Kognitive Modellierung: Menschliche Wissensrepräsentationen und Verarbeitungsstrategien

Franz Schmalhofer und Thomas Wetter

Abstract

In diesem Kapitel wird Kognitive Modellierung als ein interdisziplinäres Forschungsgebiet vorgestellt, das sich mit der Entwicklung von computerimplementierbaren Modellen beschäftigt, in denen wesentliche Eigenschaften des Wissens und der Informationsverarbeitung beim Menschen abgebildet sind. Nach einem allgemeinen Überblick über Zielsetzungen, Methoden und Vorgehensweisen, die sich auf den Gebieten der kognitiven Psychologie und der Künstlichen Intelligenz entwickelt haben, sowie der Darstellung eines Theorierahmens werden vier Modelle detaillierter besprochen: In einem Lernmodell, das in einem Intelligenten Tutoriellen System Anwendung findet und in einem Performanz-Modell der Mensch-Computer-Interaktion wird menschliches Handlungswissen beschrieben. Die beiden anderen Modelle zum Textverstehen und zur flexiblen Gedächtnisorganisation beziehen sich demgegenüber vor allem auf den Aufbau und Abruf deklarativen Wissens. Abschließend werden die vorgestellten Modelle in die historische Entwicklung eingeordnet. Möglichkeiten und Grenzen der Kognitiven Modellierung werden hinsichtlich interessant erscheinender Weiterentwicklungen diskutiert.

1. Einleitung und Überblick

Das Gebiet der Künstlichen Intelligenz wird meist unter Bezugnahme auf ursprünglich nur beim Menschen beobachtetes Verhalten definiert. So wird die **Künstliche Intelligenz** oder **KI** als die Erforschung von jenen Verhaltensabläufen verstanden, deren Planung und Durchführung Intelligenz erfordert. Der Begriff *Intelligenz* wird dabei unter Bezugnahme auf den Menschen vage abgegrenzt [Siekmann_83,Winston_84]. Da auch Teilbereiche der Psychologie, vor allem die **Kognitive Psychologie**, Intelligenz und Denken untersuchen, könnte man vermuten, daß die KI-Forschung als die jüngere Wissenschaft direkt auf älteren psychologischen Erkenntnissen aufbauen würde.

Obwohl KI und kognitive Psychologie einen ähnlichen Gegenstandsbereich erforschen, gibt es jedoch auch vielschichtige Unterschiede zwischen beiden Disziplinen. Daraus läßt sich möglicherweise erklären, daß die beiden Fächer bislang nicht in dem Maß interagiert haben, wie dies wünschenswert wäre.

1.1 Unterschiede zwischen KI und Kognitiver Psychologie

Auch wenn keine klare Grenze zwischen den beiden Gebieten gezogen werden kann, so müssen wir doch feststellen, daß KI nicht gleich Kognitiver Psychologie ist. Wichtige Unterschiede bestehen in den primären Forschungszielen und Methoden, sowie in der Interpretation von Computermodellen (computational models).

Zielsetzungen und Methoden

Während die KI eine Modellierung von Kompetenzen anstrebt, erforscht die Psychologie die Performanz des Menschen.

- Die KI sucht nach Verfahren, die zu einem intelligenten Verhalten eines Computers führen. Beispielsweise sollte ein Computer natürliche Sprache verstehen, neue Begriffe lernen können oder Expertenverhalten zeigen oder unterstützen. Die KI versucht also, intelligente Systeme zu entwickeln und deckt dabei mögliche Prinzipien von Intelligenz auf, indem sie Datenstrukturen und Algorithmen spezifiziert, die intelligentes Verhalten erwarten lassen. Entscheidend ist dabei, daß eine intelligente Leistung im Sinne eines Turing-Tests erbracht wird: Eine Implementierung des Algorithmus soll für eine Menge spezifizierter Eingaben (z. B. gesprochene Sprache) innerhalb angemessener Zeit die vergleichbare Verarbeitungsleistung erbringen wie der Mensch. Der beobachtete Systemoutput von Mensch und Computer wäre also oberflächlich betrachtet nicht voneinander unterscheidbar [Turing_63]. Ob die dabei im Computer verwendeten Strukturen, Prozesse und Heuristiken denen beim Menschen ähneln, spielt in der KI keine primäre Rolle.
- Die Kognitive Psychologie hingegen untersucht eher die internen kognitiven Verarbeitungsprozesse des Menschen. Bei einer psychologischen Theorie sollte also auch das im Modell verwendete Verfahren den Heuristiken entsprechen, die der Mensch verwendet. Beispielsweise wird ein Schachprogramm nicht dadurch zu einem psychologisch adäquaten Modell, daß es die Spielstärke menschlicher Meisterspieler erreicht. Vielmehr sollten bei einem psychologischen Modell auch die Verarbeitungsprozesse von Mensch und Programm übereinstimmen (vgl. dazu [deGroot_66]).Für psychologische Forschungen sind daher empirische und gezielte experimentelle Untersuchungen der menschlichen Kognition von großer Bedeutung.

In der KI steht die **Entwicklung und Implementierung** von Modellen im Vordergrund. Die kognitive Psychologie dagegen betont die Wichtigkeit der **empirischen Evaluation** von Modellen zur Absicherung von präzisen, allgemeingültigen Aussagen. Wegen dieser verschiedenen Schwerpunktsetzung und den daraus resultierenden unterschiedlichen Forschungsmethoden ist es für die Forscher der einen Disziplin oft schwierig, den wissenschaftlichen Fortschritt der jeweils anderen Disziplin zu nutzen [Miller_78].

Interpretation von Computermodellen

Die KI ist aus der Informatik hervorgegangen. Wie bei der Informatik bestehen auch bei der KI wissenschaftliche Erkenntnisse darin, daß mit ingenieurwissenschaftlichen Verfahren neue Systeme wie Computerhard- und -software konzipiert und erzeugt werden. Die genaue Beschreibung eines so geschaffenen Systems ist für den Informatiker im Prinzip unproblematisch, da er das System selbst entwickelt hat und daher über dessen Bestandteile und Funktionsweisen bestens informiert ist.

Darin liegt ein Unterschied zu den empirischen Wissenschaften wie der Physik oder Psychologie. Der Erfahrungswissenschaftler muß Objektbereiche untersuchen, deren Gesetzmäßigkeiten er nie mit letzter Sicherheit feststellen kann. Er muß sich daher Theorien oder Modelle über den Untersuchungsgegenstand bilden, die dann empirisch überprüft werden können. Jedoch läßt sich durch eine noch so große Anzahl von Experimenten niemals die Korrektheit eines Modells beweisen [Popper_66]. Ein einfaches Beispiel kann diesen Unterschied verdeutlichen.

- Ein Hardwarespezialist, der einen Personal Computer gebaut hat, weiß, daß die Aussage "Der Computer ist mit 640 KB Hauptspeicher bestückt" richtig ist, weil er ihn eben genau so bestückt hat. Dies ist also eine feststehende Tatsache, die keiner weiteren Überprüfung bedarf.
- Die Behauptung eines Psychologen, daß der menschliche Kurzzeit- oder Arbeitsspeicher eine Kapazität von etwa 7 Einheiten oder Chunks habe, hat jedoch einen ganz anderen Stellenwert. Damit wird keinesfalls eine faktische Behauptung über die Größe von Arealen im menschlichen Gehirn aufgestellt. "Arbeitsspeicher" wird hier als theoretischer Term eines Modells verwendet. Mit der Aussage über die Kapazität des Arbeitsspeichers ist gemeint, daß erfahrungsgemäß Modelle, die eine solche Kapazitätsbeschränkung annehmen, menschliches Verhalten gut beschreiben können. Dadurch wird jedoch nicht ausgeschlossen, daß ein weiteres Experiment Unzulänglichkeiten oder die Inkorrektheit des Modells nachweist.

In den Erfahrungswissenschaften werden theoretische Begriffe wie etwa Arbeitsspeicher innerhalb von Computermodellen zur abstrahierten und integrativen Beschreibung von empirischen Erkenntnissen verwendet. Dadurch können beim Menschen zu beobachtende Verhaltensweisen vorhergesagt werden. Aus der Sichtweise der Informatik bezeichnen genau die gleichen Terme jedoch tatsächliche Komponenten eines Geräts oder Programms. Diese unterschiedlichen Sichtweisen der gleichen Modelle verbieten einen unkritischen und oberflächlichen Informationstransfer zwischen KI und Kognitiver Psychologie.

Aus der Integration der Zielsetzungen und Sichtweisen ergeben sich jedoch auch gerade vielversprechende Erkenntnismöglichkeiten über Intelligenz. Da theoretische wie auch empirische Untersuchungen zum Verständnis der Intelligenz beitragen, können sich die Methoden und Erkenntnisse von beiden Disziplinen (ähnlich wie Mathematik und Physik im Bereich der theoretischen Physik) ergänzen und befruchten.

1.2 Synthese von KI und Kognitiver Psychologie

Im Rahmen der Kognitionswissenschaften(cognitive science) tragen viele Disziplinen (z.B. KI, Psychologie, Linguistik, Anthropologie ...) Erkenntnisse über informationsverarbeitende Systeme bei. Die **Kognitive Modellierung** als ein Teilgebiet von sowohl KI als auch Kognitiver Psychologie befaßt sich mit der **Entwicklung** von **computerimplementierbaren Modellen**, in denen wesentliche Eigenschaften **des Wissens und der Informationsverarbeitung beim Menschen** abgebildet sind. Durch Kognitive Modellierung wird also eine Synthese von KI und psychologischer Forschung angestrebt.

Ein Computermodell wird zu einem kognitiven Modell, indem Entitäten des Modells psychologischen Beobachtungen und Erkenntnissen zugeordnet werden. Da ein solches Modell auch den Anspruch erhebt, menschliches Verhalten vorherzusagen, können Kognitive Modelle aufgrund empirischer Untersuchungen weiterentwickelt werden. Die Frage, ob ein KI-Modell als ein kognitives Modell anzusehen ist, kann nicht einfach bejaht oder verneint werden, sondern wird vielmehr durch die Angabe einer Zuordnung von Aspekten der menschlichen Informationsverarbeitung zu Eigenschaften des Computermodells beantwortet.

Vorgehensweisen

Eine kognitive Modellierung kann sich im Prinzip auf zwei Weisen vollziehen.

- Für vorliegende KI-Modelle kann überprüft werden, inwieweit Menschen Informationsverarbeitungsprobleme auf gleiche oder ähnliche Weise lösen. In solchen Untersuchungen kann festgestellt werden, in welchen Aspekten ein KI-Modell auch als ein kognitives Modell angesehen werden kann (z. B. [Schank_77,Bower_79]).
- Umgekehrt gibt es auch Modellentwicklungen, deren primäres Ziel es ist, die kognitive Informationsverarbeitung beim Menschen zu beschreiben. Bei solchen Modellierungen werden zunächst psychologische Erkenntnisse, beispielsweise über das menschliche Gedächtnis, zusammengetragen. Diese Erkenntnisse bilden dann die Grundsätze und prinzipiellen Restriktionen für die durchzuführende Modellierung. Um das Informationsverarbeitungsproblem, dessen Bewältigung durch die kognitive Modellierung beschrieben werden soll, mit einem Simulationsprogramm zu lösen, werden meist Programmiertechniken aus der KI entliehen. Wie auch in der KI führt die Komplexität der Aufgabenstellungen oft dazu, daß wichtige Teile des Modells nicht implementiert werden können. Der Frage, welche Aspekte im Modell implementiert werden, kommt daher entscheidende Bedeutung zu (siehe [Wetter_85]).

Die zwei Vorgehensweisen haben die gleiche Zielsetzung. In beiden Fällen sollen grundlegende Prinzipien menschlicher Intelligenzleistungen gefunden werden. Der Nutzen eines Kognitiven Modells liegt vor allem auch darin, daß interessante und auch praxisrelevante Fragen in Angriff genommen und beantwortet werden können, die ohne eine explizite Modellierung wegen der Komplexität der menschlichen Intelligenz nicht systematisch erforscht werden können. Die beiden nächsten Abschnitte sollen beispielhaft aufzeigen, wie einerseits die psychologische Angemessenheit bestehender KI-Modelle beurteilt werden kann und wie andererseits Modellierungen der menschlichen Informationsverarbeitung schrittweise aufgebaut und eingesetzt werden können.

Psychologische Experimente zur Überprüfung von KI-Modellen

Der mögliche Nutzen experimenteller Überprüfungen von Computermodellen kann an Untersuchungen von [Swinney_79] [Swinney_84] verdeutlicht werden. Swinney hat eine Reihe von Experimenten durchgeführt, die in den wesentlichen Punkten insgesamt übereinstimmende Ergebnisse lieferten. Die Untersuchungen zeigen, daß eine einfache Anwendung von Schema-Modellen die kognitiven Prozesse der Wortdesambiguierung nicht hinreichend erklärt.

Wortdesambiguierung. Bekanntlich verstehen die Menschen Wörter, die in einer Sprache zwei oder mehrere Bedeutungen haben, aufgrund des sprachlichen Kontextes richtig. So kann sich das Wort "Bank" beispielsweise auf ein Kreditinstitut oder auch auf eine Sitzgelegenheit im Park beziehen. Die Auswahl der richtigen kontextadäquaten Bedeutung eines Wortes bezeichnet man als Wortdesambiguierung.

KI-Erklärung durch Schematheorie. Für die KI ist es naheliegend, Wortdesambiguierung mit Hilfe von Schemata zu beschreiben ([Charniak_85],S.598). Schemata sind im Prinzip genommen Ansammlungen von Konstanten und Variablen zur Erfassung von stereotypen Situationen. Darauf abgestimmte Prozeduren legen fest, welches Schema aktiviert wird und in welchen Variablen aktuell auftretende Ausprägungen abgelegt werden. In dem hier betrachteten Fall der Wortdesambiguierung wird angenommen, daß jedes Schema ein Lexikon für kontextspezifische Wörter enthält. Wie in Abb. 1 dargestellt, würde das Wort "Bank" sowohl in dem Schema "Wandern" als auch in dem Schema "Geldgeschäfte" mit der jeweils schemaadäquaten Bedeutung vorkommen. Darüberhinaus wird ein Standardlexikon postuliert, das die Wortbedeutungen enthält, die weniger stark vom Kontext geprägt werden. In dem hier betrachteten Fall wird angenommen, daß mit Hilfe des Kontextes bereits das adäquate Schema aktiviert wurde. Bei einer Aktivierung des Schemas "Wandern" würde daher das Wort Bank von Anfang an nur in dem Sinn von Rastgelegenheit verstanden. Nach dieser Schematheorie sollte also prinzipiell nur die kontextadäquate Interpretation des Wortes Bank erfolgen. Bank im Sinne von Kreditinstitut würde nie in Betracht gezogen werden. Zur Wortdesambiguierung sagt also die Schematheorie eine frühe Bedeutungsselektion voraus. Dagegen würden bei einer späten Bedeutungsselektion zuerst alle möglichen Bedeutungen eines Wortes aktiviert, bevor eine Auswahl der kontextadäquaten Bedeutung erfolgen kann.

Zur gezielten experimentellen Überprüfung dieser Schematheorie der Wortdesambiguierung werden psychologische Fakten benötigt, welche die Bestimmung der Aktivierung einer Wortbedeutung im menschlichen Gedächtnis erlauben. Dazu werden semantische Bahnungseffekte herangezogen.

Semantische Bahnungseffekte. Viele psychologische Experimente bestätigen, daß die kognitive Bearbeitung eines Wortes auch das semantische Umfeld der Wortbedeutung im Gedächtnis aktiviert. Wenn jemand beispielsweise das Wort *"Vater"* bearbeitet, wird *Kind* im Gedächtnis mit-aktiviert. Da *Kind* nicht zum semantischen Umfeld von *Röhre* gehört, wird dagegen bei der Bearbeitung des Wortes *"Röhre"* im menschlichen Gedächtnis *Kind* nicht mit-aktiviert. Die Aktivierung von *Kind* im semantischen Gedächtnis führt nun dazu, daß darauf bezogene Fragen schneller beantwortet werden als wenn keine solche **Bahnung** vorliegt. Eine solche Verkürzung von Antwortlatenzzeiten bezeichnet man als **semantischen Bahnungseffekt** *(semantic priming)*. Besonders stark treten semantische Bahnungseffekte bei lexikalischen Entscheidungsaufgaben auf. In diesen Aufgaben muß eine Person möglichst schnell entscheiden, ob eine präsentierte Buchstabenfolge (etwa "Sim", "Kind", "Bild") ein Wort ist oder nicht. Interessant ist dabei der Vergleich der Antwortlatenzzeiten bei kontextadäquaten und -inadäquaten Wörtern (hier etwa "Kind", "Bild"), während die Nichtwörter nur aus methodischen Gründen dargeboten werden.

Experiment zur Wortdesambiguierung. Von Swinney wurde der oben erläuterte Bahnungseffekt eingesetzt, um festzustellen, welche Bedeutungen bei ambiguen Wörtern zu zwei verschiedenen Verarbeitungszeitpunkten im Gedächtnis aktiviert werden. Somit konnte erforscht werden, ob Wortdesambiguierung beim Menschen durch eine frühe oder späte Bedeutungsselektion erfolgt. In den Experimenten wurden Texte wie der folgende über Kopfhörer dargeboten:

In der Region, die in letzter Zeit aufgeblüht war, gab es jetzt große Probleme. Ein (folgenreicher Firmenkonkurs, folgenreiches Sommergewitter) hatte schwerste Schäden angerichtet. Als der Mann zur Bank $_1$ kam, er$_2$ahnte er den Umfang des Schadens.

Wie durch das Wort *"Vater"* wurde mit *"folgenreicher Firmenkonkurs"* oder *"folgenreiches Sommergewitter"* eines von zwei semantischen Umfeldern aktiviert (dies entspricht der Instanziierung eines Schemas im

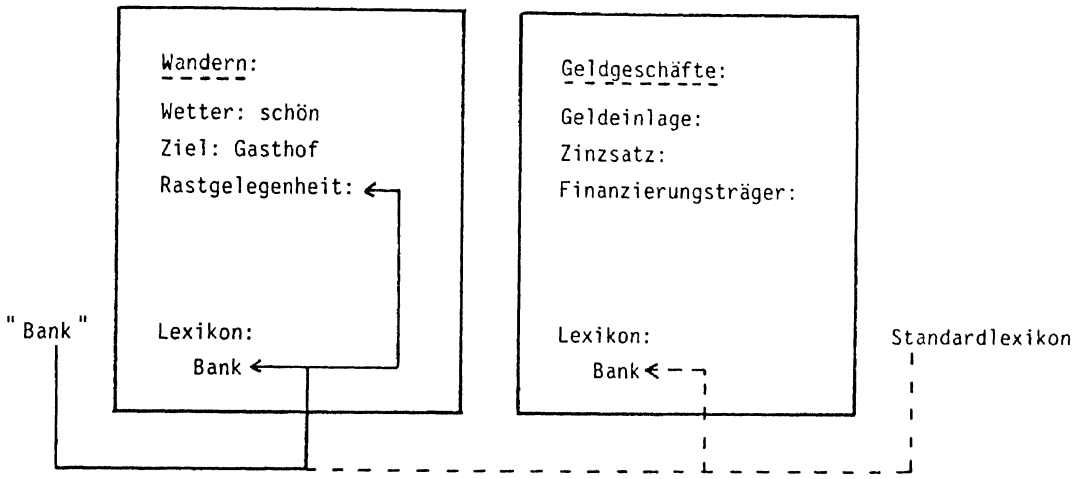

Abb. 1

Eine Schematheorie zur Erklärung von Wort-Desambiguierung nach [Charniak_85].

KI-Modell). Daraufhin folgte im Text ein ambigues Wort (z.B. "Bank"), das eine kontextadäquate und eine -inadäquate Bedeutung hat. Die lexikalische Entscheidungsaufgabe wurde direkt nach dem kritischen Wort (im Text mit 1 gekennzeichnet) oder zwei Silben nach dem kritischen Wort (im Text mit 2 gekennzeichnet) vorgegeben. In beiden Fällen erschien je nach Bedingung eine von vier verschiedenen Buchstabensequenzen am Bildschirm, ein auf Kontext 1 bezogenes Wort (Geld), ein auf Kontext 2 bezogenes Wort (Park), ein neutrales Wort (Turm) oder ein Nichtwort (Sirn).

Aufgrund der Bahnungseffekte, die sich beispielsweise durch die Verkürzung der Antwortzeiten für *"Geld"* gegenüber *"Turm"* ergeben, wurde bestimmt, inwieweit nur eine oder beide Wortbedeutungen (*Kreditinstitut* und *Sitzgelegenheit*) zu den zwei Testzeitpunkten aktiviert waren. Um eine methodisch angemessene Versuchsdurchführung zu gewährleisten, wurden die verschiedenen Meßreihen über eine hinreichend große Zahl von Versuchspersonen und über mehrere, ähnlich konstruierte Texte ausbalanciert. Da die Versuchspersonen nahezu alle Testitems richtig beantworteten, interessieren bei der Datenauswertung vor allem die Reaktionszeitunterschiede zwischen den neutralen und den auf Kontext 1 bzw. Kontext 2 bezogenen Wörtern.

Die statistisch signifikanten Ergebnisse zeigen, daß kurz nach dem Hören des kritischen Wortes sowohl die kontextadäquate als auch die kontextinadäquate Bedeutung des Wortes aktiviert wurden. Zum Zeitpunkt 2 war dagegen nur noch die kontextadäquate Bedeutung aktiv. Im Widerspruch zu den Vorhersagen der Schematheorie zeigen die Ergebnisse, daß anfangs beide Wortbedeutungen aktiviert werden und erst zu einem späteren Zeitpunkt eine Wortdesambiguierung erfolgt. Sicherlich wäre es falsch, aufgrund dieses Ergebnisses Schema-Modelle als psychologisch unangemessen zu erklären. Ein experimenteller Befund dieser Art sollte vielmehr als Anregung verstanden werden, auch andere Mechanismen der Wortdesambiguierung im Detail auszuarbeiten. Konnektionistische Modelle [Feldman_82], die im Gegensatz zu Schematheorien global eine parallele Informationsverarbeitung annehmen, stellen beispielsweise einen solchen Ansatz dar, der Swinneys experimentelle Befunde korrekt vorhersagen würde.

