen, die aus der Topologie folgen, wird im nächsten Abschnitt noch näher eingegangen.

Lernalgorithmus: Darunter versteht man jenen Algorithmus, mit dem das Netzwerk auf seine zukünftige Aufgabe trainiert wird. In Abschnitt 8.4 wird der Backpropagation Algorithmus als grundlegender Algorithmus für das Lernen in Neuronalen Netzen vorgestellt.

[3] Analog dazu kann man in der Elektrotechnik ein Netzwerk aus linearen passiven Zweipolen durch eine Ersatzschaltung eines einzigen passiven Zweipols ersetzen.

In diesem Abschnitt wurden die grundlegenden Eigenschaften eines Neuronalen Netzes besprochen. Darauf aufbauend soll im nächsten Abschnitt die Arbeitsweise von Neuronalen Netzen anhand des Modells von McCulloch und Pitts erläutert werden.

8.3 Das Perzeptron

Das Perzeptron von McCulloch und Pitts (1943) stellt den ersten Ansatz von Neuronal Computing dar. [Luger1997] Anhand dieses historischen Ansatzes soll die Arbeitsweise eines Neuronalen Netzes veranschaulicht werden. Dabei werden auch die Grenzen dieses Modells beleuchtet und ein Ausweg in der Verwendung von Multi-Layer-Perzeptronen gezeigt.

Ein Perzeptron ist ein Neuron, wie dies in Abbildung 8.1 dargestellt ist. Als Eingangssignale sind die reellen Zahlen zulässig, die Gewichte entsprechen ebenfalls reellen Zahlen. Als Aktivierungsfunktion dient eine Schwellwertfunktion mit variabler Schwelle (Threshold θ). Das Ausgangssignal des Perzeptrons ist somit diskret aus der Menge $\{0, 1\}$. [Haykin1994]

$$f(x_i) = \begin{cases} 0 & : \sum_{i=1}^{n} x_i w_i < \theta \\ 1 & : \sum_{i=1}^{n} x_i w_i \geq \theta \end{cases} \tag{8.5}$$

Zur Vereinfachung kann die Aktivierungsfunktion auch anderst angeschrieben werden:

$$f(x_i) = \begin{cases} 0 & : \sum_{i=1}^{n} x_i w_i - \theta < 0 \\ 1 & : \sum_{i=1}^{n} x_i w_i - \theta \geq 0 \end{cases} \tag{8.6}$$

Verwendet man diese Schreibweise, so kann man erkennen, daß der Threshold θ auch als zusätzliches Gewicht für einen zusätzlichen Eingang, der konstant auf 1 liegt, aufgefaßt werden kann. Ein Perzeptron mit der Realisierung des Schwellwerts durch einen zusätzlichen Eingang und einem zusätzlichen Gewicht ist in Abbildung 8.5 dargestellt. Somit ergibt sich für die Aktivierungsfunktion:

$$f(x_i) = \begin{cases} 0 & : \sum_{i=1}^{n+1} x_i w_i < 0 \\ 1 & : \sum_{i=1}^{n+1} x_i w_i \geq 0 \end{cases} \tag{8.7}$$

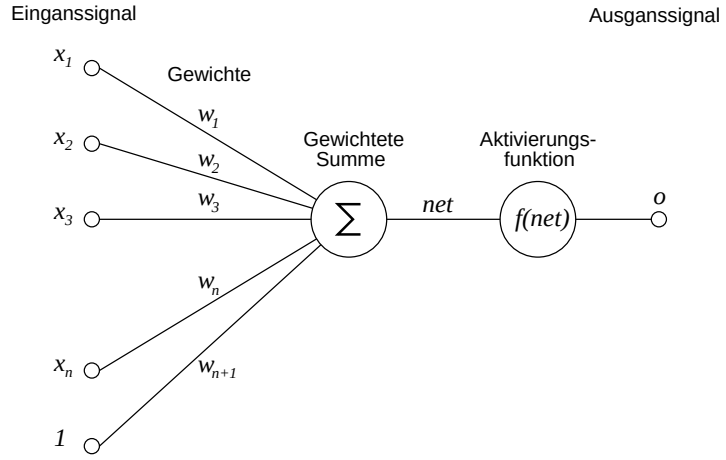

Abbildung 8.5: Perzeptron mit der Realisierung des Schwellwerts durch einen zusätzlichen Eingang und einem zusätzlichen Gewicht.

Durch diese Schreibweise, kann wie später gezeigt wird, der Perzeptron Lernalgorithmus für die Gewichte, als auch für die Schwellwerte einfach angeschrieben werden.

Wie Abbildung 8.6 zeigt, können mit der Hilfe von Perzeptronen logische Gatter realisiert werden. Die gewichtete Summe für das UND-Gatter lautet:

$$net = 1 * x + 1 * y + (-2) * 1 = x + y - 2 \tag{8.8}$$

In Tabelle 8.1 sind die gewichteten Summen und die Ausgangswerte für alle möglichen Eingangskombinationen einer UND-Verknüpfung aufgelistet. Die Tabelle für das Oder-Gatter und die Negation kann analog angeschrieben werden. [Luger1997]

Die stärke Neuronaler Netze liegt jedoch in der Eigenschaft, daß sie auf ihre spätere Aufgabe trainiert werden können. Als einfacher Lernalgorithmus soll hier der Perzeptron-Lernalgorithmus vorgestellt werden.

x	y	$net = x + y - 2$	$f(net) = o$
1	1	0	1
1	0	-1	0
0	1	-1	0
0	0	-2	0

Tabelle 8.1: Gewichtete Summen und Ausgangssignal für ein UND-Gatter realisiert mit Hilfe eines Perzeptrons. [Luger1997]

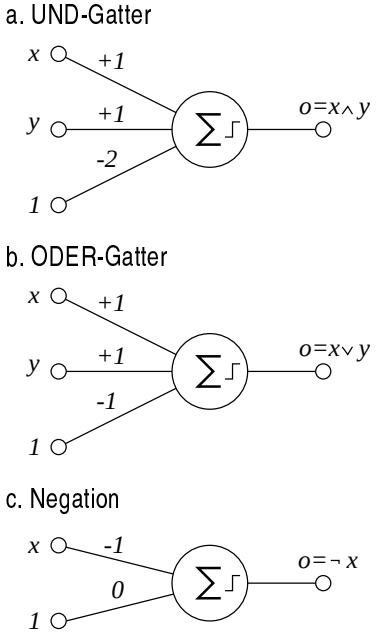

Abbildung 8.6: Logische Gatter mit Hilfe eines Perzeptrons realisiert. Abbildung a. zeigt ein UND-Gatter, Abbildung b. ein ODER-Gatter und Abbildung c. eine Negation.

8.3.1 Der Perzeptron Lernalgorithmus

Der Perzeptron Lernalgorithmus wurde erstmals 1958 von F. Rosenblatt vorgestellt. Dabei handelt es sich um ein einfaches überwachtes Lernverfahren. Zu Beginn der Lernphase werden die Gewichte zumeist zufällig initialisiert. Als ersten Lernschritt legt man am Eingang das gewünschte Eingangsmuster an und vergleicht das Ausgangssignal des Perzeptrons mit dem gewünschten Wert. Durch den Lernalgorithmus werden im nächsten Schritt die Gewichte so verändert, daß der Fehler reduziert wird. Die Änderung der Gewichte erfolgt nach folgender Formel: [Luger1997]

$$\Delta w_i = \eta(t - o)x_i \tag{8.9}$$

Δw_i	...	Gewichtsänderung des Gewichts w_i
η	...	Lernrate, bestimmt die Größe der Schritte mit denen das Gewicht verändert wird. $\eta > 0$
t	...	gewünschter Output des Trainingsbeispiels
o	...	tatsächlicher Output des Perzeptrons $o = f(net)$
x_i	...	Eingangssignal am Eingang i

Die Berechnung der neuen Gewichte erfolgt dann mit der Formel:

$$w_i^{neu} = w_i^{alt} + \Delta w_i \tag{8.10}$$

Da der Ausgang des Perzeptrons nur die diskreten Werte $\{0, 1\}$ annehmen kann, kann der Faktor $(t - o)$ aus Formel 8.9 nur die Werte $\{-1, 0, 1\}$ annehmen. Somit werden für jedes Gewicht w_i folgende Schritte vollzogen:

- Ist der gewünschte Output und der tatsächliche Output gleich, dann tue nichts.
- Ist der gewünschte Output $t = 0$ und der tatsächliche Output $o = 1$, so wird das Gewicht um $-\eta x_i$ erniedrigt.
- Ist der gewünschte Output $t = 1$ und der tatsächliche Output $o = 0$, so wird das Gewicht um ηx_i erhöht.

Mit der Lernrate η kann die Größe der Gewichtsänderung eingestellt werden. In der Praxis wählt man die Lernrate als Kompromiß zwischen möglichst großen Lernschritten (großes η) und möglichst geringer Veränderung des bereits gelernten (kleines η). Ist die Lernrate zu groß, kann ein Lernschritt zuvor gelerntes zerstören. Das Netz merkt sich jeweils das zuletzt präsentierte Muster am besten. Zuvor erlernte Muster bleiben jedoch nicht ausreichend genug gespeichert. Ist die Lernrate zu klein, sind dementsprechend viele Lernschritt notwendig. In der Praxis wird oft am Beginn der Lernphase die Lernrate groß gewählt, um mit großen Schritten näher zur Lösung zu gelangen, und mit der Fortdauer des Lernprozesses immer mehr verringert. [Köhle1990]

Die Vorgehensweise des Perzeptron Lernalgorithmus bewirkt, daß die Gewichte in der Art verändert werden, daß sich der durchschnittliche Fehler über die gesammte Menge der Testdaten minimiert. Existiert ein Satz von Gewichten w_i, sodaß für alle Trainingsdaten der korrekte Output ausgegeben wird, so konvergiert der Perzeptron Lernalgorithmus zu einer Lösung. Dies wurde 1969 von Minsky und Papert bewiesen [Luger1997]. Die Lösung muß jedoch nicht eindeutig sein, d.h. es können mehrere Sätze von Gewichten existieren, die zu den Trainingsdaten den korrekten Output liefern. Anschaulich wird dies, durch die, im folgenden eingeführte, grafische Interpretation der Arbeitsweise eines Perzeptrons.

Es stellt sich nun die Frage, welche Probleme sich mit der Hilfe eines Perzeptrons lösen lassen. Um dies zu veranschaulichen, kann ein Perzeptron als Klassifikator angesehen werden. Einem Eingangsmuster wird entweder die Zahl 0 oder 1 am Ausgang zugeordnet. Dies kann als Zuordnung zur Klasse „0" oder Klasse „1" aufgefaßt werden. Das Perzeptron trifft somit die Entscheidung, ob ein Eingangsmuster zur Klasse „0" oder „1" gehört.

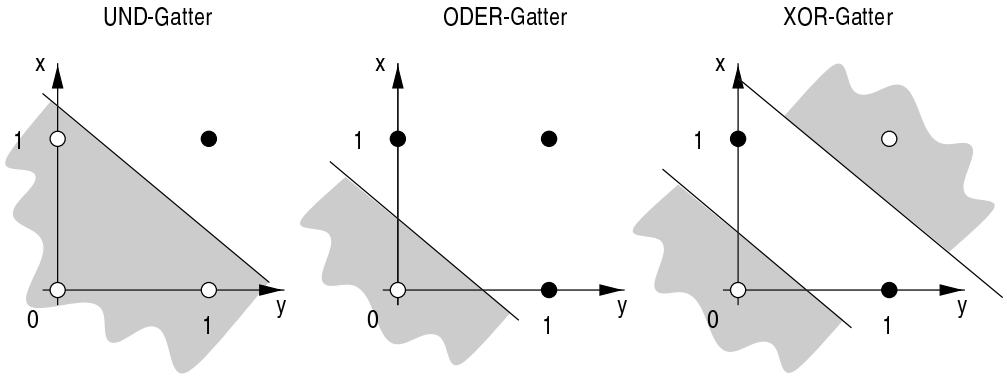

Abbildung 8.7: Zwei Klassen die sich durch eine Linie trennen lassen, sind linear separierbar.

Graphisch kann das Perzeptron als Klassifikator folgendermaßen veranschaulicht werden. Als Beispiel soll an dieser Stelle ein Perzeptron mit zwei Eingangsvariablen dienen, wie diese in Abbildung 8.6 a. und b. dargestellt sind. Ein Eingangssignal wird auf der x-Achse, das andere auf der y-Achse aufgetragen. In der Ebene, die sich so ergibt, werden die Ausgangssignale entsprechend ihren Koordinaten eingetragen. Dabei stellt ein leerer Kreis ein Element der Klasse „0", ein gefüllter Kreis ein Element der Klasse „1" dar. Dies ist in Abbildung 8.7 für das UND, ODER und das XOR-Gatter dargestellt.

Das Trainieren eines Perzeptrons kann somit als finden einer Trennung zwischen den beiden Klassen aufgefaßt werden. Für das UND- und das ODER-Gatter kann diese Trennung durch eine einzige Linie vorgenommen werden, welche die beiden Klassen trennt. In diesem Fall nennt man die Klassen linear separierbar. Für den Fall des XOR-Gatters sind dazu zwei Linien notwendig. Es läßt sich nun zeigen, daß linear separierbare Klassifikationsprobleme durch ein Perzeptron gelöst werden können. Nicht linear separierbare Klassifikationsprobleme können durch ein einfaches Perzeptron nicht gelöst werden. Dies soll anhand des Problems des XOR-Gatters demonstriert werden.

Soll ein Perzeptron das XOR-Problem lösen, so müssen 3 Gewichte w_1, w_2, w_3 existieren, sodaß Nachfolgende Ungleichungen erfüllt sind: [Rojas1993]

$$\begin{aligned}
x = 0, y = 0: &\quad 0 * w_1 + 0 * w_2 + 1 * w_3 < 0 \quad \text{Output } o = 0 \\
x = 0, y = 1: &\quad 0 * w_1 + 1 * w_2 + 1 * w_3 \geq 0 \quad \text{Output } o = 1 \\
x = 1, y = 0: &\quad 1 * w_1 + 0 * w_2 + 1 * w_3 \geq 0 \quad \text{Output } o = 1 \\
x = 1, y = 1: &\quad 1 * w_1 + 1 * w_2 + 1 * w_3 < 0 \quad \text{Output } o = 0
\end{aligned}$$

Vereinfacht kann dies angeschrieben werden als:

$$
\begin{array}{lcrl}
x=0, y=0 & : & w_3 < 0 & \text{Output } o=0 \\
x=0, y=1 & : & w_2 + w_3 \geq 0 & \text{Output } o=1 \\
x=1, y=0 & : & w_1 + w_3 \geq 0 & \text{Output } o=1 \\
x=1, y=1 & : & w_1 + w_2 + w_3 < 0 & \text{Output } o=0
\end{array}
$$

Es ist nun offensichtlich, wenn die ersten drei Ungleichungen erfüllt sind, kann Ungleichung vier nicht erfüllt werden. Es existieren somit keine Gewichte um das XOR-Problem mit Hilfe eines Perzeptrons zu lösen. [Rojas1993]

Diese Einschränkung der Anwendbarkeit des Perzeptrons wurde 1969 von Minsky und Papert nachgewiesen und führte zu einer Stagnation in der Euphorie auf dem Gebiet der Neuronalen Netze. Da sich mit Hilfe von Perzeptronen logische Gatter realisieren lassen, war bekannt, daß durch eine Hintereinanderschaltung von mehreren Gattern auch das XOR-Problem und andere komplexe Probleme gelöst werden können. Es war jedoch kein Lernalgorithmus für mehrschichtige Perzeptronennetzwerke bekannt. Erst im Jahr 1986 wurde von Rumelhart, Hinton und Williams der Back-Propagation Algorithmus als Lernverfahren für mehrschichtige Perzeptronennetzwerke (Multi Layer Perzeptron) veröffentlicht und belebte das Interesse am Forschungsgebiet Neuronaler Netze. [Rojas1993]

8.4 Der Backpropagation Algorithmus

Am Ende des vorigen Abschnitts wurde erklärt, daß Netze bei denen mehrere Perzeptronen hintereinander geschalten sind, ein breiteres Anwendungsspektrum abdecken, als ein einfaches Perzeptron. Ein solches Multilayer Perzeptron (MLP) ist in Abbildung 8.8 dargestellt. Dabei sind die Perzeptronen in Schichten (Layern) angeordnet. Das Eingangssignal wird an der Eingangs-Schicht (input layer) angelegt. Danach folgen ein oder mehrere Interne Schichten (hidden layer). Am Ende des Netzwerks folgt noch die Ausgangs-Schicht (output layer). Dabei ist jedes Neuron einer Schicht mit den Ausgängen aller Neuronen der Vorgängerschicht verbunden. Für die Angabe der Anzahl der Schichten eines Neuronalen Netzes gibt es in der Literatur keine einheitliche Vorgehensweise. In den meisten Fällen werden nur die Internen Schichten und die Ausgangsschicht gezählt. Ein Neuronales Netz mit einer Internen Schicht wird somit als zweischichtiges Neuronales Netz bezeichnet. [Haykin1994]

Die Anzahl der Internen Schichten die verwendet wird und wieviele Neuronen pro Schicht vorhanden sind, wird von der Komplexität der Aufgabe bestimmt. Grundsätzlich kann ein Perzeptron mit genügend vielen Neuronen in nur einer Internen Schicht und genügend vielen Trainingsbeispielen alle Aufgaben erfüllen,

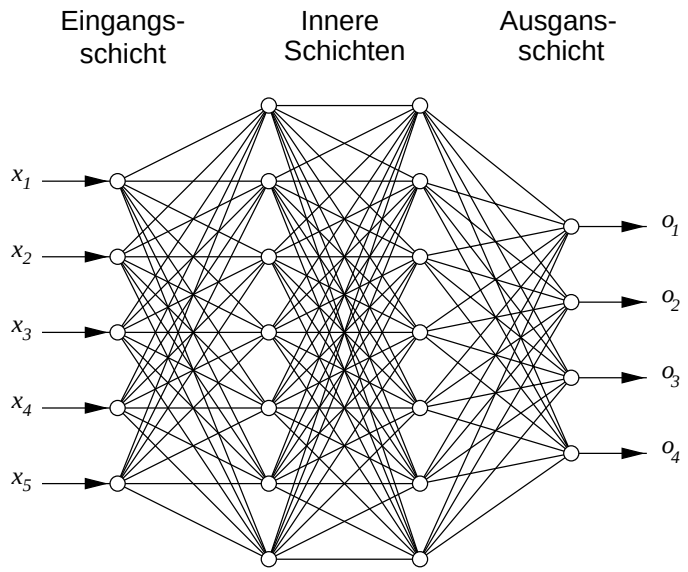

Abbildung 8.8: Beispiel eines dreischichtigen Multi Layer Perzeptrons mit fünf Eingangsneuronen, jeweils sieben Neuronen in den zwei Internen Schichten und vier Neuronen in der Ausgangsschicht.

die auch eine Turing Maschine berechnen kann. [Rojas1993] Es werden jedoch trotzdem Netzwerke mit mehreren internen Schichten verwendet, da diese unter Umständen leichter zu trainieren sind. Dies hängt jedoch von der Art der Anwendung ab.

Es erweist sich jedoch nicht als sinnvoll, ein Neuronales Netz auf eine Aufgabe zu trainieren, die algorithmisch einfacher gelöst werden kann. So ist es z.B. nicht zielführend, ein Neuronales Netz auf die Multiplikation zweier Zahlen zu trainieren. Das Netz müßte erst durch viele Trainingsbeispiele das „1x1" und dessen Anwendung erlernen. In diesem Fall erweist sich eine algorithmische Problemlösung als zielführender.

Der Backpropagation Algorithmus arbeitet in zwei Schritten. Im Feedforward Schritt wird am Eingang ein Testmuster angelegt, und der Output berechnet. Aus dem errechneten Output und dem gewünschten Output wird mit Hilfe der Fehlerfunktion der Fehler berechnet. Dieser Fehler wird nun im Backpropagation Schritt von der Ausgansschicht aus auf die Gewichte in den einzelnen Layern aufgeteilt und diese in der Art modifiziert, daß der Fehler reduziert wird. Als Fehlerfunktion wird dabei der quadratische Fehler $\tilde{E}$ verwendet.

$$\tilde{E} = \sum_k (t_k - o_k)^2 \tag{8.11}$$

o_k ... tatsächlicher Output des Neurons k der Ausgangssicht
k_k ... gewünschter Output des Neurons k für das konkrete Trainingsbeispiel

Der Backpropagation Algorithmus sucht das Minimum der Fehlerfunktion eines bestimmten Lernproblems durch Abstieg in der Gradientenrichtung entlang der Fehlerfunktion. Die Kombination der Gewichte eines Netzes, die den Berechnungsfehler minimiert, wird als Lösung des Lernproblems betrachtet. Der Gradient der Fehlerfunktion muß somit für alle Punkte des Gewichtsraums existieren, d.h. die partiellen Ableitungen der Fehlerfunktion nach den einzelnen Gewichten müssen überall definiert sein. Durch Verwendung der Sigmoiden-Funktion als Aktivierungsfunktion wird die Fehlerfunktion des Netzes stetig und überall differenzierbar. [Rojas1993] Zudem kann die Ableitung der Sigmoiden-Funktion einfach als

$$f'(x) = \left(\frac{1}{1+e^{-x}}\right)' = \frac{e^{-x}}{(1+e^{-x})^2} = f(x)(1-f(x)) \tag{8.12}$$

angeschrieben werden.

Herleitung des Backpropagation Algorithmus

An diese Stelle soll nun der Backpropagation Algorithmus für ein Multilayer Perzeptron mit einer internen Schicht hergeleitet werden. Ein solches Netzwerk ist in Abbildung 8.9 dargestellt. Die Herleitung ist angelehnt an [Pao1989].

Am Beginn der Trainingsphase werden alle Gewichte zufällig initialisiert. Im Feedforward Schritt wird ein Eingangsmuster am Eingang des Multilayer Perzeptrons angelegt und der Output des Netzes berechnet. Dazu wird zuerst der Netto-Input der Neuronen im Layer j berechnet.

$$net_j = \sum_i w_{ij} x_i \tag{8.13}$$

Der Output der Neuronen im Layer j ist somit

$$o_j = f(net_j). \tag{8.14}$$

Wobei $f(net)$ die Aktivierungsfunktion (hier die Sigmoide-Funktion) darstellt.

Im nächsten Schritt wird der Output des Layers k, der dem Output des Multilayer Perzeptrons entspricht, berechnet. Der Netto-Input der Neuronen im Layer k entspricht

$$net_k = \sum_j w_{jk} o_j \tag{8.15}$$

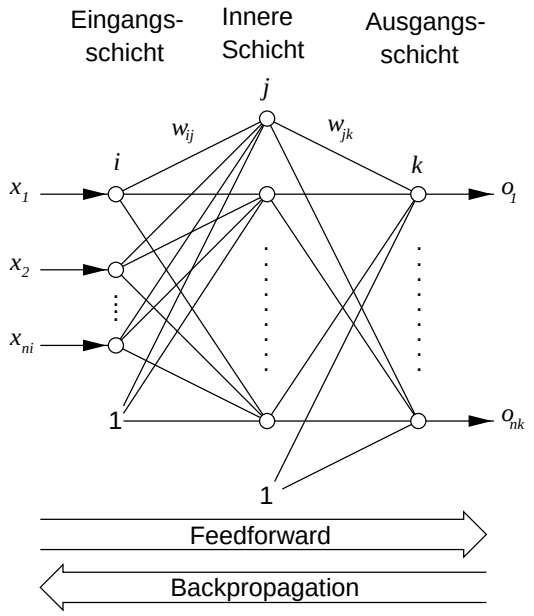

Abbildung 8.9: Beispiel eines zweischichtigen Multi Layer Perzeptrons mit n_i Eingangsneuronen, n_j Neuronen in der Internen Schicht und n_k Neuronen in der Ausgangsschicht. In der Feedforward-Richtung wird der Output für ein angelegtes Eingangsmuster berechnet. In der Backpropagation-Richtung wird in der Trainingsphase der Fehler am Output auf die Gewichte zwischen den Schichten aufgeteilt.