Personenbefragungen, Lautes Denken und Modellentwicklung

Bei der Entwicklung von Expertensystemen wird durch verschiedenartige Befragungen menschlicher Experten aufgabenrelevantes Wissen erhoben. Dadurch wird einerseits das Input-Output Verhalten eines Systems durch einen ausgewiesenen Experten legitimiert. Neben einer allgemeinen Überprüfung von Input-Output Beziehungen interessiert bei der Entwicklung eines Expertensystems jedoch auch, welches Wissen ein Experte bei einer bestimmten Aufgabenstellung aktuell einsetzt.

Von Personen abgegebene Begründungen und Beschreibungen von mentalen Operationen und Handlungen stimmen jedoch oft nicht mit dem tatsächlichen Verhalten überein [Nisbett_77] . Die Arbeit von [Ericsson_80] zeigt, daß bereits die Anwendung eines sehr allgemeinen Informationsverarbeitungsmodells entscheidend zur Beantwortung der Frage beitragen kann, welche Expertenaussagen mit den tatsächlich verwendeten kognitiven Operationen übereinstimmen.

Ein einfaches Gedächtnismodell. Auf der Grundlage vieler Gedächtnisuntersuchungen läßt sich feststellen, daß im menschlichen Gedächtnis gespeicherte Informationen verschieden kodiert und unterschiedlich leicht abrufbar sind. Zu jedem Zeitpunkt der Informationsverarbeitung gibt es einen relativ kleinen Anteil von Informationen, der schnell und leicht aktiviert werden kann. Es sind dies die Informationen, die sich gerade im Arbeitsspeicher befinden. Die Kapazität des Arbeitsspeichers beträgt zirka 7 Einheiten [Miller_56] , die als ***chunks*** bezeichnet werden. Die in einem *memory-chunk* gespeicherte Information besteht aus einer kognitiven Einheit (d.h. aus einer konkreten Datenstruktur), die jedoch mehr oder weniger komplex sein kann. Man kann sich dies so vorstellen, daß ein chunk durch einen Zeiger dargestellt wird. Der Inhalt des Arbeitsspeichers würde dann die von den Zeigern adressierten Datenstrukturen des Langzeitspeichers enthalten. Durch umfangreiches Lernen kann eine Datenstruktur größer und komplexer werden. Beispielsweise kann durch jahrelanges Üben erreicht werden, daß sich eine Person nach einmaliger Darbietung 80 zufällig aufeinanderfolgende Ziffern merken kann [Chase_82] . Zusätzlich zu den Verarbeitungsprozessen, die Informationen im Arbeitsspeicher verändern und dabei auf verbale Information zurückgreifen, müssen auch automatisierte Prozesse berücksichtigt werden.

Anwendung des Gedächtnismodells. Durch eine Anwendung dieses einfachen Modells und die Analyse vieler Gedächtnisprotokolle konnten Ericsson und Simon [Ericsson_80] angeben, welche Informationen bei den verschiedenen Befragungsverfahren verbalisiert werden. Es zeigte sich, daß sowohl der Zeitpunkt der Befragung als auch die Aufgabeninstruktion die Qualität der Befragungsdaten stark beeinflussen:

- Werden Personen instruiert, **gleichzeitig bei der Durchführung einer Aufgabe alle Gedanken auszusprechen**, die ihnen gerade durch den Kopf gehen, so werden all die Informationen verbalisiert, die zu jedem Zeitpunkt der Aufgabendurchführung im Arbeitsspeicher verbal kodiert vorliegen und gerade bearbeitet werden.
- Wird eine Person **nach Beendigung einer Aufgabe** befragt, so müssen die entsprechenden Informationen zuerst im Langzeitspeicher aufgefunden werden, bevor eine Verbalisierung stattfinden kann. Da dieser Suchprozeß nicht immer erfolgreich abläuft und nicht sämtliche relevanten Informationen in den Langzeitspeicher übergegangen sind, sind so gewonnene Beschreibungen oft sehr lückenhaft. Da ferner durch das Absuchen des Langzeitspeichers aufgrund assoziativer Verknüpfungen auch Informationen aufgefunden werden können, die während der Bearbeitung einer Aufgabe gar nicht verwendet wurden, können hier auch unrealistische Beschreibungen zustandekommen.
- Werden zu einer **systematischen Befragung** vom Fragesteller theoretische Konzeptionen eingeführt, so kann dies dazu führen, daß der Befragte diese Konzeptionen zum Auffinden einer Antwort verwendet, obwohl sie für ihn bisher irrelevant waren. Bei solchen Befragungen werden dann im Langzeitspeicher Informationen aktiviert, die bei der eigentlichen Aufgabendurchführung nicht herangezogen wurden, nun jedoch dazu verwendet werden, eine rationale Begründung zu erzeugen.
- Informationen, die im Gedächtnis nicht verbal kodiert vorliegen, müssen zuerst in eine verbale Repräsentation überführt werden, bevor sie ausgesprochen werden können. Solche Informationen sind daher mit der Methode des lauten Denkens schwer in Erfahrung zu bringen.

Überprüfungsmöglichkeiten des Modellansatzes. Auf den ersten Blick erscheint die hier gegebene Darstellung vielleicht zirkulär. (Der Modellansatz wurde aus den Beobachtungen abgeleitet und wird nun wieder verwendet, um die gleichen Beobachtungen vorherzusagen). Zirkularität ist jedoch nicht gegeben, da das Modell durch davon unabhängige Daten und Analysen abgesichert werden kann. Wird nämlich das Laute Denken bei einer Aufgabe durchgeführt, deren Struktur bekannt ist, so kann die Vollständigkeit der verbalen

Äußerungen überprüft werden.

Für eine Multiplikationsaufgabe wie 21 x 19 können beispielsweise die Strategien und Sequenzen von Operationen angegeben werden, die zu einer richtigen Lösung der Aufgabe führen. Durch einen Vergleich der Ergebnisse der Aufgabenanalyse mit den Protokolldaten des Lauten Denkens kann somit der Realitätsgehalt und die Vollständigkeit der Denkprotokolle abgesichert werden. Eine weitere Bestätigung des Modellansatzes liegt darin, daß die Resultate und Zeiterfordernisse einer Aufgabendurchführung mit und ohne Lautes Denken in wesentlichen Aspekten meist identisch sind (siehe etwa [Schmalhofer_86c]).

Bei Multiplikationsaufgaben kann man auch feststellen, daß Erwachsene beim Lauten Denken einige Operationen nicht nennen, die von Kindern explizit verbalisiert werden, wie das Multiplizieren zweier einstelliger Zahlen durch mehrere Additionsoperationen. Vermutlich sind diese Operationen bei Erwachsenen automatisiert oder durch das Nachsehen in einer mental gespeicherten Tabelle ersetzt. Es zeigt sich also, daß mit zunehmender Übung manche Operationen automatisiert oder verkürzt werden und dadurch beim Lauten Denken nicht mehr auftreten.

Um nun bei Aufgaben unbekannter Struktur möglichst zuverlässige und vollständige Informationen über das für die Lösung benötigte Wissen zu erhalten, empfiehlt es sich daher, Personen mit unterschiedlich weit fortgeschrittener Expertise direkt während der Aufgabendurchführung laut denken zu lassen. Aus der Synopse der so erhaltenene Denkprotokolle sollte sich eine vollständigere Beschreibung der zur Lösung der Aufgabe benötigten Verfahren (re-)konstruieren lassen. Daraus können auch wesentliche Erkenntnisse über die Aufgabenstruktur gewonnen werden.

Das von [Ericsson_80] angegebene Gedächtnismodell kann als Vorarbeit für eine kognitive Modellierung angesehen werden, aus der sich bereits die Nützlichkeit dieses Ansatzes erkennen läßt. Werden nun KI-Methoden herangezogen, um solche Modellskizzen auszuarbeiten und zu präzisieren, so können Computermodelle entstehen, die sehr viel detailliertere Vorhersagen liefern.

1.3 ACT als Theorierahmen für Kognitive Modellierungen

In Erweiterung zu den bisher an spezifischen Anwendungen dargestellten Ansätzen hat [Anderson_76] bei der Entwicklung von ACT (Adaptive Control of Thought) die Absicht verfolgt, ein System zu spezifizieren, das sämtliche höheren kognitiven Funktionen des Menschen modelliert. In den Jahren 1974 bis 1983 wurden mehrere Versionen von ACT beschrieben. ACT* , das hier vorgestellt wird, ist die neueste dieser Versionen [Anderson_83].

Ziele

ACT* soll nach Anderson alle kognitiven Bereiche wie Gedächtnis, Sprache, Problemlösen, Induktives Denken, Deduktives Denken etc. auf der Basis einheitlicher erster Prinzipien modellieren. Bei der Spezifikation von ACT* wurden drei Ziele verfolgt: ACT* sollte so aufgebaut werden, daß menschliches Verhalten und menschliche Lernvorgänge damit nachgebildet werden können. ACT*-Modellierungen sollten durch Problemlösen und Handeln das gleiche deklarative und prozedurale Wissen erwerben, das sich Menschen bei solchen Tätigkeiten aneignen. Eine vollständige Implementierung von ACT* müßte also den Turing-Test bestehen können. Darüberhinaus sollte ACT* bereits vorliegende empirische Erkenntnisse erklären können. Mit ACT* sollen also die Prinzipien der menschlichen Informationsverarbeitung angegeben werden, die unter den verschiedensten Aufgabenstellungen auftreten, so daß eine einheitliche Theorie über den menschlichen Verstand (mind) entsteht.

Architektur

Die grundsätzliche Architektur von ACT* ist in Abb. 2 dargestellt. Neben dem Arbeitsspeicher, in dem die jeweils aktivierten Informationen enthalten sind, werden zwei Langzeit-Speicher voneinander unterschieden, ein prozeduraler und ein deklarativer Speicher.

Prozedurales Wissen. Der prozedurale Speicher beinhaltet Produktionen, die das menschliche Handlungswissen beschreiben sollen. Eine Produktion ist im wesentlichen eine wenn-dann Regel, die aus einem Bedingungsteil und einem Aktionsteil besteht und meist in der folgenden Form dargestellt wird:

```
Wenn (Bedingung)  Dann (Aktion).
```

Der Bedingungsteil kann Spezifikationen von Zielen, Beschreibungen von kognitiven Zuständen und wahrgenommene externe Reize beinhalten. Eine Ansammlung von Produktionen bildet gemeinsam mit einem Interpreter ein Produktionssystem.

Der Interpreter vergleicht die Bedingungsteile der Produktionen mit dem Inhalt des Arbeitsspeichers. Falls der Inhalt des Arbeitsspeichers die Bedingung einer Produktion erfüllt, kann die in der Produktion angegebene Aktion ausgeführt werden. Eine Aktion kann neue Informationen einschließlich neuer Ziele in den Arbeitsspeicher schreiben, so daß im allgemeinen als nächstes der Bedingungsteil einer anderen Produktion mit dem Inhalt des Arbeitsspeichers übereinstimmen wird. Aktionen können auch beobachtbare Handlungen enthalten. Die Regel R1 beschreibt beispielsweise eine Produktion in Form einer umgangssprachlich formulierten wenn-dann Regel, die ein Student bei der Programmierung einer LISP-Funktion anwendet.

```
R1 Wenn (Ziel : Eliminiere das erste Element aus der Liste LIST1)

   Dann (Schreibe "(Cdr LIST1)" und lösche das o.g. Ziel im
         Arbeitsspeicher)
```

Falls die Bedingungen mehrerer Produktionen mit dem Inhalt des Arbeitsspeichers übereinstimmen, wird aufgrund von Konfliktresolutionsverfahren entschieden, welche Produktion ausgeführt wird.

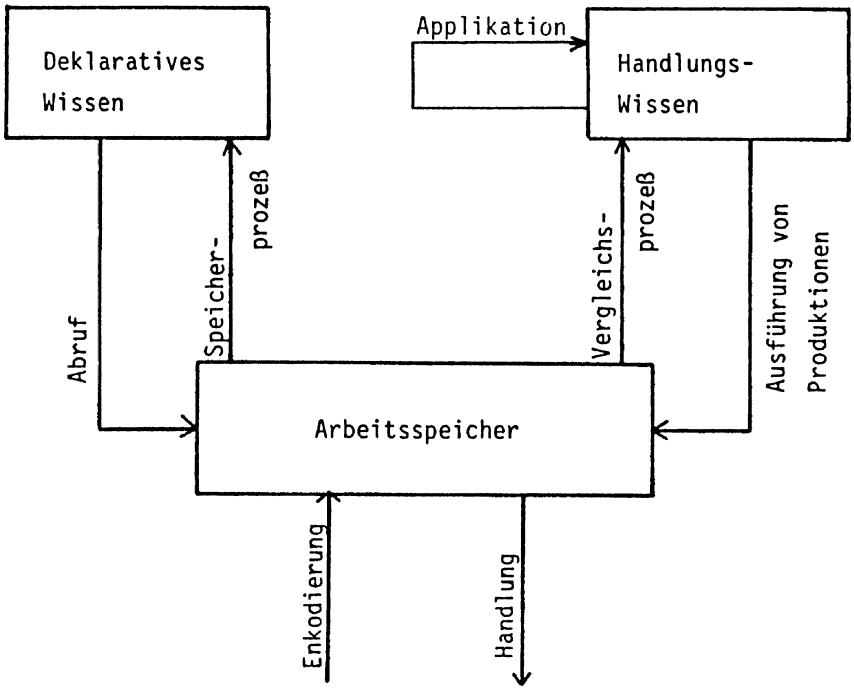

Abb. 2

Der grundsätzliche Aufbau von ACT* nach **[Anderson_83]** .

Deklaratives Wissen. In dem zweiten Langzeitspeicher von ACT* ist deklaratives Wissen in Form dreier verschiedener kognitiver Einheiten abgespeichert. Deklaratives Wissen kann repräsentiert sein durch:

- Zeichenketten
- räumliche Vorstellungen (spatial images)
- abstrakte Propositionen.

Bei einer Zeichenkette fungiert das erste und letzte Zeichen der Kette als Anker, d. h. auf ein Zeichen in der Mitte der Kette kann nicht direkt zugegriffen werden, sondern der Zugriff muß über das erste oder letzte und die daran anschließenden Zeichen erfolgen. Zeichenketten werden herangezogen, um Reihenfolgeinformationen zu kodieren.

Räumliche Vorstellungen sind dagegen Strukturen, die die Konfiguration der Elemente in einem räumlich angeordneten Feld erhalten. Räumliche Vorstellungen sind daher geeignet, Informationen über mehrdimensionale Anordungen und Strukturen zu kodieren.

Propositionen repräsentieren typischerweise die Bedeutung von Sätzen, Texten und sprachlichen Aussagen im allgemeinen.

Eine Proposition besteht aus einem Prädikat mit einem oder mehreren Argumenten. Die Bedeutung des Satzes "Anna liebt Hans" würde beispielsweise durch eine Proposition mit dem Prädikat LIEBEN und den Argumenten ANNA und HANS dargestellt. Propositionen lassen sich leicht als Listen darstellen, bei denen das erste Element das Prädikat bezeichnet und die weiteren Elemente die Argumente der Proposition, wie z.B.:

P1 (LIEBEN ANNA HANS)

Propositionen können auch als Graphen dargestellt werden. Abb. 3 zeigt, wie in ACT* propositionales Wissen dargestellt wird.

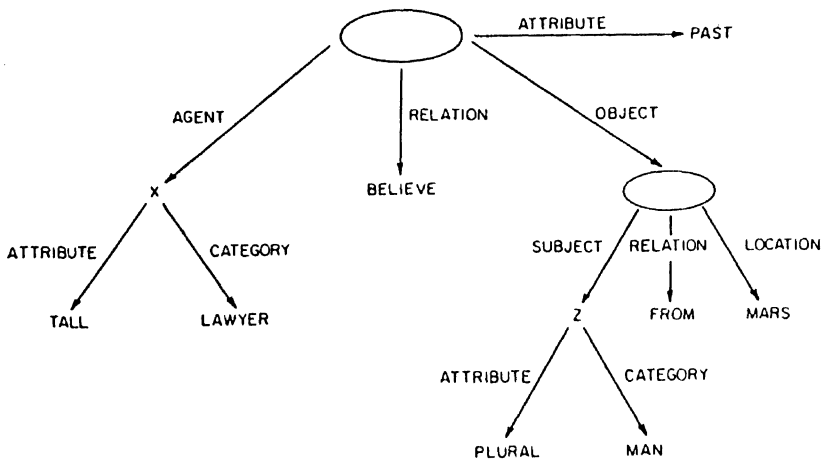

Abb. 3

Graphische Darstellung der propositionalen Enkodierung des Satzes "The tall lawyer believed the men were from Mars." aus **[Anderson_83]** S.72.

Nach der ACT-Theorie können Kombinationen von Zeichenketten, Vorstellungen und Propositionen zu hierarchisch organisierten Wissensstrukturen zusammengefaßt werden (tangled hierarchies).

Zu jedem Zeitpunkt kann in ACT* nur auf die Information zugegriffen werden, die aktiviert ist, d.h. sich im Arbeitsspeicher befindet. Im Arbeitsspeicher sind Informationen, die aus dem deklarativen Speicher aktiviert wurden und temporäre Strukturen, die durch die Enkodierung von externen Ereignissen und durch die Aktionen der ausgeführten Produktionen im Arbeitsspeicher abgelegt werden.

Verarbeitungsprozesse

Die Prozesse, die auf diesen drei Speichern operieren, sind in Abb. 2 als Pfeile dargestellt. Durch Abrufprozesse können Informationen im deklarativen Langzeitspeicher aktiviert werden und dadurch in den Arbeitsspeicher gelangen. Umgekehrt können Inhalte des Arbeitsspeichers permanent im deklarativen Speicher abgelegt werden. Durch Enkodierungsprozesse gelangen Informationen von außerhalb in das System. Die Vergleichs- und Ausführungsprozesse kennzeichnen die entsprechenden Komponenten des Produktionssysteminterpreters. Schließlich soll das System durch eine Analyse der ausgeführten Produktionen neue Produktionen lernen können. Der Produktionsspeicher soll sich also selbst modifizieren können, wie dies in Abb. 2 durch den auf den Produktionsspeicher zurückweisenden Pfeil angedeutet ist.

Weitere Annahmen

- Die deklarativen kognitiven Einheiten haben zu jedem Zeitpunkt einen gewissen **Aktivierungsgrad**, der sich laufend verändern kann.
- Die **Stärke einer Kante** zwischen zwei Knoten ergibt sich aus der Aktivierungsstärke der Knoten und der Anzahl der Kanten, die von einem Knoten ausgehen.
- Wenn ein Knoten aktiviert wurde, so breitet sich die Erregung über die Kanten fort. Die **Ausbreitungsgeschwindigkeit** wird durch eine Differentialgleichung beschrieben (vgl. [Anderson_83] , S.22).
- Jedes Element, das im Arbeitsspeicher ist, stellt eine bestimmte Zeit lang eine **Aktivierungsquelle** dar.
- Eine temporäre Struktur, die neu erzeugt wurde, hat die **Wahrscheinlichkeit p einer permanenten Speicherung**. Wenn permanente Strukturen im Arbeitsspeicher verweilen, so wird ihre Stärke um eine Einheit erhöht.
- Jede Produktion hat einen **Stärkewert**, der bei erfolgreicher Anwendung um eine Einheit erhöht wird.
- Zuerst werden **Produktionen, die in ihrem Bedingungsteil Ziele enthalten**, auf ihre Anwendbarkeit geprüft.
- Der Patternmatcher des Produktionssystems ist durch ein **Netz von Patterntests** spezifiziert. Der Aktivierungsgrad eines Patternknotens bestimmt, wie schnell das entsprechende Muster geprüft wird. Das Netz der Patterntests beinhaltet 5 Konfliktresolutionskriterien, nämlich Ausmaß der Übereinstimmung des Bedingungsteils einer Produktion mit dem Inhalt des Arbeitsspeichers, Produktionsstärke, Refraktärzeit bzgl. der Daten, Spezifizität einer Produktion und Zieldominanz. Einfach ausgedrückt bedeutet dies, daß eine sehr spezifische Produktion, deren Bestimmungsteil ein größeres Ausmaß an Übereinstimmung aufweist, die einen hohen Stärkewert hat und deren Bedingungsteil ein Ziel enthält, gegenüber Produktionen, die diese Eigenschaften nicht aufweisen, bevorzugt ausgeführt wird.
- Aus dem Trace bisher angewendeter Produktionen können durch Prozeduralisierung und Komposition **neue Produktionen** generiert werden.
- Neue Produktionen können auch durch die **Generalisierung oder Spezialisierung** (generalisation or discrimination) der Bedingungen existierender Produktionen erzeugt werden.

Um zu überprüfen, inwieweit die Ziele von ACT* erreicht werden können, müßte ACT* als Computermodell vorliegen. Anderson vermutet jedoch, daß das menschliche Produktionssystem aus Zehntausend bis zu 10 Millionen Produktionen besteht. Da ACT* daher nicht im vollen Umfang implementiert werden kann und die von Anderson vorgegebenen Spezifikationen nicht hinreichend detailliert und präzisiert sind, konnte der Beweis, daß ACT* ein suffizientes Modell ist, bisher nicht angetreten werden. Das Erreichen dieser Ziele würde auch gleichzeitig die Lösung vieler fundamentaler Forschungsprobleme der KI beinhalten.

Dennoch bildet ACT* einen nützlichen Rahmen für Kognitive Modellierungen. Spezifische Modelle, so auch die im folgenden dargestellten, können in den ACT*-Rahmen eingeordnet werden und damit besser zueinander in Beziehung gesetzt werden. Ohne einen solchen Rahmen wäre es nur schwer möglich, die Unterschiede und Ähnlichkeiten kognitiver Modelle über verschiedene Gegenstandsbereiche (Problemlösen, Lernen, Textverstehen, etc.) zu vergleichen.

Andererseits kann durch konkrete kognitive Modelle festgestellt werden, welche der global beschriebenen ACT*-Postulate unzureichend sind und welche Postulate für eine konkrete Modellierung nicht von zentraler sondern höchstens von untergeordneter Bedeutung sind.

2. Modellierung des Lernens für ein Intelligentes Tutorielles System (ITS)

Kognitive Modelle werden nicht nur aus grundwissenschaftlichen Forschungsinteressen entwickelt, sondern ermöglichen auch eine bessere Lösung praktischer Probleme. Im Bereich des computergestützten Unterrichts kann durch eine kognitive Modellierung des Lernerverhaltens ein Instruktionssystem in die Lage versetzt werden, Hilfestellungen und neues Lernmaterial entsprechend dem tatsächlichen Wissenszustand des Lernenden vorzugeben.

Durch die Einbeziehung eines kognitiven Modells werden Instruktionssysteme nicht nur in ihrer Leistung verbessert, sondern solche Systeme besitzen auch qualitative Ähnlichkeiten zu menschlichen Nachhilfelehrern oder Tutoren. Sie werden deshalb auch als Intelligente Tutorielle Systeme (ITS) bezeichnet [Sleeman_82, Spada_85].

Auf der Grundlage der ACT-Theorie wurde von Anderson und Mitarbeitern [Anderson_84] ein kognitives Modell des Lernens der Programmiersprache LISP als Vorarbeit für einen LISP-Tutor entwickelt. Mit Hilfe dieser Modellierung sollte ein Instruktionssystem in die Lage versetzt werden, eine ähnlich gute Leistung zu erbringen wie ein Nachhilfelehrer. Nachhilfelehrer können Wissen ca. viermal so schnell übermitteln wie dies für Standardunterrichtssituationen der Fall ist [Anderson_85].

2.1 Simulation des Erlernens der Programmiersprache LISP

Bei der Simulation auf der Grundlage von ACT* wird angenommen, daß menschliches Handlungswissen durch Produktionen dargestellt wird, wobei jede Produktion einen bestimmten Stärkewert mitführt. Die Simulation ist in der Produktionssystemsprache Grapes (Goal-Restricted Production System) ([GRAPES_84]) implementiert.

Psychologische Grundannahmen der Modellierung

- Der Erwerb von Programmierwissen erfolgt hauptsächlich beim Lösen von Programmieraufgaben.
- Das menschliche Problemlösen ist durch Ziel- und Teilzielbildung hierarchisch organisiert.
- Problemlösen erfolgt mit Hilfe von Analogien, durch das Einpassen einer Aufgabenstellung in Schablonen und durch das Heranziehen von strukturell ähnlichen Beispielen.
- Kapazitätsgrenzen des menschlichen Arbeitsspeichers haben einen entscheidenden Einfluß beim Lösen von Programmieraufgaben.
- Das beim Problemlösen verwendete allgemeine Wissen und die durch die Problemlösung gewonnenen Erkenntnisse können durch Komposition und Prozeduralisierung in spezifisches Programmierwissen überführt werden.

Die Grapes-Simulation geht davon aus, daß das Wissen des Studenten durch allgemein einsetzbare Produktionen und durch bereichsspezifische Produktionen dargestellt werden kann. Die Regeln R2 und R3 sind Beispiele für allgemein einsetzbare Produktionen: So gibt Produktion R2 an, wie zur Lösung eines Problems, bei dem eine gewisse Struktur erzeugt werden muß, eine Schablone eingesetzt wird.

```
R2 Wenn (Ziel: Erzeuge eine bestimmte Struktur

          und: Es ist eine  Schablone für die zu erzeugende
               Struktur vorhanden)

   Dann (Erzeuge das Ziel, die Schablone
         auf den vorliegenden Fall anzuwenden)

R3 Wenn (Ziel: Lösung eines Problems

          und: Für die Lösung eines
               ähnlichen Problems ist ein Beispiel bekannt)

   Dann (Erzeuge das Ziel, das Beispiel
         dem vorliegenden Problem anzupassen)
```

Die Regel R4 sowie die oben vorgestellte Regel R1 kennzeichnen dagegen spezifisches LISP- Wissen.

```
R4 Wenn  (Ziel: Extrahieren des ersten Elements von LIST1)

   Dann  (Schreibe "(CAR LIST1)" und lösche o.g. Ziel)
```

Mit einer Ansammlung derartiger Regeln modelliert Grapes, wie Anfänger Programmieraufgaben lösen.

Eine einfache Aufgabe

Die Grapes-Simulation soll am Beispiel des Lösens einfacher Aufgaben vorgestellt werden. In dem Beispiel soll die Funktion FIRST definiert werden, welche wie die LISP-Funktion *car* das erste Element einer Liste extrahiert. Danach soll die Funktion SECOND definiert werden.

Vorkenntnisse. Bei der folgenden Betrachtung wird davon ausgegangen, daß der Student eine kognitive Schablone für Funktionsdefinitionen in LISP und die Definition der Funktion F-TO-C als ein Beispiel für Funktionsdefinitionen kennt, nämlich:

```
Schablone: (DEFUN  <Funktionsname>

                   (<Parameter 1> <Parameter 2> . . . <Parameter n>)

                   (<Prozeßbeschreibung>))

Beispiel:  (DEFUN F-TO-C (TEMP)

           (QUOTIENT (DIFFERENCE TEMP 32) 1.8))
```

Als spezifisches LISP-Wissen werden nur die Regeln R1 und R4, die etwa beim Studium eines LISP-Buches erworben wurden, als bekannt vorausgesetzt. Auf diese Kenntnisse greift die Simulation bei der Erstellung der Funktion FIRST zurück

Das Lösen der Aufgabe. Bei der Vorgabe der Programmieraufgabe FIRST wird zuerst das Ziel gesetzt, eine Struktur für die Funktionsdefinition zu erzeugen. Dieses Ziel führt dazu, daß als nächstes die Produktion R2 "feuert". Dadurch wird die o.a. Schablone aktiviert und es werden die Teile der Lösung erzeugt, die in der Schablone bereits enthalten sind. Dabei wird < Funktionsname > durch den Namen FIRST substituiert und dadurch "(DEFUN FIRST" geschrieben.

Die Spezifikation der Parameterliste ergibt sich jedoch nicht aus der Schablone. Hierzu liegt in der Modellierung noch kein spezifisches Wissen vor, so daß als nächstes eine allgemein einsetzbare Produktion

feuert. Es ist dies die Produktion R3, die ein konkretes zuvor betrachtetes Beispiel ins Spiel bringt. Aus der Beispielsfunktion F-TO-C wird die korrekte Oberflächendarstellung der Parameterliste aufgefunden. Bei der jetzigen Funktionsdefinition spielt LIST1 die gleiche Rolle (d.h. die eines Parameters der Funktion, wie TEMP bei der Definition von F-TO-C). Somit wird "(LIST1)" niedergeschrieben. In ähnlicher Weise erfolgt durch die Analyse des zuvor betrachteten Beispiels die Spezifikation "(car (list1))". Als erste vollständige, aber inkorrekte Lösung ergibt sich somit "(DEFUN FIRST (LIST1) (CAR (LIST1)))".

Beim Überprüfen dieser ersten Lösung durch Eingabe in das LISP- System ergibt sich die Fehlermeldung "undefined function LIST1". Eine Produktionsregel, die spezifiziert, daß durch Einfügen von Quote diese Fehlermeldung vermieden werden kann, führt zu einer weiteren Lösung, die ebenfalls ein falsches Resultat liefert. Durch das unmittelbare Ausprobieren einfacher Beispiele wie "(car '(LIST)" und "(car LIST)" wird schließlich die richtige Lösung gefunden. Die Zielstruktur für die Simulation der Lösung der vorgegebenen Programmieraufgabe ist in Abb. 4 dargestellt.

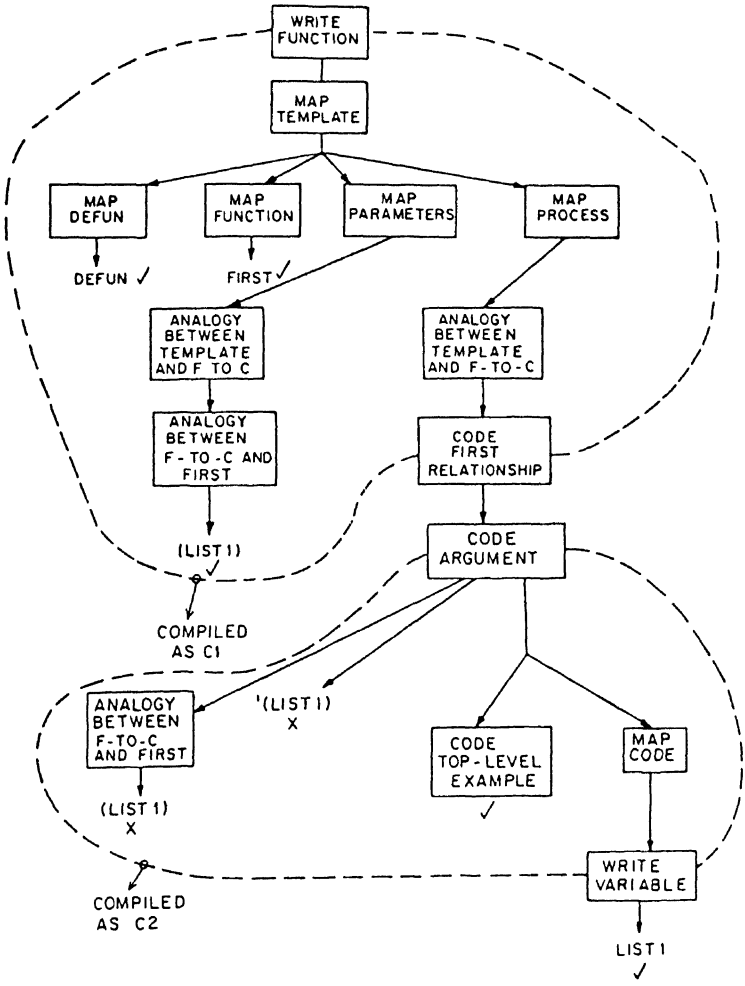

Abb. 4

Struktur von Zielen und Teilzielen bei der Kodierung der Funktion FIRST (aus [Anderson_84] , S.94) . Die Pfeile zeigen an, wie Ziele durch Produktionen in Unterziele zerlegt wurden. Die zwei mit gebrochener Linie umgrenzten Teile führen zu den kompilierten Produktionen R-C1 bzw. R-C2.

Wissenskompilierung

Nach dem Lösen dieser Aufgabe wird das durch Problemlöseprozesse erarbeitete Wissen in das Produktionssystem aufgenommen. Die in Abb. 4 mit gestrichelten Linien umgrenzten Teile werden als C1 und C2 zusammengefaßt und zu den Produktionen R-C1 und R-C2 kompiliert.

```
R-C1: Wenn (Ziel: Schreibe Funktion von einer Variablen)

      Dann (Schreibe "(DEFUN <Funktionsname>(variable)" und setze als
           Unterziel, die Relation zu kodieren, die von der
           Funktion  berechnet werden soll; schreibe dann ")" )

R-C2: Wenn (Ziel: Kodierung eines Arguments

           und:  Das Argument korrespondiert
                 mit der formalen Variable)

      Dann (Schreibe: Variablenname)
```

Bei einer weiteren Programmieraufgabe muß dieses Wissen nun nicht mehr durch Problemlöseprozesse erarbeitet werden, sondern ist durch die speziellen Produktionen direkt einsetzbar. Wird als nächstes die Funktion SECOND definiert, so wird daher der bei der Kodierung des Arguments zuvor aufgetretene Fehler vermieden, da einmal korrigierte Fehler bei der Wissenskompilierung nicht mit übernommen werden.

Die gleichen Grundsätze, die hier bei einer äußerst einfachen Programmieraufgabe dargestellt wurden, lassen sich auch bei etwas schwierigeren Aufgaben wie rekursiven Funktionsdefinitionen sowohl in der Grapes-Simulation als auch in den Verbalisierungen von Studenten wiederfinden. Den einfachen Funktionsschablonen entsprechend, gibt es vermutlich auch allgemeine Rekursionsschablonen [Vorberg_86].

Beim Programmieren einer LISP-Funktion lassen sich folgende Phasen unterscheiden:

- Eine Suchphase, in der Vorwissen aus der Mathematik (beispielsweise bei der Definition von Mengenoperationen wie dem Bilden der Potenzmenge) und Wissen aus dem vorherigen Lernmaterial herangezogen wird. Diese Suchphase wird durch eine kritische Einsicht beendet, welche ein Verfahren für die Lösung des Problems beinhaltet.
- Die eigentliche Kodierung der Funktion. Dabei können Informationen aus dem Arbeitsspeicher verloren gehen, so daß Ziele vergessen oder LISP-Wissen nicht abgerufen werden kann. Dann müssen Aufgabeninstruktionen oder LISP-Informationen nachgelesen werden.
- Eine Überprüfung der kodierten Funktion und iterative Wiederholungen der drei Schritte.

Durch die Grapes-Simulation wurde angegeben, welche Problemlösestrategien beim Lösen von Programmieraufgaben eingesetzt werden. Dadurch werden auch die Stärken und Schwächen dieser Prozesse erkennbar, so daß diese von Intelligenten Tutoriellen Systemen nun gezielt ins Spiel gebracht bzw. kompensiert werden können.

2.2 Überprüfung des Modells durch Anwendung in einem ITS

Die Grapes-Simulation wurde anhand von Protokollen des Lauten Denkens entwickelt und empirisch überprüft. Durch den Einsatz der Grapes-Simulation in einem ITS kann jedoch noch ein sehr viel strengerer Test der empirischen Angemessenheit und des praktischen Nutzens des Modells durchgeführt werden.Von [Reiser_85] wurde ein LISP-Tutor entwickelt, in dem das Lernverhalten des Studenten durch das vorgestellte Kognitive Modell beschrieben wurde. Bei der Entwicklung des LISP-Tutors wurden darüber hinaus folgende psychologische Grundsätze und Lehrstrategien berücksichtigt :

- Zu jedem Zeitpunkt des Lernens sollte die zugrundeliegende Zielstruktur der Lösung einer Aufgabe dem Studenten explizit übermittelt werden. Deshalb werden zu jedem Zeitpunkt der Aufgabenlösung am Bildschirm des Tutors Ziele und Teilziele aufgeführt. Dabei werden die Ziele und Teilziele des Lernenden auf der Grundlage der Grapes-Simulation diagnostiziert.

- Instruktionen sollen im Problemlösekontext vorgegeben werden. Aufgrund der kognitiven Modellierung kann der Tutor dem Studenten die kritischen Informationen vermitteln, die ihm gerade zur Lösung einer Aufgabe fehlen.
- Es soll ein abstraktes Verständnis des Problemlösewissens übermittelt werden. Durch die Explikation der Zielstruktur zur Lösung einer Aufgabe werden Problemlösestrategien übermittelt.
- Der Tutor soll die Beanspruchung des Arbeitsspeichers reduzieren. Dazu werden Funktionsdefinitionsschablonen, die mit einem Strukturneditor vom Lernenden ergänzt und verändert werden können, am Bildschirm vorgegeben und müssen so nicht aus dem Gedächtnis abgerufen werden.
- Dem Lernenden soll unmittelbare Rückmeldung gegeben werden. Aufgrund des kognitiven Modells kann der Tutor Hilfestellungen in Bezug auf das Wissen geben, das einen Fehler erzeugt hat.
- Aufgrund des kognitiven Modells kann die Größe der Informationsstückchen dem Bedürfnis des Lernenden angepaßt werden. Die angestrebten Programmierfertigkeiten können so schrittweise aufgebaut werden.

In dem LISP-Tutor wird der Lernfortschritt des Studenten im kognitiven Modell nachverfolgt. Jede Handlung des Studenten wird in das Modell übertragen. Dadurch wird eine dynamische Modellierung der Lernfortschritte erreicht.

Der Umfang des Lehrstoffes, der vom LISP-Tutor übermittelt wird, reicht von elementaren LISP-Funktionen bis zu rekursiven Funktionsdefinitionen. Der Einsatz des LISP-Tutors im Programmierunterricht zeigt, daß Studenten mit dem Computertutor fast zweimal so schnell lernen wie bei den sonst üblichen Instruktionsverfahren [Anderson_85]. Studenten, die von menschlichen Tutoren unterrichtet werden, lernen jedoch noch etwas schneller.

2.3 Diskussion

Durch die Grapes-Simulation wird der Wissenszustand und der Wissenszugewinn eines Studenten durch eine kognitive Modellierung nachgebildet. Die Modellierung basiert auf einigen ACT*-Postulaten, insbesondere auf den Annahmen zur Repräsentation von prozeduralem Wissen durch Produktionen und die darauf bezogenen Lernmechanismen. Der erfolgreiche Einsatz des LISP-Tutors im universitären Unterricht zeigt, wie Computerinstruktionsprogramme durch eine explizite kognitive Modellierung auf eine qualitativ höhere Stufe gestellt werden können. In Andersons LISP-Tutor, der kommerziell vertrieben wird, erfolgen jedoch noch nicht sämtliche Rückmeldungen an den Lernenden auf der Basis des kognitiven Modells. Die Möglichkeiten der kognitiven Modellierung wurden also hier noch nicht voll ausgenutzt. Möglicherweise können solche Tutoren auch durch eine ergänzende Modellierung deklarativen Wissens (z.B. Berücksichtigung des episodischen Gedächtnisses) in ihrer Effektivität noch weiter verbessert werden [Weber_86] .

In der Grapes-Simulation wird ein Individuum durch die von ihm eingesetzten Produktionen und deren Stärkewerte charakterisiert. Aufgrund des so diagnostizierten Wissenszustands eines Studenten kann dann entschieden werden, durch welche Programmieraufgaben das noch fehlende Wissen aufgebaut werden kann. Die Grapes-Simulation leistet somit eine Umsetzung des beobachteten Lernverhaltens in die kognitiven Strukturen und Zielhierarchien, die dieses Verhalten erzeugt haben. Dadurch kann ein Student hinsichtlich eines angestrebten Lernerfolgs sehr viel gezielter instruiert werden. Unklar bleibt jedoch, ob diese Simulation darüberhinaus auch selbsttätig lernt, also den Lernerfolg, den sie beschreibt, auch gleichzeitig selbst erzielt. In diesem Fall müßte die Simulation mit den gleichen Kenntnissen, die einem Studenten zur Verfügung stehen, auch denselben Lernerfolg erzielen. In anderen Worten, es bleibt ungewiß, ob mit Grapes ein allgemeingültiger Lernmechanismus programmiert wurde, oder ob die Grapes-Simulation nur für das Erlernen einiger LISP-Kenntnisse eine explizite Beschreibung von spezifischen Lernvorgängen darstellt.

3. Modellierung von Handlungswissen in der Mensch-Computer-Interaktion

Das Design von ergonomisch günstigen Mensch-Computer-Schnittstellen erfordert Kenntnisse über das Wissen, das Personen bei interaktiver Arbeit am Computer verwenden. Das folgende Modell strebt eine formale Beschreibung dieses Handhabungswissens an. Wie schon in GRAPES ist es auch hier plausibel, ein Produktionssystem als Modellierungsmittel zu verwenden. Basierend auf den einzelnen Elementaraktionen der Produktionen wird ferner untersucht, ob Transfer-Lernen stattfindet, d.h. ob Elementaraktionen, die in einem Kontext gelernt wurden, in jedem folgenden Kontext bekannt sind und nicht unter Aufwand zusätzlicher Lernzeit erneut gelernt werden müssen. Wenn solches Transferlernen tatsächlich stattfindet, sollten Mensch-Computer-Schnittstellen so gestaltet sein, daß die auszuführenden Aufgaben möglichst viele Elementaraktionen gemeinsam haben.

3.1 Grundlegende Modellannahmen und Konkretisierung bei einer Aufgabenklasse

Zu den Grundannahmen einer Klasse von Modellen ([Newell_57,Moran_81,Card_83,Kieras_85,Polson_85, Polson_86]) gehört, daß sich das Lösen von Aufgaben beschreiben läßt mittels **Produktionen**, deren Bedingungen und Aktionen darstellbar sind durch eine endliche Menge festgelegter elementarer Symbole. **Bedingung** wird formalisiert als logische Verknüpfung (und,nicht) von Elementar-Bedingungen über

- das aktuelle Ziel
- Inhalte des Arbeitsspeichers
- externe Reize

Eine **Aktion** verändert den Arbeitsspeicher oder die Umwelt durch Elementaraktionen der Arten

- Ändern des Ziels
- Ändern sonstiger Inhalte des Arbeitsspeichers
- Informationsaufnahme
- Durchführen von Handlungen (z.B. Dateneingabe)

Abb. 5 zeigt ein Beispiel zweier Produktionen, die sich auf eine Aufgabe in einem Textverarbeitungssystem beziehen (vgl. [Kieras_85,Polson_85]).

Elementare Symbole sind hier FUNCTION, ENTITY, ***word*** usw.. Sie werden in diesem Modell als Ganzheiten gesehen. In anderen Modellen könnten sich Produktionen auf andere elementare Symbole beziehen, beispielsweise auf motorische Einzelschritte zum Bedienen einer Taste. Statt DO-KEYSTROKE ... würde dann ADD-GOAL do-keystroke in *Task 1.2* auftreten, und es müßten in einer weiteren Aufgabe (*task*) Koordinaten der Hand und der Tastatur als elementare Symbole miteinander verknüpft sein. Zusätzlich zur Verwendung von Produktionen muß also entsprechend den Zielen der Modellierung eine Beschreibungsebene festgesetzt werden. Mit den Symbolen dieser Ebene wird das Handlungswissen für den gewählten Aufgabenbereich durch Produktionen für alle Aufgabentypen beschrieben (Aufgabenanalyse).

Das Handlungswissen zum Lösen einer solchen Aufgabe besteht gemäß den Grundannahmen dieses Modells aus der Kenntnis der Bedingungen und Aktionen aller Teilziele, d.h. der Produktionen. Wenn eine Person das in dieser Weise repräsentierte Wissen beherrscht, kann sie die Aufgabe lösen.

Bei den Aufgaben verwenden Personen oft unterschiedliche Lösungswege (beispielsweise Cursor-Bewegung mittels Cursor-Tasten oder mittels Suchbegriffen). Im Modell wird dieser Tatsache dadurch Rechnung getragen, daß mehrere Sequenzen von Produktionen nebeneinander Bestand haben.

Abb. 5

(nach [Kieras_85,Polson_85]). Zum Löschen der doppelten Wörter "space" und "comma" werden die Aufgabentypen: *start and finish "Delete one word"*, *Execute "Delete one word"* und *Move cursor* durch das Eintragen der entsprechenden Ziele in den Arbeitsspeicher aktiviert.

In der vorgestellten Anwendung geht es darum, Parameter des Produktionssystem-Modells daraufhin zu prüfen, ob sie geeignet sind, Zahlwerte über das Lern- und Arbeitsverhalten vorherzusagen, die sich bei der Beobachtung menschlicher Versuchspersonen messen lassen.