Für die weitere Herleitung haben die verwendeten Varialblen folgende Bedeutung:

x_i	...	Eingangssignal am Eingang i
w_{ij}	...	Gewichte zwischen der Eingangsschicht und der inneren Schicht
net_j	...	Netto-Input des Neurons j der inneren Schicht
o_j	...	Output des Neurons j der inneren Schicht und zugleich Eingangssignal für die Neuronen der Ausgangssicht
w_{jk}	...	Gewichte zwischen der inneren Schicht und der Ausgangssicht
net_k	...	Netto-Input des Neurons k der Ausgangssicht
o_k	...	tatsächlicher Output des Neurons k der Ausgangssicht
k_k	...	gewünschter Output des Neurons k für das konkrete Trainingsbeispiel

und der korrespondierende Output lautet

$$o_k = f(net_k). \tag{8.16}$$

Im Backpropagation Schritt wird nun mit Hilfe der Fehlerfunktion der quadratische Fehler E berechnet.

$$E = \frac{1}{2}\sum_k (t_k - o_k)^2 \tag{8.17}$$

Der Faktor $\frac{1}{2}$ wurde zur Erleichterung der nachfolgenden Rechnung eingeführt. Eine Strategie zur Korrektur der Gewichte und somit zur Reduzierung des Fehlers ist, daß man die inkrementale Gewichtsänderung Δw_{jk} proportional zu $-\frac{\partial E}{\partial w_{jk}}$ wählt. Somit folgt

$$\Delta w_{jk} = -\eta \frac{\partial E}{\partial w_{jk}}. \tag{8.18}$$

Wobei η, analog zum Perzeptron Lernalgorithmus, die Lernrate darstellt. Der Fehler E ist jedoch in Gleichung 8.17 als Funktion des Outputs o_k ausgedrückt. Wobei

$$o_k = f(net_k) \tag{8.19}$$

und net_k der Netto-Input des k-ten Neurons definiert ist durch die gewichtete Summe des Outputs des vorhergehenden Layers.

$$net_k = \sum_j w_{jk} o_j \tag{8.20}$$

Mit Hilfe der Kettenregel kann die partielle Ableitung $\frac{\partial E}{\partial w_{jk}}$ ausgedrückt werden durch

$$\frac{\partial E}{\partial w_{jk}} = \frac{\partial E}{\partial net_k}\,\frac{\partial net_k}{\partial w_{jk}}. \tag{8.21}$$

Setzt man in den Ausdruck $\frac{\partial net_k}{\partial w_{jk}}$ Gleichung 8.20 ein, so erhält man

$$\frac{\partial net_k}{\partial w_{jk}} = \frac{\partial}{\partial w_{jk}} \sum_j w_{jk} o_j. \tag{8.22}$$

Da die partielle Ableitung nur nach einer Komponente der Summe gebildet wird, kann man schreiben

$$\frac{\partial net_k}{\partial w_{jk}} = \frac{\partial}{\partial w_{jk}}\, w_{jk} o_j = o_j. \tag{8.23}$$

Definiert man nun δ_k als

$$\delta_k = -\frac{\partial E}{\partial net_k} \tag{8.24}$$

und setzt dann in Gleichung 8.18 ein, so erhält man

$$\Delta w_{jk} = \eta\, \delta_k o_j. \tag{8.25}$$

Um den Ausdruck $\delta_k = -\frac{\partial E}{\partial net_k}$ zu berechnen, verwendet man wieder die Kettenregel.

$$\delta_k = -\frac{\partial E}{\partial net_k} = -\frac{\partial E}{\partial o_k}\,\frac{\partial o_k}{\partial net_k} \tag{8.26}$$

Wobei man für den ersten Faktor

$$\frac{\partial E}{\partial o_k} = \frac{\partial}{\partial o_k}\,\frac{1}{2}\,\sum_m (t_m - o_m)^2 \tag{8.27}$$

schreiben kann, wenn man die Definition der Fehlerfunktion aus Gleichung 8.17 einsetzt. Auch hier wird die partielle Ableitung nur nach einer Komponente der Summen gebildet. Darum kann man vereinfacht schreiben:

$$\frac{\partial E}{\partial o_k} = \frac{\partial}{\partial o_k}\,\frac{1}{2}(t_k - o_k)^2 = -(t_k - o_k) \tag{8.28}$$

Für den zweiten Faktor aus Gleichung 8.26 gilt

$$\frac{\partial o_k}{\partial net_k} = f'(net_k). \tag{8.29}$$

Da als Aktivierungsfunktion die Sigmoide-Funktion verwendet wird, kann man nach Gleichung 8.12 schreiben

$$\frac{\partial o_k}{\partial net_k} = f'(net_k) = f(net_k)(1 - f(net_k)) = o_k(1 - o_k). \tag{8.30}$$

Somit erhält man für δ_k

$$\delta_k = (t_k - o_k)\, o_k(1 - o_k). \tag{8.31}$$

Eingesetzt in Gleichung 8.25 erhält man für die Änderung der Gewichte

$$\Delta w_{jk} = \eta\,(t_k - o_k)\, o_k(1 - o_k)\, o_j. \tag{8.32}$$

Diese Formel gilt allerdings nur für die Gewichte der Neuronen in der **Ausgangsschicht**. Für die Gewichte der Internen Neuronen setzt man analog an.

$$\begin{aligned}
\Delta w_{ij} &= -\eta\,\frac{\partial E}{\partial w_{ij}} \\
&= -\eta\,\frac{\partial E}{\partial net_j}\,\frac{\partial net_j}{\partial w_{ij}} \\
&= -\eta\,\frac{\partial E}{\partial net_j}\, o_i \\
&= \eta\left(-\frac{\partial E}{\partial o_j}\,\frac{\partial o_j}{\partial net_j}\right) o_i = \eta\,\overbrace{\left(-\frac{\partial E}{\partial o_j}\right) f'(net_j)}^{\delta_j}\, o_i \\
&= \eta\,\delta_j o_i
\end{aligned} \tag{8.33}$$

Für den Fall, daß nur eine Interne Schicht vorhanden ist, entspricht in obiger Gleichung $o_i = x_i$. Da sich der Faktor $\frac{\partial E}{\partial o_j}$ nicht direkt berechnen läßt, schreibt man ihn unter Zuhilfenahme von bekannten oder berechenbaren Ausdrücken an.

$$-\frac{\partial E}{\partial o_j} = -\sum_k \frac{\partial E}{\partial net_k}\,\frac{\partial net_k}{\partial o_j} = \sum_k \left(-\frac{\partial E}{\partial net_k}\right)\,\frac{\partial}{\partial o_j}\sum_m w_{mk}o_m$$

$$= \sum_k \overbrace{\left(-\frac{\partial E}{\partial net_k}\right)}^{\delta_k \text{ nach Definition 8.24}} w_{jk} = \sum_k \delta_k w_{jk} \quad (8.34)$$

Aus Gleichung 8.33 und 8.34 folgt nun somit

$$\delta_j = f'(net_j)\sum_k \delta_k w_{jk}. \quad (8.35)$$

Das δ eines Inneren Neurons kann somit aus den δ's der Vorgängerschicht berechnet werden. Beginnt man bei der letzten Schicht, der Ausgangsschicht, kann man sich aus Geichung 8.31 δ_k berechnen. Anschließend kann man den Fehler ,,rückwärts-propagieren" um die δ's für die Inneren Schichten zu berechnen.

Da als Aktivierungsfunktion die Sigmoide-Funktion verwendet wird gilt nach Gleichung 8.12

$$f'(net_j) = f(net_j)(1 - f(net_j)) = o_j(1 - o_j). \quad (8.36)$$

Somit kann für δ_j

$$\delta_j = o_j(1 - o_j)\sum_k \delta_k w_{jk} \quad (8.37)$$

geschrieben werden. Für die Änderung der Gewichte in der Inneren Schicht ergibt sich somit

$$\Delta w_{ij} = \eta\, o_j(1 - o_j)\, o_i \sum_k \delta_k w_{jk}. \quad (8.38)$$

Wobei für ein Netzwerk mit nur einer Inneren Schicht $o_i = x_i$ gilt.

Zusammenfassend erfolgt die Gewichtsänderung während des Lernprozesses eines Multi Layer Perzeptrons mit Hilfe des Backpropagation Algorithmus nach folgenden Formeln:

$$\delta_k = (t_k - o_k)\, o_k(1 - o_k)\,; \quad \Delta w_{jk} = \eta\, \delta_k o_j \quad \text{für die Ausgangsschicht} \quad (8.39)$$

$$\delta_j = o_j(1 - o_j)\sum_k \delta_k w_{jk}\,; \quad \Delta w_{ij} = \eta\, \delta_j o_i \quad \text{für die inneren Schichten} \quad (8.40)$$

Für die Berechnung der δ_j's der inneren Schichten werden, nach obigen Formeln, jeweils die δ_k's der Vorgängerschicht in Backpropagation-Richtung verwendet. Der Fehler wird somit vom Ausgang zum Eingang ,,backpropagiert".

Bei der Realisierung eines Neuronalen Netzes mit der Hilfe eines Multi Layer Perzeptrons muß darauf geachtet werden, daß die Topologie des Netzes in einem sinnvollen Zusammenhang mit der Anzahl der zur verfügungstehenden Trainingsbeispiele steht. Bei der Festlegung der Topologie wird festgelegt, aus wievielen Schichten ein Multi Layer Perzeptron bestehen soll und wieviele Neuronen sich in jeder internen Schicht befinden. Somit wird die Anzahl der Gewichte des Netzes festgelegt. Um jetzt ein sinnvolles trainieren des Netzes zu ermöglichen, sollte die Anzahl der Trainingsbeispiele um den Faktor 10 größer sein, als die Anzahl der Gewichte des gesamten Netzwerks. Ist obiger Zusammenhang nicht erfüllt, so existieren zuviele freie Parameter (Gewichte) im Netzwerk, alsdaß ein sinnvolles Training möglich wäre. [Haykin1994]

In Neuronalen Netzen ist das Wissen das aus den Trainingsdaten erlernt wurde in den Gewichten verteilt über das Netz gespeichert. Aus diesem gespeicherten Wissen kann der Mensch jedoch leider keine Rückschlüsse auf Gesetzmäßigkeiten und Regeln ziehen. Das erlernte Wissen ist somit für den Menschen nicht unmittelbar zugänglich, das Neuronale Netz arbeitet als „Black Box". Im folgenden Kapitel soll aus der Verbindung von Fuzzy Logic und Neuronalen Netzen ein Ansatz gezeigt werden, bei dem unscharfe Regeln ähnlich wie bei Neuronalen Netzen erlernt werden können. Durch die Verwendung von Neuronalen Fuzzy Systemen wird es somit möglich, Einblicke in das vom System erlernte Wissen zu erhalten. [Kruse1997]

Kapitel 9

Neuronale Fuzzy Systeme

In den beiden vorangegangenen Abschnitten über Fuzzy Logic und Neuronale Netze wurde einerseits ein regelbasiertes System vorgestellt, welches durch die Verwendung von linguistischen Werten einfach zu modellieren ist, und andererseits ein lernfähiges System, welches mit der Hilfe von Trainingsdaten auf seine zukünftige Aufgabe trainiert wird. Durch die Kombination von Neuronalen Netzen und der Fuzzy Logic in Neuronalen Fuzzy Systemen wird nun versucht, die Schwächen der einen Technik durch die Stärken der anderen zu kompensieren. Daher ist es zu Beginn dieses Abschnitts notwendig, die Vor- und Nachteile der Fuzzy Logic und der Neuronalen Netze zu analysieren. Eine entsprechende gegenüberstellung ist in Tabelle 9.1 zu finden. [Seraphin1994]

Betrachtet man die oben angeführten Vor- und Nachteile der beiden Techniken, wird deutlich, daß eine geeignete Kombination der beiden Ansätze in der Lage ist, die Vorteile zu vereinen und Nachteile zu kompensieren. Das Hauptargument für eine Kombination der Fuzzy Logic mit Neuronalen Netzen ist deren Lernfähigkeit. Der Neuro-Fuzzy-Ansatz versucht somit die Transparenz der intuitiven Regeln von Fuzzy Systemen mit der Lernfähigkeit von Neuronalen Netzen zu vereinen. Ein daraus entstandenes System sollte in der Lage sein, linguistische Regeln und/oder Zugehörigkeitsfunktionen zu erlernen oder bestehende zu optimieren. [Nauck1994]

Bei der Kombination von Neuronalen Netzen und Fuzzy Systemen lassen sich grundsätzlich zwei Ansätze unterscheiden. In *kooperativen Systemen* arbeitet das Neuronale Netz gänzlich unabhängig vom Fuzzy System. Die Kopplung der beiden Techniken besteht darin, daß Parameter des Fuzzy Systems durch das Neuronale Netz bestimmt bzw. optimiert werden. Das somit erzeugte Fuzzy System arbeitet im Betrieb ohne das Neuronale Netz. In *hybriden Systemen* werden hingegen beide Prinzipien vereinigt. Man kann ein hybrides System sowohl als spezielles Neuronales Netz, wie in Kapitel 8 ausgeführt, als auch als konventionelles

Neuronale Netze	**Fuzzy Logic**
Vorteile	
• Kein mathematisches Modell notwendig • Kein Regelwissen notwendig • Es stehen unterschiedliche Lernalgorithmen zu Verfügung	• Kein mathematisches Modell notwendig • A-priori (Regel-)Wissen nutzbar • Einfache Interpretation und Implementierung
Nachteile	
• Black-Box Verhalten • Kein Regelwissen extrahierbar • Anpassung an veränderte Aufgabenstellung ist eventuell schwierig und kann eine Wiederholung des Lernvorgangs erfordern • Kein A-priori Wissen verwendbar • Der Lernvorgang konvergiert nicht garantiert	• Regelwissen muß verfügbar sein • Nicht Lernfähig • Keine formalen Methoden für das „Tuning" bekannt • Anpassung an veränderte Parameter eventuell schwierig • Ein „Tuning"-Versuch kann erfolglos bleiben

Tabelle 9.1: Gegenüberstellung der Vor- und Nachteile von Neuronalen Netzen und der Fuzzy Logic. [Nauck1994]

Fuzzy System, wie in Kapitel 7 vorgestellt, interpretieren. Eine Trennung der Teilsysteme der einen oder anderen Art ist nicht mehr möglich. [Kruse1997]

Mit dem Begriff „Neuro Fuzzy" werden jedoch auch Systeme bezeichnet, die getrennt ein Fuzzy System und ein Neuronales Netz zur Behandlung von Teilen des Gesamtproblems enthalten; z.B. kann das Neuronale Netz jenen Teil des Problems behandeln, über den kein Regelwissen vorhanden ist, das Fuzzy System bearbeitet jenen Teil, in dem Wissen über die Zusammenhänge der Größen bekannt ist. [Kruse1997]

Nachfolgend werden kooperative und hybride Systeme näher besprochen.

9.1 Kooperative Systeme

Ein Fuzzy System ist einerseits bestimmt durch die Definition der unscharfen Mengen, das heißt der Zugehörigkeitsfunktionen die hinter den linguistischen Ausdrücken, wie z.B. „niedrige Temperatur" stehen, und andererseits durch die Angabe der Regeln, die die linguistischen Begriffe miteinander verknüpfen. Sowohl die Mengen als auch die Regeln sind durch wenige Parameter beschreibbar. Ein kooperatives Neuro Fuzzy System kann nun entweder unscharfe Mengen bei festelegten Regeln lernen oder die Regeln, bei unveränderlichen unscharfen Mengen. Variiert man die unscharfen Mengen zugleich mit den Regeln, so existieren für kooperative Systeme zuviele freie Parameter. Diese komplexe Aufgabe wird erst durch die Verwendung von hybriden Systemen, die in Abschnitt 9.2 betrachtet werden, bewältigt. Es kann jedoch sinnvoll sein, zwischen beiden Lernverfahren abzuwechseln. [Kruse1997]

Lernen von unsicheren Mengen

Im Fall des Erlenens von unsicheren Mengen geht man davon aus, daß die Regeln, mit denen das System arbeitet, korrekt sind. Eine Fehlfunktion des Systems wird somit auf die nichtopimale Modellierung der Fuzzy Mengen zurückgeführt. In einer Anwendung läßt man z.B. die Regel „Wenn die Temperatur niedrig ist, drehe das Ventil weit auf." unangetastet und veranlaßt das System seine Vorstellung von dem linguistischen Begriff „niedrige Temperatur" zu revidieren. Zu diesem Zweck kann man ein modifiziertes Backpropagation-Verfahren anwenden, welches Anstelle der Gewichte die Lage und Form der Zugehörigkeitsfunktion verändert. Man verwendet somit nicht explizit ein Neuronales Netz, sondern nur dessen Lernalgorithmus.

Da es sich beim Backpropagation-Verfahren, wie in Abschnitt 8.4 erläutert, um ein Gradientenverfahren handelt, muß die entstehende Fehlerfunktion am

Ausgang differenzierbar sein. Aus diesem Grund dürfen für die Fuzzymengen nur differenzierbare Zugehörigkeitsfunktionen verwendet werden. Da die Dreiecks- und die Trapezfunktion diese Eigenschaft nicht erfüllen, sind sie für die Anwendung in kooperativen Neuro Fuzzy Systemen ungeeignet. An ihrer Stelle wird vorzugsweise die Gaus'sche Glockenkurve als Zugehörigkeitsfunktion verwendet. Weiters müssen die in Abschnitt 7 vorgestellten Verfahren zur Inferenz und zur Komposition durch Verfahren ersetzt werden, die die Differenzierbarkeit der Fehlerfunktion aufrecht erhalten. Daher wird anstelle der Minimumsbildung bei der Inferenz die Produktbildung verwendet. Die Maximumsbildung bei der Komposition und die anschließende Defuzzyfizierung wird durch einen Regler des Sugeno-Typs[1] ersetzt. Bei einem Regler des Sugeno-Typs handelt es sich um eine Einheit, die direkt aus den Werten der Inferenzbildung mit der Hilfe einer Funktion einen Ausgangswert ermittelt. Eine Defuzzyfizierung ist dabei nicht mehr notwendig. [Nauck1994]

Bei Systemen, die die Fuzzy Mengen durch Lernen bestimmen, muß darauf geachtet werden, daß die wichtigsten Eigenschaften von Fuzzy Systemen, die Transparenz und Interpretierbarkeit, durch den Lernprozeß nicht beeinträchtigt werden. Es muß z.B. darauf geachtet werden, daß zwei Regeln in denen der linguistische Ausdruck „niedrige Temperatur" vorkommt, die entsprechenden Zugehörigkeitsfunktionen nicht unabhängig voneinander verändern und somit unterschiedlich interpretieren. Weiters muß darauf geachtet werden, daß die Fuzzy Menge für den Ausdruck „niedrige Temperatur" nicht in einen Bereich verschoben wird, den man umgangssprachlich als heiß bezeichnen würde. Dies würde der ursprünglichen Intuition bei der Erstellung der Regeln widersprechen. Dies kann durch eine geeignete Beschränkung des erlaubten Variationsbereichs erfolgen. [Kruse1997]

Lernen von Regeln

Bei der Strategie des Erlernens von Regeln bei vorgegebenen unscharfen Mengen wird dem zu trainierendem System eine große Anzahl von Trainingsbeispielen vorgelegt. Für das oben verwendete Beispiel der Heizungsregelung wären das Paare aus Temperaturwerten und der daraufhin gewählten Stellung des Heizkörperventils. Dies kann auch interpretiert werden, als würde das System einem erfahrenem Bediener bei seiner Arbeit beobachten. Während der Lernphase wird nicht das eigentliche Fuzzy System modifiziert, sondern ein Neuronales Netz trainiert. Im Verlauf des Lernprozesses kristallisieren sich gewisse Regelmäßigkeiten im Verhalten des Bedieners heraus. So wird sich z.B ein Neuron auf die Erkennung der Kombination von niedriger Temperatur und weit offenem Heizungsventil spezialisieren, falls diese Kombination oft genug in den Trainingsdaten vorkommt.

[1] Nähere Informationen zum Sugeno-Regler finden sich unter [Nauck1994].

Erst nach dem Abschluß der Lernprozesses werden in einem gesonderten Rechenschritt die gefundenen Regelmäßigkeiten mit Hilfe der vorgegebenen unscharfen Mengen in Regeln ausgedrückt. Auf diesem Weg wird für ein System ohne Vorwissen eine Regelbasis gewonnen. Aus der Beobachtung des Systems ,,Wenn X vorliegt, dann stellt der Bediener Y ein." wird nach der Umsetzung auf die unscharfen Mengen die Regel ,,Wenn X vorliegt, stelle Y ein.". [Kruse1997]

Die oben beschriebenen Systeme lernen vor dem eigentlichen Betrieb. Man spricht auch von *Offline Systemen*. Das resultierende Fuzzy System gelangt erst zum Einsatz, wenn der Lernprozeß bereits abgeschlossen ist. Ein *Online System* lernt hingegen während des laufenden Betriebs. Zu Beginn wird ein solches Online System mit einer Regelbasis ausgestattet, in der grob entworfene Fuzzy Mengen miteinander verknüpft werden. Während das System arbeitet wird nun seine Leistung bewertet. Weicht die Ausgabe erheblich vom gewünschten Wert ab, so wird eine Fehlerkorrektur notwendig. Diese kann z.B. mit der Hilfe eines modifizierten Backpropagation Algorithmus erfolgen. Die Fehlerkorrektur kann sich dabei entweder auf die unscharfen Mengen oder auf die Regeln auswirken.

Ein Online System ist grundsätzlich nur in Fällen anwendbar, in denen ein Versagen des Systems keinen großen Schaden anrichten kann. So darf z.B. eine Regelung für einen Kühlraum nie eine Temperatur von $+20°$ C zulassen, um erst beim Auftreten dieser Situation eine Regel für diesen Systemzustand zu lernen. Als Lösung dieses Problems kann man ein System vor dem Einsatz mit Hilfe einer Simulation lernen lassen.

Der Vorteil von Online Systemen ist, daß sie in der Lage sind, während des Betriebs weiter zu lernen. Dadurch können sie sich auf Änderungen in ihrer Aufgabenstellung anpassen. Bei solchen Online Systemen wird das Neuronale Netz, bzw. der neuronale Lernalgorithmus nicht nach dem Abschluß des Trainings entfernt. [Kruse1997]

9.2 Hybride Systeme

In hybriden Neuro Fuzzy Systemen sind die Eigenschaften der Fuzzy Logic und der Neuronalen Netze untrennbar vereint. Zu den wesentlichen Eigenschaften dieses Ansatzes zählen:

- Wissen wird in transparenten Regeln gespeichert
- Die Verwendung von Linguistische Ausdrücke
- Die Lernfähigkeit

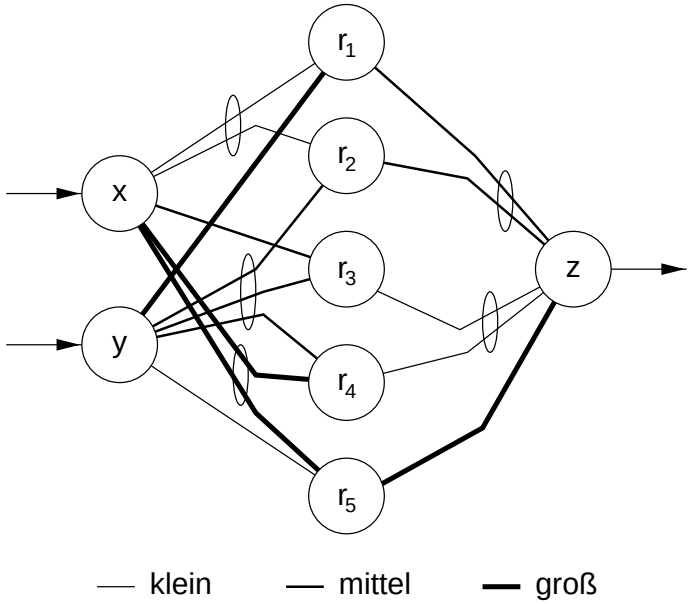

Abbildung 9.1: Hybrides Neuro Fuzzysystem nach dem NEFCON-Modell. Die Verbindungen zwischen den Neuronen entsprechen den unscharfen Mengen. Durch die Strichstärke der Verbindung ist die Zugehörigkeit zu einer bestimmten Fuzzy Menge ausgedrückt. Die Neuronen der Eingangsschicht entsprechen den Eingangsgrößen, die der inneren Schicht den Fuzzy Regeln, die der Ausgangsschicht, der Ausgangsgröße. Die Regel 1 lautet somit: „Wenn x klein ist und y groß ist, dann ist z mittel." [Kruse1997]

Vereint werden diese Eigenschaften im formalen Modell des Fuzzy-Perzeptrons. Das Fuzzy-Perzeptron ist ähnlich einem zweischichtigen Multi Layer Perzeptron aufgebaut. Dabei macht man sich zunutze, daß die Abarbeitung innerhalb eines Fuzzy Systems in zwei Schritten erfolgt. Im Ersten Schritt werden die Regeln ausgewertet (Inferenz), im zweiten Schritt erfolgt die Vereinigung dieser Auswertung (Komposition). Jeder dieser Schritte besteht dabei aus einer Anzahl getrennt ausführbarer Einzelaktionen. Diese Einzelaktionen können parallel in einem modifizieren Neuron abgearbeitet werden, sofern die Verbindungen entsprechend der des Fuzzy Systems geknüpft sind. [Seraphin1994]

Das Fuzzy-Perzeptron ist, wie oben bereits erwähnt, von einem zweischichtigen Multi Layer Perzeptron abgeleitet. Anstelle der Gewichte der einzelnen Verbindungen treten beim Fuzzy-Perzeptron die unscharfen Mengen. Anstelle der gewichteten Summe und der Aktivierungsfunktion tritt bei den Neuronen in der inneren Schicht die Fuzzyfizierung und die Inferenz, bei den Neuronen in der Ausgangsschicht die Komposition und die Defuzzyfizierung. Ein solches Fuzzy-Perzeptron ist in Abbildung 9.1 dargestellt.

Damit ein solches Modell immer eindeutig als Fuzzy System interpretiert werden kann, müssen ähnliche Einschränkungen wie bei kooperativen Systemen getroffen werden. So muß darauf geachtet werden, daß sich die unscharfen Mengen nicht in einem Bereich verschieben, der nicht mehr der Interpretation des linguistischen Wertes entspricht. Weiters stellt man durch Kopplung von Verbindungen sicher, daß sich anfangs gleiche unscharfe Mengen, während des Lernprozesses, nicht unterschiedlich entwickeln. Diese Kopplung ist in Abbildung 9.1 durch Ringe um die gekoppelten Verbindungen dargestellt. [Nauck1994]

Der Lernalgorithmus zur Anpassung der Fuzzy-Mengen beruht auf einer Variation des Backpropagation Algorithmus. Bei dem an der Otto-von-Guericke-Universität Magdeburg entwickelten NEFCON-Modell (Neuro Fuzzy Control) wird der Fehler am Ausgang des Systems nicht durch einen konkreten Wert beschrieben, sondern durch eine unscharfe Menge. Dadurch erhält der Benutzer die Freiheit, nicht nur die Regeln umgangssprachlich auszudrücken, sondern auch das Ziel des Systems. Dies kommt vielen Anwendungsfällen in der Realität entgegen. So es ist z.B. nicht notwendig, daß die Regelung einer Raumheizung die Temperatur auf ein Zehntel Grad genau auf 22° C hält. Als Ziel des Systems genügt die Formulierung „Wenn die Temperatur angenehm ist, dann ist die Leistung des Systems hoch (und somit der Fehler gering)." Der linguistische Wert „angenehme Temperatur" wird dabei durch eine Fuzzy Menge beschrieben. Das System wertet diese Regel wie alle anderen aus und zieht das Ergebnis zur Korrektur der Fuzzy Mengen heran.

Für das Erlernen der Fuzzy Regeln stehen zwei Methoden zur Verfügung: Man beginnt ohne Regeln und fügt eine Regel nach der anderen hinzu bis das System eine zufriedenstellende Leistung erbringt, oder man beginnt mit einer Basis aus allen möglichen erzeugbaren Regeln und streicht nicht benötigte nacheinander wieder heraus. Die erste der oben angeführten Methoden ist weniger aufwendig, jedoch nicht für das online-lernen anwendbar, da ein solches System zu Beginn der Lernphase nur durch Raten die einzufügende Regel bestimmen kann um die Regelbasis in weiterer Folge langwierig nachzubessern.

Die Zweite Methode ist für online lernende Modelle geeignet. Die Komplexität eines solchen Systems steigt jedoch rasch mit der Anzahl der Ein- und Ausgangsvariablen und der Anzahl der Fuzzy Mengen, da für jede Kombination eine Regel generiert werden muß. Für ein Beispiel mit zwei Eingangsvariablen, einer Ausgangsvariable und einer Fuzzy Partitionierung von je acht Fuzzy Mengen zu jeder Variable existieren bereits 512 mögliche Regeln. Erhöht man die Zahl der Eingangsvariablen auf vier, so sind es bereits 32.768 Regeln. Die Komplexität des Systems steigt somit exponentiell mit der Anzahl der Freiheitsgrade. Daraus ist ersichtlich, daß die Regelbasis nur für kleine Systeme vollständig erlernbar ist. In vielen Fällen lassen sich jedoch für einen Teil des Problems Regeln angeben, sodaß nur die fehlenden Regeln hinzugelernt werden müssen. [Kruse1997]

Die Vorgänge während des Lernvorganges eines hybriden Neuro Fuzzy Systems, welches mit einer Basis aus allen möglichen erzeugbaren Regeln startet, soll an dieser Stelle mit Hilfe des Beispiels der Heizungsregelung erläutert werden. Im ersten Schritt wird durch die Definition der Fuzzy Mengen festgelegt, welche Temperaturen man als angenehm empfindet. Dies kann z.B. durch die Einteilung der Temperaturskala in die linguistischen Werte „zu niedrig", „angenehm" und „zu hoch" erfolgen. Anschließend formuliert man das Ziel des Systems, die Raumtemperatur im angenehmen Bereich zu halten.

Da das System zu Beginn des Lernprozesses noch keine Informationen über die korrekte Vorgehensweise besitzt, wird man anfangs eine „chaotische" Heizperiode überstehen müssen, in der das System herauszufinden versucht, welche Fuzzy Regeln für eine angenehme Raumtemperatur relevant sind. In dieser Phase überprüft das System zu welchem Grad die aktuelle Raumtemperatur mit der vorgegebenen Vorstellung einer angenehmen Temperatur übereinstimmt. Besteht keine ausreichend große Übereinstimmung, so weiß das System, daß es sich in einem Fehlzustand befindet und bewertet jene Regeln schlecht, die zu diesem Zustand geführt haben. Entspricht die Temperatur jedoch der Vorstellung von „angenehm", so werden die dafür verantwortlichen Regeln als gut bewertet. Nach einer Weile hat das System durch Probieren und Bewerten selbständig eine Regelbasis für das Problem gefunden. In einem weiteren Schritt kann nun noch eine Optimierung des Systems durch eine Anpassung der Fuzzy Mengen erfolgen. Dies erfolgt, wie oben angeführt, mit der Hilfe eines modifizierten Backpropagation Algorithmus. [Kruse1997]

Nach dem Abschluß der Lernphase kann man sich die Regeln und die Fuzzy Mengen anzeigen lassen und auf ihre Plausibilität hin überprüfen. Man erhält somit nicht nur eine Lösung des Problems, sondern auch Wissen über die Gesetzmäßigkeiten der Lösung. Ein hybrides Neuro Fuzzy System kann somit auch zum Wissenserwerb eingesetzt werden. Dabei entwickelt das System, ähnlich dem Problemlösungsverhalten des Menschen, durch Vorgabe eines Ziels und durch die Beobachtung des Prozeßverlaufs eine geeignete Vorgehensweise. Die Möglichkeiten des Wissenserwebs mit der Hilfe von hybriden Neuro Fuzzy Systemen sind jedoch auf kleine, eingegrenzte Probleme beschränkt, da für größere Probleme die Komplexität exponentiell anwächst. [Kruse1997]

Kapitel 10

Intelligente Agenten

In den vorhergegangenen Abschnitten wurden verschiedene Techniken der künstlichen Intelligenz vorgestellt, die in der Wissensverarbeitung ihre Anwendung finden. Viele Problemstellungen der Wissensverarbeitung sind jedoch sehr komplex und umfangreich. Man denke nur an die effektive Informationssuche im Internet. In vielen Fällen ist es daher nicht möglich, ein konventionelles Programm zu erstellen, welches seine Aufgabenstellung in zufriedenstellender Qualität löst. Unter konventionellen Programmen werden an dieser Stelle Systemen verstanden, welche die Lösung des Gesamtproblems mit der Hilfe eines komplexen zentralen Programms verfolgen. Diese Programme können unter Umständen auch in einzelne Module gegliedert sein. [Jennings1998]

Die Verwendung von intelligenten Agenten stellt nun einen Ansatz dar, mit dessen Hilfe komplexe, verteilte Probleme bewältigt werden können. Zu diesem Zweck wird das Gesamtproblem in kleinere Teilprobleme zerlegt, die, von eventuell verteilten, autonomen Programmen, den Agenten, ausgeführt werden. Im folgenden Abschnitt sollen kurz einige Problemstellungen angerissen werden, welche einen Überblick über die Motivation zur Verwendung von intelligenten Agenten geben sollen.

10.1 Motivation für die Entwicklung von intelligenten Agenten

Das Internet stellt einen stark skalierten, ständig in Wachstum begriffenen Wissenspeicher dar, in dem die Informationsressourcen verteilt über das Netz auf unterschiedlichen Rechnern gespeichert sind. Im Internet existiert keine übergeordnete Instanz, welche den laufenden Zuwachs an Informationen koordiniert und inhaltlich erschließt. Ein konventionelles Software-System ist somit nicht in der

Lage, eine umfassende Suchhilfe — für alle im Internet zugänglichen Informationen — anzubieten. Um zu einer Lösung für dieses Problem zu gelangen, wird an der Entwicklung von Systemen gearbeitet, die auf der Verwendung von verteilten, eventuell mobilen, intelligenten Agenten aufbauen. [Jennings1998]

Ein weiteres Problem, welches mit der Hilfe von intelligenten Agenten zu lösen versucht wird, ist einerseits die steigende Verbreitung von Computersystemen und der damit verbundenen steigenden Anzahl der Benutzer, und andererseits der steigende Funktionsumfang der Programme. Mit der wachsenden Benutzerzahl sinkt die Qualifikation der Benutzer. Im gleichen Zuge werden jedoch die Programme immer umfangreicher und komplexer, und fordern somit eine längere Einschulungszeit. Um modernen Computersysteme, trotz des Funktionsumfangs für den einfachen Anwender benutzbar zu gestalten, werden dem Anwender intelligente Agenten zur Seite gestellt, welche ihm eine Hilfestellung geben und bei der Bedienung unterstützen sollen. [Caglayan1998]

Ein weiteres Problem, welches an dieser Stelle zur Motivation angeführt werden soll, ist die Vernetzung von Haushaltsgeräten. Die Haushaltsgeräte werden innerhalb einer Wohneinheit z.B. über das Leitungsnetz miteinander vernetzt. In jedem Gerät ist ein Agent integriert, welcher mit den anderen Geräten und einem zentralen Rechner kommuniziert. Ziel der Kommunikation ist es, z.B. die Verbrauchslast gleichmäßig über den Tag zu verteilen, Energie zu sparen und die Sicherheit zu erhöhen.

10.2 Definition von intelligenten Agenten

Die in den obigen Beispielen angedeuteten Eigenschaften von Agenten sollen in diesem Abschnitt in einer allgemeinen Definition von Agenten festgehalten werden. Die Forschung auf dem Gebiet der intelligenten Agenten stellt ein Teilgebiet der künstlichen Intelligenz dar. [Nwana1998] Ähnlich der fehlenden einheitlichen Definition von künstlichen Intelligenz, existiert auch für den Begriff intelligenter Agent in der Literatur keine einheitliche Definition. Um sich einer Definition des Begriffes Agent anzunähern, soll zu Beginn die Definition in Websters World Dictionary betrachtet werden:

> *„Agent: Eine Person oder Sache, die in der Lage oder ermächtigt ist, im Auftrag dritter zu handeln."* [Caglayan1998]

Diese allgemeine Definition hebt bereits zwei Schlüsseattribute von intelligenten Agenten hervor: [Caglayan1998]

- Ein Agent führt Dinge aus.

- Ein Agent handelt im Auftrag einer Person oder Sache.

Diese beiden zentralen Attribute sind bei allen intelligenten Agenten zu finden. Bertachtet man im nächsten Schritt eine Definition aus der wissenschaftlichen Fachliteratur, so erhält man bereits eine sehr gute Vorstellung dessen, was man sich unter einem Agenten vorzustellen hat.

> *"Ein Agent ist ein Computersystem, welches in einer Umgebung arbeitet, in der das System autonom Aktionen ausführen kann, um die ihm gestellte Aufgabe zu erfüllen."*[1] [Jennings1998]

Diese Definition schließt sowohl Hardware als auch Software Agenten mit ein. Unter Hardware Agenten z.B. sind autonom arbeitende Roboter zu verstehen. In der weiteren Arbeit liegt der Schwerpunk bei der Betrachtung von Software Agenten. Auf Hardware Agenten wird nicht weiter eingegangen. In der oben getroffenen Definition sind folgende, für Agenten charakteristische Eigenschaften enthalten: [Jennings1998]

- **Autonomie:** Ein Agent soll in der Lage sein, ohne direkte Interaktion des Benutzers, oder eines anderen Agenten, zu handeln. Weiters muß ein Agent die Kontrolle über seine Aktionen und seinen internen Status besitzen. Dieses Verständnis von Autonomie soll anhand eines Vergleichs mit dem objektorientierten Programmierparadigma veranschaulicht werden. In einem Objekt sind die Variablen gekapselt, das heißt daß nur das Objekt selbst die Kontrolle über sie besitzt, da man diese nur ansprechen und modifizieren kann, indem man Methoden verwendet, welche das Objekt bereit stellt. Ähnlich den Objekten ist auch in einem Agenten der interne Zustand gekapselt. Doch im Unterschied zu Objekten kann ein Agent selbst über sein Verhalten entscheiden. Ein Agent kann somit selbst entscheiden, ob er seinen internen Zustand verändert oder nicht. Ein Objekt besitzt diese Wahlmöglichkeit nicht. Wenn eine Methode des Objekts aufgerufen wird, so muß diese auch ausgeführt werden. Ein Agent dagegen besitzt, im Gegensatz zu Objekten, die Freiheit selbst zu entschieden, ob und wie er auf eine Anfrage reagiert. [Jennings1998]

  Der Agent nimmt somit nicht nur blind Kommandos entgegen, sondern rechnet auch damit, daß der menschliche Benutzer Fehler macht, wichtige Informationen ausläßt oder daß Mehrdeutigkeiten aufgedeckt werden

[1] *„An intelligent agent is a computer system that is capable of flexible autonomous action in order to meet its design objectives."* [Jennings1998]

müssen. Diese Unklarheiten müssen dann durch geeignete Mittel (z.B. Nachfrage beim Benutzer, Heranziehen einer Wissensbasis, usw.) beseitigt werden. Der Agent darf sogar gewisse Anfragen ablehnen, z.B. falls durch sie zu einer ungünstigen Zeit das Netz zu stark belastet oder anderen Benutzern geschadet würde.

- **Reaktiv:** Ein Agent muß in der Lage sein, seine Umwelt wahrzunehmen und auf Änderungen in dieser zu reagieren. Abhängig von der Aufgabe des Agenten kann die Umwelt z.B. der realen Welt, dem Internet, dem Benutzer, oder auch anderen Agenten entsprechen.

- **Proaktiv:** Agenten sollten nicht nur einfach auf ihre Umwelt reagieren, sie sollten auch in der Lagen sein, selbst, wenn nötig, zielgerichtet die Initiative zu ergreifen.

- **Sozial:** Agenten sollen in der Lage sein, mit ihre Umwelt zu interagieren. Die Interaktion kann mit anderen Agenten oder mit einem Benutzer erfolgen. Ziel der Interaktion ist es, das gestellte Problem zu lösen oder andere Agenten bei der Lösung ihres Problems zu unterstützen.

Unter einem *agentenbasierten System* wird ein System verstanden, welches auf abstrakter Ebene auf den oben angeführten Attributen beruht. Im Prinzip kann ein System als agentenbasierten System konzeptioniert sein, jedoch ohne Software-Strukturen implementiert werden, die dem Agenten-Paradigma entsprechen. Ähnlich existiert auch die Möglichkeit, objektorientierte Programme, ohne die Verwendung einer objektorientierten Entwicklungsumgebung, zu erstellen. Jedoch wäre diese Vorgehensweise unüblich und konterproduktiv. Aus diesem Grund werden agentenbasierten Systeme sowohl als solche designed und mit Hilfe der Agenten-Paradigmen implementiert. Ein agentenbasiertes System besteht somit im Regelfall aus mindestens einem autonomen Agenten. Dabei stellt ein Multi-Agentensystem, welches aus mehreren interagierenden Agenten besteht, ein signifikant komplexeres System als ein Einzel-Agentensystem dar. [Jennings1998]

Die Forschung auf dem Gebiet der intelligenten Agenten ist noch recht jung. Das erste Konzept eines Software Agenten stammt aus einer Arbeit auf dem Gebiet der verteilten künstlichen Intelligenz. Carl Hewitt veröffentlichte 1977 in seinem Artikel ,,Concurrent Actor Model" ein Modell, in dem eigenständige, interaktive und parallel exekutierbare Objekte, mit gekapselten internen Zustand, miteinander kommunizieren können. In den 80'er Jahren begann die eigentliche Grundlagenforschung auf dem Gebiet der Software Agenten. Dabei wurden die Interaktion und die Kommunikation zwischen Agenten, die Auf- und Verteilung von Aufgaben, die Kooperation und die Konfliktlösung via Verhandlung erforscht. Ziel dieser Forschung war die Spezifikation, die Analyse, das Design und die Integration von Systemen aus mehreren kooperierenden Agenten. [Nwana1998] Mit

dem Beginn der 90'er Jahre wurde damit begonnen, die bisher gewonnenen Erkenntnisse in konkrete Anwendungen umzusetzen. Welche Bedeutung die Forscher der Agenten-Technologie in der Zukunft zumessen, wird aus folgenden Aussagen deutlich:

> *„Agenten stellen das nächste bedeutende Programmierpradigma dar und werden bis zum Jahr 2000 jeden Markt durchdringen."* [2] Janca, 1995 [Jennings1998]

> *„Das Marktforschungsunternehmen Ovum*[3] *sagt für das Jahr 2000 einen Markt mit einem Volumen von 4 Milliarden Dollar voraus. Die Agenten werden vorallem im Bereich der Telekommunikation, der Konsumgüterindustrie, beim Militär [...] eingesetzt."* [Caglayan1998]

10.3 Modell eines Agenten

Um eine bessere Vorstellung über den Aufbau und die Arbeitsweise eines Agenten zu erhalten, soll in diesem Abschnitt ein Schema für ein Agentenmodell vorgestellt werden. Es handelt sich dabei um ein einfaches Modell aus der Sicht des Anwenders, mit dessen Hilfe die funktionale Architektur, das Zusammenspiel zwischen den einzelnen Systemkomponenten und die Interaktion mit dem Benutzer veranschaulicht werden sollen. Aus dem, in Abbildung 10.1 dargestellten Modell gehen die Qualifikation auf der Aufgabenebene, das Wissen und die Kommunikationsqualifikation als zentrale Elemente eines intelligenten Agenten hervor. Auf diese zentralen Elemente soll nun näher eingegangen werden. [Caglayan1998]

10.3.1 Qualifikation auf der Aufgabenebene

Die Qualifikation auf der Aufgabenebene setzt sich aus Funktionen zusammen, die ein intelligenter Agent benötigt, um die ihm gestellte Aufgabe zu erfüllen. So muß z.B. der Hilfsagent eines Betriebssystems in der Lage sein, das Benutzerverhalten zu beobachten, um, wenn nötig, Hilfestellungen am Bildschirm einblenden zu können. Ein Internet-Suchagent hingegen muß die Fähigkeit besitzen, auf die Informationsressourcen des Internets zuzugreifen. Der Agent muß somit in der Lage sein, sein Umfeld mit der Hilfe von Sensoren erfassen zu können und bei der Ausführung seiner Aufgabe in diesem Umfeld zu agieren.

[2] *„Agents are the next major computing paradigm and will be pervasive in every market by the year 2000."* Janca, 1995 [Jennings1998]

[3] http://www.ovum.com/

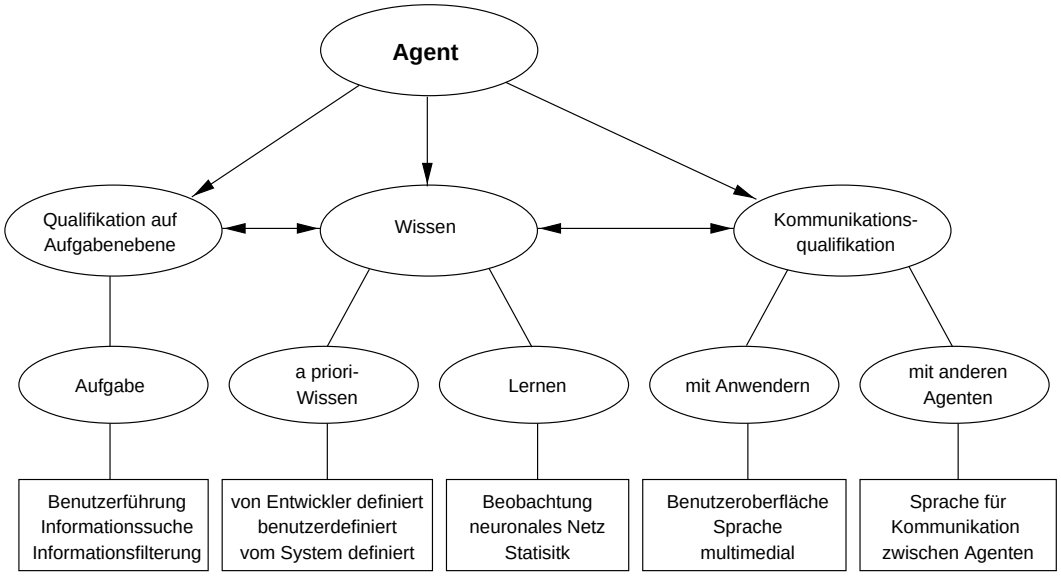

Abbildung 10.1: Modell eines intelligenten Agenten. [Caglayan1998]

10.3.2 Das Wissen

Im Wissensbereich eines Agenten ist das Wissen über die Umwelt des Agenten, über die Zusammenhänge, die in dieser Umwelt gültig sind, und die Ziele des Agenten gespeichert. Weiters gehören zu Wissensbereich auch Techniken und Strategien, die ein Agent benötigt, um die ihm gestellte Aufgabe zu erfüllen. Bei den in den Agenten verwendeten Wissensrepräsentationen und Techniken handelt es sich um die, in den vorhergegangenen Abschnitten vorgestellten Wissensrepräsentationen und Techniken der künstlichen Intelligenz. Beim Wissen können zwei Arten unterschieden werden, a-priori Wissen und vom Agenten erlerntes Wissen (siehe auch Abbildung 10.1). Beide Arten von Wissen können auch zugleich in einem Agenten vorkommen. [Caglayan1998]

- **A-priori Wissen:** Beim a-priori Wissen handelt es sich um Wissen, welches entweder vom Entwickler, vom Benutzer oder vom umgebenden System vorgegeben wird.

  - Vom Entwickler vorgegeben: Der Entwickler kann Wissen in vielen verschiedenen Formen implementieren. Dies kann z.B. in der Form einer Wissensbasis mit der dazugehörigen Inferenzmaschine geschehen, in Form eines Neuronalen Netzes, welches vor dem Einsatz vom Entwickler trainiert wurde oder auch in Form einer Suchstrategie. Es stehen dafür sämtliche Techniken der künstlichen Intelligenz und der Wissens-

repräsentation zur Verfügung, die in den vorhergegangenen Kapiteln vorgestellt wurden.

  - Vom Benutzer vorgeben: Dabei handelt es sich meist, um vom Benutzer selbst definierte einfache Regeln. Ein Beispiel dafür ist die automatisierte Verschiebung von Emails in bestimmte Folder, in Abhängigkeit vom Inhalt der Felder „Form", „To", „Subject" oder „Body". Die dazu notwendigen einfachen Regeln werden vom Benutzer mit der Hilfe eines Dialogfelds des Email-Clients erzeugt.
  - Vom umgebenden System vorgegeben: Darunter wird z.B. das Wissen um die Ansprechpartner in einer Netzwerkumgebung verstanden. Dazu zählt auch das Wissen, welche Dienste der Agent von anderen beziehen kann oder welche Datenbanken zur Verfügung stehen.

- **Erlerntes Wissen:** Lernende Agenten beobachten ihre Umwelt nicht nur um auf Veränderungen in ihr reagieren zu können, sondern auch um aus ihrer Umwelt zu lernen. Dabei wird, zusätzlich zum vorhandenen a-priori Wissen, Wissen aus der Umwelt exzerpiert, um das Verhalten des Agenten an das geänderte Umfeld anzupassen oder um regelmäßige Abläufe zu erkennen und diese zu automatisieren.