3.2 Überprüfung des Modells

Ableitungen aus dem Modell

Für den Aufgabenbereich "Editieren von Texten" haben [Kieras_85,Polson_85] Vorhersagen aus dem Modell abgeleitet. Zu diesem Zweck wurden ausgewählte Aufgabentypen in Produktionen beschrieben. Eine Überprüfung des Modells erfolgt durch Vergleich von Modellparametern mit experimentell bestimmten Meßgrößen. Neben den Produktionen werden *recognize-act-Zyklen* und Arbeitsspeicher-Belastung zur Schätzung herangezogen. Ein **recognize-act-Zyklus** ist das Prüfen der Bedingungen einer Produktionsregel und Durchführen der Aktion. Der **Arbeitsspeicher** wird benutzt als Gedächtnis für Zwischenziele und sonstige Merkinhalte Diese Parameter werden herangezogen, um die folgenden experimentellen Meßgrößen vorherzusagen:

Als maßgeblich für die **Lernzeit** einer Aufgabe wird die Anzahl der Produktionen angesehen, die sich das Individuum **neu** einprägen muß. Aus der Reihenfolge des Lernens der Aufgabentypen läßt sich ablesen, welche Produktionen jeweils neu sind. Variiert man diese Reihenfolge, so variiert auch die Anzahl neuer Produktionen. Diese Anzahl dient als Vorhersageparameter des Modells.

Bei Personen, die einen Aufgabenbereich beherrschen, wird die **Bearbeitungszeit** für eine Aufgabe als abhängig angesehen von der Anzahl der recognize-act-Zyklen und der jeweiligen Anzahl von Einträgen im Arbeitsspeicher. [1] Beim Durchführen der Simulation ergibt sich die Anzahl der recognize-act-Zyklen aus der Anzahl der ausgeführten Produktionen. Die Belegung des Arbeitsspeichers resultiert aus der Sequenz der ADD bzw. DELETE Operationen.

Sind in einem System verschiedene Methoden für einen Aufgabentyp implementiert (z.B. Cursor positionieren mittels Cursor-Tasten oder durch Suchbegriff), so unterscheiden sich diese Implementierungen in der Regel in ihren Produktionen und in ihrer Belastung des Arbeitsspeichers. Das hat zur Folge, daß für unterschiedliche Implementierungen unterschiedliche Bearbeitungszeiten vorhergesagt werden. Dies kann experimentell durch Implementierungen ausgenutzt werden , die angepaßt sind an Bedürfnisse und Fertigkeiten unterschiedlicher Benutzergruppen, z.B. neu angelernte Benutzer und solche mit längerer Systemerfahrung. Formal hat das hier vorgestellte Modell daher zwei unterschiedliche Mengen von Produktionen für den gleichen Aufgabenbereich. Eine soll das Wissen einer frisch angelernten Person beschreiben, die andere das einer Person mit langer Systemerfahrung. Interessant ist natürlich, wie eine Person ihr Wissen vom ersten zum zweiten Produktionssystem fortentwickelt. Zwar macht Polson dazu keine Angaben, aber es sind zwei Wege denkbar. Der erste besteht im Erwerb zusätzlicher effektiver Methoden. Ein zweiter Weg ist in Einklang mit [Anderson_83] in folgender Weise möglich:

> In einer frühen Phase der Systembenutzung werden die Produktionen eine nach der anderen interpretativ verarbeitet. Eine große Zahl von Produktionen in einer Lösungssequenz hat eine entsprechend große kognitive Belastung zur Folge. Bei der weiteren Arbeit mit dem System werden Sequenzen von Produktionen zusammengefaßt, d.h. als Einheit gespeichert; man spricht auch von ***compiled methods***. In dieser Phase spielt die Anzahl der Produktionen eine untergeordnete Rolle, entscheidend ist nur noch die Anzahl der ***compiled methods***.

Wenn die zweite Hypothese, wie der Mensch sein Handlungswissen ausbaut, richtig ist, dann muß sich die Arbeit eines relativen Anfängers durch die Parameter solcher Implementierungen beschreiben lassen, die viele isolierte Produktionen enthalten, während in der Arbeit des Erfahrenen prozeduralisierte Sequenzen von Produktionen vorkommen. Die Ökonomisierung des Erfahrenen besteht also darin, daß er auch lange Sequenzen automatisiert, was ihm die Möglichkeit gibt, Methoden mit geringer Anzahl von recognize-act-Zyklen zu benutzen.

---

1 Experimente belegen, daß Suchzeiten zur Anzahl der Einträge im Arbeitsspeicher proportional sind ([Sternberg_69]). Dies wird hier auch für das Vergleichen der Elementarbedingungen von Produktionen mit dem Arbeitsspeicherinhalt postuliert.

Ferner bietet das Modell die Möglichkeit, **Bedienungsfehler** mit einer zu großen Anzahl von Einträgen im Arbeitsspeicher in Zusammenhang zu bringen. Von dieser Möglichkeit wird in den folgenden Experimenten kein Gebrauch gemacht.

Empirische Untersuchungen

Lernexperiment. Ziel dieses Experiments ist es, zu überprüfen, ob die Anzahl neuer Produktionen innerhalb einer Aufgabe geeignet ist, die Lernzeit für die Aufgabe vorherzusagen.

Methode. Versuchspersonen ohne EDV-Kenntnisse (zwischen 15 und 44 je Versuchsbedingung) erhalten eine kurze Einführung in Tastatur und Bildschirm eines Computer-Terminals. Dann sollen sie an diesem Terminal 5 bis 7 vom Versuchsleiter zusammengestellte Aufgabentypen (Buchstabierprüfung, Zeilennumerierung ändern, Überschrift prüfen, Name und Kommentar ändern, Zeichenabstand ändern, Diskette duplizieren, drucken) eine nach der anderen in einer festen Reihenfolge lernen. Während des Lernens werden sie auf Fehler sofort hingewiesen. Der Zeitpunkt des Lernerfolgs ist durch die korrekte Lösung von 3 Aufgaben definiert. Er dient als **Meßgröße** (abhängige Variable) des Lernexperiments. Erst nach dem Lernerfolg darf eine Versuchsperson mit dem Lernen des nächsten Aufgabentyps beginnen. Dies bietet die Gewähr, daß eine Versuchsperson, die ihren n + 1-ten Aufgabentyp lernt, ihre Aufgabentypen 1 bis n beherrscht. Verschiedene Gruppen lernen die Aufgabentypen in planmäßig unterschiedlicher Reihenfolge.

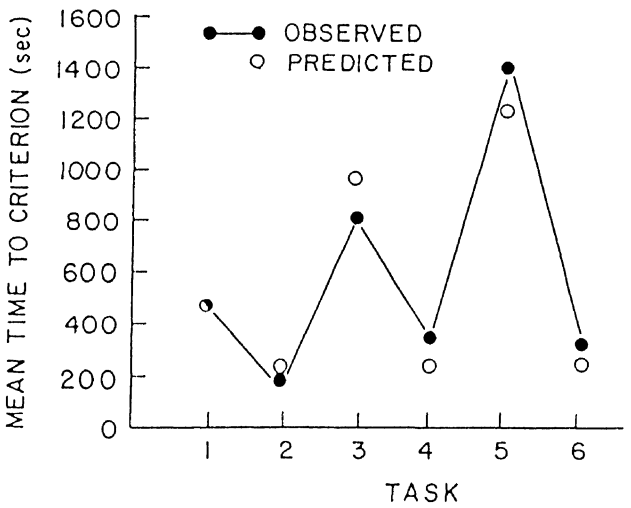

Abb. 6

(von [Polson_86]) Für sechs Aufgabentypen sind beobachtete (•) und vorhergesagte (°) Zeit im Mittel über die verschiedenen Positionen in der Lernreihenfolge dargestellt.

Ergebnisse. Für die verschiedenen Aufgabentypen lagen die gemessenen Lernzeiten zwischen 200 und 1400 Sekunden. Sie werden mit den aus dem Modell abgeleiteten Vorhersagewerten in einer gemeinsamen Graphik aufgetragen (Abb. 6). Die Graphik zeigt eine gute Übereinstimmung zwischen den Werten des Modellparameters und den gemessenen Zeiten. Insbesondere stellt sich heraus, daß es Aufgabentypen gibt, bei denen viel Wissen aus schon bekannten Aufgabentypen verwendet werden kann (etwa task 4), und solche, die unabhängig von der Menge der bereits beherrschten Aufgabentypen das Lernen vieler neuer Produktionen erfordern (etwa task 5). Letztere zeichnen sich dadurch aus, daß die gemessene wie auch die vorhergesagte Lernzeit lang ist, unabhängig davon, ob der Aufgabentyp früh oder spät präsentiert wird.

Arbeitsexperiment. Hauptziel dieses Experiments ist die Überprüfung der Hypothese, daß sich die Bearbeitungszeit für einen Aufgabentyp vorhersagen läßt aus der Anzahl der recognize-act-Zyklen und der Auslastung des Arbeitsspeichers im betreffenden Aufgabentyp. Ferner wird geprüft, worin sich die Arbeit eines fortgeschrittenen Benutzers von der eines relativen Anfängers unterscheidet.

Methode. Zunächst erhalten Versuchspersonen ohne EDV-Kenntnisse (n = 8) Gelegenheit, alle für die später präsentierten Aufgaben erforderlichen Aufgabentypen am Terminal zu erlernen. Im Gegensatz zum Lernexperiment sind gelegentlich zwei oder mehr Verfahren zur Lösung eines Aufgabentyps implementiert, zwischen denen die Versuchspersonen frei wählen dürfen. In der Lernphase wird jedoch nur die Beherrschung mindestens einer Methode zu jedem Aufgabentyp kontrolliert. Nach Abschluß der Lernphase beginnt das eigentliche Experiment mit Arbeitssitzungen an acht aufeinanderfolgenden Tagen. Die Versuchspersonen erhalten täglich Aufgaben, deren Lösung in einer genau spezifizierten Häufigkeit die Verwendung bestimmter Aufgabentypen erfordert. **Meßgrößen** des Arbeitsexperiments sind die mittleren Bearbeitungszeiten für vier Aufgabentypen am ersten und achten Arbeitstag nach Abschluß der Lernphase.

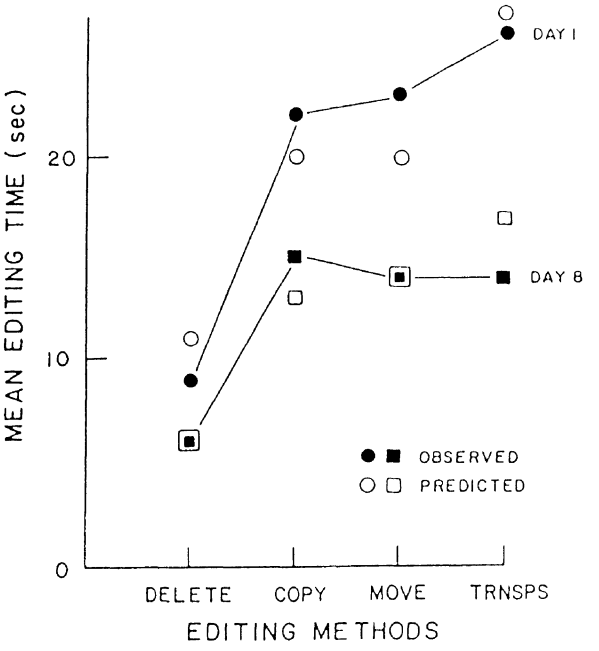

Abb. 7

(von [Polson_85]) Für vier Aufgabentypen sind beobachtete (•,▪) und vorhergesagte (°,□) Bearbeitungszeiten am ersten und achten Tag dargestellt.

Ergebnisse. Die gemessenen Bearbeitungszeiten werden in einer gemeinsamen Graphik aufgetragen mit den rechnerisch (multiple Regression) bestimmten Vorhersagewerten des Modells, und zwar getrennt für den 1. und 8. Tag. Die Vorhersagewerte für den 1. Tag entstammen dem Produktionssystem, welches dem vermuteten Wissen des relativen Anfängers nachgebildet wurde, die vom 8. Tag dem Produktionssystem des erfahrenen Benutzers. Abb. 7 zeigt für Anfänger wie Geübte gleichermaßen eine gute Übereinstimmung zwischen Vorhersage und Messung. Insbesondere spiegelt in beiden Fällen die Vorhersage die Gesamtspanne zwischen leichten (delete) und schweren (trnsps) Aufgaben gut wieder. Damit werden zwei Hypothesen des Modells gestützt.

1. Die Anzahl der recognize-act-Zyklen und die jeweilige Anzahl von Einträgen im Arbeitsspeicher haben entscheidenden Einfluß auf die Bearbeitungszeit für eine Aufgabe.
2. Bekanntes Wissen wird in neue Kontexte transferiert.

3.3 Beziehung zu ähnlichen Ansätzen

Die Beschreibung von Kontrollstrukturen menschlichen Handelns durch Produktionssysteme geht zurück auf [Newell_72]. Newell und Simon untersuchten beispielsweise Spielstrategien beim Schach. Anwendungen auf die Mensch-Computer-Interaktion finden sich in den neueren Arbeiten [Moran_81] , [Card_83] .

Allen diskutierten Untersuchungen ist gemeinsam, daß das Handlungswissen zum Lösen einer Aufgabe aus der Kenntnis der Bedingungen und Aktionen aller Produktionsregeln besteht. Wenn eine Person das in dieser Weise repräsentierte Wissen beherrscht, kann sie die Aufgabe lösen.

So verfährt auch [Moran_81] mit der CLG (Command Language Grammar). Dort werden Bedingungen, Aktionen und Ziele in mehreren Ebenen ***(levels)*** angesiedelt, die je einer beim Benutzer vorausgesetzten Semantik entsprechen. Im (obersten) *task-level* werden Aufgabenformulierungen auf abstrakte EDV-Konzepte wie *file* bezogen. Über zwei Zwischenebenen (*semantic level, syntactic level*) werden Aufgaben übersetzt in konkret am System auszuführende Sequenzen von Aktionen und Systemantworten des interaction level (prompts, <return>). Aufgabenlösung vollzieht sich hier durch Formulieren eines Ziels auf einem hohen *level* und sukzessives Übersetzen und Ausarbeiten in Interaktionsziele und -sequenzen. Methoden und Ziele sind je mit einer Beschreibung versehen, die sich auf Entitäten eines benachbarten *level* beziehen. Wichtig an CLG ist, daß inhaltliche Aufgabenkonzepte (LOOK-FOR-NEW-MAIL) mittels der Beschreibungen mit den syntaktischen und den Bedienungskonzepten (PROMPT for specification of MAIL-NO) verknüpft werden. Diese vertikale Integration entspricht der Erfahrung, daß viele reale Aufgaben auf einem abstrakteren *level* formuliert und dann für die Ausführung in konkrete Handlungen umgesetzt werden. Das GOMS (**G**oals,**O**perations,**M**ethods,**S**election) Modell in [Card_83] stellt das gleiche Thema aus einer etwas anderen Sichtweise dar. Mit im Vordergrund stehen Mechanismen zur Komposition von Handlungssequenzen (methods) aus einfachen Aktionen (operations) sowie das Vorgehen im Falle mehrerer zutreffender Bedingungsteile (selection).

Problematisch an allen Produktionssystemen ist die Voraussetzung, daß eine Person, welche eine Aufgabe löst, **genau die** Produktionen des Modells beherrscht, oder anders ausgedrückt, daß **diese** Produktionen eine eindeutige Darstellung der Aufgabenlösung sind. Diese Zuordnung von Aufgabentyp zu Produktions-Darstellung ist nicht frei von Willkür. Denn alternative Regelsätze können nur mittels Plausibilitätsüberlegungen (beispielsweise über maximale Anzahlen von Elementaraktionen in einer Produktion oder maximale Anzahl von Arbeitsspeicher-Einträgen) abgewogen, nicht aber unwiderlegbar ausgeschlossen werden. Zwar gibt es bei Rückführung von Produktionssystemen auf formale Sprachen Normalformen [Newell_72] [Hopcroft_79], aber diese sind nur in einem mathematischen Sinne ausgezeichnet, und dies hat nichts damit zu tun, ob sie auch gute kognitive Modelle sind. Jedoch deutet sich an, daß hier das experimentelle Ergebnis, d.h. die für ein psychologisches Experiment recht gute Übereinstimmung zwischen Modellvorhersage und Meßgröße, die jeweilige Wahl des Produktionssystems aus einer größeren Zahl möglicher Alternativen nachträglich rechtfertigt. Dies gilt in gleicher Weise für spätere Replikationen und Variationen der Experimente.

Positiv hervorzuheben ist, daß das Modell von [Polson_86] ausdrücklich die Vorhersage experimenteller Befunde anstrebt und hierzu mit sehr wenigen Parametern auskommt. Dies wird ermöglicht durch einen sehr strengen experimentellen Ansatz, der viele Verhaltensweisen eines spontanen Lernens und Bearbeitens strikt unterbindet und so mögliche Ursachen von Variation in den Daten ausschließt. Unberücksichtigt bleiben z.B. Planungsprozesse in Begriffen aus dem EDV-freien Arbeitsumfeld (z.B. Anlagen zu einem Brief zusammenstellen) bzw. generell die Benutzung jeglicher anderer Wissensquellen. Der streng reglementierte Lernvorgang erlaubt der Versuchsperson nicht, ihren persönlichen Lernstil anzuwenden.

Nimmt man jedoch diese Einschränkungen in Kauf, so bestätigen die Experimente, daß mit der Konzentration auf neue Produktionen für die Lernexperimente sowie auf recognize-act-Zyklen und Arbeitsspeicher-Belastung für die Arbeitsexperimente entscheidende Parameter für die kognitive Komplexität von Mensch-Computer-Schnittstellen identifiziert sind, d.h. daß vermutete andere Einflußgrößen für den hier modellierten Bereich in ihrer Bedeutung deutlich zurücktreten. Auch finden sich Hinweise, daß Anfänger und Erfahrene spezifisch unterschiedliche Repräsentationen des Wissens heranziehen, die sich in der Komplexität der Methoden unterscheiden.

Insgesamt ist hier für ein kleines geschlossenes Gebiet der Schritt hin zu einer quantitativen Theorie gelungen. Die spezielle Anwendung hat darüber hinaus eine Bedeutung für die Software-Ergonomie, d.h. das Design guter Mensch-Maschine-Schnittstellen. Denn schon vor einer Implementierung können durch eine Aufgabenanalyse mittels Produktionen Aufgabentypen identifiziert werden, die nur mit kognitiv komplexen Methoden gelöst werden können. Durch Umstrukturierung des Systems können u.U. für diese Aufgabentypen kognitiv einfachere Methoden bereitgestellt werden ([Polson_86]). Dies spart Entwicklungs- und Testzeit. Mit diesem Ansatz können z.B. verschiedene Arten von Editoren, wie etwa Graphik- und Texteditoren konsistent gestaltet werden. Dadurch würde ein möglichst großer Wissenstransfer zwischen beiden Editoren ermöglicht (siehe z.B. [Ziegler_86]).

4. Textverstehen und Textverständlichkeit

Texte sind Informationsquellen, die bei den oben beschriebenen Produktionsregel-Modellen des Erwerbs und der Anwendung von Handlungswissen zwar völlig außer acht gelassen wurden, jedoch trotzdem eine wichtige Rolle spielen. Textverständlichkeit ist dabei nicht nur bei der Gestaltung von Computer-Dokumenten, sondern bei jeglichen Beschreibungen ein allgemeines praktisches Problem. Die Verständlichkeit von Texten wurde von [Kintsch_78], [Miller_80] und [Kintsch_85] durch eine Modellierung von Verstehensprozessen untersucht.

4.1 Komponenten des Textverstehens

Selbst wenn Texte nicht in akustischer Form, d.h. als gesprochene Sprache, sondern bereits geschrieben vorliegen, sind zum Verstehen eines Textes mehrere Verarbeitungsprozesse erforderlich. Einige dieser Prozesse sind beispielsweise: Lexikalische Bestimmung und Desambiguierung von Wörtern, syntaktisches und semantisches Parsen von Sätzen, referentielle Identifikation von Pronomina, Aufbau der Kohärenz und Bedeutung von Textsegmenten, pragmatische Analyse - beispielsweise unter Heranziehung eines Partnermodells ([Wahlster_82]). Textverstehen ist also kein einheitlicher Prozeß, sondern vollzieht sich auf mehreren Ebenen und wird von verschiedenen Verarbeitungskomponenten oder Modulen getragen.

Wie auch psychologische Experimente belegen, sind einige dieser Komponenten bereichsspezifisch, während andere auf allgemeinen heuristischen Verfahren basieren [Schmalhofer_83]. Eine kognitive Modellierung bereichsspezifischer Komponenten muß für jedes Gegenstandsgebiet separat entwickelt und überprüft werden. So kann das Bereichswissen, das zum Verstehen von Unfallberichten herangezogen wird, ganz anders strukturiert sein als das relevante Bereichswissen über ein Computersystem. Heuristische Verstehensprozesse sind dagegen nicht unmittelbar an ein spezifisches Wissensgebiet gebunden, so daß eine kognitive Modellierung dieser Verarbeitungsprozesse möglicherweise bereichsübergreifend eingesetzt werden kann. Das hier vorgestellte Modell beschreibt einzelne bereichsübergreifende und einzelne bereichsspezifische Module.

Heuristische Prozesse der semantischen Kohärenzbildung

Das Modell von ([Kintsch_78]) beschreibt, wie beim Lesen semantische Repräsentationen eines Textes im Gedächtnis aufgebaut werden und wie aus der Text-Leser Interaktion die Textverständlichkeit bestimmt werden kann.

Ein Text wird dabei zunächst nach spezifischen Regeln in seine Propositionen zerlegt (vgl. [Turner_78]). Durch diese Propositionalisierung werden also Texte in eine Ansammlung gedanklicher Einheiten deklarativen Wissens überführt, die eingangs bei der Besprechung von ACT* bereits vorgestellt wurden. Aus dieser Ansammlung wird dann unter Berücksichtigung der Limitierung des menschlichen Arbeitsspeichers eine kohärente Bedeutungsrepräsentation des Textes (Mikro-Struktur) gebildet. Da der Text nicht als ganzes verarbeitet werden kann, sind dazu mehrere Verarbeitungszyklen erforderlich. In jedem Zyklus sind nur wenige Propositionen im Arbeitsspeicher, die in zwei Teilmengen unterteilt sind:

- Propositionen, die in diesem Schritt neu bearbeitet werden
- Propositionen, die aus dem vorangegangenen Schritt übernommen wurden.