  Ein Beispiel für ein System, welches das Benutzerverhalten beobachtet, um Regelmäßigkeiten darin zu erkennen, ist Open Sesame!. Open Sesame! beobachtet im Apple MacOS[4] alle High-Level-Benutzeraktionen auf sich wiederholende Muster und bietet dem Anwender an, diese zu automatisieren. Beobachtet Open Sesame! z.B., daß ein Anwender immer nach dem Einschalten des Computers den Email-Client startet, so bietet ihm das System an, diese Aufgabe für ihn zu übenehmen. [Caglayan1998]

  Internet-Suchdienste stellen z.B. Systeme dar, die aus der Veränderung ihrer Umwelt lernen. Sie lernen ihren Suchindex aus der Beobachtung des Internets durch ihren Indizier-Prozeß. Dies kann so aufgefaßt werden, daß sich die Suchdienste die Antworten auf mögliche Anfragen bereits im Voraus zurechtlegen, um bei einer konkreten Anfrage schneller reagieren zu können. [Caglayan1998]

  Als Daten für den Lernprozeß von lernenden Agenten können z.B. Ereignisprotokolle dienen. Diese Protokolle werden auf Trends und Korrelationen untersucht. Dazu können statistische Methoden herangezogen werden. Neuronale Netze können zur Identifizierung annähernd gleicher Abfolgen im Verhalten des Benutzers eingesetzt werden. Diese Technik wird z.B. im oben beschriebenen Open Sesame! System verwendet. Ein Nachteil bei der Verwendung von Neuronalen Netzen ist die lange Trainingsphase. Der Agent

[4] http://www.apple.com

muß den Benutzer über eine lange Zeit beobachten, um die auftretenden Regelmäßigkeiten zu erlernen. [Caglayan1998]

10.3.3 Kommunikationsfähigkeit

Die Kommunikationsfähigkeit eines Agenten bezieht sich einerseits auf die Kommunikation mit dem Benutzer über die Benutzerschnittstelle und andererseits auf die Kommunikation zwischen einzelnen Agenten, sofern es sich um ein Multi-Agentsystem handelt. Die Kommunikation mit dem Benutzer findet gewöhnlich über die graphischen Benutzeroberfläche, z.B. über Dialogfelder, statt. Eine multimediale Aufbereitung der Kommunikation mit der Hilfe von Graphiken, Video oder Audio kann die Kommunikationserfahrung mit dem Anwender erweitern und einem Agenten eine „Persönlichkeit" vermitteln. [Caglayan1998]

Für die Kommunikation zwischen den Agenten in einem Multi-Agentsystem wird eine spezielle Kommunikationssprache verwendet. Beispiele für solche Sprachen sind: TeleScript[5], SmallTalkAgents[6], Agent Tcl[7] (Tool Command Language), Knowledge Query and Manipulation Language[8] (KQML) oder Knowledge Interface Format[9] (KIF). [Caglayan1998]

Im Vorhandensein von vielen verschieden Kommunikationssprachen liegt die Gefahr, daß man bei der Entwicklung eines Projekts auf eine Sprache setzt, deren Weiterentwicklung in der näheren Zukunft eingestellt wird. Diesem Zustand versucht die Advanced Research Projects Agency[10] (ARPA) mit dem Knowledge Sharing Effort[11] (KSE) Projekt entgegen zu wirken. KSE beschäftigt sich mit der Entwicklung von Vereinbarungen, welche es erlauben verteilte Wissensbasen oder Wissensbasierte Systeme zu gestalten. Dabei wird auch die Wiederverwertbarkeit der Wissensbasen berücksichtigt. Das Ziel ist die Definition, die Entwicklung und das Testen von Systemen, die es den Entwicklern erlauben größere und umfangreichere Systeme zu verwirklichen, als es mit Einzelsystemen möglich wäre. [Finin1994]

[5] http://www.genmagic.com/
[6] http://www.oti.com/links/smaltalk.htm
[7] http://cuiwww.unige.ch/eao/www/TclTk.html
[8] http://www.cs.umbc.edu/kqml/
[9] http://www.cs.umbc.edu/kse/kif/
[10] http://www.arpa.mil/
[11] http://www.cs.umbc.edu/kse/

10.4 Intelligenz durch Kommunikation

In den vorhergegangenen Abschnitten wurde des öfteren der Begriff ,,intelligenter Agent" verwendet. Damit ist jedoch nicht impliziert, daß jeder intelligente Agent auf einer Technik der künstlichen Intelligenz aufbauen muß. Mit dem Begriff intelligent wird vielmehr das Gesamtsystem aus Agenten und Kommunikation bezeichnet. Die einzelnen Agenten müssen somit kein Neuronales Netz oder ein Expertensystem enthalten. Die ,,Intelligenz" in einem Agenten-System kann vielmehr auch durch die Interaktion und die Kommunikation zwischen den Agenten entstehen. Dies soll anhand eines einfachen Beispiels veranschaulicht werden.

Ein Anwender plant sich einen Personal Computer mit 400 MHz Prozessortakt und 128 MByte RAM zu kaufen. Aus diesem Grund beauftragt er einen Software Agenten Preis und Produktinformationen im Internet einzuholen. Als Ergebnis erwartet der Anwender eine übersichtliche Auflistung der Angebote, unter Berücksichtigung des Zolls, der Versankosten und der Lieferdauer.

In einem ersten Schritt wir der Agent einen Suchagenten kontaktieren, um Adressen von Online-Computer-Shops zu erfragen. Im nächsten Schritt wird der Agent Kontakt mit den Agenten der Online-Shops aufnehmen und ihnen in einer Anfrage die Leistungsmerkmale des gewünschten Computers mitteilen. Als Antwort erhält der Agent Produktinformationen, den Preis und zusätzliche Lieferinformationen, wie Versankosten, Zahlungsmöglichkeiten, Lieferzeit und Auslieferungsland. Das Auslieferungsland ist wichtig, da der Agent auch einen eventuellen Einfuhrzoll mitberücksichtigen soll. In einem weiteren Schritt wird der Agent den Agenten des Finanzministeriums kontaktieren um die nötigen Zollinformationen einzuholen. Weiters kontaktiert der Agent den Agenten einer Online-Bank, um den aktuell gültigen Wechselkurs zu ermitteln. Die Adresse der Online-Bank hat der Agent zuvor von einem Suchagenten erhalten. Im letzten Schritt erfolgt die Preisberechnung und die übersichtliche Aufstellung der gesammelten Daten. Zusammengefaßt sehen die Schritte, die der Agent ausführt, folgendermaßen aus:

- Kontaktiere Suchagenten und hole Adressen von Online-Shops ein.
- Kontaktiere Agenten der Online-Shops, sende die Produktanfrage und hole Preis, Produkt und Lieferinformationen ein.
- Kontaktiere Agenten des Finanzministeriums und hole Zollinformationen ein.
- Kontaktiere Suchagenten und hole Adressen von Online-Banken ein.
- Kontaktiere Agenten der Online-Banken und hole den aktuellen Wechselkurs ein.

- Führe die Preisberechnung durch.
- Stelle die gesammelten Informationen übersichtlich zusammen.

Keiner dieser Einzelschritte benötigt ein Verfahren der künstlichen Intelligenz. Die erbrachte Leistung des Agenten kann man jedoch durchaus als intelligent bezeichnen. „Entstanden" ist diese Intelligenz aus der Interaktion mit den Service-Agenten im Netz. Auch jeder dieser Service-Agenten kann ohne die Verwendung von künstlicher Intelligenz implementiert werden. Dieses Beispiels zeigt auch die Stärken des Agenten Paradigmas auf. Durch die Zusammenarbeit vieler einfacher Einzelsysteme entstand ein Gesamtsystem, welches in der Lage ist, auch komplexe Probleme zu lösen. Aus diese Einschätzung heraus, erklären sich auch die optimistischen Prognosen für die Zukunft der Agenten-Technologie, wie sie am Ende der Abschnitts 10.2 angeführt wurden.

Erweitert man im obigen Beispiel den Auftrag an den Agenten um den konkreten Kaufauftrag beim Billigstbieter, so wird klar, welches Vertrauen der Auftraggeber in die Fähigkeiten seines Agenten und in die Glaubwürdigkeit der beteiligten Service-Agenten besitzen muß. Bereits bei dem Beispiel ohne den konkreten Kaufauftrag muß der anfragende Agent auf die erhaltenen Informationen vertrauen können. Service-Anbieter könnten in Versuchung kommen, durch die Angabe von falschen Daten interessierte Kunden auf ihre Internetseiten zu locken. Probleme die die Sicherheit und das Vertrauen von Agenten betreffen, werden in Abschnitt 10.6 näher besprochen. Zunächst wird jedoch das Konzept von mobilen Agenten vorgestellt.

10.5 Mobile Agenten

Bei den bisher vorgestellten Agenten handelt es sich um Programme, die lokal auf dem Client-Rechner ausgeführt werden und eventuell mit Agenten auf dem lokalen Rechner oder anderen Rechnern kommunizieren. Mobile Agenten werden hingegen von einem Client-Rechner an einen oder mehrere Remote-Server gesendet, um dort ausgeführt zu werden. Nach der Abarbeitung kehren die Programme mit dem Ergebnis zum Client-Rechner zurück. Mit der Ausführung eines Programmes auf einem fremden Rechner sind natürlich große Sicherheitsrisiken verbunden. Es existieren jedoch Anwendungen, für die die Fähigkeit von mobilen Agenten, Programme auf einem entfernten Server auszuführen, ein signifikanter Vorteil sind. [Caglayan1998]

- Unterstützung für mobile Computer: Mobile Computer die nur temporär an ein Netz angeschlossen sind, können mobile Agenten absetzen, die in der

offline Periode Aufträge des Benutzers erledigen. Ist der Benutzer später wieder an das Netz angeschlossen, kehren die Agenten mitsamt den Ergebnissen zurück.

- Suche in großen, unstrukturierten oder nicht indexierten Wissensspeichern: Wenn ein Benutzer in einem großen, unstrukturierten oder nicht indexierten Wissenspeicher eine Suche durchführen möchte, kann es effizienter sein, einen Suchagenten loszuschicken, der die Suche lokal auf dem Remote-Server durchführt. Der Wissensspeicher kann z.B. eine Datenbank oder eine digitale Bibliothek sein. Auf diese Weise ist es auch möglich, Suchen in einem Wissenspeicher durchzuführen, für die vor Ort keine Vorkehrungen getroffen wurden. Der Suchagent kehrt nach der Suche mit den gefilterten Informationen zum Client zurück. Durch die Verwendung von mobilen Agenten kann somit die Netzwerkbelastung gesenkt werden.

- Echtzeitinteraktion: Wenn es für eine Server-Anwendung wichtig ist, daß in Echtzeit auf Veränderungen der Umwelt reagiert wird, so kann die Zeitverzögerung, die durch die Übertragung der Daten vom Client zum Server und zurück entsteht, störend wirken. Die zeitkritische Server-Anwendung kann z.B. eine Roboter Steuerung sein. Sendet der Client jedoch das Steuerprogramm an den Server, so kann der Server direkt, den Wünschen des Clients entsprechend, auf die Umwelt reagieren. Die Übertragungsverzögerungen werden somit eliminiert.

- Ressourcenauslastung: Durch die Verteilung von Aufgaben im Netz können die vorhandene Rechenressourcen gleichmäßiger ausgelastet werden.

Programme für mobile Agenten können in einer Compiler-Sprache oder in einer Interpreter-Sprache geschrieben sein. Damit mobile Agenten auch in heterogenen Netzen, in denen unterschiedlichste Rechnerplattformen integriert sind, arbeiten können, sollten sie vorzugsweise in einer Interpreter-Sprache implementiert sein oder innerhalb einer virtuellen Maschine laufen (z.B. Java). Dadurch wird zwar auf Rechenleistung verzichtet, jedoch die meisten rechenaufwendigen Schritte finden nicht im Agentenprogramm selbst statt, sondern greifen auf Funktionen, die in der virtuellen Maschine oder im Interpreter implementiert sind, zurück. Zudem wird bei Java durch die „Just-in-Time"-Compilierung eine Leistungsteigerung erreicht. Interpretierte Sprachen bieten auch den Vorteil der späteren Bindung: Dies ermöglicht dem mobilen Agenten, Funktionen und Klassen zu referenzieren, die zwar nicht auf dem aktuellen System, aber auf dem Zielsystem zur Verfügung stehen. Beispiele für Sprachen für mobile Agenten sind: Java[12], TeleScript[13] und Tcl[14]. [Caglayan1998]

[12] http://java.sun.com/

[13] http://www.genmagic.com/

[14] http://cuiwww.unige.ch/eao/www/TclTk.html

Ein sehr wesentlicher Punkt bei der Arbeit mit mobilen Agenten ist der Sicherheitsaspekt, da der Server, auf dem der mobile Agent arbeiten soll, die Intention des Agenten nicht mit absoluter Genauigkeit feststellen kann. Näher wird auf die Sicherheitsaspekte im nachfolgenden Abschnitt eingegangen.

10.6 Agentensicherheit

In den vorhergegangenen Abschnitten wurden immer wieder Sicherheitsfragen und das nötige Vertrauen in Agenten erwähnt. Wie wichtig die Betrachtung dieser Fragen ist, zeigt schon der Vergleich mit einem menschlichen Agenten. Erteilt ein Auftraggeber einem menschlichen Agenten einen Auftrag, so vertraut er auf die Fähigkeit des Agenten, den Auftrag zu erledigen. Dazu muß der Agent mit den nötigen Mitteln und Rechten ausgestattet werden. Nach der Erledigung des Auftrags durch den Agenten muß der Auftraggeber auf die Richtigkeit der erbrachten Leistung vertrauen. Um die Mißbrauchsmöglichkeiten des Agenten einzuschänken und seine Aktionen nachvollziehen zu können, wird der Auftraggeber gewisse Sicherheitsmechanismen vorsehen.

In Agentensystemen, die in öffentlich zugänglichen Netzen arbeiten, sind Fragen des Datenschutzes von ganz besonderer Wichtigkeit. Ein Agent, der als persönlicher Assistent eines Benutzers agiert, benötigt zur Erfüllung seiner Aufgaben umfangreiches Wissen über die Interessen seines Benutzers. Mobile intelligente Agenten müssen dieses Wissen ständig mit sich führen, während sie sich in einem Netzwerk von Ort zu Ort bewegen. Man könnte solche Agenten als im Internet wandelnde Interessenprofile einzelner Personen ansehen. Da Benutzer die gespeicherten Daten vor willkürlichem fremden Zugriff schützen möchten, müssen dafür geeignete Schutzmaßnahmen, wie z.B. die Verschlüsselung, vorgesehen werden. [Caglayan1998]

Die in einem Agenten gespeicherten Daten müssen auch vor Manipulation durch Dritte geschützt werden können. Sonst kann ein Benutzer nicht davon ausgehen, daß die von seinem Agenten gelieferten Informationen unverfälscht und authentisch sind.

Diese Sicherheitsfragen stellen sich in Systemen von Agenten, die untereinander, ohne explizite Benachrichtigung der Benutzer, Daten austauschen, als noch schwieriger dar. Solchen Systemen liegt der sogenannte „knowledge sharing approach" zugrunde. Ein Agent muß wissen, welche Daten er bedenkenlos weitergeben kann und welche unter allen Umständen zu schützen sind. [Caglayan1998]

Es sind auch Szenarien denkbar, in denen intelligente Agenten „digitales Geld" mit sich führen. Damit könnten sie die Informationsdienste anderer Agenten bezahlen, aber auch selbst, durch das Bereitstellen von Information für andere

Agenten, Geld bekommen. Solche Agenten erfordern einen besonderen Schutz gegen Manipulation.

Desweiteren müssen Agenten auf die Identität ihres Kommunikationspartners vertrauen können. Es wäre z.B. denkbar, daß ein Agent eine falsche Identität vorgibt, um so zu Informationen zu gelangen oder einen finanziellen Vorteil daraus zu ziehen.

Um den oben angeführten Mißbrauchsmöglichkeiten entgegenzuwirken, ist es notwendig, daß die Agentenkommunikation verschlüsselt stattfindet, bzw. mobile Agenten verschlüsselt über das Netz übertragen werden. Zudem müssen digitale Signaturen und Zertifikate verwendet werden, um die Identität des eigenen Agenten belegen zu können und die des Kommunikationspartners zu überprüfen. Diese Aspekte müssen auch bei der Entwicklung von Agenten-Kommunikationssprachen mitberücksichtigt werden. [Caglayan1998]

Mobile Agenten bringen noch ein weiteres Sicherheitsrisiko mit sich. Auf der Server-Plattform wird der Programmcode des mobilen Agenten ausgeführt. Durch geeignete Maßnahmen muß nun sichergestellt werden, daß der Agent keinen Schaden an der Server-Plattform anrichten kann. Dies kann z.B. geschehen, indem man den Code nur innerhalb einer abgekapselten virtuellen Maschine ausführen läßt, wie dies z.B. bei Java geschieht. [Caglayan1998]

Trotz all der oben angeführten Sicherheitsbedenken stellt der Einsatz von intelligenten Agenten ein großes Potential für verteilte Problemlösungen dar. Im nachfolgenden Abschnitt soll anhand von einigen konkreten Forschungsprojekten die Vielseitigkeit der Agententechnologie und ihre Anwendungsmöglichkeiten vorgestellt werden.

10.7 Beispiele für Agenten

10.7.1 BargainFinder

Der BargainFinder[15] ist ein experimenteller virtueller Einkaufsagent für CDs im Internet und weist ähnliche Funktionen auf wie der Beispielagent aus Abschnitt 10.4 Intelligenz durch Kommunikation. Entwickelt wurde der BargainFinder von Anderson Consulting. Der BargainFinder benutzt, ähnlich wie die Meta-Suchmaschinen, eine parallel Abfragearchitektur, um damit die Preise und die Verfügbarkeit der vom Benutzer gesuchten Audio CDs abzufragen.

Der Kunde übergibt den BargainFinder den Titel und den Interpreten der gewünschten CD. Der Agent nimmt die Produktanfrage des Kunden und legt sie

[15] http://bf.cstar.ac.com/

parallel einer Gruppe von Online-Geschäften vor. Dabei füllt der Agent bei jedem Geschäft die entsprechenden Formulare aus. Der BargainFinder filtert in den Anfrageergebnissen den Header und die Werbung heraus und extrahiert die angegebenen Preise. Anschließend faßt er die extrahierten Ergebnisse zusammen und übergibt die Zusammenfassung dem Kunden. In der Antwort meldet der Agent das Geschäft und den Preis der CD, oder daß ein Geschäft die CD nicht verfügbar hat, oder ein Geschäft dem BargainFinder den Zugriff verwehrt hat. Manche Online-Musikgeschäfte blockieren den BargainFinder, da sie nicht möchten, daß die Preise und die Verfügbarkeit eines Produktes automatisiert abgefragt werden. [Caglayan1998]

Die Vorteile von Einkaufsagenten lassen sich wiefolgt zusammenfassen:

- Einkaufsagenten bieten für verschiedene Online-Geschäfte eine einheitliche Schnittstelle an. Damit entfällt für den Kunden die Notwendigkeit, bei jedem Geschäft die verschiedene Formular einzeln auszufüllen.
- Einkaufsagenten helfen die besten Preise zu finden und geben Auskunft über die Verfügbarkeit eines Produkts.

Virtuelle Geschäfte erlauben jedoch den Agenten den Zugriff nur sehr widerwillig, da sie meisten Anbieter kein Interesse haben, daß der Kunde auf einfache weise einen Überblick über die Preise am Markt erhält. Schätzungen gehen davon aus, daß die Agenten-Technologie den Handel im Internet von Grund auf verändern wird. [Caglayan1998]

10.7.2 CIG Searchbots

Bei CIG (cooperative information gathering) handelt es sich um eine Forschungsarbeit der University of Massachusetts. CIG ist ein System, in dem mehrere Agenten zusammenarbeiten, um Informationen zu filtern. Der CIG Searchbot gehört somit zu den Multi-Agentensystemen.

Auf die Anfrage eines Benutzers hin führen mehrere Agenten eine Suche an mehreren entfernten Orten im WWW durch. Ein Teil dieser Suche wird von existierenden Suchdiensten, wie z.B. Lycos oder Infoseek, abgedeckt, die von den CIG Searchbots angesprochen werden. Einzelne CIG Searchbots sind Experten auf speziellen Gebieten - sogenannte „domain experts". Sie bestimmen, an welchen Orten nach Informationen gesucht wird, um möglichst gute Antworten mit möglichst geringen Suchkosten zu erhalten. [CIG1998]

Für den Benutzer liegt der Vorteil gegenüber herkömmlichen Tools, wie Lycos oder Infoseek darin, daß er seine Anfragen auf einem hohen Abstraktionsniveau

stellen kann, und sich nicht mit den Einzelheiten der Suchstrategie befassen muß. In der Endfassung des Systems soll es für den Benutzer möglich sein Anfragen wie z.B. *„Finde Informationen über Tabellenkalkulationen für Windows und OS/2"* zu stellen. Daraufhin werden die Agenten einen Plan aufstellen, eine koordinierte Suche durchführen, während der Suche Teilresultate benutzen, um weitere Informationsquellen zu lokalisieren, die Resultate zusammenfügen und dem Benutzer nach einer vorgegebenen Zeitspanne präsentieren. Der Benutzer kann Beschränkungen bezüglich der Suchzeit, der Anzahl der aufgesuchten Server und der Menge der empfangenen Daten angeben. [CIG1998]

Besonders die Benutzung der Teilresultate einer Suche zur Verfeinerung und Verbesserung der Suchstrategie, hebt den Ansatz der CIG Searchbots von der Suche mit Systemen wie Lycos oder Infoseek ab.

Die Hauptcharakteristika der CIG Searchbots lassen sich kurz so darstellen: [CIG1998]

- Zielorientierte Suche auf hohem Niveau
- Benutzung von bereits entdeckten Informationen während einer Suche, um weitere ausfindig zu machen.
- Parallele Suche mit mehreren zusammenarbeitenden Agenten
- Selbstanpassende und dynamische Suchstrategien
- Aktive Suche, keine statische offline Index-Suche wie bei Lycos

Bei der Bewertung eines Systems wie den CIG Searchbots muß beachtet werden, daß eine online Suche nach jeder Anfrage eines Benutzers, sehr ressourcenintensiv ist. Insbesondere wenn mehrere Agenten parallel arbeiten. Für Systeme dieser Art müssen noch geeignete Restriktionen gefunden werden, um die Netzlast nicht zu hoch werden zu lassen. [CIG1998]

10.7.3 BASAR

BASAR[16] (Building Agents Supporting Adaptive Retrieval) ist ein Projekt des GMD Institute for Applied Information Technology[17] (FIT). BASAR stellt ein Multi-Agentensystem dar, mit dessen Hilfe ein Benutzer im World Wide Web einen persönlichen Informations-Raum einrichten und gestalten kann.

[16] http://fit.gmd.de/hci/projects/basar/
[17] http://fit.gmd.de/

Die Grundtechniken um Informationen im World Wide Web abzurufen und wiederzufinden sind suchen, browsen und das Setzen von Bookmarks. Die Suche erfolgt mit der Hilfe von Suchdiensten, wie sie in Abschnitt 2.4.3 Suchdienste vorgestellt wurden. Die Bookmarks erlauben es, das WWW zu personifizieren, d.h. mit der Hilfe der Bookmarks kann ein Benutzer sein persönliches „Information-Image" des Webs erstellen. Da das Internet ein dynamisches ständig im Wachstum begriffenes System darstellt, muß der Benutzer fortwährend darauf bedacht sein, sein persönliches „Information-Image" aktuell zu halten. Dazu ist es notwendig, daß neue Bookmarks hinzugefügt werden, alte und selten benutzte gelöscht werden und bestehende upgedated werden. Dies stellt eine sehr anspruchsvolle und zeitaufwendige Aufgabe dar. [Thomas1997]

BASAR ist nun der Prototyp eines Systems, der versucht diese Aufgabe zu automatisieren. BASAR soll den Benutzer dabei unterstützten, seinen persönlichen Informations-Raum zu managen und übernimmt dabei folgende Aufgaben: [Thomas1997]

- Aktuell halten des persönlichen Informations-Raum durch fortlaufendes updaten der Links.
- Reduktion der Anzahl der Links durch Löschen der Einträge von selten oder nie besuchten Web-Adressen.
- Unterstützung bei der Anpassung des persönlichen Informations-Raums an das Wachstum des Webs durch Unterstützung zur effektiven Suche mit der Hilfe von Suchdiensten.
- Unterstützung bei der Wiederauffindung von Informationen mit der Hilfe der History.

Grundkonzept von BASAR

Um dem Benutzer die oben genannte Funktionalitäten zur Verfügung zu stellen, baut BASAR auf folgende drei Grundkonzepte: Active Views, Software Agents und Usage Profiles. [Thomas1997]

Active View: Die Active Views entsprechen den persönlichen Informations-Räumen unter BASAR. Sie bestehen aus einer Menge von Bookmarks zu einem Interessensgebiet und einer Menge von Agenten. Die Agenten haben dabei die Aufgabe, die Informations-Räumen entsprechend den Wünschen des Benutzers zu verwalten.

Software Agents: BASAR stellt eine Umgebung zur Verfügung, die es dem Benutzer erlaubt, bestimmte Aufgaben an Agenten zu delegieren. Dabei assistieren bestimmte Agenten immer nur einem Benutzer, über den sie Informationen sammeln, wie z.B. Interessen, Gewohnheiten und Aufgaben. Mit der Zeit wird das Bild, das sich die BASAR-Agenten über den Benutzer und seine Nutzungsweise des Webs machen können, immer detaillierter. Dieses Bild wird im Usage Profile (Benutzungsprofil) gespeichert.

Usage Profile: Im Usage Profile sind die Interessen, Gewohnheiten und Aufgaben des Benutzers gespeichert. BASAR erstellt dieses User-Modell aufgrund von Beobachtungen und expliziten Fragen an den User. Dies ermöglicht es dem System sich an den Benutzer anzupassen.

Agententypen

Bei BASAR handelt es sich um ein Multi-Agentensystem, in dem drei Grundtypen von intelligenten Agenten zusammenarbeiten: Interface-Agents, Task-Agents und Network-Agents. Diese drei Agententypen sollen im folgenden kurz vorgestellt werden: [Thomas1997]

- **Interface-Agents:** Interface-Agents sind die einzigen, die direkt mit dem Benutzer kommunizieren. Sie stellen die Kommunikation zwischen dem Benutzer und den beiden anderen Agentenarten her. Wenn ein Task-Agent oder ein Network-Agent den Benutzer kontaktieren möchte, stellt er eine Anfrage an den Interface-Agent. Der Interface-Agents bestimmt mit Hilfe der Einstellungen im Usage Profile, wann, wo und wie die Kommunikation mit dem Benutzer stattfindet. Die Kommunikation kann zum Beispiel durch ein blinkendes Icon realisiert werden, wenn der Benutzer anwesend ist, oder durch E-Mail, wenn der Benutzer abwesend ist. Mit einem neuen Active View wird automatisch auch ein Interfaceagent geschaffen, der auf den Einstellungen im Usage Profile beruht.

- **Task-Agents:** Diese Agenten unterstützen das Filtern von Informationen, den Aufbau persönlicher Informationsräume, sowie das Finden relevanter Information und den Zugriff darauf. Es gibt verschiedene vordefinierten Task-Agents, die dem Benutzer ohne weitere Konfiguration zur Verfügung stehen:

  - Clean-Up-Agent: Er ist für die Bookmarks verantwortlich. Er deckt sogenannte dead Links auf. Das sind Links auf Dokumente, auf die nicht mehr zugegriffen werden kann. Außerdem besteht die Möglichkeit, Bookmark-Einträge mit einem Verfallsdatum zu versehen: Ist ein

Bookmark-Eintrag seit n Monaten nicht mehr angewählt worden, so wird gefragt, ob dieser gelöscht werden soll.

- Search-Agent: Unterstützt Benutzer im Gebrauch von verschiedenen Suchmaschinen und in der Auswertung ihrer Ergebnisse. Er sammelt Information, indem er eine oder mehrere Suchmaschinen startet und die erhaltenen Dokumente anhand des Usage Profiles und des Kontextes auswertet. Abbildung 10.2 zeigt das Interface des Search-Agents.
- Filter-Agent: Verknüpft die gefundenen Informationen mit den Anforderungen des Benutzers, die im Usage Profile gespeichert sind, und filtert sie entsprechend.
- Blackboard-Agent: Dieser Agent erzeugt eine Webseite, die als ein schwarzes Brett für eine Active View benutzt wird. Der Benutzer kann sich damit den momentanen Zustand des für ihn zu einem bestimmten Thema aufgebauten Informations-Raums vergegenwärtigen.
- Monitor-Agent: Überwacht vom Benutzer bestimmte Webseiten und informiert diesen, wenn Veränderungen vorgenommen wurden.

- **Netzwerk-Agenten:** Sie besitzen Wissen über die Client/Server-Architektur des Internet, über die Aufenthaltsorte von Suchmaschinen, über erreichbare Server und über Zeitzonen.

Die verschiedenen Arten von Agenten braucht der Benutzer nicht zu kennen, da er einfach durch E-Mail, Icons und Dialogboxen mit dem System kommuniziert. Der Interfaceagent präsentiert ihm ein einheitliches Aussehen des Systems.

Das Multi-Agentensystem BASAR verbessert die Handhabung des World Wide Web auf mehrfache Weise. Das System verwandelt durch das Konzept der Active Views passive Information (z.B. Bookmarks) in aktive. Passive Information kann veraltern oder ungültig werden. Aktive Information wird durch intelligente Agenten auf dem neuesten Stand gehalten. Der Benutzer wird benachrichtigt, wenn neue Information zugänglich wird. [Thomas1997]

Benutzer können Aufgaben an ihre Agenten delegieren. So kann ein Agent z.B. aus dem Ergebnis einer Suche diejenigen Links auswählen, welche noch nicht betrachtet worden sind. Agenten können wiederholt die gleichen Aktionen ausführen. Zum Beispiel ist es möglich, automatisch jede Woche von einem bestimmten Server bestimmte Daten anzufordern. Benutzer können Agenten den Zeitpunkt überlassen, an dem bestimmte große Mengen an Daten übertragen werden sollen. Dann kann sich ein Agent die nach seiner Erfahrung günstigste Zeit aussuchen. Dadurch kann eine gleichmäßigere Auslastung des WWW erreicht werden.

Ein Beispielsystem für die Anwendung von BASAR findet sich auf der Internet-Seite: `http://fit.gmd.de/hci/projects/basar/`

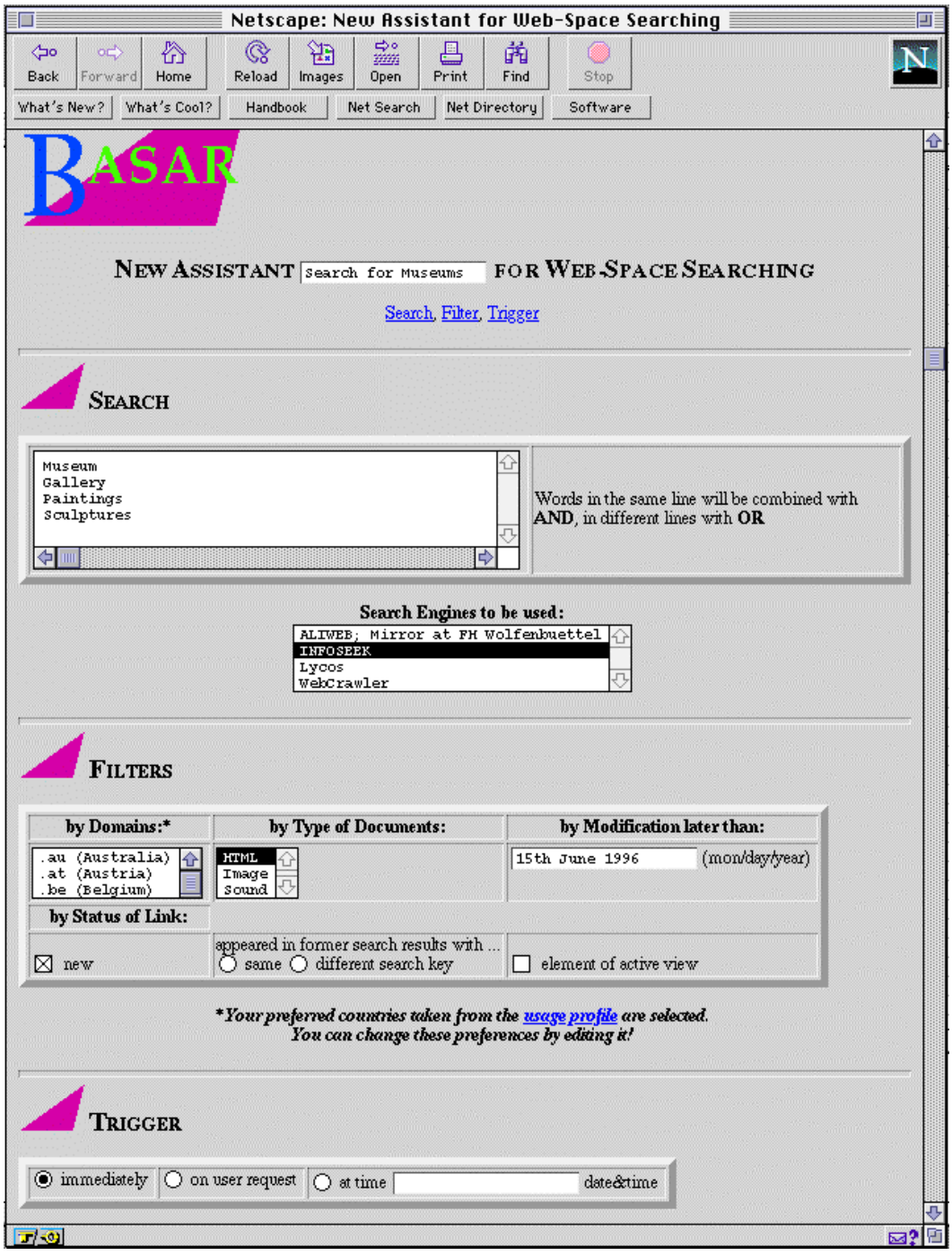

Abbildung 10.2: Interface des Basar Search-Agents. Es lassen sich die Schlagworte, der Suchdienst, die möglichen Top-Level-Domains, das Modifikationsdatum und andere Parameter angeben. Die Ergebnisse der einzelnen Suchdienste werden entsprechend den Einstellungen gefiltert und aufbereitet und in der Active View der Suche dargestellt. [Thomas1997]

Teil II

Der Gestaltungsbereich

Kapitel 11

Der Virtual Tutor

Im Untersuchungsbreich (Teil I) der vorliegenden Arbeit wurden die Grundbegriffe der Wissensverarbeitung eingeführt. Darüber hinaus wurden verschiedenste Techniken der Künstlichen Intelligenz, die in der Wissensverarbeitung ihre Anwendung finden, vorgestellt. Aufbauend, auf das im Untersuchungsbereich erarbeitete Wissen, wurde im Gestaltungsbereich ein interaktiver dialogbasierter Lernbehelf entwickelt. In der weiteren Arbeit wird dieser Lernbehelf als *„Virtual Tutor"* bezeichnet. Der Virtual Tutor stellt somit einen interaktiven Lernbehelf dar, mit dem der Benutzer über einen Frage-Antwort-Dialog kommuniziert.

11.1 Zielsetzung

Ziel des Virtual Tutors ist es, als Mittel zur problembezogenen Lernunterstützung im Web Based Trainingssystem GENTLE[1] zu dienen. Als konkreter Problemfall dient beim Virtual Tutor die Aufgabe, sich ausgehend von einer speziellen Problemstellung, für eine geeignete Technik der Künstlichen Intelligenz zu entscheiden. Zu diesem Zweck stellt der Virtual Tutor gezielt Fragen über die Art der Problemstellung und ist in der Lage, Auskunft darüber zu geben, warum das System die Frage an den Benutzer gerichtet hat und welche Auswirkungen die Antwort auf das Ergebnis haben wird. Hat der Virtual Tutor genügen Informationen über die Problemstellung vom Benutzer erhalten, so gibt das System jene Technik der Künstlichen Intelligenz aus, welche für das gegebene Problem am geeignetesten ist. Dabei wählt der Virtual Tutor, in der vorliegenden Testimplementierung, aus den folgenden vier Techniken aus: Expertensystem, Fuzzy Logic, Neuronales Netz, NeuroFuzzy. Weiters ist das System auch in der Lage, seine Entscheidung für eine spezielle Technik zu begründen. Darüber hinaus stellt

[1] `http://wbt.iicm.edu`

der Virtual Tutor die Möglichkeit zur Verfügung, sämtliche Begründungen und Erläuterungen mit Verweisen auf eine Hintergrundbibliothek zu versehen. Als Hintergrundbibliothek dient das Wissen, welches im Untersuchungsbereich erarbeitet wurde. Dieses Wissen wurde als HTML-Seiten aufbereitet und steht über das WBT-System GENTLE im Netz zur Verfügung. Zum besseren Verständnis des theoretischen Wissen enthält die Hintergrundbibliothek zudem auch multimedial aufbereitete Wissenseinheiten.

Der Virtual Tutor stellt ein *Coginitiv Tool* dar. Unter einem Cognitiv Tool wird nach [Reeves1997] „im weitesten Sinne eine Technologie verstanden, welche das menschliche Denken, das Problemlösen und das Lernen unterstützt."[2] Beispiele für Cognitiv Tools sind z.B. eine einfache Einkaufliste, welche dem Menschen bei der Erinnerung unterstützt oder ein mathematische Formel, der sich ein Mensch bedient um bestimmte Zusammenhänge von Größen auszudrücken. Der Virtual Tutor stellt somit ein Coginitiv Tool dar, welches in der Hochschulausbildung seine Anwendung finden soll. Um den Vorteil, welche ein Software System als Cognitiv Tool zu bieten in der Lage ist, voll auszuschöpfen, muß darauf geachtet werden, daß dem Studierenden die Möglichkeit geboten wird, umfassend mit dem System zu interagieren. [Reeves1997]

Durch die Interaktion mit einem elektronischen Lernbehelf über den Frage-Antwort-Dialog erhöht sich der Nutzen für den Studierenden, gegenüber der Benutzung eines einfachen starren Lernbehelfs. [Dietinger1999] Über den Dialog mit dem Virtual Tutor wird die Theorie aus dem Untersuchungsbereich in einer neuen Weise aufbereitet. Somit findet nach dem Studium der Theorie eine Stoffwiederholung statt, bei der der Studierende die in der Lage ist, die bereits gelernte Theorie aus einem neuen Blickwinkel zu betrachten. Der Studierende wird durch den Virtual Tutor in die Lage versetzt, sich problemorientiert mit dem Wissensgebiet auseinanderzusetzten. Zudem wird dem Studierenden die Möglichkeit gegeben, selbst ein wissensbasiertes System zu bedienen und Erfahrungen mit einem solchen zu sammeln.

Der Virtual Tutor ist als Java-Applet konzipiert. Dadurch ist es den Studierenden möglich, das System einfach und ohne Installationaufwand zu starten. Um den Vorteil voll ausschöpfen zu können, ist darauf zu achten, daß der Virtual Tutor in einem, zum Zeitpunkt des erstellen der Vorliegenden Arbeit, gängigen Web-Browser lauffähig ist.

Die Implementierung des Virtual Tutors baut auf Techniken auf, welche im Untersuchungsbereich vorgestellt wurden. Welche Techniken der künstlichen Intelligenz im Virtual Tutor zur Anwendung kommen, wird in Abschnitt 11.2 näher ausgeführt. Zudem bietet der Virtual Tutor die Möglichkeit, auf das Wissen aus

[2] *„In the broadest sense, cognitive tools are any technologies that enhance our thinking, problem-solving, and learning."*[Reeves1997]

dem Untersuchungsbereich in der Form von Verweisen zuzugreifen. Das im Untersuchungsbereich erarbeitete Wissen dient somit einerseits als Hintergrundbibliothek für den Virtual Tutor, andererseits als Grundlage für die Implementierung des Virtual Tutors als wissensbasierten System.

Weiters ist bei der Implementierung des Virtual Tutors drauf zu achten, daß diese möglichst modular erfolgt. Durch diese Anforderung kann erreicht werden, daß sich das System möglichst einfach für andere Wissensgebiete adaptieren läßt. Die modulare Implementierung erleichtert auch die späterer Erweiterbarkeit und sie Wartbarkeit des Systems.

Aus den oben angeführten Betrachtungen ergeben sich für den Virtual Tutor folgende Anforderungen, welche bei der Implementierung berücksichtigt wurden:

- Der Virtual Tutor muß in der Lage sein, die an den Benutzer gerichtete Frage zu begründen.
- Der Virtual Tutor muß in der Lage sein, das Zustandekommen der Lösung zu erläutern.
- Innerhalb der oben angeführten Erläuterungen und Erklärungen muß es möglich sein, auf die entsprechenden Kapiteln im Untersuchungsbereich zu verlinken.
- Der Untersuchungsbereich ist in HTML-Seiten aufzubereiten und als Hintergrundbibliothek für das System zugängig zu machen.
- Die Kommunikation mit dem Virtual Tutor hat über einen Frage-Antwort-Dialog zu erfolgen. Zu diesem Zweck ist einen entsprechende Benutzerschnittstelle zu implementieren.
- Der Virtual Tutor muß innerhalb eines, zum Zeitpunkt des Erstellens der vorliegenden Arbeit, allgemein verfügbaren Web-Browser ausführbar sein. (z.B. Netscape 4.x, MS Explorer 5.x)
- Der Virtual Tutor ist so zu implementieren, daß sich das System einfach für andere Wissensgebiete adaptieren läßt.
- Weiters ist auf die zukünftige Erweiterbarkeit Rücksicht zu nehmen.

Im nachfolgenden Abschnitt wird das Grundkonzept des Virtual Tutors vorgestellt, welches die oben aufgelisteten Systemanforderungen erfüllt.

11.2 Das Grundkonzept

Aus den, im vorhergegangenen Abschnitt erarbeiteten Anforderungen an die Implementierung des Virtual Tutors geht hervor, daß die Fähigkeit zur Begründung der ausgeführten Schritte und des Ergebnisses, eine zentrale Rolle einnimmt. Erst die Fähigkeit zur Begründung der ausgeführten Schritte ermöglicht es, den Virtual Tutor als problemorientierten Lernbehelf einzusetzen. Wie in Abschnitt 6.3 ausgeführt, bieten Expertensysteme als wissensbasierte Systeme die Möglichkeit, die ausgeführten Schritte zu begründen. Für die Ausgabe der Begründungen ist die Erklärungskomponente innerhalb des Expertensystems zuständig. Weiters unterstützen Expertensysteme die oben angeführte Forderung nach dem modularen Aufbau des Systems und der Realisierung der Benutzerschnittstelle als Frage-Antwort-Dialog. Aus diesen Gründen wurde der Virtual Tutor als Expertensystem implementiert.

11.2.1 Aufbau des Expertensystems

Der Virtual Tutor stellt ein System dar, welches in seiner ersten Testimplementierung einem Benutzer bei der Entscheidung unterstützen soll, welche Technik der Künstlichen Intelligenz für seine Problemstellung am geeignetesten ist. Um diese Aufgabe zu erfüllen, muß der Virtual Tutor drei unterschiedliche Arten von Wissen besitzen: [Pivec1997]

- **Wissen über die Problemstellung:** Der Virtual Tutor benötigt Wissen über die Art der Problemstellung, welche mit der Hilfe von Künstlicher Intelligenz bewältigt werden soll. Dieses Wissen erhält das System über einen Frage-Antwort-Dialog mit dem Benutzer in dem das System gezielt Fragen zu bestimmten Fakten der Aufgabenstellung stellt.
- **Erklärungswissen:** Dabei handelt es sich um Wissen, um die angeführten Schritte zu erläutern und das Ergebnis zu erklären. Mit der Hilfe dieses Wissens ist es möglich das Verständnis für die zugrundeliegende Theorie zu vertiefen.
- **Wissen über das Fachgebiet:** Dieses Wissen ist in den Regeln der Wissensbasis gespeichert. Mit der Hilfe dieses Wissens findet die eigentliche Wissensverarbeitung innerhalb des Virtual Tutors statt. Das Wissen über das Fachgebiet bestimmt die Auswahl der Fragen im Frage-Antwort-Dialog, bis hin zur Berechnung der ausgegebenen Lösung.

Ausgehend von den Antworten des Benutzers im Frage-Antwort-Dialog erstellt der Virtual Tutor intern ein Bild der Problemstellung. Das interne Bild der

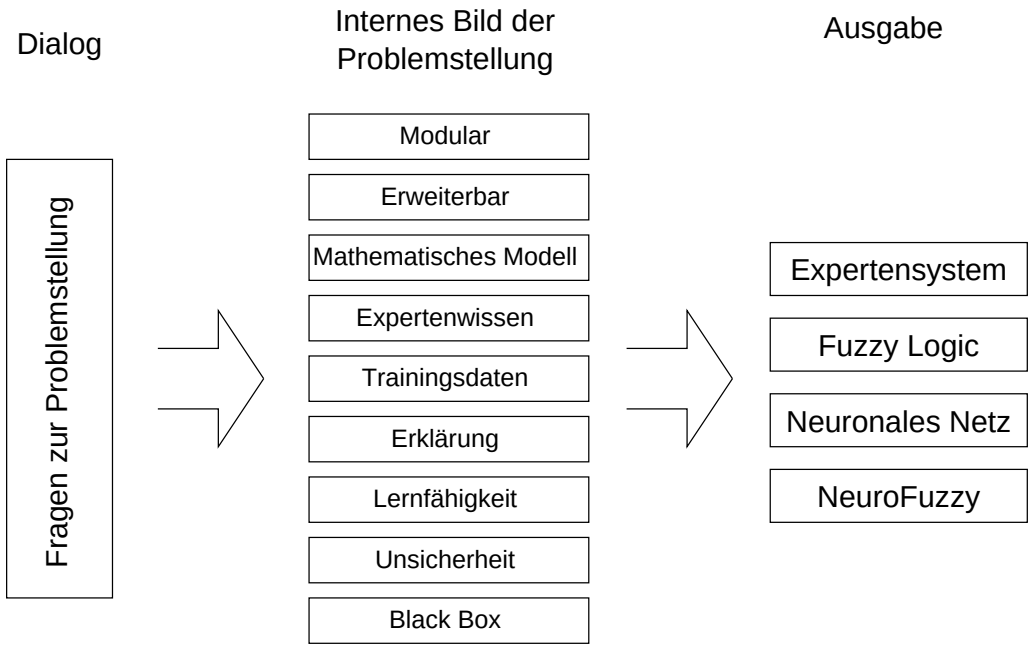

Abbildung 11.1: Arbeitsweise des Virtual Tutors. Die Wissensverarbeitung erfolgt in drei Schritten: Dem Dialog mit dem Benutzer, dem Erstellen des internen Bildes der Problemstellung und der Ausgabe.

Problemstellung besteht aus Attributen wie: Modular, Erweiterbar, Mathematisches Modell, usw. Eine genaue Aufstellung aller verwendeten Attribute findet sich in Abbildung 11.1. Dieses Bild dient im letzten Schritt der Wissensverarbeitung bei der Auswahl der geeignetesten Technik der Künstlichen Intelligenz.

Das Expertensystem des Virtual Tutors arbeitet somit in folgenden drei Schritten:

1. Dialog mit dem Benutzer über die Problemstellung.
2. Erstellen des internen Bildes der Problemstellung des Benutzers.
3. Auswahl einer Technik der Künstlichen Intelligenz.

Das interne Bild der Problemstellung ist aus Abbildung 11.1 ersichtlich. In dieser Abbildung ist auch die dreischrittige Arbeitsweise des Virtual Tutors visualisiert.

Der Dialog des Virtual Tutors mit dem Benutzer erfolgt in Form von Fragen über den Problembereich. Die Auswahl der gestellten Fragen erfolgt gezielt, da nicht jede Frage zu jedem Zeitpunkt sinnvoll ist. So ist z.B. die Frage ,,Stufen Sie auf einer Skala von 0 bis 10 (0 $\widehat{=}$ gering; 10 $\widehat{=}$ sehr aufwendig) den Aufwand ein,

ein mathematisches Modell für das Problem zu erstellen." nur dann sinnvoll, wenn die Frage ,,Läßt sich das Problem mit der Hilfe eines mathematischen Modells beschreiben? (ja/nein/warum)" mit ja beantwortet wurde.

Wie aus der oben angeführten Frage ,,Stufen Sie auf einer Skala von 0 bis 10 (0 $\widehat{=}$ gering; 10 $\widehat{=}$ sehr aufwendig) den Aufwand ein, ein mathematisches Modell für das Problem zu erstellen." ersichtlich ist, lassen sich nicht alle Fragen mit einem exakten ja bzw. nein beantworten. Die vom Benutzer eingegebenen Fakten sind somit mit einer gewissen Unsicherheit verknüpft. Um diese Unsicherheiten im Virtual Tutor speichern und verarbeiten zu können, baut das System nicht auf der Verwendung von einfachen ,,Wenn-Dann-Regeln" auf, sondern benutzt Certainty Factors zur Behandlung von Unsicherheiten. Die Theorie zu den Certainty Factors wurde in Abschnitt 6.8 im Untersuchungsbereich behandelt.

Ein weiterer Grund für die Implementierung der Certainty Factors ist, daß jede Eingabe vom Benutzer nur einen Hinweis auf das interne Bild der Problemstellung geben kann. So ergibt sich z.B. durch die Antwort auf die Frage ,,Lassen sich über das Problem exakt formulierbare Zusammenhänge angeben? (ja/nein/warum)" nur ein Hinweis darauf, ob es für die Problemstellung notwendig ist, daß das gesuchte System mit Unsicherheiten umgehen kann. Wird die Frage mit ,,ja" beantwortet, so ist es sehr wahrscheinlich, daß das System mit exakten Regeln auskommt, wird die Frage jedoch mit ,,nein" beantwortet so erhält das System ein Indiz dafür, daß die Berücksichtigung von Unsicherheit für die Problemlösung notwendig ist. Dieser Tatsache wurde durch die Gewichtung der Regeln mit Certainty Factors Rechnung getragen.

Aus den im vorhergegangen Abschnitt 11.1 erarbeiteten Anforderungen an den Virtual Tutor geht hervor, daß die Bedienung des System aus dem Web-Browser heraus erfolgen soll. Aus diesem Grund wurde für die Implementierung die Expertensystem-Shell Jess gewählt, da diese vollständig in Java geschrieben ist. Im nachfolgendem Abschitt wird eine kurze Vorstellung der Expertensystem-Shell Jess gegeben und die konkrete Implementierung des Virtual Tutors vorgestellt.

11.3 Expertensystem-Shell Jess

Bei Jess[3] (Java Expert System Shell) handelt es sich um eine vollständig in Java geschriebene Expertensystem-Shell, welche von Ernest Friedman-Hill am Sandia National Laboratories in Livermore, CA entwickelt wurde. Als Anstoß für die

[3] `http://herzberg.ca.sandia.gov/Jess/`

Entwicklung von Jess diente die Expertensystem-Shell CLIPS. Bei CLIPS[4] handelt es sich um eine an der NASA entwickelte, frei erhältliche Expertensystem Entwicklungsumgebung. Mittlerweile wurde aus Jess jedoch ein eigenständiges Software-Projekt. Aus diesem Grund sind die beiden Sprachen auch nicht mehr kompatibel zueinander. Zum Zeitpunkt des Erstellens der vorliegenden Arbeit lag Jess in der Version Jess 5.0b1 vor, welche auch für die Implementierung des Virtual Tutors verwendet wurde.