Das Modell gibt an, wieviele und welche Propositionen zu Beginn eines Verarbeitungsschritts eingelesen werden, wie die Mikro-Stuktur aus den sich im Arbeitsspeicher befindlichen Propositionen erstellt wird und welche Propositionen für den nächsten Zyklus aktiv im Arbeitsspeicher erhalten bleiben. Eine weitere

Modellkomponente, auf die hier nicht weiter eingegangen werden soll, beschreibt, wie durch die Bildung einer Makro-Struktur die Hauptpunkte eines Textes identifiziert werden.

Textsegmentierung

Die Informationsstückchen (chunks), die in dem jeweils nächsten Zyklus neu bearbeitet werden sollen,werden durch die Reihenfolge der Wörter im Text und durch die Struktur der Propositionen festgelegt. Dazu werden die Wörter des Textes sequentiell durchgegangen und für jedes Wort wird das zugehörige Konzept in der entsprechenden Proposition aufgesucht. Für die so gefundene Proposition wird dann entschieden, ob sie noch aufgenommen oder ob mit dieser Proposition ein neuer chunk begonnen werden soll. Mit jedem Satzanfang wird ein neuer chunk eröffnet. Bei Beginn eines neuen chunks werden mindestens I Wörter gelesen, wobei der Parameter des Modells I beispielsweise 19 sein kann. Ein chunk muß mindestens zwei Propositionen enthalten. Darüberhinaus werden zur Textsegmentierung die folgenden heuristischen Regeln herangezogen, die in der angegebenen Reihenfolge abgearbeitet werden:

- Eine Proposition, die ausschließlich weitere Propositionen als Argumente enthält, wird nur aufgenommen, wenn ihre Argumente (eingebettete Propositionen) bereits im chunk sind.
- Falls die Wörter, die die Argumente der Proposition ausdrücken, im Text bereits gelesen wurden, wird die Proposition aufgenommen.
- Falls die angesprochene Proposition sich mit einer weiteren Proposition, die bereits im chunk ist, in einem Einbettungsverhältnis befindet, so wird sie aufgenommen.
- Falls die angesprochene Proposition mit der direkt vorher aufgenommenen Proposition ein gemeinsames Argument hat, wird sie aufgenommen.
- Falls keine der vorangegangenen Regeln zutrifft, wird mit der angesprochenen Proposition ein neuer chunk eröffnet.

Die Textsegmentierung - die Aufteilung der Ansammlung von Propositionen - in mehrere chunks basiert somit auf syntaktischen und semantischen Eigenschaften und einem Parameter I, der den Leser und die Schwierigkeit des Textes kennzeichnet. Die chunks enthalten nun die Propositionen, mit denen die Mikro-Struktur im jeweils nächsten Zyklus erweitert wird.

Aufbau der Mikro-Struktur im begrenzten Arbeitsspeicher

Von den im ersten Zyklus neu eingelesenen Propositionen wird eine als Wurzel des zu konstruierenden Kohärenzgraphen ausgewählt und bildet somit die Stufe 1 der **Texthierarchie**. Diese Auswahl erfolgt oft aufgrund des Themas des Textes, wie es etwa durch die Überschrift gegeben ist. Alle Propositionen, die mit der Wurzelproposition ein Argument gemeinsam haben, ein Argument der Wurzelproposition darstellen oder die Wurzelproposition als Argument enthalten (Überlappungskriterium), werden mit ihr verknüpft. Daraufhin werden die verbleibenden Propositionen auf der Basis des Überlappungskriteriums mit einer Proposition der zweiten, (dritten, usw.) Stufe verknüpft. Wie auch in den folgenden Zyklen werden die neuen Propositionen somit direkt oder indirekt mit der Wurzel verbunden.

In den weiteren Zyklen werden die neu eingelesenen Propositionen an die aus den vorangegangenen Zyklen erhaltenen (alten) Propositionen geknüpft. Unter Beachtung des Überlappungskriteriums wird dabei jede Proposition möglichst nahe an die Wurzel angeknüpft.

In jedem Zyklus wird somit durch die Bearbeitung der im Arbeitsspeicher verfügbaren Propositionen die semantische Textkohärenz schrittweise aufgebaut. In der Modellierung erfolgt das Auffinden semantischer Beziehungen zwischen Propositionen, indem geprüft wird, ob eine Proposition als Argument einer anderen Proposition auftritt oder ob zwei Propositionen ein gemeinsames Argument besitzen. In beiden Fällen wird die Beziehung zwischen den Propositionen durch eine Kante dargestellt. Da jede neue Proposition nur an eine Proposition angeknüpft wird, entsteht so eine Baumstruktur, welche die semantische Texthierarchie darstellt. Wenn eine Proposition mit keiner der im Arbeitsspeicher verfügbaren Propositionen verbunden werden kann, so wird der Langzeitspeicher nach einer Proposition abgesucht, mit der eine solche Verbindung möglich ist und diese Proposition re-instanziiert, d. h. in den Arbeitsspeicher zurückgeholt.

In einem Verarbeitungszyklus können somit folgende Fälle auftreten:

1. Sämtliche neuen Propositionen können **ohne Absuche** des Langzeitspeichers integriert werden.
2. **Re-instanziierung:** Sämtliche neuen Propositionen können nach Re-instanziierungen von zuvor bearbeiteten Propositionen integriert werden.
3. **Inferenz erforderlich:** Neue Propositionen können nicht integriert werden,
   a. weil beim Leser zusätzliches Vorwissen aktiviert werden muß, um eine Proposition zu integrieren
   b. weil der Leser nicht das nötige Vorwissen besitzt, um eine Proposition zu integrieren
   c. weil der Text an dieser Stelle für jeden Leser inkohärent ist.

Das Modell postuliert, daß durch die Fälle 2 und 3 die Verständlichkeit eines Textes erschwert wird. Die Häufigkeit des Auftretens dieser Fälle hängt insbesondere auch davon ab, welche Propositionen am Ende eines Zyklus aktiv im Arbeitsspeicher erhalten bleiben.

Gedächtnisstrategie

Ebenfalls ein heuristisches Verfahren, die leading-edge Strategie, bestimmt, welche Propositionen aus dem vorangegangenen Schritt im Arbeitsspeicher erhalten bleiben. Diese Strategie selegiert Propositionen, die aufgrund der bisher konstruierten Textrepräsentation in der Hierarchie weit oben stehen. Ein weiteres Kriterium der leading-edge Strategie besteht darin, daß später eingelesene Propositionen bevorzugt werden. Insgesamt werden s (Modellparameter) Propositionen in den nächsten Schritt übertragen. Wenn eine erhaltene Proposition eine weitere als Argument enthält, so wird auch diese Proposition im Arbeitsspeicher behalten. Aufgrund inhaltlicher Zusammenhänge kann also die Kapazität des Arbeitsspeichers ausgedehnt werden (vgl. dazu obige Erörterung über Arbeitsspeicher).

Im folgenden soll die Funktionsweise des Modells an einem Beispiel veranschaulicht werden. Abb. 8 zeigt die ersten zwei Zyklen bei der Bearbeitung eines Textes, der mit den folgenden Sätzen beginnt:

- The origins of belly dancing lie in the fertility rites practiced in Egypt long before the time of the pharaohs.
- From Egypt belly dancing spread among the harems of Turkish sultans and their nobles.

Als Ergebnis der Textsegmentierung ergeben sich die folgenden chunks von Propositionen:

- P1 (ORIGINS BELLY-DANCING P3)
- P2 (LIE P1)
- P3 (FERTILITY RITES)
- P4 (PRACTICED P3 EGYPT)
- P5 (BEFORE P4 P7)
- P6 (LONG P5)
- P7 (TIME-OF PHARAOHS)

Ende des ersten Satzes, sowie des ersten Chunks

- P8 (SPREAD BELLY-DANCING EGYPT HAREM)
- P9 (POSSESS P10 HAREM)
- P10 (CONJUNCTION SULTAN NOBLES)
- P11 (TURKISH SULTAN)
- Ende des zweiten Satzes, sowie des zweiten Chunks

Wie Abb. 8 zeigt, werden bei der Konstruktion des Kohärenzgraphen im ersten Zyklus die Propositionen P1-P7 bearbeitet. Am Ende des Zyklus bleiben P1, P3 und P4 erhalten. Im zweiten Zyklus werden die Propositionen P8-P11 angefügt und P1 sowie P8 bleiben für den nächsten Zyklus im Arbeitsspeicher erhalten.

Bereichsspezifische Wissensrepräsentationen

Der Aufbau einer bereichsspezifischen Wissensrepräsentation, eines sogenannten Mentalen Modells der Situation oder Situationsmodell [vanDijk_83], spielt insbesondere dann eine wichtige Rolle, wenn Textinformationen zum Handeln in einem Aufgabenbereich eingesetzt werden sollen. Für das Verstehen und Lösen von Textaufgaben durch Schulkinder der 3. Klasse wurde das oben beschriebene Modell von [Kintsch_85] entsprechend erweitert. Die Erweiterung beschreibt, wie aus Texten eine propositionale Repräsentation und ein Situationsmodell, das zum Lösen der Textaufgabe herangezogen wird, aufgebaut

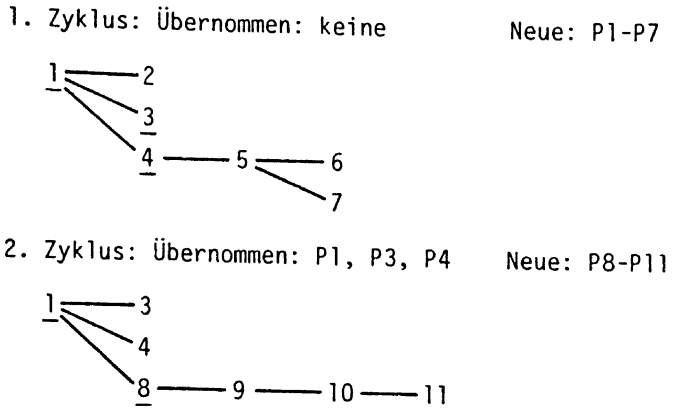

Abb. 8

Konstruktion des Kohärenzgraphen in der Simulation von **[Miller_80]**.

werden und wie diese beiden Komponenten miteinander interagieren. Dadurch kann vorhergesagt werden, welche Arten von Fehlern beim Versagen der einen oder anderen Komponente entstehen würden.

Das Modell nimmt an, daß zum Verstehen der in Betracht gezogenen Textaufgaben auf Wissen über Mengen und Mengenrelationen zurückgegriffen wird. Das Wissen über Mengen wird durch einfache Mengenschemata mit 3 slots dargestellt:

- Ein Objektslot, der angibt, um welche Art von Objekt es sich handelt.
- Ein Quantitätsslot, der die Anzahl der Objekte vermerkt.
- Ein Spezifikationsslot, der dazu dient, verschiedene Mengen z. B. aufgrund des Besitzers der Objekte und der Zeit voneinander zu unterscheiden.

Das Wissen über Mengenrelationen wird durch übergeordnete Schemata dargestellt, in deren Slots Instanzen von einfachen Mengenschemata abgelegt werden können. Das Vereinigungsmengenschema ist ein solches übergeordnetes Schema, das die Beziehung von zwei Teilmengen zur Vereinigungsmenge darstellt. Das Wissen über Mengen und Mengenrelationen wird sowohl zur Konstruktion der propositionalen Textrepräsentation als auch beim Aufbau der Problemrepräsentation verwendet. Die Instanziierung der Mengenschemata und das Auffüllen der Slots wird von Produktionsregeln durchgeführt, die aufgrund der Propositionen, die sich im Arbeitsspeicher befinden, mehrere Aktionen ausführen und so Textmikrostruktur und Situationsmodell konstruieren. Beim Aufbau des Situationsmodells werden die Propositionen auf die problemrelevanten Informationen reduziert und in den entsprechenden Slots abgelegt.

Beispielsweise werden für die Textaufgabe

Joe has three marbles.
Tom has five marbles.
How many marbles do they have alltogether?

durch die Anwendung mehrerer Produktionsregeln aus den ersten beiden Sätzen, die in Abb. 9 dargestellten Instanzen der Mengenschemata (Set1, Set2) als Teile des Situationsmodells erzeugt.

Gleichzeitig wird aus den Sätzen eine propositionale Textrepräsentation aufgebaut, wobei die Propositionen auch den Mengenschemata zugeordnet sind. Prädikate dieser Propositionen können nun auch übergeordnete Schemata aktivieren. So wird durch die Propositionen des 3. Satzes ein Vereinigungsschema instanziiert (Superset).

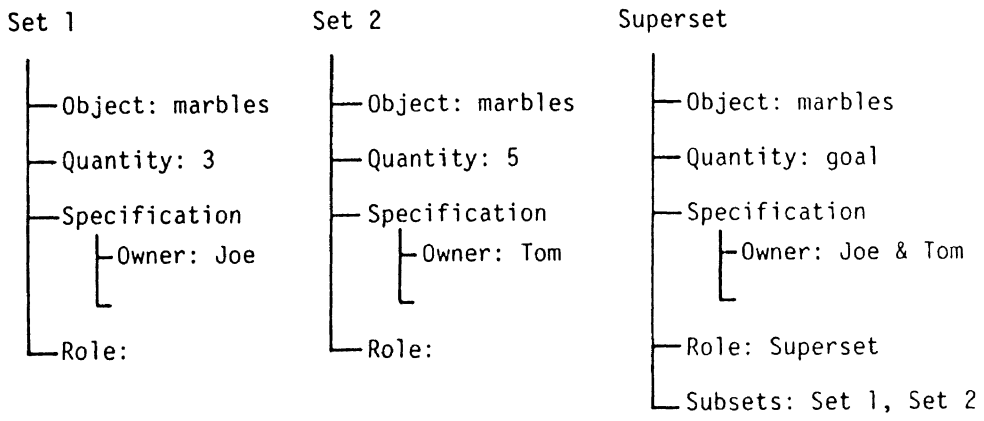

Abb. 9

Beispiele für Schemata beim Lösen einer einfachen Textaufgabe (nach **[Kintsch_85]**).

Bei der Instanziierung des Vereinigungsmengenschemas wird das Füllen der Untermengenslots als Teilziel erzeugt. Das Teilziel wird durch Zuweisung von Set1 und Set2 erreicht. Die Aufgabe wird nun durch die Berechnung der Anzahl der Objekte in der Vereinigungsmenge gelöst.

Anstelle des Vereinigungsmengenschemas müssen für andere Textprobleme andere Schemata wie z.B. das Transfer- oder Differenzschema verwendet werden. Das Modell gibt somit an, welches Begriffswissen, welche mathematischen Kenntnisse und welche Verarbeitungsstrategien zur Lösung von Textaufgaben hinreichen und darüber hinaus psychologisch plausibel sind. Obwohl die erweiterte Modellierung zunächst auf das Lösen einfacher Textaufgaben begrenzt ist, so lassen sich jedoch nach den gleichen Grundsätzen Modelle für weitere Gebiete, wie etwa für das Erwerben von Computerkenntnissen spezifizieren (z. B. [Schmalhofer_86c]).

4.2 Experimentelle Überprüfung

In mehreren Untersuchungen wurden sowohl allgemeine Modellannahmen als auch spezifische, aus der Computersimulation abgeleitete Vorhersagen durch psychologische Experimente überprüft.

Distanz der Konzepte (Argumente) einer oder zweier verschiedener Propositionen. Die Annahme, daß Propositionen beim Textverstehen kognitive Einheiten darstellen, wurde bestätigt, indem gezeigt wurde, daß die Konzepte einer Proposition im Gedächtnis nahe beisammen abgelegt sind und daß Propositionen meist ganzheitlich abgerufen werden. Wie eingangs ausgeführt wurde (Abschnitt 1.2) kann das Bahnungsparadigma herangezogen werden, um festzustellen, welche Begriffe im Gedächtnis nahe beieinander liegen. Mit diesem Paradigma hat [Ratcliff_78] untersucht, ob die Wörter, die in der Textoberfläche nahe nebeneinander stehen und zu unterschiedlichen Propositionen gehören, oder die Wörter, die zu ein und derselben Proposition gehören, im Gedächtnis zusammen abgespeichert sind.

In der Untersuchung wurden nach dem Lesen eines Textes Wörter jeweils einzeln vorgegeben. Die Versuchspersonen mußten dann durch das Drücken einer Taste entscheiden, ob das Wort im Text vorkam oder nicht. Bei dieser einfachen Aufgabe, bei der kaum Fehler gemacht wurden, waren die Wiedererkennungslatenzzeiten für ein Wort, das im Text vorkam, davon abhängig, welches Wort zuvor dargeboten wurde. Stammte das vorausgehende Wort aus der gleichen Proposition, so war die Latenzzeit statistisch signifikant (20 ms) kürzer, als wenn das vorausgehende Wort nur in der Textoberfläche nahe an dem getesteten Wort auftrat und nicht der gleichen Proposition angehörte. Solche Situationen lassen sich

für experimentelle Untersuchungen leicht konstruieren. Die Ergebnisse der Experimente von [Ratcliff_78] zeigen also, daß die Teile einer Proposition im Gedächtnis nahe zusammen abgespeichert sind.

Ganzheitlicher Gedächtnisabruf. [Goetz_81] konnten zeigen, daß beim Reproduzieren von Texten Teile einer Proposition (Prädikat, Argumente) kaum alleine abgerufen werden. So fanden sie in ihren Untersuchungen, daß die bedingte Wahrscheinlichkeit, nach **einem Bestandteil nicht alle weiteren Teile einer Proposition zu reproduzieren, kleiner als 0,1 war. Diese und ähnliche Untersuchungen belegen, daß bei der menschlichen Textverarbeitung Propositionen kognitive Einheiten darstellen.**

Semantische Distanz. Durch Bahnungseffekte, die mit dem gleichen Verfahren beobachtet wurden, konnten McKoon und Ratcliff ([McKoon_80]) auch zeigen, daß Propositionen, die nach dem Modell nur durch wenige Kanten verknüpft sind, im Gedächtnis tatsächlich näher beisammen liegen als Propositionen, zwischen denen eine größere Anzahl von Kanten liegt.

Textreproduktion. Neben den Annahmen über die Einheiten der kognitiven Verarbeitung wurden auch Modellvorhersagen der Erinnerungsleistung überprüft. Das Modell nimmt an, daß eine Proposition, die an einem Zyklus teilnimmt, später mit Wahrscheinlichkeit p erinnert wird. Eine Proposition die an n Zyklen teilgenommen hat, wird somit mit Wahrscheinlichkeit $1-(1-p)^n$ erinnert. Da die verschiedenen Propositionen an unterschiedlich vielen Zyklen teilnehmen, ergeben sich aus dem Modell Vorhersagen, welche Propositionen besser erinnert werden als andere.

Zur Vorhersage der Reproduktionswahrscheinlichkeit einer Proposition wird nur der Parameter p aus den Daten geschätzt. Der Wert n ist direkt aus der Simulation bekannt. Da Propositionen, die nahe bei der Wurzel der Texthierarchie stehen, allgemein an mehreren Zyklen teilnehmen, sollten diese Propositionen besser erinnert werden.

Abb. 10 zeigt die Ergebnisse von [Miller_80] , die diese Überprüfung vorgenommen haben: Die vorhergesagten und empirisch beobachteten relativen Reproduktionshäufigkeiten für die Propositionen von 7 verschiedenen Hierarchieebenen des Textes sind dargestellt. Obwohl in dieser Untersuchung wegen der Kürze der Texte der Hierarchieeffekt nicht in der üblichen Größe zu beobachten war, ergab sich eine gute Übereinstimmung zwischen den vorhergesagten und beobachteten Werten.

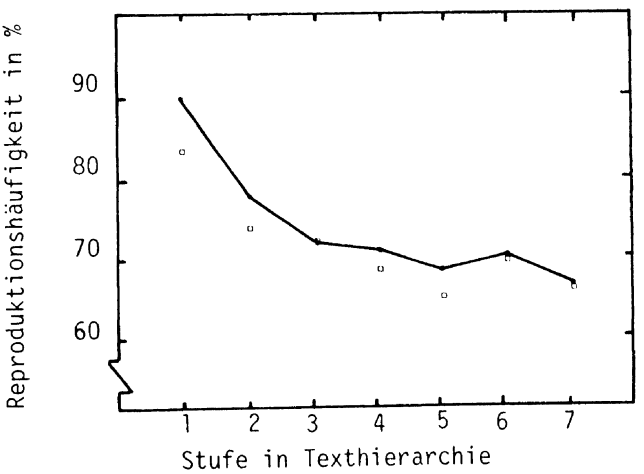

Abb. 10

Prozent der vom Modell vorhergesagten (Rechtecke) und von den Personen tatsächlich reproduzierten Propositionen (durchgezogene Linie) in Abhängigkeit von der Texthierarchiestufe (nach [Miller_80]).

Überprüfung der Inhalte des Arbeitsspeichers. Eine auf den Punkt der Gedächtnisstrategie gezielte Überprüfung des Modells wurde von [Fletcher_81] vorgenommen. Dazu wurde mit dem

Simulationsprogramm berechnet, welche Propositionen sich beim Lesen der Worte eines chunks im Arbeitsspeicher befinden sollen. Propositionen, die nach dem Modell bereits einmal bearbeitet wurden, lassen sich danach unterteilen in

(a) Propositionen, die nach dem Modell im jetzigen Zyklus noch im Arbeitsspeicher sind, und
(b) Propositionen, die aus dem Arbeitsspeicher verdrängt wurden.

Beim Lesen der zuvor bestimmten Textstelle wurde jeweils ein Wort, das sich auf die entsprechende Proposition bezieht, vorgegeben. Die Aufgabe der Versuchsperson bestand darin, das Wort, das im Text auf dieses Wort folgte, zu nennen. Die Ergebnisse zeigten, daß für Propositionen, die sich nach dem Modell noch im Arbeitsspeicher befanden (Gruppe (a)), signifikant mehr Wörter richtig reproduziert wurden (45%) als für Propositionen der Gruppe (b) (27%). Die vom Modell vorhergesagten Unterschiede werden durch die empirischen Daten somit deutlich bestätigt.

Vorhersagen der Textverständlichkeit. Die bisher vorgestellten Experimente, die sich auf die heuristischen Verarbeitungsprozesse bezogen, dienten vor allem zur unmittelbaren Modellüberprüfung.