Wie in den vorangegangenen Abschnitten bereits erwähnt wurde, muß der Virtual Tutor in der Lage sein, als Java-Applet in einem Web-Browser ausgeführt werden zu können. Aus diesem Grund wurde Jess als Plattform für die Implementierung des Virtual Tutors gewählt. Zudem ist Jess für Forschungszwecke frei erhältlich und steht im Internet zum Download[5] bereit. Der Download umfaßt auch den vollständigen Sourcecode von Jess. Dies ermöglicht es, das System nach den eigenen Erfordernissen zu erweitern.

Um eine Vorstellung von der Arbeitsweise eine wissensbasierten System unter Jess zu erhalten, soll an dieser Stelle ein kurzer Überblick über die in Jess verwendeten Sprachelemente gegeben werden.

11.3.1 Die Sprache Jess

Der Syntax der Sprache Jess ist stark an den Syntax der Sprache LISP angelehnt. Eine ausführliche Einführung in die Sprache Jess und eine Auflistung der verfügbaren Funktionen findet sich unter [Friedman1999]. Jedes gültige Sprachelement hat den Syntax eines Funktionsaufrufs. Funktionsaufrufe in Jess verwenden die Prefix-Notaton. Die Addition ,,2+3" sieht in Jess folgendermaßen aus:

```
Jess> (+ 2 3)
    5
```

Für die Kurzeinführung in Jess gelten folgende Konventionen: Zeilen die mit dem ,,`>Jess`" Prompt beginnen stellen Eingaben des Benutzers dar. Die darauffolgenden Zeilen entsprechen der Jess Ausgabe. Weiters gilt der Rest der Zeile, der einem Strichpunkt (`;`) folgt, als Kommentar.

[4] Die offizielle NASA CLIPS Homepage: `http://www.siliconvalleyone.com/clips.htm` Weitere nützliche Links zu CLIPS: `http://home.hkstar.com/~skoo/expert.htm`, `http://www.ghg.net/clips/CLIPS.html`

[5] `http://herzberg.ca.sandia.gov/jess/`

Fakten

Ein Expertensystem besteht, wie in Kapitel 6 besprochen, aus Fakten und Regeln. In Jess stehen unterschiedliche Arten von Fakten zur Auswahl; dazu zählen einfache Fakten und Frames. Einfache Fakten werden mit Hilfe der „`assert`"-Funktion folgendermaßen definiert.

```
Jess> (assert (father-of Miriam Peter)) ; Definition des Faktums
                                        ; father-of
<Fact-0>
Jess> (facts)                           ; Jess soll alle bekannten
                                        ; Fakten ausgeben
f-0   (father-of Miriam Peter)
For a total of 1 facts.
```

Mit der Hilfe von Frames ist es in Jess möglich, zusammengehörenden Fakten zu Objekten zu gruppieren. Der theoretische Hintergrund zu Frames wurde bereits in Abschnitt 3.5 erarbeitet. Vor der Verwendung von Frames müssen diese mit Hilfe der „`deftemplate`"-Anweisung definiert werden. Dabei stellen Slots Felder dar, die einen Wert aufnehmen können.

```
Jess> (deftemplate automobile           ; Name des Frames
        "A specific car."               ; Kommentar
        (slot make)
        (slot model)
        (slot year)
        (slot color (default white)))   ; Für den Slot color gilt
                                        ; der Defaultwert white
TRUE
```

Nach der Definition des Frames können darauf aufbauende Fakten definiert werden.

```
Jess> (assert (automobile (make Chrysler)
              (model LeBaron) (year 1997)))
<Fact-0>
Jess> (facts)
f-0   (automobile (make Chrysler) (model LeBaron)
                  (year 1997) (color white))
For a total of 1 facts.
TRUE
```

Bei der Zuweisung zum Faktum `automobile` wurde kein Wert für den Slot `color` angegeben. Aus diesem Grund nimmt der Slot den Defaultwert `white` an.

Regeln

Regeln werden in Jess über die `defrule` Funktion definiert.

```
Jess> (reset)                    ; Löschen aller Fakten
TRUE
Jess> (assert (father-of Miriam Peter))
<Fact-1>
Jess> (assert (father-of John Peter))
<Fact-2>
Jess> (assert (father-of Karin John))
<Fact-3>
Jess> (defrule is-father ""      ; Nane der Regel: is-father
        (father-of ?x Peter)     ; Vorbedingung der Regel
      =>
        (printout t ?x crlf))    ; gibt die Varialble ?x aus
TRUE
Jess> (run)                      ; Startet das Programm
John
Miriam
2                                ; Insgesamt hat zweimal eine
                                 ; Regel gefeuert
```

In der Regel `is-father` bedeutet die Vorbedingung `father-of ?x Peter`, daß die Regel feuern soll, wenn es ein Faktum `father-of` existiert, welches im zweiten Feld `Peter` stehen hat. Der Wert des ersten Feldes wird dann der Variable `?x` zugewiesen. Im obigen Beispiel ist die Vorbedingung für zwei Fakten erfüllt. Die Ausgabe lautet somit `John, Miriam`.

Variablennamen beginnen in Jess grundsätzlich mit einem ,,?". Im oben angeführten einfachen Regelbeispiel wurde bereits die Anwendung einer Regel gezeigt. In diesem Beispiel wurde der Variable `?x` ein Wert über die Vorbedingung der Regel zugewiesen. In Jess können jedoch auch direkt Variablen Werte zugewiesen werden. Dies erfolgt über die `bind` Anweisung.

```
Jess> (bind ?x 1)                ; entspricht ?x = 1
1
Jess> (printout t ?x crlf)       ; ?x ausgeben
1
Jess> (bind ?x (+ 1 1))          ; entspricht ?x = 1+1
2
Jess> (printout t ?x crlf)       ; ?x ausgeben
```

Nach dieser kurzen Einführung in die Arbeitsweise von Jess soll im nächsten Abschnitt die Regelbasis des Virtual Tutors vorgestellt werden.

11.3.2 Die Regeln des Virtual Tutors

Mit der im vorhergegangenen Abschnitt vorgestellten Kurzeinführung in Jess, sollte ein Grundverständnis für die Art der Wissensverarbeitung in Jess vermittelt werden. Aufbauend auf dieses Verständnis soll in diesem Abschnitt die interne Arbeitsweise des Virtual Tutors erläutert werden. Dieser Einblick soll anhand von exemplarischen Beispielregeln erfolgen.

Wie in Abschnitt 11.2.1 erläutert ist die Implementierung von Certainty Factors für die Funktion des Virtual Tutors notwendig. Es werden somit nicht nur Fakten im System gespeichert, sondern auch der dazugehörige Certainty Factor. Der Name des Faktums und der Certainty Factor stellen somit zusammengehörige Werte dar, welche bei der Implementierung in einem Frame gespeichert werden. Die Definition des Frames sieht dabei folgendermaßen aus.

```
(deftemplate fact
  (slot name)       ; Name des Faktums
  (slot value))     ; Certainty Factor
```

Der Virtual Tutor führt mit dem Studierenden einen Dialog, in dem er dem Studierenden Fragen zur Problemstellung stellt. Für jede dieser Fragen ist eine spezielle Regel in der Wissensbasis des Systems zuständig. Ein Beispiel für eine solche Regel sieht folgendermaßen aus:

```
(defrule Mathematisches_Modell_rule     ; Name der Regel
  "2"                                   ; Kommentar
  (not (Mathematisches_Modell_fact ?))  ; Frage noch nicht
                                        ; gestellt
=>
  (bind ?question "Läßt sich das Problem mit der Hilfe eines
        mathematischen Modells beschreiben? (ja/nein/warum) ")
  (bind ?$possibilities (create$ "ja" "nein" "warum"))
  (bind ?explanation "Läßt sich ein Problem durch ein
        mathematisches Modell beschreiben, so ist es möglich,
        das Modell direkt in ein
        <a href=wissens/content/node11.html>
        Expertensystem</a> zu integrieren. Ein lernfähiges
```

```
        System müßte dieses mathematische Modell erst in
        vielen Trainingsschritten
        <a href=wissens/content/node36.html>
        erlernen</a>.")
  (bind ?answer (ask-question ?question ?$possibilities))

 ; solange die Frage mit warum beantwortet wird, gebe die
 ; Erklärung aus und stelle erneut die Frage.
  (while (eq ?answer "warum") do
    (printout t ?explanation crlf)
    (bind ?answer (ask-question ?question ?$possibilities)))

 ; Die Frage wurde mit ja oder nein beantwortet.
 ; Ordne entsprechend der Antwort den Faken einen Certainty
 ; Factor zu.
  (if (eq ?answer "ja") then
    (assert (fact (name Mathematisches_Modell) (value 1.0)))
    (assert (fact (name Expertenwissen) (value 0.3)))
    (assert (Mathematisches_Modell_fact TRUE)))
  (if (eq ?answer "nein") then
    (assert (fact (name Mathematisches_Modell) (value -1.0)))
    (assert (Mathematisches_Modell_fact FALSE)))
);end
```

Bei der Funktion `ask-question` in der oben angeführten Regel handelt es sich um eine selbst definierte Funktion, welche die Frage am Bildschirm ausgibt, die Eingabe einliest und überprüft, ob die Eingabe einem gültigen Wert entspricht. Welche Werte die Funktion als gültig akzeptiert, wird bei der Variablenübergabe angegeben. Entspricht die Eingabe keinem gültigen Wert, so wird die Frage erneut gestellt.

Der in der Variable `?explanation` gespeicherte Text stellt den Erklärungstext zu dieser Regel dar. Dieser Text wird ausgegeben, wenn der Benutzer eine Frage mit Faktum beantwortet. In diesem Text sind zusätzlich Link-Informationen enthalten, auf die im nächsten Abschnitt näher eingegangen wird.

Bewerten zwei unterschiedliche Regeln ein Faktum mit dem gleichen Namen, ist es, wie im Kapitel 6.8 ausgeführt, notwendig, daß die beiden Certainty Factors zu einem kombiniert werden. Der dafür notwendige Schritt wird parallele Kombination genannt. Für die parallele Kombination wird die in Abschnitt 6.8.1 angeführte Formel 6.5 angewendet. Aus Anschaulichkeitsgründen sei die Formel

hier nocheinmal angeführt:

$$CF\,(h \mid E_1, E_2) \;=\; \begin{cases} s + t - s \cdot t & \text{Fall 1: falls } s, t \geq 0 \\ \dfrac{s + t}{1 - \min\{|s|, |t|\}} & \text{Fall 2: falls } s \cdot t \in (-1, 0] \\ \text{undefiniert} & \text{Fall 3: falls } s \cdot t = -1 \\ s + t + s \cdot t & \text{Fall 4: falls } s, t < 0 \end{cases} \tag{11.1}$$

Wobei s und t den Certainty Factors der beiden zu kombinierenden Fakten entspricht und $CF\,(h \mid E_1, E_2)$ dem neu bewerteten Certainty Factor.

Die Implementierung der Certainty Factors in Jess erfolgt über eine Regel, die genau dann feuert, wenn zwei Fakten mit dem selben Namen existiern. Feuert die Regel, so wird entsprechend der Formel 11.1 eine Fallunterscheidung vorgenommen. Die Implementierung der paralellen Kombination in Jess sieht damit folgendermaßen aus.

```
(defrule Kombination_rule
  "Kombiniert die Certainty Faktors zweier Fakten"
  (declare (salience 50))          ; Priorität der Regel

 ; Vorbedingung: Existieren zwei Fakten mit gleichem Namen?
  ?fact1 <- (fact (name ?name) (value ?val1))
  ?fact2 <- (fact (name ?name) (value ?val2&:(neq ?fact1 ?fact2)))
=>
 ; Wenn ja, kombiniere die beiden Certainty Factors und lösche
 ; ein Faktum
  (if (and (>= ?val1 0) (>= ?val2 0)) then     ; Fall 1
    (modify ?fact1 (value (+ ?val1 (- ?val2 (* ?val1 ?val2)))))
  else
    (if (and (<= (* ?val1 ?val2) 0) (<> (* ?val1 ?val2) -1.0))
        then                                    ; Fall 2
      (modify ?fact1 (value (/ (+ ?val1 ?val2)
                               (- 1 (min (abs ?val1) (abs ?val2))))))
    else
      (if (= (* ?val1 ?val2) -1.0) then         ; Fall 3
        (assert (undefiniert ?name))
      else
        (if (and(< ?val1 0) (< ?val2 0)) then   ; Fall 4
          (modify ?fact1 (value (+ ?val1
                                   (+ ?val2 (* ?val1 ?val2)))))
        );if
```

```
      );if
    );if
  );if
  (retract ?fact2)                    ; löscht ein Faktum
);end
```

Wie in Abschnitt 11.2.1 ausgeführt arbeitet der Virtual Tutor in drei Schritten. Nach dem Frage-Antwort-Dialog mit dem Benutzer bildet der Virtual Tutor im zweiten Schritt ein internes Bild der Problemstellung. Im dritten Schritt analysiert der Virtual Tutor das interne Bild und berechnet die Relevanz der einzelnen Techniken der Künstlichen Intelligenz. Unter Relevanz wird dabei die Eignung der KI-Technik zur Lösung des gegebenen Problems verstanden. Die Berechnung der Relevanz erfolgt unter Zuhilfenahme eines Certainty Factors. Dabei gibt der Certainty Factor an, welche Bedeutung ein Zustand des internen Bildes für eine bestimmte Technik der künstlichen Intelligenz hat. Bei dem Certainty Factor handelt es sich um einen festen Certainty Factor einer Regel. Eine Beispielregel sieht somit folgendermaßen aus:

Wenn *modular*
Dann *(-0,3) Neuronales Netz*

Die obige Regel sagt aus, daß wenn das Faktum ,,modular" im internen Bild der Problemstellung existiert, ein Neuronales Netz mit dem Certainty Factor -0,3 geeignet ist. Die Abarbeitung der Regel erfolgt nach der in Abschnitt 6.8.1 angeführten Formel 6.4. Dabei wird der Certainty Factor des Faktums mit dem Certainty Factor der Regel multipliziert. Zur Berechnung der Relevanz muß somit zu jedem Faktum des internen Bildes der Problemstellung, dessen Bedeutung für die jeweilige Technik der Künstlichen Intelligenz bekannt sein. Die dazu notwendigen Certainty Factors sind in Tabelle 11.1 übersichtlich dargestellt. Ermittelt wurden die Certainty Factors aus den Erfahrungen, die bei der Erarbeitung der Theorie des Untersuchungsbereiches gesammelt wurden. Anschließend erfolgte eine Feinabstimmung der Faktoren durch Probeläufe des Systems.

Im nachfolgenden Abschnitt wird auf die Implementierung der Benutzerschnittstelle des Virtual Tutors eingegangen.

11.3.3 Benutzerschnittstelle

Zur Erstellung der Benutzeroberfläche wurden zusätzlich, zu den, zum Jess Grundsystem zählenden Klassen, Klassen und Methoden hinzugefügt, die für die speziellen Anforderungen des Virtual Tutors benötigt wurden. Zu den speziellen

	Neuronales Netz	**Experten System**	**Fuzzy Logic**	**NeuroFuzzy**
modular	-0.6	0.5	0.7	0.3
erweiterbar	-0.2	0.4	0.3	0.2
Mathematisches Modell	-0.3	0	0	-0.2
Mathematisches Modell Aufwand	0.7	-0.5	0.8	0.4
Expertenwissen	-0.3	0.7	0.7	0.5
Trainingsdaten	0.95	0	0	0.9
Erklärung	-0.9	0.5	0.3	0.3
Lernfähigkeit	0.6	-0.5	-0.5	0.8
Unsicherheit	0.5	-0.3	0.5	0.5
Black Box	0.3	-0.5	-0.5	0

Tabelle 11.1: Die in den Spalten angegebenen Faktoren stellen die Bedeutung dar, welche ein Faktum des internen Bildes der Problemstellung für die Anwendbarkeit einer bestimmten Technik der Künstlichen Intelligenz hat.

Anforderungen gehören die Exekutierbarkeit als Applet in einer Web-Browser-Umgebung, die Möglichkeit Hyperlinks in den ausgegebenen Text zu integrieren und das Verweisen auf die Theorie aus dem Untersuchungsbereich.

Bei der Implementierung der Benutzeroberfläche muß darauf Rücksicht genommen werden, daß der Virtual Tutor in einer Web-Browser-Umgebung ausführbar ist. Diese Forderung bietet dem Studierenden den Vorteil, einfach, ohne Installationsaufwand, auf den Virtual Tutor zugreifen zu können. In den, zum Zeitpunkt des Erstellens des Virtual Tutors verbreiteten Web-Browsern (Netscape 4.x, MS Explorer 5.x) ist die Java Virtual Machine 1.1 implementiert. Aus diesem Grund wurde der Virtual Tutor in Java 1.1 entwickelt. Durch diese Festlegung ergeben sich Einschränkungen in den Gestaltungsmöglichkeiten der Benutzeroberfläche. Es kann nicht auf die Java Swing[6] Klassen zurückgegriffen werden, da diese erst mit Java 1.2 eingeführt wurden. Somit stehen die umfangreichen Möglichkeiten zur Gestaltung einer Benutzeroberfläche, welche die Java Swing Klassen bieten, nicht zur Verfügung. Für die Plattformen MS Windows[7] und Sun[8] Solaris stellt Sun ein Plug-in[9] zum Download zur Verfügung, welches es ermöglicht, Java 1.2

[6] `http://java.sun.com/docs/books/tutorial/uiswing/`
[7] `http://www.microsoft.com/`
[8] `http://www.sun.com`
[9] `http://java.sun.com/products/plugin/index.html`

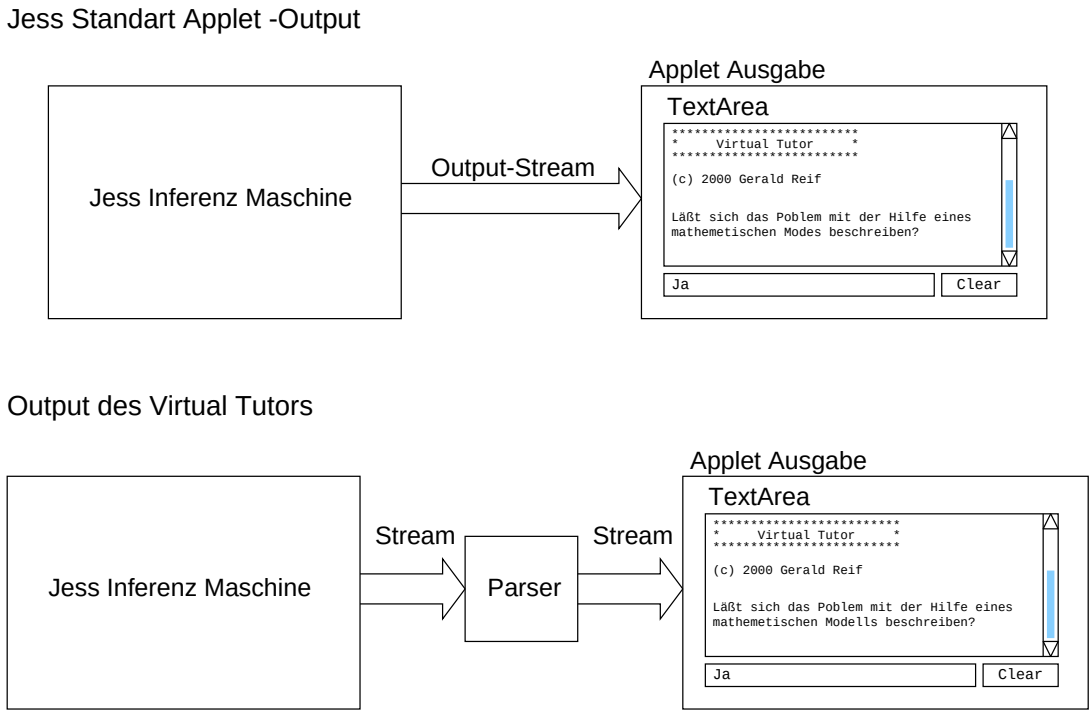

Abbildung 11.2: Bei der Implementierung des Virtual Tutors wurde in den Output-Stream von der Inferenzmaschine zum `TextArea`-Objekt ein Parser zwischengeschaltet. Der Parser hat die Aufgabe, Link-Informationen aus dem Output-Stream zu filtern und in einer URL-Liste zu speichern.

Applets in einem Web-Browser auszuführen. Doch muß dieses Plug-in erst am Client-Rechner installiert werden und steht somit im Gegensatz zu den oben angeführten Argument für die Implementierung als Applet.

Output-Stream

Jess bietet standardmäßig die Möglichkeit ein Jess Programm als Applet auszuführen. Wird diese Art der Programmausführung gewählt, so wird der Output der Inferenzmaschine in einen Output-Stream umgewandelt und zu einem `java.awt.TextArea`-Objekt gelenkt. Dieser Output-Stream ist schematisch in Abbildung 11.2 dargestellt. Das `TextArea`-Objekt stellt eine scrollbare Textbox am Bildschirm dar. Die Größe diese Textbox läßt sich in der Anzahl der Zeilen und Spalten angeben.

Bei der Implementierung des Virtual Tutor wurde in den Output-Stream zwischen der Inferenzmaschine und dem `TextArea`-Objekt ein Parser zwischengeschaltet. Der Parser hat die Aufgabe, Linkinformationen aus dem Output-Stream

zu filtern und das Link-Ziel und die Position des Links im Textfenster in einer Liste zu speichern. Das Speichern der Link-Informationen erfolgt in Objekten der `UrlList`-Klasse.

Parser

Wie bereits oben ausgeführt, hat der Parser die Aufgabe, Link-Informationen aus dem Output-Stream zu filtern. Die Link-Informationen haben dabei folgendes Format:

`<a href=`*`Object-Name`*`>`*`Link-Text`*`</a>`

Wobei der *Object-Name* dem Objektnamen in der WBT-Umgebung GENTLE entspricht. Dabei kann als *Object-Name* einerseits die Hyperwave GOid, oder der Document Name (z.B. `Wissens/content/node34.html`) verwendet werden. Aus dem *Object-Name* und den Parametern, welche an das Applet übergeben werden, setzt das Applet dann den vollständigen Uniform Ressource Locator (URL) zusammen. Die Übergabeparameter an das Java-Applet werden im nachfolgenden Abschnitt besprochen. Ein vom Applet generierter, im WBT-System GENTLE gültiger URL sieht folgendermaßen aus:

```
http://wbt-3.iicm.edu/wbt/v1/core/app/hwt/mod/ce2;course=wissens
&oid=Wissens/content/node34.html
```

Unter dem *Link-Text* wird der Mausklick-sensible Textteil des ausgegebenen Textes versanden. Dieser Text wird wie unten ausgeführt durch spezielle Textmarker markiert.

Der Parser löscht die Link-Informationen aus dem Output-Stream und fügt vor und hinher dem Link-Text Markierungen ein, sodaß dieser Text, bei der Ausgabe am Bildschirm als Link erkenntlich wird. Für die Linkmarkierung wurden die Textzeichen „`[>`" und „`<]`" gewählt. Die Linkmarkierung erfolgt mit Textzeichen, da das `TextArea`-Objekt unter Java 1.1 keine Methoden zur Verfügung stellt, die Textfarbe oder den Textstil gezielt für bestimmte Textstellen zu verändern.

Der Textparser führt im Output-String eine Textsuche nach der Linkkennzeichnung „`<a href=`" durch. Zu diesem Zweck bedient sich der Parser des Algorithmus von Knuth-Morris-Pratt. Dieser Algorithmus bietet den Vorteil, daß er die Stringsuche in der linearen Zeit $O(n)$ durchführt. Wobei n die Länge des zu durchsuchenden Textes darstellt. [Cormen1998]

Event-Handling

Zur Überwachung der Mausaktiviatäten im Ausgabebereich des Applets wurde zum `TextArea`-Objekt ein `MouseListener` hinzugefügt. Der `MouseListener`

ermöglicht es Aktionen, für bestimmte Maustastenereignisse, zu definieren. Der hinzugefügte Event-Handler reagiert auf den `MouseReleased`-Event, der dem Loslassen einer gerückten Maustaste entspricht. Nach dem Loslassen der Maustaste wird die aktuelle Cursor-Position bestimmt und mit Hilfe der gespeicherten Linkpositionen ermittelt, ob sich dieses Ereignis im Bereich einer Linkmarkierung ereignet hat. Ist dies der Fall, so wird der Web-Browser dazu veranlaßt, die entsprechende Internetseite zu laden. Die gespeicherten Linkinformationen befinden sich, wie oben erwähnt, in Objekten der `UrlList`-Klasse.

In den vorangegangenen Abschnitten wurden die Spezifikationen und die Implementierung des Virtual Tutors vorgestellt. Im nachfolgenden Abschnitt sollen nun das Starten und die Bedienung des Virtual Tutors in der WBT-Umgebung GENTLE vorgestellt werden.