Eine weitere Überprüfungsmöglichkeit besteht in der Anwendung des Modells zur Vorhersage von Textverständlichkeit, einmal in Bezug auf die Leichtigkeit, mit der der Text enkodiert und erinnert wird und zum anderen in Bezug auf die Schwierigkeiten, die beim Umsetzen der Textinformation in eine handlungsrelevante Wissensrepräsentation entstehen.

So haben Miller und Kintsch aus der Simulation der Mikrostrukturkonstruktion für 20 Texte Kennwerte abgeleitet und diese Werte mit den Lesezeiten und Textreproduktionen von 600 Personen verglichen. Dabei zeigte sich, daß sowohl die Anzahl der Re-instanziierungen als auch die Anzahl der erforderlichen Inferenzen die Lesezeit signifikant verlängerte. Bei Texten, die nach dem Modell weniger Inferenzen erforderten, wurden bei der Textwiedergabe mehr Propositionen reproduziert. Darüberhinaus korrelierten die beiden Modellkennwerte mit den Textverständlichkeitseinschätzungen der Personen selbst. (Die Korrelationskoeffizienten waren 0,57 für Inferenzen und 0,61 für Re-instanziierungen). Dies belegt die Bedeutung der heuristischen Verarbeitungsprozesse beim Textverstehen.

Selbstverständlich spielen für die Textverständlichkeit noch weitere Faktoren eine Rolle. Dazu gehören auch die Verstehensprozesse, die im erweiterten Modell beschrieben werden. Dazu gibt es jedoch noch keine ähnlich detaillierten experimentellen Überprüfungen. Die Modellvorhersagen stimmen jedoch mit vorliegenden Ergebnissen allgemein überein ([Kintsch_85],S.122) .

4.3 Diskussion

Miller und Kintsch ([Miller_80]) geben an, welche Informationen gemeinsam im Arbeitsspeicher verweilen müssen, damit die essentiellen Zusammenhänge, die in einem Text ausgedrückt werden, vom Leser kognitiv überhaupt rekonstruiert werden können. Durch die Erzeugung von Verknüpfungen zwischen Propositionen gibt die Simulation nur eine grobe Beschreibung des Resultats der Verarbeitung an, ohne dabei jedoch die entscheidenden Analysen selbst durchzuführen. Das Simulationsprogramm versteht also den Text nicht selbst, sondern beschreibt die Gedächtnisstrategien, die menschliches Textverstehen ermöglichen.

Ähnlich wie ein Cache Memory Manager, der selbst keine substantielle Informationsverarbeitung vornimmt, sondern "nur" festsetzt, welche Informationen verfügbar gehalten werden sollen, werden hier Textverarbeitungsstrategien spezifiziert, die dafür sorgen, daß die Propositionen, die kognitiv verarbeitet werden sollen, auch tatsächlich im Arbeitsspeicher sind. Solche Strategien müssen natürlich an die zu verarbeitenden Informationen angepaßt sein. Deshalb ist es nicht überraschend, daß in den Strategien Prinzipien der Textgestaltung verborgen sind.

In der Simulation von [Miller_80] sind einige der interessanteren Prozesse des Kintsch-Modells ausgeklammert oder werden wie die Propositionalisierung außerhalb der Simulation vollzogen. Zwar gibt es computergestützte Umsetzungsmöglichkeiten von Texten in Propositionen, jedoch müssen auch bei solchen Programmen ([Groen_85]) die eigentlich kritischen Schritte vom Menschen erledigt werden. Es handelt sich hierbei nur um ein Werkzeug und nicht um eine Modellierung des Parsens.

Für den propagierten modularen Ansatz kann die Miller-Simulation ([Miller_80]) als ein Rahmen angesehen werden, innerhalb dessen weitere substantiellere Verstehensprozesse modelliert werden können. Die Kintsch-Modellierung beschreitet diesen Weg, indem sie wiederum unter den Restriktionen eines begrenzten Gedächtnisses Verstehensprozesse soweit modelliert, daß die Simulation einfache Aufgaben in einer auch psychologisch plausiblen Weise lösen kann.

Bei dieser Simulation zeigt sich jedoch auch, daß der modulare Ansatz nur zu einem gewissen Ausmaß erfolgreich ist. So mußte bei dem erweiterten Modell eine etwas verfeinerte Notation für Propositionen verwendet werden. Ferner wurde für die Konstruktion der Textmikrostruktur explizit ein Schema benutzt. Von diesen Veränderungen im Detail jedoch einmal abgesehen, kann man feststellen, daß sich gerade auch hinsichtlich der umfangreichen experimentellen Absicherungen der modulare Ansatz der Verstehensmodellierung als erfolgreich erwiesen hat.

5. Flexible Gedächtnisorganisation

Wie auch in den Skriptmodellen [Schank_77] wurde in der zuerst dargestellten Modellierung von Kintsch hauptsächlich die Wissensenkodierung thematisiert. Es ist jedoch anzunehmen, daß der Abruf und die Enkodierung von Wissen stark miteinander interagieren [Lehnert_84]. Der im folgenden dargestellte Ansatz, der eine Erweiterung und Modifizierung der Skriptmodelle darstellt, trägt dieser Interaktion Rechnung. Gegenstand des Modells [Kolodner_83a,Kolodner_83b] ist es, die Bedeutung von Begriffen so in eine hierarchische Gedächtnisstruktur abzubilden, daß bestimmte Merkmale menschlichen Merkens und Erinnerns qualitativ nachgebildet werden:

- Die **Suchzeit** nach einem Eintrag im Gedächtnis ist weitgehend **unabhängig von** der aktuellen **Anzahl der Einträge**.

  Dies entspricht der Erfahrungstatsache, daß ein Fachmann auf einem Gebiet trotz größeren Wissensumfangs Details gleich schnell zur Verfügung hat wie ein Anfänger.

- Das **Vergessen** einzelner Einträge ist eine **Konsequenz der gewählten Datenstruktur** und hängt im Einzelfall ab
  - vom Eintrag selbst
  - von der Art und Anzahl der danach gespeicherten Einträge
  - von den Parametern der Suchanfrage.

  Dies entspricht der Erfahrungstatsache, daß manche Details in einer Situation verfügbar sind, in einer anderen hingegen nicht, während andere Details ständig verfügbar sind. Z.B. erinnert man sich, wo man einen Gegenstand hingelegt hat, wenn man nach einer Tätigkeit gefragt wird, die man damit ausgeführt hat, aber möglicherweise nicht, wenn man nach dem Ort selbst gefragt wird.

Dieser Ansatz orientiert sich also an der zwar fehlerbehafteten aber hocheffektiven menschlichen Verarbeitung bei gewissen Aufgaben und versucht durch formale Beschreibung hierzu geeigneter Datenstrukturen und Algorithmen diese Effektivität der Informatik nutzbar zu machen.

Die Konzepte werden am Beispiel von Ereignissen aus dem Leben von Diplomaten erläutert. Die allgemeinere Anwendbarkeit zeigt sich im nächsten Kapitel sowohl an anderen Beispielgegenständen als auch daran, daß die Gedächtnisprozesse mit anderen kognitiven Prozessen verknüpft werden können.

5.1 Rekonstruktives Gedächtnis

Datenstrukturen

Entscheidendes Gestaltungskriterium der Datenstrukturen ist die "benachbarte" Speicherung ähnlicher Merkinhalte unter der Nebenbedingung, daß dennoch unterschiedliche Ereignisse unterscheidbar bleiben. Daher finden sich Datenstrukturen für die individuellen Merkmale des einzelnen Ereignisses wie auch solche zur Zusammenfassung von Merkmalen gleichartiger Ereignisse. Zu diesem Zweck verwenden wir folgende Bezeichnungen (vgl. [Kolodner_83a,Kolodner_83b,Schank_80]):

MOP memory organization packet; enthält gemeinsame Merkmale von untergeordneten MOP's und EV's
EV event; einzelnes Ereignis

Zur Beschreibung gemeinsamer und unterscheidender Merkmale werden zwei Grundtypen von Strukturierungsmitteln verwendet:

Norm gemeinsames Merkmal einer Klasse von MOP's oder EV's
Index unterscheidendes Merkmal zwischen den Mitgliedern einer Klasse

Normen und Indizes sind jeweils gekennzeichnet durch **Name** und **Wert**. Normname und Normwert haben eine vergleichbare Funktion wie *slot* und *slot*-Eintrag in einem *script* oder wie Variablenname und Variablenwert in einem *record*. Indexname und Indexwert kennzeichnen ein unterscheidendes Merkmal eines MOP's oder EV's. Folglich ist ein Paar (Indexname, Indexwert) verbunden mit einem *pointer* auf das betreffende MOP oder EV.

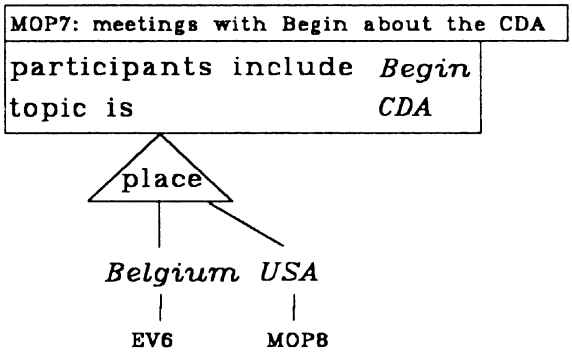

Abb. 11

Normnamen sind **participants include, topic is**. Indexname ist **place**, mit den Werten ***Belgium und USA***. CDA bezeichnet das Camp David Abkommen. (nach **[Kolodner_83a]**).

Zwei grundlegende Unterschiede zur klassischen *record*-Struktur sollen jedoch aufgezeigt werden:

1. Mehrere *records* in einem Datensatz beschreiben in der Regel Objekte einer Klasse, etwa Ereignisse, wobei jedes dadurch beschrieben wird, daß sich die Ausprägungen der Variablen unterscheiden.

   MOP's und EV's hingegen können Ereignisse, Klassen von Ereignissen, Klassen von Klassen von Ereignissen usw. in variierender Tiefe und Breite darstellen.

2. Bei *records* wird vor der ersten Benutzung eines Datensatzes eine feste Anzahl fester Variablen- und *pointer*-Namen definiert, z.B. Ort, Gegenstand, Teilnehmer, sowie ggfs. die Wertebereiche festgelegt.

Normnamen und Indexnamen dagegen werden erst zur Laufzeit aus den Beschreibungen des Gegenstandsbereichs generiert. Dies heißt im Einzelfall, daß eine Reihe von Ereignissen (z.B. Verhandlungen) beschrieben sein kann durch Ort, Gegenstand, Teilnehmer usw., so daß sich diese Merkmalsnamen empfehlen als Norm- oder Indexnamen. Andere Ereignisse (etwa Preisverleihungen) mögen durch verleihende Institution, Empfänger, Art des Festaktes usw. charakterisiert sein, so daß diese Begriffe aus dem Gegenstandsbereich als Norm- und Indexnamen neu einzuführen sind. Dies sei deswegen noch einmal besonders hervorgehoben, weil es einerseits charakteristisch ist für die Offenheit des Gegenstandsbereichs, andererseits sich hier entscheidende Grenzen der Anwendbarkeit zeigen werden.

Die bisher eher technisch orientierte Darstellung soll nicht den Eindruck vermitteln, daß das menschliche Gehirn aus Arealen vom Typ MOP und solchen vom Typ EV besteht. Vielmehr dient sie dem Ziel, eine präzise Beschreibungsgrundlage zu haben für Prozesse auf diesem Datenmodell, die dann in Beziehung gesetzt werden können zu am Menschen beobachtbaren Gedächtnisprozessen als Leistungen des Gehirns. Denn nur Gedächtnisäußerungen, nicht das Gedächtnis selbst, sind der Beobachtung zugänglich (siehe [Muthig_85] S.334).

Gedächtnisprozesse

In den zwei folgenden Abschnitten werden die zwei grundlegenden Prozeßtypen separat behandelt.

Speicherprozeß Ein neues Ereignis als EV in die bestehende Struktur einbauen; **Merken**
Suchprozeß Eine Anfrage durch Angabe eines bzw. des zutreffenden Ereignisses beantworten; **Erinnern**

Die separate Behandlung ist in einem technischen Sinne so zu verstehen, daß das Modell grundsätzlich entweder in einem Speicher- oder einem Suchmodus arbeitet. Dies darf aber nicht darüber hinwegtäuschen, daß das Einbauen eines neuen Merkinhalts in die bestehende Wissensstruktur, d.h. das Speichern, immer ein Durchsuchen der bestehenden Merkinhalte zum Zwecke der geeigneten Lokalisierung des neuen Eintrags in der Nähe ähnlicher Einträge beinhaltet. Jedoch ist diese Art von Suchanfrage leicht zu bearbeiten, da die Suchschlüssel als Beschreibungen des neuen Ereignisses explizit gegeben sind.

Speicherprozeß

Grundgedanke ist es, ein neues EV so zwischen MOP's einzubauen, daß Speicherung und Zugriff möglichst ökonomisch sind. Da ein Vorprozeß zur Analyse einer natürlichsprachlichen Darstellung der Ereignisse nicht Gegenstand des Modells ist, müssen wir voraussetzen, daß die Ereignisse (EV's) vorkodiert vorliegen in der konzeptuellen Beschreibung:

Ereignisname
mehrere Paare (Merkmalsname,Merkmalswert)
z.B.
Camp David meeting
(participants include *Carter*)
(participants include *Begin*)
(participants include *Sadat*)
(place *Camp David*)
(topic *Camp David Accord*).

Ausgehend von der Situation

- ein oder einige MOP's und EV's im Speicher (siehe Abb. 11)[2]
- ein neues EV in konzeptueller Beschreibung

---

2 EV's sind eigentlich MOP's "der Stufe 0", d. h. MOP's, die noch nicht generalisiert wurden. Deshalb ist die Unterscheidung zwischen EV's und MOP's nicht zwingend.

setzen wir uns das Ziel, das EV so einzubauen, daß es von allen schon eingetragenen EV's unterscheidbar ist. Hierzu benötigt der Speicherprozeß folgende Prozeduren:

1. Als Aufhängepunkt für das EV ein geeignetes MOP_0 bestimmen.
   Ein MOP_0. ist geeignet, wenn einige der Paare (Merkmalsname,Merkmalswert) des EV mit einigen Paaren (Normname,Normwert) von MOP_0 übereinstimmen.
2. Indizes zur Unterscheidung unterhalb MOP_0 generieren ***(indizieren)***.
   Hierzu dienen die restlichen Paare (Merkmalsname,Merkmalswert) von EV.

   Der als Indizierung bezeichnete Subprozeß hat folgenden Aufbau:
   a. Merkmalsnamen des EV bestimmen, die auch bei MOP_0 als Normnamen vorkommen, für die aber gilt:
      Merkmalswert ≠ Normwert
   b. Aus diesen Merkmalsnamen "gute" auswählen; insbesondere muß das Paar (Merkmalsname,Merkmalswert) das EV eindeutig beschreiben.

Für "gute" Merkmalsnamen gibt es neben der am reinen Datensatz exakt nachweisbaren lokalen Eindeutigkeit noch heuristische Kriterien. Deren Anwendung ist ein erster Rückgriff auf ein Hintergrundwissen, das im Suchprozeß eine noch größere Rolle spielen wird. Heuristische Kriterien können sich auf die wechselseitige Vorhersagequalität von Merkmalen beziehen. Z.B. kann man aus der Nationalität der Verhandlungspartner, nicht aber aus dem diplomatischen Rang, auf die betroffenen Nationen schließen. Man wird also bei der Festsetzung des Indexnamens der Nationalität gegenüber dem diplomatischen Rang den Vorzug geben.

Ist in der bestehenden Wissensstruktur der Knoten MOP_0 mit mehreren mit dem neuen Ereignis gemeinsamen Merkmalen gefunden, erfolgt zum Einbau eine Ergänzung bzw. Anpassung der Zeiger, die den (Indexname,Indexwert)-Paaren zugeordnet sind. Hier können drei Situationen vorliegen, die drei Verfahrensvarianten zur Folge haben:

1. Ein Paar (Merkmalsname,Merkmalswert) kommt bei MOP_0 nicht als Index vor (siehe Abb. 12).
   Dann wird das EV direkt eingetragen mit (Merkmalsname, Merkmalswert, EV) als Indizierung.

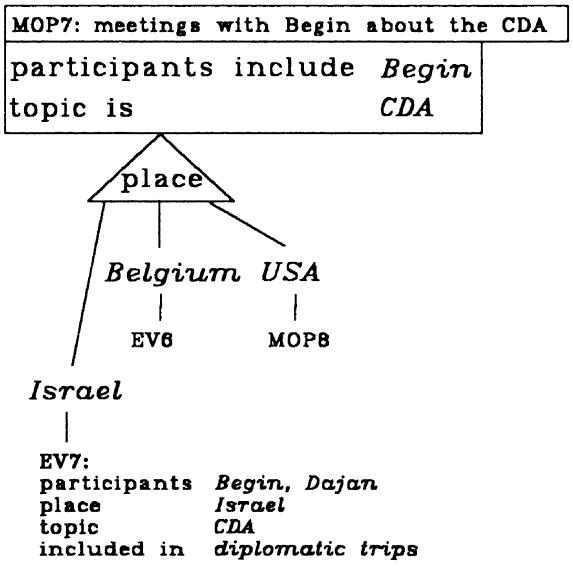

Abb. 12

(place Israel) tritt erstmals als Index auf (nach [Kolodner_83a]).

2. Es existiert unter MOP_0 bereits genau ein anderes, aber gleich indiziertes EV′ (siehe Abb. 13), d.h. EV′ wird erreicht mittels
   (Indexname, Indexwert, EV′)
   und es ist
   Indexname = Merkmalsname
   Indexwert = Merkmalswert
   - Dann wird unterhalb MOP_0 an der Stelle von EV′ ein neues MOP′ eingebaut mit der Indizierung (Merkmalsname, Merkmalswert, MOP′) sowie mit dem gemeinsamen Merkmal von EV und EV′ als neue Norm.
   - In einem weiteren Schritt müssen dann unterhalb MOP′ differenzierende Merkmale zwischen EV und EV′ zur Bildung von Indizes gefunden werden.

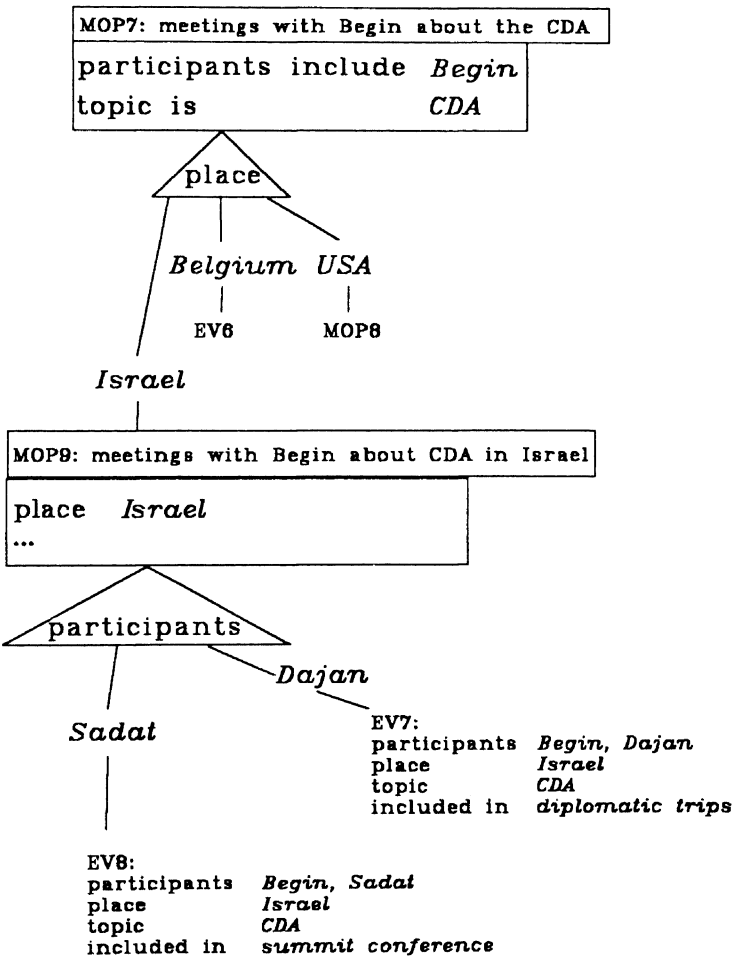

Abb. 13

(place Israel) tritt erneut als Merkmal auf und wird als Norm in das neue MOP9 übernommen (nach **[Kolodner_83a]**).

3. Unterhalb von MOP_0 existiert ein MOP′ mit den Merkmalen von EV als Indizes (ohne Abbildung). Dann muß EV unterhalb von MOP′ erneut indiziert werden.

Kontrollprozesse. Keine der bisherigen *pointer-* und Variablenmanipulationen löscht einen einmal benutzten Namen oder Wert. Strukturänderungen haben als maximale Auswirkung, daß ursprünglich als Indizes verwendete Paare in ein vorgeschaltetes MOP als Norm übernommen werden. Dies wird als **primäre Verallgemeinerung** bezeichnet und ist ein erster Kontrollprozeß auf der Datenstruktur.

Entscheidend in Hinblick auf die Eigenschaft des Modells, situations- und anfrageabhängiges Vergessen zu realisieren, ist ein anderer Kontrollprozeß. Er prüft bei jeder Norm vom Moment der Einführung an, wie viele konforme und wie viele abweichende Ereignisse nachfolgen. Überwiegen die Abweichungen, so wird die Norm aus dem MOP gelöscht. Dies hat für den Suchprozeß die Konsequenz, daß ein Suchen eines Ereignisses unter **dieser** Norm fehlschlägt. Ein Ereignis kann jedoch an mehreren Stellen in der Wissensstruktur eingehängt werden (siehe Abb. 14). Daraus wird deutlich, daß das Modell solche Einträge nicht findet, die unterhalb gelöschter Normen angeordnet sind, daß aber anders formulierte Suchanfragen zum Erfolg führen können.