11.4 Das System: Virtual Tutor

11.4.1 Der Aufruf der Virtual Tutors

Der Virtual Tutor wurde, wie in den vorangegangenen Abschnitten angeführt, als Java-Applet implementiert. Der Virtual Tutor wird somit durch den Aufruf einer HTML-Seite gestartet, von der aus das Java-Applet aufgerufen wird. Beim Aufruf des Virtual Tutors werden dem Java-Applet Parameter übergeben. Die Übergabeparameter werden vom Applet dazu benötigt, zusammen mit den Link-Informationen in den ausgegebenen Erklärungen, wie im vorangegengenen Abschnitt erläutert, einen im WBT-System GENTLE gültigen URL zu erzeugen. Ein URL hat in GENTLE z.B. folgendes aussehen:

```
http://wbt-3.iicm.edu/wbt/v1/core/app/hwt/mod/ce2;course=wissens
&oid=Wissens/content/node34.html
```

Wobei die in der ersten Zeile des URLs angegebenen Daten für alle Verweise auf das Hintergrundwissen gleich sind. Um die Link-Information in den Erklärungstexten, die im Jess-Programm gespeichert sind, möglichst kurz zu gestalten, wird nur der *Object-Name* als Link-Information gespeichert. Auf die Definition der Link-Information wurde bereits in Abschnitt 11.3.3 ausführlich eingegangen. Der restliche URL wird mit Hilfe der Übergabeparameter zusammengestellt. Eine genaue Auflistung der Übergabeparameter, deren Default-Werte und deren Bedeutung ist aus Tabelle 11.2 ersichtlich. Auf den entsprechenden Default-Wert wird zurückgegriffen, wenn beim Aufruf des Virtual Tutors ein Parameter nicht übergeben wird. Die Default-Werte entsprechen den, zum Zeitpunkt des Erstellens der vorliegenden Arbeit, in GENTLE gültigen Konventionen.

Name	Default-Wert	Beschreibung
INPUT	kein Default-Wert	Name des Jess Programms. Das Jess-Programm enthält das eigentliche Experten System.
COURSE	wissens	Name des Kurses in der GENTLE-Umgebung.
URL_MASK	wbt/v1/core/app/hwt/mod/ce2;	Pfad der GENTLE-Module.
TARGET	awac_main	Name des Browserfensters, in dem die Hintergrundinformation angezeigt werden soll. Dieser Parameter entspricht dem TARGET-Attribut des HTML-Link-Tags <a>.

Tabelle 11.2: Übergabeparameter an das Java-Applet Virtual Tutor. Die Default-Werte entsprechen den derzeit in GENTLE gültigen Konventionen.

Die für den Aufruf des Virtual Tutors notwendigen HTML-Tags und die zu übergebenden Parameter sind nachfolgend angeführt:

```
<applet code=jess.ConsoleApplet width=600 height=500>
  <param name=INPUT value=exp.clp>
  <param name=COURSE value=wissens>
  <param name=URL_MASK value=wbt/v1/core/app/hwt/mod/ce2;>
  <param name=TARGET value=awac_main>
  <b><i>Jess</i></b>would appear here if you had a Java-aware
  browser.
</applet>
```

Nach dem Aufruf des Virtual Tutors in der GENTLE Umgebung wird ein neues Browserfenster geöffnet und in diesem das Java-Applet geladen und gestartet. Das Öffnen eines neuen Browser-Fensters erfolgt mit Hilfe der JavaScript window.open-Methode. Der gestartete Virtual Tutor in der WBT-Umgebung GENTLE ist in Abbildung 11.3 zu sehen.

11.4.2 Die Bedienung

Der Virtual Tutor wird, wie im vorangegangenen Abschnitt erwähnt, durch das Aufrufen einer HTML-Seite aktiviert, von der aus das Java-Applet gestartet wird.

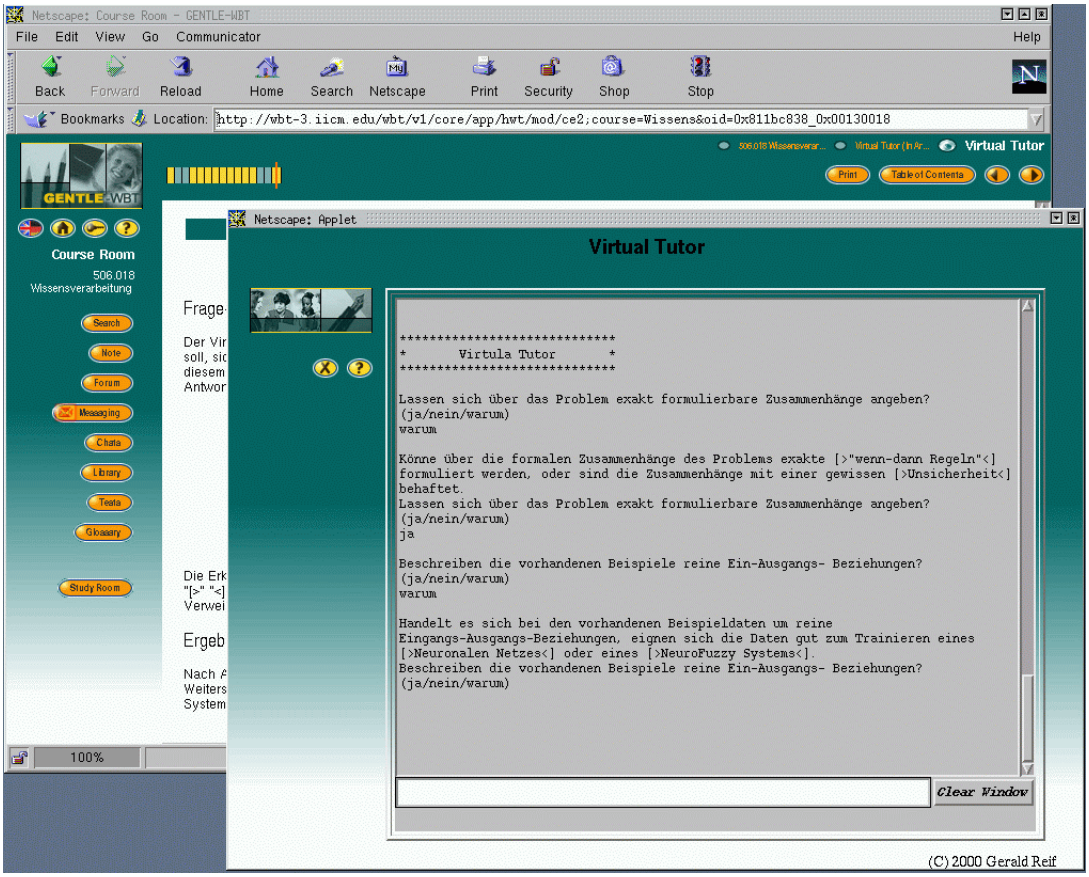

Abbildung 11.3: Nach dem Start erscheint der Virtual Tutor in einem separaten Browser-Fernster.

Nach dem Ladevorgang wird der Virtual Tutor automatisch gestartet. Das Erscheinungsbild des Virtual Tutors ist in Abbildung 11.3 zu sehen. Im unteren Bereich des Applets befindet sich die Eingabezeile, von der aus sämtlichen Eingaben an den Virtual Tutor über die Tastatur erfolgen. Um in der Eingabezeile schreiben zu können, muß zuvor den Cursor in die Eingabezeile gesetzt werden. Dies geschieht über einen Mausklick in die Eingabezeile.

Neben der Eingabezeile befindet sich ein Button mit der Aufschrift „Clear Window". Betätigt man diesen Button, so werden sämtliche Ausgaben im Ausgabebereich gelöscht. Der Ausgabebereich ist jener Bereich, der sich über der Eingabezeile und dem „Clear Window" Button befindet. Im Ausgabebereich des Virtual Tutors werden sämtliche Fragen, Erklärungen und Ergebnisse ausgegeben. Sollen vorangegangenen Ausgaben im Ausgabebereich nicht mehr sichtbar sein, so kann man diese durch Betätigen der Scroll-Leiste auf der rechten Seite des Ausgabebereichs wieder sichtbar machen.

11.4.2.1 Frage-Antwort-Dialog

Der Virtual Tutor stellt ein Expertensystem zur Lernunterstützung auf dem Gebiet der Wissensverarbeitung dar, der dem Studierenden helfen soll, sich ausgehend von einer bestimmten Problemstellung, sich für die geeignete Technik der künstlichen Intelligenz zu entscheiden. Zu diesem Zweck führt der Virtual Tutor einen Frage-Antwort Dialog mit dem Studierenden. Die Fragen werden im Ausgabebereich des Applets wiedergegeben. Zu jeder Frage wird ein kurzer Hilfstext mit den jeweils möglichen Antworten ausgebeben. Die möglichen Antworten sind in Tabelle 11.3 zusammengefaßt.

Hilfetext	**Antworten**	**Erklärung**
(ja/nein/warum)	ja	Die Frage wird mit ja beantwortet.
	nein	Die Frage wird mit ja beantwortet.
	Warum	Ein Erklärungstext zur Frage wird ausgegeben.
(0..10/warum)	0..10	Ein Zahlenwert zwischen 0 und 10 wird erwartet.
	Warum	Ein Erklärungstext zur Frage wird ausgegeben

Tabelle 11.3: Mögliche Antworten im Frage-Antwort-Dialog des Virtual Tutors.

Die ausgegebenen Erklärungstexte zu den Fragen sind mit Verweisen (Hyperlinks) auf das theoretische Hintergrundwissen versehen. Die Verweise sind durch

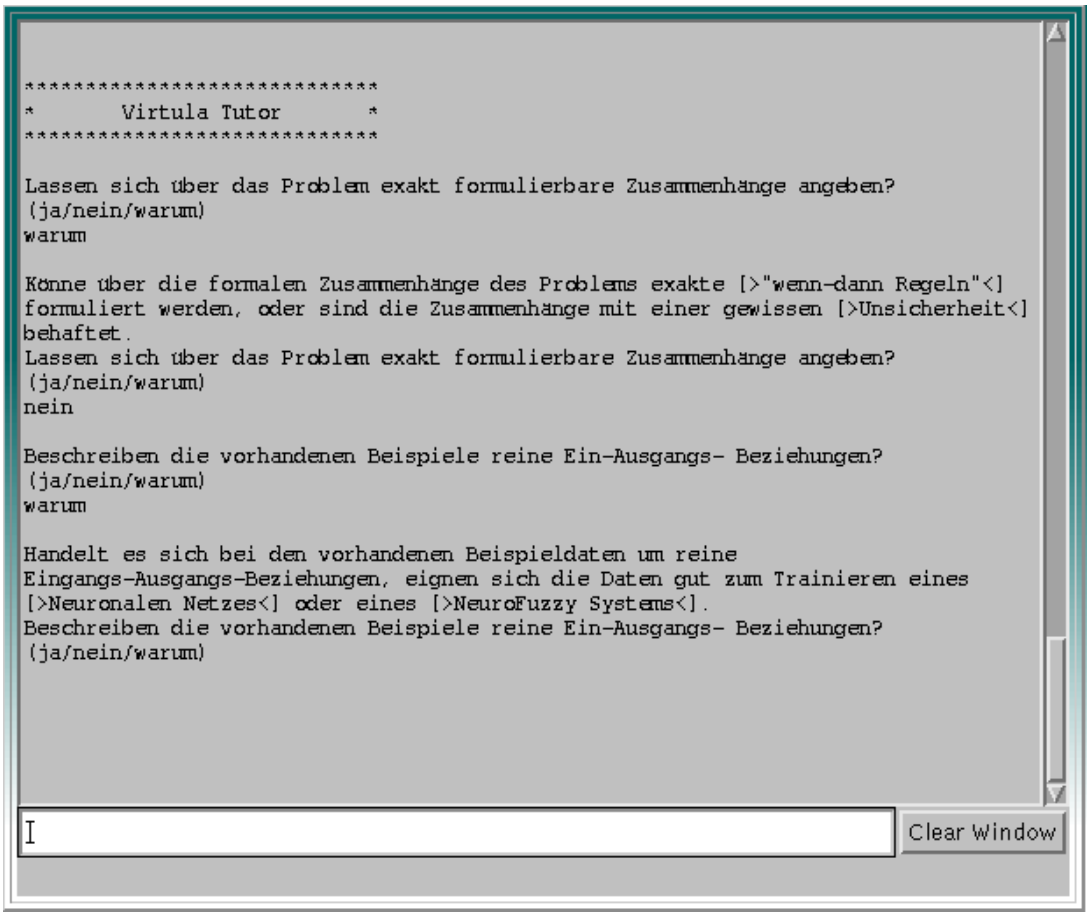

Abbildung 11.4: Der Virtual Tutor im Detail. Die Erklärungstexte im Virtual Tutor sind mit Verweisen auf theoretisches Hintergrundwissen versehen. Die Verweise sind durch die Textsymbole „[>" „<]" markiert.

„[>" „<]" Zeichen markiert. Ein Beisbiel dafür ist in Abbildung 11.4 zu sehen. Ein Mausklick (mit der linken Taste) auf den Textbereich zwischen diesen beiden Markierungen öffnet den Verweis. In Abbildung 11.5 ist z.B. das multimedial aufbereitete Hintergundwissen zur Fuzzy Regelabarbeitung zu sehen. Die multimediale Aufbereitung erfolgte mit Hilfe von Shockwave[10] der Firma Macromedia[11].

11.4.2.2 Ergebnis

Nach dem Abschluß des Frage-Antwort-Dialogs wird das Ergebnis ausgegeben. Die Ausgabe erfolgt sortiert, nach der Relevanz der Systeme. Die Ausgabe des

[10] http://www.shockwave.com/
[11] http://www.macromedia.com/

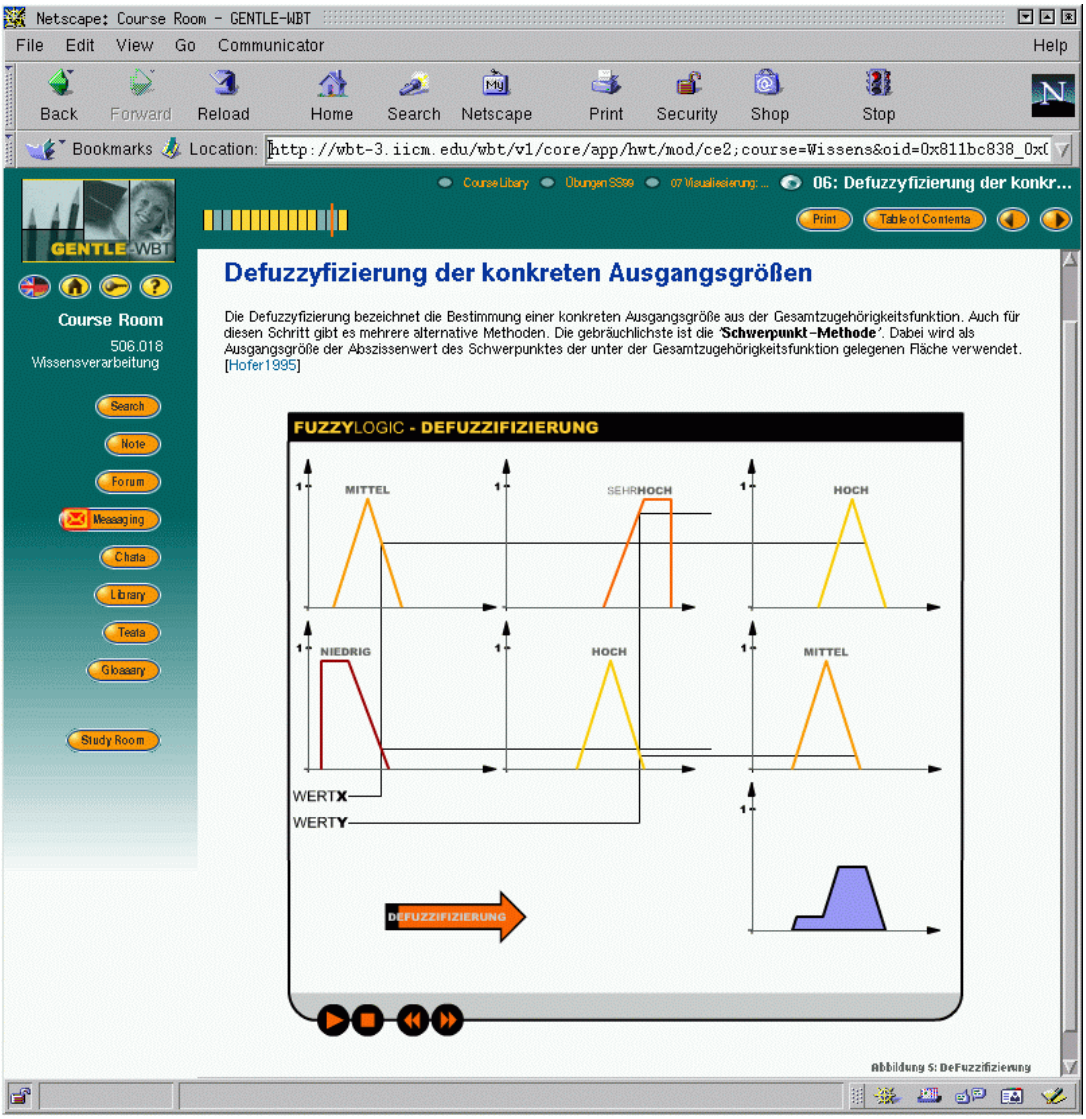

Abbildung 11.5: Beispiel einer Multimedial aufbereiteten Lerneinheit. Die Regelabarbeitung in Fuzzy Logic Systemen wurde mit der Hilfe von Shockwave multimedial aufbereitet.

Hilfetext	Antworten	Erklärung
(es/fl/nn/nf/start)	es	Eine Kurzinformation zu Expertensystemen wird ausgegeben.
	fl	Eine Kurzinformation zu Fuzzy Logic wird ausgegeben.
	nn	Eine Kurzinformation zu Neuronales Netz wird ausgegeben.
	nf	Eine Kurzinformation zu NeuroFuzzy wird ausgegeben.
	start	Der Virtual Tutor wird neu gestartet.

Tabelle 11.4: Eingaben um Informationen zu den einzelnen Techniken der Künstlichen Intelligenz abzurufen.

Ergebnisses ist in Abbildung 11.6 zu sehen. Weiters können zu den einzelnen Techniken der Künstlichen Intelligenz Informationen abgerufen werden. Die dazu notwendigen Eingaben sind aus Tabelle 11.4 ersichtlich. Auch diese Erklärungen sind mit Verweisen auf das multimedial aufbereitete Hintergrundwissen versehen.

11.5 Ausblick

In den vorangegangenen Anschnitten wurde auf die Funktion, die Arbeitsweise und die Implementierung des Virtual Tutors eingegangen. In diesem Abschnitt sollen Ansatzpunkte für künftige Verbesserungen und Erweiterungen angeführt werden. Zudem werden Vorschläge zur Evaluierung des Virtual Tutors diskutiert.

11.5.1 Benutzerschnittstelle

Wie in Abschnitt 11.3 besprochen wurde, führt die Einschränkung auf die Möglichkeiten, welche Java in der Version 1.1 zur Gestaltung der Benutzeroberfläche zur Verfügung stellt, zu einer Beschränkung bei den Gestaltungsmöglichkeiten der Mensch-Maschine-Schnittstelle. Die für die Dialogausgabe verwendete Java-Klasse (`java.awt.TextArea`) stellt uner Java 1.1 keine Methoden zur Verfügung, die Textfarbe oder den Textstil gezielt für bestimmte Textstellen zu verändern. Dies wäre jedoch wünschenswert, da sich ohne diese Möglichkeit die Links in den Erklärungstexten nur durch die Verwendung von Textzeichen markieren lassen.

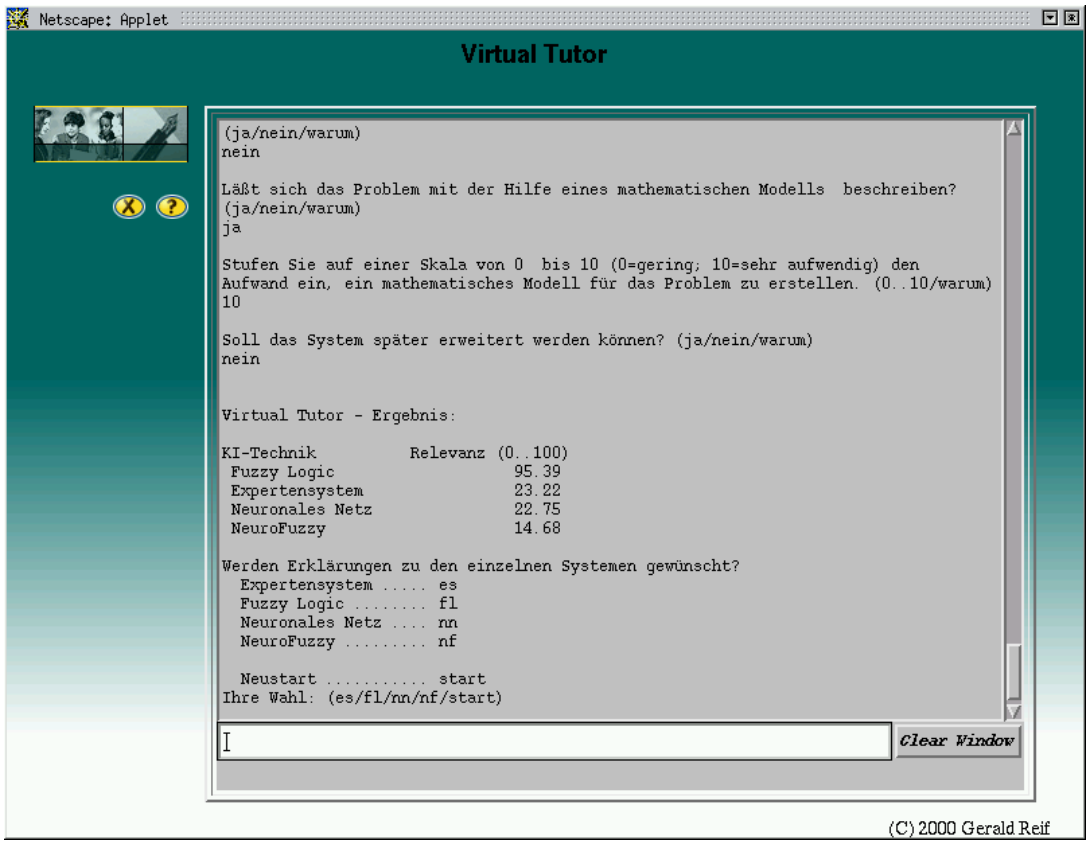

Abbildung 11.6: Ergebnisausgabe des Virtual Tutors. Die Techniken der Künstlichen Intelligenz werden nach ihre Relevanz sortiert ausgegeben.

Die Java 1.2 Klassenbibliothek bietet mit den Swing[12]-Klassen jedoch weit mehr Möglichkeiten eine Benutzeroberfläche zu ansprechend gestalten. Auf den zum Zeitpunkt der Implementierung der Virtual Tutors gängigen Web-Browsern (Netscape 4.x, MS Explorer 5.x) ist jedoch nur die Java Virtual Machine 1.1 implementiert. Für die Zukunft ist zu erwarten, daß die Nachfolgegeneration von Web-Browsern Java 1.2 unterstützen. Damit eröffnet sich ein großes Potential, die Benutzerschnittstelle anwenderfreundlich zu gestalten. Mögliche Ansatzpunkte dazu sind:

- Hervorheben von Links durch Änderung der Fontfarbe und der Fontstils.
- Einbinden von erläuternden Graphiken in die Erklärungstexte.
- Umstellung vom testbasierten Input-Dialog hin zu einem Mausgesteuerten Input-Dialog über Buttons.

Ein weiterer Ansatzpunkt für die Verbesserung der Benutzerschnittstelle ist die Einführung neuer Tags, welche bei der Ausgabe berücksichtigt werden. In der vorliegenden Version des Virtual Tutors wird der ausgegebene Text einzig nach dem Link-Tag (`<a href=url></a>`) geparst. Durch die Einführung neuer Tags wird die Möglichkeit geschaffen, die Ausgabe des Expertensystems ansprechend zu gestaltet. Beispiele für solche Tags sind: Tags zur Formatierung der Text-Ausgabe, Tags zum Einbinden von Graphiken, Tags zur Strukturierung der Output-Bereichs, usw.

11.5.2 Surflets

Der Virtual Tutor ist in der vorliegenden Version in zwei Teile gegliedert:

- Dem Applet, welches der Expertensystem-Shell entspricht. Das Appelt besteht aus der Inferenzmaschine und der Benutzerschnitstellt.
- Dem Jess-Programm, welches der Wissensbasis für das Expertensystem entspricht. Das Jess-Porgramm besteht aus der Regelbasis und der Erklärungskomponente des Systems.

Beim Starten des Virtual Tutors wird sowohl das Applet als auch das Jess-Programm vollständig über das Netz zum Web-Browser übertragen. Da sämtliche Class-Files des Applets zusammen über 500 KByte Speicher benötigen, führt dies zu größeren Wartezeiten.

[12] `http://java.sun.com/docs/books/tutorial/uiswing/`

Um die Ladezeit des Virtual Tutors zu verkürzen, sollte das Programm in einer Nachfolgeversion in drei Teile gegliedert werden:

- Surflet: Besteht aus der Inferenzmaschine. Das Surflet wird vom Server ausgeführt.
- Applet: Beinhaltet nur die Benutzeroberfläche. Das Applet wird über das Netz zum Web-Client übertragen und dort ausgeführt. Das Applet steht mit dem Surflet über das Netz in Verbindung.
- Jess-Programm (eigentliches Jess-Programm): wird vom Surflet (Inferenzmaschine) als Wissensbasis geladen.

Durch die in der obigen Auflistung vorgeschlagene Dreiteilung des Systems wird erreicht, daß nur das Applet, welches die Benutzerschnittstelle enthält, über das Netz übertragen werden muß. Die Übertragungsdauer läßt sich dadurch stark reduzieren.

11.5.3 Dynamische Hintergrundbibliohtek

In der vorliegenden ersten Testimplementierung des Virtual Tutors, wird in den Erklärungstexten ausschließlich auf Hintergrundwissen verwiesen, welches sich innerhalb der WBT-Umgebung GENTLE befindet. In einer weiteren Ausbaustufe des Virtual Tutors wäre es wünschenswert, auf Hintergrundwissen aus einer dynamischen Hintergrundbibliothek, welcher ausgewählte Internet-Server zugrundeliegen, zurückgreifen zu können. Zur Realisierung der dynamischen Hintergrundbibliothek kann die am Institut für Informationsverarbeitung und Computerunterstützte neue Medien[13] (IICM) entwickelte Suchmaschine xFind[14] herangezogen werden. Mit der Hilfe von xFind ist es möglich, vordefinierte Suchen auf ausgewählten Internet-Servern durchzuführen. So ist es durch die Verwendung von xFind möglich, innerhalb von Erklärungstexten auf vordefinierte Suchen z.B. nach dem Begriff „Neuronales Netz", „Backpropagation", usw. auf ausgewählten Servern durchzuführen. Dem Studierenden wird somit die Möglichkeit geboten, sich einfach Hintergrundinformationen zu einem Suchbegriff zu beschaffen. Dabei muß sich der Studierende nicht durch Ergebnislisten mit weit über Zehntausend Einträgen arbeiten, welche herkömmliche Suchdienste, wie sie in Abschnitt 2.4.3 besprochen wurden, als Ergebnis liefern. Zudem bietet die Selektion der zudurchsuchenden Server einen Qualitätsfilter.

[13] `http://www.iicm.edu`
[14] `http://xfind.iicm.edu`

11.5.4 Evaluierung

Ziel der vorliegenden Arbeit ist es, einem Studierenden einen Überblick über das Gebiet der Wissensverarbeitung zu geben. Zu diesem Zweck wurde im Untersuchungsbereich das Forschungsgebiet der Wissensverarbeitung theoretisch aufbereitet und im Gestaltungsbereich ein interaktives dialogbasiertes System zur Wissensvertiefung entwickelt. Durch die Interaktion mit dem elektronischen Lernbehelf über den Frage-Antwort-Dialog erhöht sich der Nutzen für den Studierenden gegenüber der Verwendung eines starren traditionellen Lernbehelfs. [Reeves1997] Erreicht wird dieser Mehrwert durch die Möglichkeit, daß sich der Studierende durch den Dialog mit dem System intensiv mit dem Stoffgebiet beschäftigen kann. Dem Sudierenden wird somit die Möglichkeit geboten, sich dem Wissenschaftsgebiet der Wissensverarbeitung auf eine problemorientierte Art anzunähern, wie dies durch das konventionelle Studium eines gedruckten Skriptums nicht möglich wäre. Um festzustellen, ob dieser Mehrwert tatsächlich existiert, und wenn ja, wie hoch dieser einzuschätzen ist, sollte eine Evaluierung des Systems durchgeführt werden.

Für die Durchführung der Evaluierung des Virtual Tutors sollten zwei Gruppen von Studierenden gebildet werden, wobei die Gruppe 1 mit der Unterstützung des Virtual Tutors arbeitet und die Gruppe 2 ohne. Nach dem Abschluß der Lernphase, für die beiden Gruppen gleich viel Zeit zur Verfügung stand, werden die Studierenden einer Prüfung über das erlernte Stoffgebiet unterzogen. Der Prüfer fertigt bei jeder Prüfung Notizen über das Verständnis des Stoffgebietes an. Dabei ist es wichtig, daß der Prüfer nicht weiß, welcher der beiden Gruppen die Studierenden angehören. Nur so ist eingewisses Maß an Objektivität zu gewährleisten. Nach dem Abschluß der Prüfungen werden die Notizen des Prüfers ausgewertet. Zusätzlich sollte jede Gruppe, die mit dem Virtual Tutor gearbeitet hat, einen Fragebogen über ihre Erfahrungen mit dem System ausfüllen. Die aus der Evaluierung gewonnenen Erkenntnisse sollten anschließend in die Erweiterung des Virtual Tutors einfließen.

Kapitel 12

Zusammenfassung

Der Untersuchungsbereich

Der Mensch in der modernen Informationsgesellschaft sieht sich zunehmend konfrontiert mit dem Problem der rasch wachsenden Wissensmenge. Beispielhaft dafür wurde in Abschnitt 2.5 das exponentielle Wachstum des Wissens in der Wissenschaft diskutiert. Aber auch in anderen Lebensbereichen ist es für den Menschen problematisch, die, in einer Situation notwendigen, handlungsrelevanten Informationen zu erhalten. Die Wissensverarbeitung hat nun die Aufgabe, den Menschen im Umgang mit Wissen zu unterstützen. In Kapitel 2 des Untersuchungsbereichs wurde versucht den Begriff Wissensverarbeitung zu definieren, und die dafür notwendigen Begriffe wie z.B. Daten, Wissen, Information, Bedeutung, usw. vorgestellt.

Um den Menschen im Umgang mit Wissen zu unterstützen, bedient sich die Wissensverarbeitung der Methoden der Künstlichen Intelligenz. In Kapitel 3 wurde eingangs versucht, den Begriff Künstliche Intelligenz zu definieren und anschließend Methoden zur Repräsentation von Wissen vorgestellt. Weiters wurde auf die Notwendigkeit eingegangen, unsicheres Wissen zu speichern und zu verarbeiten.

In den nachfolgenden Kapiteln wurden Techniken der Künstlichen Intelligenz vorgestellt. In Kapitel 4 wurde das Problemlösen durch Suchen behandelt. Es wurden dabei unterschiedliche Algorithmen zur blinden Suche, zur heuristische informierten Suche und zur Suche nach dem optimalen Lösungspfad vorgestellt. Weiters wurde in Kapitel 5 Algorithmen vorgestellt, die es Computern ermöglichen, Strategiespiele zu spielen. In Kapitel 6 wurden zunächst die Aufgaben und Ziele eines Expertensystems definiert. Es wurde auf die unterschiedlichen Arten von Wissen menschlichen Experten eingegangen und gezeigt wie deren Wissen formalisiert und in ein Expertensystem übertragen werden kann. Zudem wurde

mit den Certainty Factors eine Technik vorgestellt, mit deren Hilfe es möglich ist, unsicheres Wissen in Expertensystemen zu verarbeiten. In Kapitel 7 wurde Eingangs die mathematische Theorie der Fuzzy Sets vorgestellt. Darauf aufbauend wurde die Arbeitsweise eines Fuzzy Regelsystems erläutert. In Kapitel 8 wurde anhand des Perzeptrons die Grundlagen Neuronaler Netze vorgestellt und in weiterer Folge der Backpropagation Algorithmus für mehrschichtige Neuronale Netze hergeleitet. Anhand der Vor- und Nachteile von Neuronalen Netzen und der Fuzzy Logic wurden in Kapitel 9 NeuroFuzzy-Systeme vorgestellt.

Intelligente Software Agenten wurden in Kapitel 10 als Möglichkeit vorgestellt, komplexe, verteilte Probleme zu lösen. Dabei bauen die Agenten auf die, in den vorangegangenen Kapiteln vorgestellten, Techniken der Künstlichen Intelligenz auf. Nach der Definition von Agenten wurden die Möglichkeiten besprochen, die sich für die Agenten durch die Kommunikationsfähigkeit ergeben.

Der Gestaltungsbereich

Aufbauend auf den Untersuchungsbereich der Arbeit wurde im Gestaltungsbereich ein interaktiver, dialogbasierter Lernbehelf, der „Virtual Tutor", entwickelt. Der Virtual Tutor stellt ein Expertensystem dar, welches dem Studierenden bei der Auswahl einer Technik der Künstlichen Intelligenz zur Lösung seines Problems helfen soll. Über einen Frage-Antwort Dialog bildet sich der Virtual Tutor ein Bild über die Problemstellung und gibt eine Reihung der geeigneten Techniken aus. Um den Lerneffekt für den Studierenden zu steigern, ist das System in der Lage, Erklärungstexte zu den einzelnen Verarbeitungsschritten zu geben.

Die Implementierung des Virtual Tutors erfolgte als Java-Applet. Somit ist die Anwendung jederzeit für den Studierenden über das Netz erreichbar. Zudem bietet die Realisierung als Applet auch den Vorteil, daß sämtliche Erklärungstexte mit Verweisen zu einer Hintergrundbibliothek versehen werden können. Als Hintergrundbibliothek dient das als HTML-Seiten aufbereitete Wissen, welches im Untersuchungsbereich erarbeitet wurde. Im Kapitel 11 wurde das Konzept des Virtual Tutors diskutiert, die Implementierung schrittweise vorgestellt und ein Ausblick für zukünftige Erweiterungsmöglichkeiten des Systems gegeben.

Anhang A

Verzeichnisse

Abbildungsverzeichnis

Tabellenverzeichnis

Literaturverzeichnis

[Alexa1998] Presseaussendung Alexa Intenet: *Web spawns 1.5 million pages daily according to findings from Alexa Internet*; `http://www.alexa.com/company/inthenews/webfacts.html` (Stand 31.08.1998)

[Altrock1993] Altrock, Constantin von: *Fuzzy Ligic am Praxisbeispiel, in: Spektrum der Wissenschaften*; Heidelberg, März/1993.

[BE1978] *Bertelsmann Universallexikon*; Bertelsmann Lexikonverlag Gütersloh, 1978.

[Böhringer1988] Böhringer, Bernhard; Chiopris Carlo; Futo, Ivan: *Wissensbasierte Systeme mit Prolog*; Addiso-Wesley Verlag Bonn, 1988.

[Caglayan1998] Caglayan, Alper K.; Harrison, Colin G.: *Intelligente Software-Agenten. Grundlagen, Technik und praktische Anwendung im Unternehmen*; Hanser Verlag München Wien, 1998.

[Chowdhury1999] Chowdhury, G. G.: *Introduction to modern information retrieval*; Verlag Library Association Publ. London, 1999.

[CIG1998] CIG Searchbots. University of Massachusetts at Amherst: *CIG Searchbots. Cooperative Information Gathering*; `http://dis.cs.umass.edu/research/searchbots.html` (Stand 05.08.1997)

[Cormen1998] Cormen, Thomas H.; Leiserson, Charles E.; Rivest, Ronald L.: *Introduction to Algorithms*; MIT Press Cambridge, Mass. [u.a.], 1998.

[Dietinger1999] Dietinger, Thomas; Gütl, Christian; Maurer, Hermann; Scherbakov, Nick; Schmaranz Klaus: *Kriterien für ein flexibles System für die Unterstützung von Ausbildungsaufgaben mit moderner Web-Technologie. In: Heilmann, Heidi; [u.a.] (Hrsg.): HMD 205, Praxis der Wirtschaftsinformatik. Multimediale Bildungssysteme*; Hüthig Verlag Heidelberg, 1999.

[Dobrov1980] Dobrov, Gennadij M.: *Wissenschaft. Grundlagen ihrer Organisation und Leitung*; Akadademischer Verlag Berlin, 1980.

[Ebel1998] Ebel, Hans F.; Bliefert, Claus: *Schreiben und Publizieren in den Naturwissenschaften. Neuer Schwerpunkt: elektronisches Publizieren* 4., völlig neu bearb. Aufl.; Verlag Wiley-VCH Weinheim [u.a.], 1998.

[Ernst1996] Ernst, Roman: *Untersuchung verschiedener Problemlösungsmethoden in einem Experten- und Tutorsystem zur makroskopischen Bestimmung krautiger Blütenpflanzen*;
`http://www.biozentrum.uni-wuerzburg.de/~rernst/diplarb/h_dipl_titel.htm`
(Stand 15.02.1998)

[EB1999] *Encyclopædia Britannica*;
`http://www.eb.com` (Stand 19.03.1999)

[Friedman1999] Friedman-Hill, Ernest J.: *Jess, The Java Expert System Shell*;
`http://herzberg.ca.sandia.gov/jess/docs/index.html`
(Stand 21.09.1999)

[Finin1994] Tim Finin *KQML as an Agent Communication Language*;
`http://www.cs.umbc.edu/kqml/papers/kqml-acl-html/root2.html`
(Stand 03.10.1994)

[Gottlob1990] Gottlob, Georg; Frühwirt, Thomas; Horn, Werner: *Expertensysteme*; Springer Verlag Wien, 1990.

[Grötschel1995] Grötschel Martin, Lügger Joachim, Zimmermann Uwe: *Wissenschaftliche Information am Wendepunkt? Zwänge, Krisen und Chancen aus Sicht der Mathematik (Kurzfassung), am 4th Weinheim Librarian Meeting 1995*;
`http://www.wiley-vch.de/books/lib_meet_95/bib04.html`
(Stand 20.08.1999)

[Hacker1992] Hacker, Rupert: *Bibliothekarisches Grundwissen* 6., völlig neu bearb. Aufl.; Verlag Saur München [u.a.], 1992.

[Hauffe1997] Hauffe, Heinz: *Die elektronische Revolution und ihre Auswirkung auf Verlage und Bibliotheken*;
`http://info.uibk.ac.at/sci-org/voeb/vor9402.html`
(Stand 27.11.1997)

[Haykin1994] Haykin, Simon: *Neuronal Networs. A Comprehensive Foundation*; Macmillan Ciollege Publishing Company New York, 1994.

[Heinsohn1999] Heinsohn, Jochen; Socher-Ambrosius, Rolf: *Wissensverarbeitung. Eine Einführung*; Spektrum, Akademischer Verlag Heidelberg [u.a.], 1999.

[Herrmann1997] Herrmann, Jürgen: *Maschinelles Lernen und Wissensbasierte Systeme*; Springer Verlag Berin Heidelberg, 1997.

[Herter1990] Herter, Eberhard; Lörcher, Wolfgang: *Nachrichtentechnink. Übertragung–Vermittlung–Verarbeitung*; Hanser Verlag München Wien, 1990.

[Hinton1992] Hinton, Geoffrey E.: *Wie Neuronale Netze aus Erfahrung lernen, in: Spektrum der Wissenschaften*; Heidelberg, November/1992.

[Hofer1995] Hofer, A.: *Klassische Regler - Fuzzy-Regler: Ein Vergleich, in: Telematik. Zeitschrift des Telematik-Ingenieursverband TIV*; Graz, 2/3 1995.

[ISC2000] Internet Software Consortium; *Internet Domain Survey*; `http://www.isc.org/ds/WWW-9907/report.html` (Stand 19. Jänner 2000)

[Jennings1998] Jennings, Nicholas R.; Wooldridge, Michael J.: *Applications of Intelligent Agents. In: Jennings, Nicholas R.; Wooldridge, Michael J. (Hrsg.): Agent Technology. Foundations, Applications, and Markets*; Springer Verlag Berlin Heidelberg New York, 1998.

[JUCS] Journal of Universal Computer Science. Springer Verlag `http://www.iicm.edu/jucs`

[Köhle1990] Köhle, Monika: *Neurale Netze*; Springer Verlag Wien, 1990.

[Kupka1997] Kupka Ingebert, David Michael *Theorie der Wissensverarbeitung, Sichtweisen und Problemstellungen.* TU-Clausthal, `http://www.in.tu-clausthal.de/~mdavid/TdWV/Kapitel1-1.html`

[Kurzweil1999] Kurzweil, Ray: *Homo S@piens. Leben im 21. Jahrhundert – Was bleibt vom Menschen*; Verlag Kiepenheuer & Witsch Köln, 1999.

[Kruse1997] Kruse, Rudolf; Nauck, Detlef; Klawonn, Frank: *Neuronale Fuzzy-Systeme, in: Spektrum der Wissenschaften. Dossier: Kopf oder Computer*; Heidelberg, 4/1997.

[Lassila1997] Lissila, Ora E.: *Introduction to RDF Metadata*; `http://www.w3.org/TR/NOTE-rdf-simple-intro.html` (Stand 13.11.1997)

[Lassila1998] Lassila, Ora: *Web Metadata. A Matter od Semantics, in: IEEE Internet Compuning*; New York, vol.2, no.4; Juli-August/1998.

[LF] Langenscheidts Fremdwörterbuch online `http://www.langenscheidt.aol.de/`

[LOM1998] IEEE P1484.12 Learning Objects Metadata Working Group: *Learning Objects Metadata (LOM) Draft Document v2.5*; `http://ltsc.ieee.org/doc/wg12/LOMdoc2_5a.doc` (Stand 23.12.1998)

[Luger1997] Luger, Georg; Stubblefield, William: *Artificial Intelligence. Structures and Strategies for complex problem solving*; Addison Wesley Verlag Reading, Mass., 1997.

[Lynch1998] Lynch, Clifford: *Strategien der Informationssuche, in: Spektrum der Wissenschaften. Dossier: Die Welt im Internet*; Heidelberg, 1/1998.

[Meadow1992] Meadow, Charles T.: *Text Information Retrival Systems*; Academic Press San Diego, 1992.

[Nauck1994] Nauck, Detlef; Klawonn, Frank; Kruse, Rudolf: *Neuronale Netze und Fuzzy-Systeme. Grundlagen des Konnektionismus, Neuronaler Fuzzy-Systeme und der Kopplung mit wissensbasierten Methoden*; Vieweg Verlag Braunschweig Wiesbaden, 1994.

[Neumann1997] Neumann, Bernd; Nauck: *Natürliche und künstliche Intelligenz. Computerintelligenz heute — unser Thron wackelt, in: Spektrum der Wissenschaften. Dossier: Kopf oder Computer*; Heidelberg, 4/1997.

[Neussl1998] Neussl, Dietmar *Weiterentwicklung von Werkzeugen zur Wissensauffindung im World-Wide-Web. Ergänzung des Harvest-Systems im Hinblick auf Fehlertoleranz, Konfigurierbarkeit, Keyword-Relevanz und Ranking*; Graz, Technische Universität, Inst. für Informationsverarbeitung und Computergestützte neue Medien, Dipl.-Arb., 1998

[Nilsson1982] Nilsson, J. Nils: *Principles of Artificial Intelligence*; Springer Verlag Berlin Heidelberg, 1982.

[Nwana1998] Nwana, H.S.; Ndumu, D.T.: *A Brief Introduction to Software Agent Technology, In: Jennings, Nicholas R.; Wooldridge, Michael J. (Hrsg.): Agent Technology. Foundations, Applications, and Markets*; Springer Verlag Berlin Heidelberg New York, 1998.

[OCLC1999] Online Computer Library Center: *Electronic Collections Online*; `http://www.oclc.org/oclc/menu/eco.htm` (Stand 23.08.1999)

[Pao1989] Pao, Yoh-Han: *Adaptive Pattern Recognition and Neural Networks*; Addison-Wesley Verlag Reading, 1989.

[Pivec1997] Pivec, Maja: *Knowledge Based Tools: An Example in Knowledge Modeling. In Halim, Zahran; Ottmann, Thomas; Razak, Zaidah (Hrsg.): Proceedings of ICCE97 (International Conference on Computers in Education 1997)*; Association for the Advancement of Computing in Education (AACE) Charlottesville, 1997.

[Puppe1991] Puppe, Frank: *Einführung in Expertensysteme*; Springer Verlag Berlin Heidelberg, 1991.

[Rauch1982] Rauch, Wolf:*Büro-Informations-Systeme. Sozialwissenschaftliche Aspekte der Büro-Automatisierung durch Informations-Systeme*; Verlag Böhlau Wien, Graz [u.a.] 1982.

[Rauch1993] Rauch, Wolf:*Einführung in die Informationswissenschaft Teil 1: Informationsvermittlung.*

[Rauch1996] Rauch, Wolf:*Einführung in die Informationswissenschaft Teil 2: Informationsmanagement.*

[Rauch1994] Rauch, Wolf:*Einführung in die Informationswissenschaft Teil 3: Sozio-Ökonomisches Umfeld.*

[Reeves1997] Reeves, Thomas C.: *Using the WWW as a Cognitive Tool in Higher Education. In Halim, Zahran; Ottmann, Thomas; Razak, Zaidah (Hrsg.): Proceedings of ICCE97 (International Conference on Computers in Education 1997)*; Association for the Advancement of Computing in Education (AACE) Charlottesville, 1997.

[Rich1986] Rich, Elaine: *Artificial intelligence*; McGraw-Hill Verlag New York, 1983.

[Rojas1993] Rojas, Raùl: *Theorie der neuronalen Netze. Eine systematische Einführung*; Springer Verlag Berlin Heidelberg New York, 1993.

[Russell1995] Russell, Stuart J.; Norvig, Peter: *Artificial Intelligence. A modern Approach*; Prentice Hall Verlag, Upper Saddle, NJ, 1995.

[Sander1998] Sander-Beuermann, Wolfgang: *Schatzsucher. Die Internet-Suchmaschinen der Zukunft, in: c't. Magazin für computer und Technik; Seite 178-184*; Hannover, 13/1998.

[Schnupp1986] Schnupp, Peter; Leibrandt Ute: *Expertensysteme. Nicht nur für Informatiker*; Springer Verlag Berlin Heidelberg, 1986.

[Seraphin1994] Seraphin, Marco: *Neuronale Netze und Fuzzy-Logik. Verknüpfung der Verfahren, Anwendungen, Vor- und Nachteile, Simulationsprogramm*; Franzis Verlag München, 1994.

[Stix1998a] Stix, Gary: *Publizieren mit Lichtgeschwindigkeit, in: Spektrum der Wissenschaften. Dossier: Die Welt im Internet*; Heidelberg, 1/1998.

[Stix1998b] Stix, Gary: *Das Auffinden von Bildern, in: Spektrum der Wissenschaften. Dossier: Die Welt im Internet*; Heidelberg, 1/1998.

[SEW1999] Search Engine Watch: *Search Engine Coverage Study Published*; `http://www.searchenginewatch.com/sereport/99/08-size.html`
(Stand 02.08.1999)

[Stubenrauch1998] Stubenrauch, Robert; Vickery, Barbara; Ruppert, Ato: *LIBERATION: A Value-Added Digital Libary*;
`http://www.iicm.edu/stubenrauch/ECDL98`
(Stand September 1998)

[Thomas1997] Thomas, Christoph; Fischer, Gerhard: *Using agents to personalize the Web. In: Moore, J.; Edmonds, E.; Puerta, A. (Hrsg.): Proceedings IUI '97 - International Conference on Intelligent User Interfaces.*; New York: ACM, 1997.

[Tröger1998] Tröger, Beate: *Das Internet in der Lehr- und Wissenschaftspraxis. Aufgaben und Zielsetzungen für Wissenschaftliche Bibliotheken*; `http://eldorado.uni-dortmund.de:8080/bib/97/troeger2/htmldoc` (Stand 09.12.1998)

[Umstätter1998] Umstätter, Walther: *Die Zukunft des Buches, in: Spektrum der Wissenschaften. Dossier: Die Welt im Internet*; Heidelberg, 1/1998.

[Wersig1971] Wersig, Gernot: *Information - Kommunikation - Dokumentation. Ein Beitrag zur Orientierung der Informations- und Dokumentationswissenschaft*; Verl. Dokumentation München-Pullach [u.a.], 1971.

[Winston1993] Winston, Patrick Henry: *Artificial intelligence*; Addison-Wesley Verlag Reading, 1993.

[ZDB1998] Deutsche Bibliotheksinstitut: *Zeitschriftendatenbank (ZDB). Kurzinfo*;
`http://www.dbi-berlin.de/de/ibas/zdb/zdb01.htm`
(Stand 26.05.1998)

[Zimmer1998] Zimmer, Dieter E.: *Die langsame Lösung vom Papier. Die digitale Bibliothek (IV) - Eine Artikelserie für Nutzer und verächter der Computernetze*;
`http://www.zeit.de/tag/digbib/digbib4.html`
(Stand 30.6.1998)

[Zimmermann1993] Zimmermann, Hans-Jürgen: *Prinzipien der Fuzzy Ligic, in: Spektrum der Wissenschaften*; Heidelberg, 3/1995.

Anhang B

CD-ROM

Inhalt:

- Referenzen, soweit in elektronischer Form verfügbar
- Vorliegende Arbeit
- Sourcecode