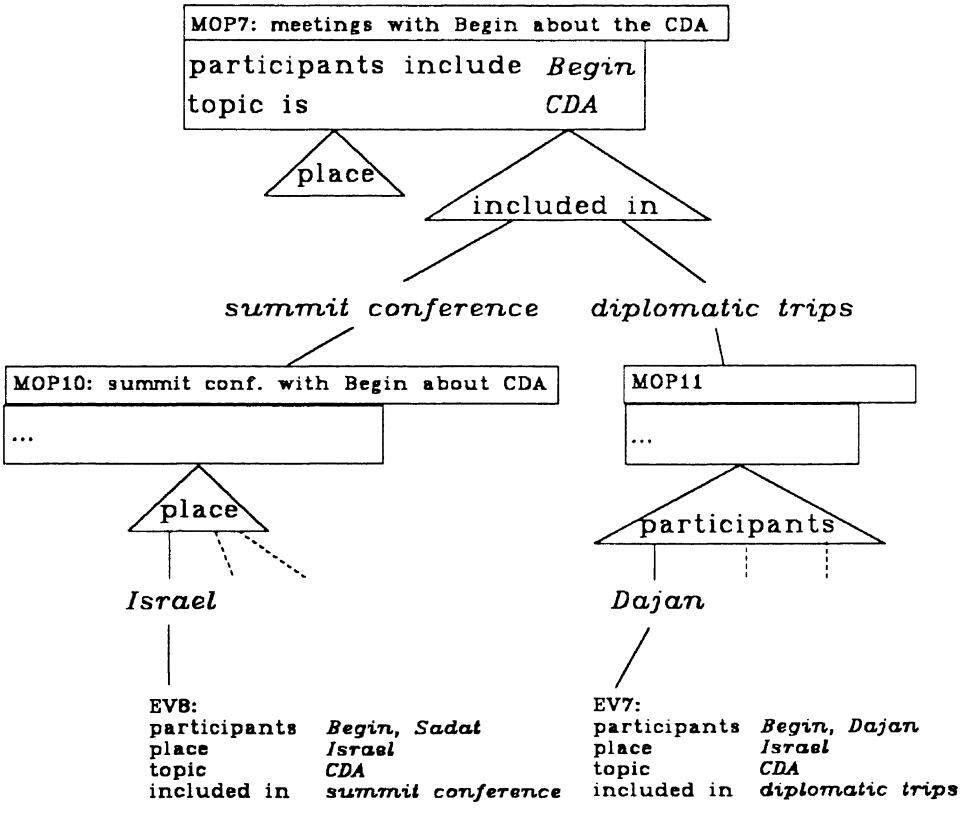

Abb. 14

Die Ereignisse EV7 und EV8 sind auch über die Norm **included in** zu erreichen (nach **[Kolodner_83a]**).

Das Ergebnis nach dem Speichern von Ereignissen ist eine baumartige Wissensstruktur. **Blätter** sind die Ereignisse EV, sonstige **Knoten** sind die MOP's und **Kanten** sind die Paare (Indexname,Indexwert). Ein Ereignis kann mehrfach (d.h. in mehreren Blättern) gespeichert werden, wenn in einem Knoten mehrere differenzierende Indizes gefunden wurden.

Suchprozeß

Erfolgreiches Suchen heißt: Durchlaufen des Baumes von einem Einstiegsknoten zu einem Blatt, d.h. die Rekonstruktion von Merkmalsnamen und Merkmalswerten, welche die Objekte des Baumes markieren. Dabei ist aufzählendes Suchen (etwa aller bisher als Normname benutzten Variablen und zu jeder Variablen die Aufzählung aller aufgetretenen Werte) aus zwei Gründen ausgeschlossen:

1. Der Mensch ist in der Regel nicht in der Lage, sein Gedächtnis durch Aufzählen aller denkbaren Ereignisse zu durchsuchen. So erinnert sich beispielsweise kaum jemand an alle seine Reisen. Durch die Erinnerung an ein bestimmtes Ereignis auf einer Reise kann jedoch eine beim Versuch der Aufzählung übersehene Reise auf einmal wieder "im Gedächtnis sein".
2. Aufzählendes Suchen steht im Widerspruch zur Forderung über die von der Größe des Wissens unabhängige Suchzeit.

Es müssen daher Techniken beschrieben werden, die gezielter als durch Aufzählung die Aufgabe lösen, Merkmalsnamen und -werte zu generieren, die dann als Argumente bei der Baumtraversierung verwendet werden. Dabei sind unterschiedliche Arten von Anfragen zuzulassen, etwa Alternativfragen (Antwort: ja oder nein), Fragen nach einem Ort oder Fragen danach, wie viele Ereignisse eines Typs bekannt sind. Letztere sind natürlich besonders kritisch, da auch hier die Beantwortung ohne Aufzählung angestrebt wird.

Mehr noch als bei der Bestimmung "guter" Merkmalsnamen zur Indizierung spielt hier Hintergrundwissen eine Rolle. Wir wollen, um dies verdeutlichen zu können, zusätzlich zu den Beispielen aus den Abb. 11 bis Abb. 14 annehmen, daß Ereignisse auch unter einem MOP mit Namen *sightseeing* (etwa von Kulturdenkmälern, Industrieanlagen usw.) gespeichert sein können. Wir stellen nun drei rekonstruktive Techniken dar, die auf die möglichen Unbekannten im Suchprozeß abzielen:

Namen von MOP's. Wenn sich bei der Anfrage: "Wo haben sie Ölfelder gesehen?" kein Einstiegs-MOP mit Namen Ölfelder findet, ist mit **Hintergrundwissen** der Art: *Ölfeld ISA Industrieanlage* und *Industrieanlage Ziel_von Besichtigung (sightseeing)* der Einstieg beim Knoten *sightseeing* möglich.

Index-Namen. Bei einer Anfrage: "Wie viele *meetings* haben Sie in Israel gehabt?" ist, wie auch sonst, eine Aufzählung unzulässig. Daher müssen Suchwege konstruiert werden, welche geeignet sein könnten, einzelne *meetings* zu Ergebnissen zu haben. Beispielsweise kann aus dem Merkmalswert *Israel* des Merkmalsnamens *Nation* mittels *Hintergrundwissen über die Vorhersagequalität von Merkmalsnamen* die Nationalität der Teilnehmer erschlossen werden. Dies erlaubt einen Sucheinstieg mit dem Ziel, ein *meeting* mit einem Israelischen Diplomaten zu identifizieren. Sofern dieses *meeting* in Israel stattfand, ist es eines der Ereignisse, die zur Beantwortung der Anfrage beitragen. Entsprechend kann das System verfahren mit dem *Hintergrundwissen*, daß *topics, die Israel betreffen, in Israel abgehandelt werden können*, also eine Vorhersagequalität des Merkmalsnamens *topic* auf den Merkmalsnamen *place* usw. .

Index-Werte. In Fortsetzung der obigen Anfrage, nun mit dem Ziel, zur bekannten Nationalität Individuen zu finden, kann mittels *Hintergrundwissen über Klassen* wie *Israeli, Ägypter, Amerikaner* usw. eine Menge von Personen, d.h. Indexwerte generiert werden, die auf die gesuchten Ereignisse verweisen (im Beispiel Dajan auf EV7, Begin auf EV8).

Allen echten Suchvorgängen gemeinsam ist der Rückgriff auf Hintergrundwissen sehr unterschiedlicher Ausprägung. Da im Verlauf des Speicherprozesses, wo ja auch Suchaspekte eine kleine Rolle spielen, verbindliche Merkmalsnamen und -werte in den konzeptuellen Beschreibungen schon vorhanden sind, sind die schwierigen rekonstruktiven Techniken dort nicht erforderlich.

5.2 Bewertung des Modells und geschichtliche Einordnung

Wie bei Kolodner geht es auch bei früheren Arbeiten von Schank und Mitarbeitern vor allem darum, qualitative Konsequenzen von Modellannahmen zu zeigen, weniger darum, eine experimentelle Überprüfung vorzunehmen.

Vorläufer

CDT. In der **C**onceptual **D**ependency **T**heory (CDT, [Schank_69]) spielten Relationen zwischen Objekten bzw. Rollen von Objekten bzgl. Relationen eine große Rolle. Insbesondere wurden Kausalitäten zeitweise intensiv studiert. Jedoch wurde dann klar, daß viele Ursache-Wirkung-Ketten, insbesondere auch im sozialen Bereich, als Gewohnheiten oder Standard-Handlungssequenzen abgespeichert sind. Dies entlastet den Menschen bei der Bewältigung der überwiegenden Mehrzahl der Alltagsaufgaben (siehe z.B. [Ackermann_86,Frese_85]).

Scripts. Dem wurde in der Weiterentwicklung der CDT dadurch Rechnung getragen, daß man ***scripts*** als neue Repräsentationsform einführte [Schank_77]. Sie vereinen kausale, assoziative, relationale usw. Konzepte in Merkeinheiten, wobei hier nicht so sehr die Art des Zusammenhangs, sondern die Tatsache des Zusammenhängens von Objekten im Vordergrund steht. Eine *script*-Darstellung ist eine erste Form hierarchischer Anordnung von Wissen. Sie hat die Konsequenz, daß zusammengehöriges Wissen (durch seine Zusammenfassung in einem *script*) immer zusammen verfügbar ist, aber nur als Ganzes, d.h. bezogen auf eine Anfrage die relevanten Details zusammen mit allen irrelevanten. Auch stellen zwei *scripts* zwei getrennte Einheiten dar, deren Ähnlichkeit zueinander in der Speicherstruktur nicht repräsentiert ist. Nicht nur in dieser Hinsicht sind *scripts* wie Schemata, die im Zusammenhang mit der Wortdesambiguierung verwendet wurden. Wie schon dort dargelegt, ist die Anwendbarkeit von *scripts* bzw. Schemata zumindest nicht universell gegeben.

In einem nächsten Entwicklungsschritt wurden "kompilierte *scripts*" durch einen Prozeß ergänzt, der zusätzlich erforderliche *scripts* oder *script*-Fragmente ad hoc generiert ([Schank_80]). Dieser Prozeß greift auf Inhalte eines wiederum hierarchisch angelegten Gedächtnisses mit fest vorgegebenen Hierarchieebenen zurück. Während beim "kompilierten" *script* alle Merkmale eines Ereignisses geschlossen, aber isoliert von ähnlichen Ereignissen gespeichert sind, kann jetzt ein *script* aus allgemeinerem Wissen aus hohen Hierarchieebenen und spezifischerem Wissen aus tieferen Hierarchieebenen aufgebaut werden, z. B. *themes* und *goals* [Schank_80]. Der Gewinn an Flexibilität ist offensichtlich. Suchpfade zielen auf tatsächlich relevante Inhalte, ohne daß unwichtige Begleitumstände, die nur durch die Koinzidenz in einem Ereignis, nicht aber systematisch mit dem Suchgegenstand zusammenhängen, mit rekonstruiert werden. Auch ist es sicher ökonomisch und daher eine sinnvolle Annahme über menschliches Verhalten, das Gemeinsame vieler Ereignisse als Normen einmal und nicht bei jedem Ereignis neu zu repräsentieren und die individuellen Merkmale erst beim Ereignis selbst zu speichern.

Heutiger Stand

Vor diesem Hintergrund lassen sich nun die spezifischen Eigenschaften der flexiblen Gedächtnisorganisation, wie sie hier vorgestellt wurde, formulieren:

1. Das Wissen ist hierarchisch aufgebaut. Die Hierachieebenen sind durch die MOP's und die EV's gegeben.
2. Gemeinsames Wissen über mehrere Objekte ist in Normen zusammengefaßt.
3. Ähnliche Objekte sind dadurch kenntlich, daß sie viele gemeinsame Normen haben und folglich in benachbarten Knoten gespeichert sind.
4. MOP's und Normen, d.h. Hierarchie-**Struktur** und -**Bedeutung** werden ständig zur Laufzeit den neuen Daten angepaßt.

Im Prinzip ist daher die Möglichkeit gegeben, Wissen ökonomisch zu speichern und die Ökonomie auch bei grundlegenden Änderungen im zu speichernden Wissen selbstregulierend aufrechtzuerhalten. Diese Selbstregulation - in Form der Kontrollprozesse - hat natürlich auch die Konsequenz des Vergessens, was den Wert des kognitiven Modells nicht in Frage stellt, da auch der Mensch vergißt.

Dennoch sei auf einige Aspekte hingewiesen, die noch Erweiterungen oder Änderungen des Modells erforderlich machen. Dies betrifft zunächst das Format der Dateneingabe: Natürliche Sprache als eine dominierende Repräsentation menschlichen Wissens kann derzeit lediglich in Verbindung mit anderen Systemen verarbeitet werden, z. B. FRUMP [DeJong_79]. Gemessen an menschlicher Gedächtnissuche ist es eine weitere ernstzunehmende Einschränkung, daß Speichern und Suchen als streng getrennte Prozesse realisiert sind. Denn im menschlichen Gedächtnis hat ein erfolgreiches Suchen (d.h. ein erfolgreicher Lauf

im Suchmodus) eine Auffrischung des Suchpfades zur Folge, d.h. ein Suchlauf ist qualitativ auch immer ein Speicher-Auffrischungslauf.

Das Hauptproblem jedoch dürfte darin liegen, daß das Modell für eine gute Performanz auf Hintergrundwissen angewiesen ist. Dies gilt beim Speicherprozeß für die Bestimmung "guter" Merkmalsnamen und, was viel wichtiger ist, beim Suchprozeß zur Generierung von plausiblen Merkmalsnamen und -werten. Ist das geeignete Hintergrundwissen vorhanden (etwa über Angehörige bestimmter Nationen), so kann in der Tat das Auffinden von MOP's und EV's im Graphen gezielt und in annähernd von der Wissensmenge unabhängiger Zeit absolviert werden (wenn es auch natürlich einer weitaus präziseren algorithmischen Beschreibung bedarf). Daher soll keinesfalls gesagt werden, daß die Verwendung von Hintergrundwissen in einem kognitiven Gedächtnismodell unangemessen ist. Es ist jedoch erforderlich, die konkrete Strukturierung und inhaltliche Beschreibung einer Wissensbasis anzugeben. Schnell durchsuchbar kann diese Hintergrund-Wissensbank nur sein, wenn sie klein ist, es sei denn, man setzt auch dort eine Organisation in MOP's und EV's voraus. Dann aber steht dahinter eine neue Hintergrund- Wissensbank usw.. Da die letztendliche Verankerung ungeklärt ist, muß man wohl annehmen - und dann kann man es auch direkt im ersten Schritt tun - daß das Hintergrundwissen in einer der etablierten Techniken realisiert und damit mit einer mit der Größe wachsenden Suchzeit behaftet ist. Das heißt, daß die Hintergrund-Wissensbank den Gegenstandsbereich begrenzt. Dies steht jedoch in Widerspruch zur Forderung nach Offenheit.

6. Rückblick und Ausblick

Im Hauptteil dieses Beitrages wurden einige Modellansätze der kognitiven Modellierung vorgestellt. Sie alle stellen in etwa den Forschungsstand dar, der Anfang der 80-ger Jahre publikationsreif war. Abschließend wollen wir

- die Modellansätze unter einigen inhaltlichen Aspekten vergleichen und mit einigen Denkanstößen und Diskussionsanregungen über Möglichkeiten und Grenzen des Faches Kognitive Modellierung (die z.T. für die gesamte KI gelten) versehen
- diesen Stand kurz in die historische Entwicklung einordnen
- einen Ausblick geben auf die interessant scheinenden Weiterentwicklungen einzelner Konzepte.

6.1 Inhaltlicher Vergleich der Ansätze

Der Vergleich der beschriebenen Ansätze erfolgt nach zwei Maßstäben, nämlich bezogen auf die ACT*-Annahmen und im Hinblick darauf, ob eine offene oder eine geschlossene Welt modelliert wird.

Zuordnung zu ACT*-Annahmen. In aller Kürze und ohne Anspruch auf Vollständigkeit soll der folgende Absatz aufzeigen, wie ACT* als Theorierahmen zur Einordnung kognitiver Modelle dienen kann. So läßt sich leicht nachvollziehen, daß das Grapes-Modell und die Arbeiten [Kieras_85,Polson_85] auf einem rein prozeduralen Langzeitspeicher und einer zielgerichteten Aktivierung der Produktionen basieren. Bei Grapes finden sich ferner Stärkewerte für Produktionen und Mechanismen zur Generierung neuer Produktionen, Komponenten also, die eine Feinanpassung des bestehenden Wissens an das Datenmaterial ermöglichen. Das Textverstehensmodell benutzt prozedurales und propositionales Wissen. Aktivierungsgrade von Knoten und Wahrscheinlichkeiten permanenter Speicherung gehen ein in Abhängigkeit vom Knotengrad einer Proposition im Kohärenzgraphen und der Häufigkeit ihrer Referenzierung. Im Modell zur flexiblen Gedächtnisorganisation dominiert ein deklarativer Speicher, dessen Einträge abstrakten Propositionen ähneln. Hauptgegenstand sind Speicherungs-, d.h. Enkodierungsprozesse. Durch die im Modell vorhandenen Kontrollmechanismen erfolgt, wenn auch indirekt, eine Steuerung von Aktivierungsgraden.

Merkmale der Gegenstandsbereiche. Die Gegenstandsbereiche der diskutierten Modelle unterscheiden sich nicht nur inhaltlich, sondern auch hinsichtlich der Frage, ob die entstehenden Repräsentationen eine geschlossene Welt darstellen oder geeignet sind, beliebige Wissensinhalte zu verarbeiten.

Typisches Beispiel eines geschlossenen Systems ist das Produktionssystem aus [Kieras_85,Polson_85]. Außer den von den Autoren formulierten Produktionsregeln kommt definitiv kein weiteres Wissen vor.

Bei Produktionssystemen läßt sich die Geschlossenheit auch formal zeigen durch Rückführung auf formale Sprachen ([Newell_72,Hopcroft_79]). Hier wird also durch Angabe des Alphabets eine geschlossene Welt definiert. Wenn jedoch, wie in Grapes, ein Daten-getriebener Einbau neuer Produktionen möglich ist, ist der Zugang zu einer offenen Welt gegeben.

Bei den Textverstehens-Versuchen ist die Situation ähnlich zwiespältig. Das reine Textkohärenz-Modell reagiert auf beliebige input-strings aus einer offenen Welt durch Versuche, mehrfach auftretende Muster zu identifizieren. Es ist zunächst in der Verarbeitungsmöglichkeit nicht auf eine geschlossene Welt beschränkt. Sollen jedoch Textaufgaben gelöst werden, muß der Eingabetext sich auf die endliche Menge von Symbolen beschränken, welche zur Aktivierung der Mengenschemata fest vorgesehen sind. Auch das Modell zum flexiblen Gedächtnis läßt zwei Sichtweisen bezüglich der Offenheit oder Geschlossenheit des Gegenstandsbereiches zu. Im Prinzip arbeiten alle Prozesse auf beliebigen Daten, d.h. auf einer offenen Welt, aber Effizienz wird erst erreicht bei Einschränkung auf diejenige Welt, zu der Hintergrundwissen vorhanden ist.

6.2 Geschichte und neueste Entwicklungen

Die 50-er Jahre

Zwei Veröffentlichungen aus dem Jahre 1957 [Newell_57,Chomsky_57] markieren Anfangspunkte der Bemühungen, Kognition formal zu beschreiben.

GPS. Der **G**eneral **P**roblem **S**olver von **Newell, Simon** und **Shaw** stellt einen Versuch dar, menschliches Problemlösen als Informationsverarbeitungsprozeß explizit zu beschreiben. Wegweisend sind die Annahmen einer symbolischen Repräsentation des Problems und die Trennung der Gedächtnisstruktur von den Prozessen zur Manipulation der symbolischen Gedächtnisinhalte [3]. Erste Anwendungen waren Spiele wie Schach [4]. Kern der Wissensbasis ist eine Liste der Zustände, die das Spiel annehmen kann, eine Menge ausgezeichneter (Ziel- oder Gewinn-) Zustände, ein Abstandsmaß zwischen Zuständen und Gewinnzuständen sowie eine Menge von Operationen. Auf dieser Wissensbasis absolviert der GPS eine Tiefensuche, indem er vom aktuellen Zustand aus einen zielnäheren Zustand wählt, je nach Vorhandensein eines Zuges diesen ausführt oder rekursiv ein Zwischenziel auswählt und erneut nach einem Zug sucht.

Wir stellen hier schon typische Merkmale späterer regelorientierter Ansätze fest - Bedingungen betreffen Abstände und Existenz von Zügen, Aktionen starten Züge oder erzeugen Zwischenziele - müssen aber bemerken, daß die eigentliche Problemlösung schon in der Angabe der Abstandsmaße liegt [5] und daß eine rekursive Suche beliebiger wenn auch endlicher Tiefe sicher wegen der Begrenztheit des Arbeitsspeichers vom Menschen nicht geleistet wird.

Transformationsgrammatiken. Formale Systeme zur Beschreibung von natürlicher Sprache wurden zur gleichen Zeit von Chomsky ([Chomsky_57]) vorgelegt. Die Transformationsregeln seiner generativen Grammatik dienen jedoch in erster Linie dazu, Ausdrücke formal zu beschreiben. Da die natürliche Sprache ein Schlüssel zu einer Vielzahl kognitiver Funktionen ist, haben experimentelle Psychologen überprüft, ob Transformationsgrammatiken auch kognitive Prozesse beschreiben. Nach einer anfänglich vermuteten Stützung dieser Hypothese (z.b. [Miller_64]), haben weitere Experimente und Analysen jedoch gezeigt, daß Transformationsgrammatiken als psychologische Modelle inadäquat sind [Fodor_74] .

Sprache wird vom Menschen nicht erworben und benutzt als formales System, sondern als Mittel der Kommunikation zweier oder mehrerer kognitiver Systeme. Als Kommunikation funktioniert Sprache auch

---

3 Es ist sicher kein Zufall, daß zur gleichen Zeit LISP entstand, mehr als ein Jahrzehnt früher als die mehr am formalen System der Prädikatenlogik orientierte Sprache PROLOG.

4 sofern man von dem Vorläufer LT [Newell_56] zum Beweis mathematischer Sätze absieht

5 was in späteren GPS-Realisierungen auch vom System übernommen wurde, [Banerij_84,Ernst_69]

unter Mißachtung der formalen Regeln, wie das Sprechen mit radebrechenden Ausländern zeigt. Daher ist ein breit angelegtes kognitives Modell für das Sprachverstehen vielleicht noch wichtiger als für die bisher aufgezeigten Anwendungen. Sicher wird hier aber die gelegentliche Fehleranfälligkeit menschlichen Sprachverstehens zu berücksichtigen sein. **Gemeinsam** ist diesen beiden frühen Ansätzen, daß

- die gewählten Wissensstrukturen (Abstandsmaße, Ersetzungsregeln) formaler und starrer sind als beim Menschen zu erwarten
- kein Zusatzwissen herangezogen wird (z.B. typische Endspiele im Schach, bzw. Semantik oder Pragmatik der Sprache). Beide Ansätze konzentrieren sich auf die Prozeßaspekte der Tiefensuche bzw. des Parsens.

Ihr **Hauptunterschied** hingegen besteht darin, daß Elemente des GPS in neueren kognitiven Modellen weiterleben, während bei der Transformationsgrammatik die kognitive Interpretation hinter der formal-linguistischen Anwendung zurücktrat.

Die 60-er Jahre

Neben Detail-Weiterentwicklungen der genannten Ansätze aus den 50-er Jahren verdient ein Ergebnis von [Minsky_67] Beachtung. Es bedeutet den Nachweis, daß die Berechenbarkeitsbegriffe aus [Turing_36, Church_36,Post_36] äquivalent sind. Damit ist die Informatik-Seite der KI in der glücklichen (oder unglücklichen) Situation, daß die Grenzen des je mit dem Computer Berechenbaren abgesteckt sind. Zusammen mit (hier nicht ausgeführten) Ergebnissen der Komplexitätstheorie (siehe z.B. [Hopcroft_79]) sind nun die Grenzen der Informatik geschlossen beschreibbar, in deutlichem Gegensatz zu einer naturwissenschaftlichen Kognitionsforschung, die (zumindest im Sinne des kritischen Rationalismus) nie zu einer geschlossenen Beschreibung ihres Forschungsgegenstandes kommen kann ([Popper_66]).

Die 70-er Jahre

Die Entwicklung ist u.a. gekennzeichnet durch zwei Strömungen:

- Gedächtnis und Wissen werden zunehmend als eigenständige Forschungsinhalte ernst genommen und drängen zeitweilig das Interesse an Problemlöseverfahren spürbar in den Hintergrund.
- An vielen Stellen wird der direkte Vergleich theoretisch abgeleiteter Ergebnisse mit bekannten experimentellen Befunden oder passend zur Theorie konzipierten Experimenten gesucht.

Zum ersten Typ zählen neben Schank u.a. [Rumelhart_72,Norman_75,Miller_76], zum zweiten besonders [Newell_72] und die vorgestellten Arbeit von Polson und Kieras. Beide Strömungen sind außer bei Kintsch vertreten durch Anderson [Anderson_73]. Es führt zuweit, jeden dieser Ansätze auch nur oberflächlich zu charakterisieren, daher sei auf die ausgezeichnete Darstellung in [Lachman_79] verwiesen (siehe auch [Cohen_83]). Die folgende Tabelle stellt die historische Perspektive noch einmal im Überblick dar.

Verlagerung der Forschungsschwerpunkte

50-er und 60-er	Jedes Modell läßt sich durch einen einfachen Prozeßtyp charakterisieren, der oft auch präzise mathematische Analysen zuläßt. Ziel ist das Lösen formal beschreibbarer Probleme.
70-er	Es werden unterschiedliche Methoden der Wissensrepräsentation exploriert. Ziel ist die Darstellung auch komplexer, schwach strukturierter Sachverhalte sowie ***retrieval*** und lokale Inferenzen auf solchen Wissensbasen.
80-er	Es zeichnet sich ein Trend zu wissensbasierten Problemlösemethoden auf schwach strukturierten Räumen ab. Wissen wird eingesetzt, sofern es vorhanden ist. Andernfalls wird auf sogenannte ***weak methods***, d.h. allgemeine Problemlöseheuristiken zurückgegriffen.

Trends der 80-er Jahre

Der Hinweis auf die Entwicklung in den 80-er Jahren läßt sich am Beispiel der Arbeiten von [Rosenbloom_85, Kolodner_86] darstellen.

Verwendung von Erfahrung beim Problemlösen. Die Weiterentwicklung der Konzepte von Kolodner ist dadurch gekennzeichnet, daß die Merkinhalte (bisher "neutrale" Ereignisse oder Situationen) durch Diagnose-, Entscheidungs- oder ähnliche Situationen ersetzt werden, Situationen also, die Beschreibungen von "Präzedenz"-Fällen und die dort getroffenen Maßnahmen oder Entscheidungen als Merkmale haben. Für Problemlösesituationen kann man nun nicht davon ausgehen, daß für jeden Fall schon ein Präzedenzfall vorhanden ist. Daher enthalten die verschiedenen Implementierungen des Konzeptes zusätzlich einen allgemeinen Problemlösemodul (z.B. regelbasiert), der Grundregeln des betreffenden Bereiches enthält.

Nun kann das Zusammenspiel zwischen Speicherprozeß, Suchprozeß und Problemlöseprozeß (in unterschiedlichen Implementierungen nach unterschiedlichen Strategien) in der Weise ablaufen, daß

- ähnliche Fälle gesucht werden
- der Problemlöser gemäß dem Grad der Abweichung weiteres Wissen verarbeitet
- ein Lösungs-"Ereignis" vorgeschlagen wird
- im Erfolgsfall die Speicherung erfolgt
- im Mißerfolgsfall nach den Gründen, d.h. in der Regel nach Indizes gesucht wird.

Hier gehen als wichtige Elemente ein, daß

- das System, solange es irgendwie kann, auf Erfahrungen zurückgreift und erst im Mißerfolgsfall neue Kausalketten aufbaut
- ähnliche Fälle benachbart gespeichert und nur durch Indizes unterschieden werden
- das System selbsttätig lernt

und als wichtigster Aspekt im Sinne der historischen Diskussion

- Implementierungen vorliegen, welche die Integration schwach strukturierten Wissens mit realistischen Problemlöseprozessen darstellen.

R1 Soar. Der Ansatz [Rosenbloom_85] geht aus einer technischen Umgebung hervor, hat aber vergleichbare Relevanz für **wissensintegriertes Problemlösen**, wie bei [Kolodner_86] dargestellt wird. Ausgangspunkt ist das VAX-Konfigurationssystem R1 [McDermott_82], ein wissensintensives regelbasiertes System, das sich in praktischem Einsatz befindet. Das Ausgangssystem erreicht die geforderte *performance* dadurch, daß Experten neben Basisregeln auch eine Vielzahl spezieller Regeln für häufig auftretende Teilkonfigurationen formuliert haben.

Die Erweiterung R1/Soar ([Rosenbloom_85]) dagegen beginnt mit einem allgemeinen regelbasierten Problemlöser (ähnlich OPS5) und nur wenigen Basisregeln. Ferner ist die Konfigurationsaufgabe in ***problem spaces*** partitioniert, und es existieren Kontrollmechanismen, die folgendes leisten:

1. Bei jedem Konfigurationslauf wird buchgeführt, mit welchen Variablen ein *problem space* betreten und mit welchen er verlassen wurde, sowie, ob während der Ersetzungen im *problem space* Variablen von außerhalb des *problem space* benutzt oder manipuliert wurden.

   Ist letzteres nicht der Fall, generiert das System eine spezielle Regel, die genau die Manipulationen zwischen Betreten und Verlassen des *problem space* zusammenfaßt. Dies entspricht der bereits in Abschnitt 2 besprochenen Wissenskompilierung, siehe Abb. 4.

2. Der Regelinterpreter sucht zuerst nach solchen speziellen Regeln und greift erst bei Mißerfolg auf die Standardmethoden zurück.

Mit dieser Konfiguration wurden drei Varianten erprobt:

1. Soar mit Basisregeln, ohne Kontrollmechanismen 1. und 2..
2. Soar mit Basisregeln und von Fachleuten geschriebenen speziellen Regeln und mit Kontrollmechanismus 2..
3. Soar mit Basisregeln und beiden Kontrollmechanismen.

Nachdem die 3. Variante eine gewisse Anzahl von Lernsituationen absolviert hat, ist sie ähnlich schnell wie die Variante 2 und das ursprüngliche R1, während die 1. Variante die Konfigurationsaufgaben nur langsam löst.

Hier bewährt sich also, wie auch bei [Kolodner_86] eine Architektur, die Problemlösetechniken mit komplexen Wissensrepräsentationen integriert. Die Implementierung lernt mittels der vom Fachmann vorgenommenen Partitionierung in *problem spaces* aus positiven Beispielen, ohne jedoch Mechanismen zur Beseitigung von Regeln zu haben, die aufgrund exotischer Beispiele gebildet, aber nie wieder verwendet wurden. Die Beschränkung auf Regeln in dieser Wissensrepräsentation schränkt den Wert als kognitives Modell zwar ein, um so beachtenswerter ist der Beitrag aber aus der Sicht der Wissensakquisition. Jedoch wurde bei [Kolodner_86,Rosenbloom_85] als wichtiger Aspekt hervorgehoben, daß Lernbeispiele dem "nackten" Problemlöser den Erwerb von Präzedenzfällen bzw. speziellen Regeln ermöglichen. Verfahren dieser Art, ähnlich den induktiven Verfahren im engeren Sinne (z.B. [Michalski_84]) , sind behaftet mit einer Stichprobenabhängigkeit, die [6] dazu führen kann, daß eine Ansammlung von Regeln bzw. MOP's das System belastet, die nur für zufällig am Anfang präsentierte exotische Fälle zutreffen. Die Angemessenheit des erworbenen Wissens hängt also von einer sinnvoll getroffenen Auswahl von Lernbeispielen ab.

6.3 Schlußfolgerungen

Das Forschungsgebiet der Kognitiven Modellierung ist in den Bereichen der Künstlichen Intelligenz und der Kognitiven Psychologie verwurzelt. Es bietet sich an, das Methodenspektrum beider Disziplinen zu bereichern und Forschungsergebnisse wechselseitig nutzbar zu machen.

Sicht der Psychologie. In der Psychologie wurden kognitive Modelle häufig wegen ihrer Komplexität und den damit verbundenen Problemen der empirischen Überprüfung kritisiert. Allerdings werden in den letzten Jahren die Vorteile von kognitiven Modellen gegenüber mathematisch formulierten stimulus-response Theorien und verbal beschriebenen Theorieansätzen, wie sie etwa die Gestalttheorie hervorbrachte, zunehmend anerkannt. In diesem Sinne ist der Wert der kognitiven Modellierung für die Psychologie sicher kaum mehr bestritten, und wir begnügen uns unter Hinweis auf die einleitenden methodischen Erwägungen auf die folgende tabellarische Zusammenstellung der Anwendungsmöglichkeiten:

- Zugang zu kognitiven Prozessen des Individuums, im Gegensatz zu psychometrischen Verfahren, die Individuen in Bezug auf eine Normalpopulation beschreiben.
- Theorienbildung durch qualitatives Modellieren und qualitative Übereinstimmung mit beobachteten Phänomenen.
- Theorie-Überprüfung durch detaillierte (und dann in der Regel auf ein enges Feld begrenzte) Modelle und quantitative Übereinstimmung mit beobachteten Phänomenen.
- Anwendung der Ergebnisse in der Software-Ergonomie und zur Entwicklung von Intelligenten Tutoriellen Systemen usw..

Sicht der Informatik. Es fällt auf, daß keiner der referierten Beiträge eine Einordnung mittels klassischer Kriterien der Informatik[7] anbietet. Dies gilt mit Einschränkungen auch bzgl. vieler Kategorien der KI [8]. Vielmehr ist es wohl eher so, daß zu einer gewählten psychologischen Fragestellung nach geeigneten Beschreibungshilfsmitteln aus der Informatik gesucht wird bzw. alternative Repräsentationsformen wie z.B. die MOP's entwickelt werden. Dies wirft natürlich die Frage auf, welchen Wert derartige Untersuchungen für die Informatik bzw. für die KI haben.

---

6 weniger in geschlossenen Welten, siehe etwa LEX von [Mitchel_84]

7 etwa Entscheidbarkeit, Komplexität, Logik-Kalkül, Typen-Hierarchie formaler Sprachen, Datenstrukturen, Algorithmen

8 etwa Repräsentationsparadigmen wie frames, logische Formeln, Strategien wie Vorwärts-, Rückwärts-Inferenz, Breiten- oder Tiefensuche

Nun ist es unbestritten, daß menschliche Kognition in bestimmten Punkten effektiver ist als bekannte Techniken der Informatik, etwa bei Speicherung und Abruf von Expertenwissen oder beim Sprach- oder Bildverstehen. Wenn die Informatik Zugang zu dieser Effektivität gewinnen will, ist es sogar zwingend, daß sie ihre etablierten Methoden und Maßstäbe zunächst außer acht läßt - denn innerhalb dieser Methoden und Maßstäbe ließ sich die gewünschte Effektivität ja gerade nicht erzielen - und unbelastet die Nachbildung beim Menschen beobachteter Prozesse und Phänomene versucht. Die so verstandene kognitive Modellierung ([Kolodner_83a,Kolodner_83b] ist ein Beispiel dieser Art) mag also Hinweise auf ökonomischere Datenstrukturen und effizientere Verfahren geben. Die Grenzen des dabei Erreichbaren sind natürlich durch [Minsky_67] abgesteckt. Andererseits zeigt die Beobachtung menschlicher kognitiver Arbeit - bei hoher mittlerer Effektivität, zumindest in den oben angesprochenen Bereichen - gelegentliche Fehlleistungen auf. Vielleicht kann man das eine - **Effektivität** - nicht ohne das andere - **gelegentliche Fehler** - erreichen. Vielleicht braucht die Informatik zur Überschreitung ihrer derzeitigen Grenzen eine Theorie **fast immer korrekter Verfahren** und Berechenbarkeits- und Komplexitätsbegriffe auf einer solchen Theorie [9].

Neben diesen Bemerkungen, die die Informatik als Ganzes betreffen, gibt es Anwendungen der kognitiven Modellierung für Teilgebiete. So ist es sicher für die Software-Ergonomie wichtig zu wissen, was eine Person über ein Anwendungsprogramm oder eine Programmiersprache zur Benutzung wissen muß. Kognitive Modellierung bietet eine formale Beschreibung dieses Wissens. Das formale Modell ist ein für den Programmdesigner leicht zu handhabendes Mittel zur Analyse und Bestimmung kognitiv komplexer Aspekte.

Auch für die Expertensystem-Entwicklung bietet die kognitive Modellierung Hilfestellung. Denn für die Akzeptanz von Expertensystemen ist die Angepaßtheit der Systemoberfläche an die Denkstrukturen des Experten von entscheidender Bedeutung. Ein System, das den Experten zur Unterbrechung oder Umstrukturierung eigener Gedankengänge zwingt, kann und darf er mit Rücksicht auf die Qualität seiner Arbeit nicht akzeptieren [10]. Der Expertensystem-Designer benötigt also Zugang zu den Denkstrukturen des Experten. Hier bietet sich - neben empirischen Methoden der Psychologie zur Aufdeckung dieser Strukturen - die kognitive Modellierung als Beschreibungssprache an.

Danksagungen

Für Anregungen und kritische Diskussionen möchten wir uns sehr herzlich bei Dietrich Albert, Ulrich Hoppe, Hein Lehmann, Klaus Muthig, Peter Reimann, Hans Spada und Stefan Wrobel bedanken. Stefan Back, Stefan Boschert und Erik Farin haben uns dankenswerter Weise bei der technischen Aufbereitung und Überarbeitung des Manuskripts geholfen.

Adressen

Dr. Franz Schmalhofer
Psychologisches Institut der Universität Freiburg
Niemensstr.10
D-7800 Freiburg i.Br.

Dr. Thomas Wetter
Wissenschaftliches Zentrum der IBM
Wilckensstr.1a
D-6900 Heidelberg

---

9 In ähnlicher Weise wurden in der Mathematik die Unzulänglichkeiten der Riemann'schen Integrationstheorie überwunden durch die Lebesgue-Theorie, in der auf kleinen Ausnahmebereichen Funktionen beliebige Eigenschaften haben dürfen.

10 Generell belasten Eingriffe in etablierte Arbeitsinhalte und -sequenzen in vielfacher Weise (siehe z.B. [Frese_85]). Bei Experten können jedoch wegen der Komplexität der Aufgaben die Konsequenzen besonders gravierend sein.

Bibliographie

[Ackermann_86] Ackermann, D., A Pilot Study on the Effects of Individualization in Man-Computer-Interaction, Proceedings 2nd IFAC-IFIP-IFORS-IEA Conference on Analysis, Design, and Evaluation of Man-Machine-Systems, Varese 1985. Pergamon Press. (London, 1986).

[Anderson_73] Anderson, J.R. and Bower, G.H., Human associative memory, Winston (Washington DC, 1973).

[Anderson_76] Anderson, J.R., Language, Memory and Thought, Erlbaum (Hillsdale NJ, 1976).

[Anderson_83] Anderson, J.R., The architecture of cognition, Harvard University Press (Cambridge, Mass., London, 1983).

[Anderson_84] Anderson, J.R., Farrell, R., and Sauers, R., Learning to program in LISP, Cognitive Science **8** (1984) pp. 87-129.

[Anderson_85] Anderson, J.R., Boyle, C.B., and Reiser, B.J., Intelligent tutoring systems, Science **228** (1985) pp. 456-462.

[Banerij_84] Banerij, R.B., GPS and the psychology of the Rubik cubist: A study in reasoning about actions, Elithorn and Banerij (eds.) Artificial and human intelligence, North Holland (Amsterdam, 1984) pp. 67-79.

[Bower_79] Bower, G.H., Black, J.B., and Turner, T.J., Scripts in memory for text, Cognitive Psychology **11** (1979) pp. 177-220.

[Card_83] Card, S.K., Moran, T.P., and Newell, A., The psychology of human-computer interaction, Erlbaum (Hillsdale NJ, 1983).

[Charniak_85] Charniak, E. and McDermott, D., Introduction to artificial intelligence, Addison-Wesley Publishing Company (Reading, Mass., 1985).

[Chase_82] Chase, W.G. and Ericsson, K.A., Skill and working memory, Bower (ed.) The Psychology of learning and motivation, Academic Press **16** (New York, 1982).

[Chomsky_57] Chomsky, A.N., Syntactic structures, Mouton (The Hague, 1957).

[Church_36] Church, A., An unsolvable problem of elementary number theory, American Journal of Mathematics **58** (1936) pp. 345-363.

[Cohen_83] Cohen, P.R. and Feigenbaum, E.A., Models of cognition, P.R. Cohen and E.A. Feigenbaum (eds.), The handbook of artificial intelligence, Pitman **3** (London, 1983) pp. 1-74.

[vanDijk_83] van Dijk,T.A. and Kintsch,W., Strategies of Discourse Comprehension, Academic Press (New York, 1983).

[deGroot_66] de Groot, A., Perception and memory versus thought: some old ideas and recent findings, B. Kleinmuntz (ed.) Problem solving, Wiley (New York, 1966).

[DeJong_79] De Jong,G., Prediction and substantiation: A new approach to natural language processing, Cognitive Science **3** (1979) pp. 251-273.

[Ericsson_80] Ericsson, K.A. and Simon, H.A., Verbal reports as data, Psychological Review **87** (1980) pp. 215-251.

[Ernst_69] Ernst, G.W. and Newell, A., GPS: A case study in generality and problem solving, Academic Press (New York, 1969).

[Feldman_82] Feldman,J.A. and Ballard,D.H., Connectionist Models and their Properties, Cognitive Science **6** (1982) pp. 205-254.

[Fletcher_81] Fletcher, C.R., Short-term memory processes in text comprehension, Journal of Verbal Learning and Verbal Behavior **20** (1981) pp. 564-574.

[Fodor_74] Fodor, J.A., Bever, T.G., and Garrett, M.F., The psychology of language, McGraw-Hill (New York, 1974).

[Frese_85] Frese, M., A Theory of Control: Implications for Software Design and Training for Computer Aided Work, Universität Muenchen, Institut für Psychologie, Bereich Organisations- und Wirtschaftspsychologie **1** (1985).

[Goetz_81] Goetz,E.T. and Anderson,R.C. and Schallert,D.L., The representation of sentences in memory, Journal of verbal Learning and verbal Behavior **20** (1981) pp. 369-385.

[GRAPES_84] GRAPES user's manual, Advanced Computer Tutoring, Inc. (Pittsburgh, 1984).

[Groen_85] Groen,G.J. and Frederiksen,C.H. and Dillinger,M.L., A propositional analyst's assistant, Behavior Research Methods and Instrumentation **16(2)** (1985) pp. 154-157.

[Hopcroft_79] Hopcroft, J.E. and Ullman, J.D., Introduction to Automata Theory, Languages, and Computation, Addison-Wesley (Reading, Mass., 1979).

[Kieras_85] Kieras, D. and Polson P.G., An approach to the formal analysis of user complexity, International Journal of Man-Machine Studies **22** (1985) pp. 365-394.

[Kintsch_78] Kintsch, W. and van Dijk, T.A., Toward a model of text comprehension and production, Psychological Review **85** (1978) pp. 363-394.

[Kintsch_85] Kintsch,W. and Greeno,J.G., Understanding and solving word arithmetic problems, Psychological Review **92** (1985) pp. 109-129.

[Kolodner_83a] Kolodner, J.L., Maintaining organization in a dynamic long-term memory, Cognitive Science **7** (1983) pp. 243-280.

[Kolodner_83b] Kolodner, J.L., Reconstructive memory: A computer model, Cognitive Science **7** (1983) pp. 281-328.

[Kolodner_86] Kolodner, J.L., Experimental processes in natural problem solving, Manuscript submitted for publication.

[Lachman_79] Lachman, R., Lachman, J.L., and Butterfield, E.C., Cognitive psychology and information processing: An introduction, Wiley (New York, 1979).

[Lehnert_84] Lehnert,W.G. and Robertson,S.P. and Black,J.P., Memory Interactions during Question Answering , in: Learning and Comprehension of Text, Mandl,H. et.al.(Hrsg.), Lawrence Erlbaum (Hillsdale,New Jersey, 1984).

[McDermott_82] McDermott, J., R1: A rule-based configurer of computer systems, Artificial Intelligence **19** (1982) pp. 39-88.

[McKoon_80] McKoon,G. and Ratcliff,R., Priming in Item Recognition : The Organization of Propositions in Memory for Text, Journal of verbal Learning and verbal Behavior **19** (1980) pp. 369-386.

[Michalski_84] Michalski, R.S., Carbonnel, J.G., and Mitchel, T.M. (eds.), Machine Learning, Springer (Berlin, 1984).

[Miller_56] Miller, G.A., The magical number seven, plus or minus two: some limits on our capacity for processing information, Psychological Review **63** (1956) pp. 81-97.

[Miller_64] Miller, G.A. and McKean, K.O., A chronometric study of some relations between sentences, Quarterly J. Experimental Psychology **16** (1964) pp. 297-308.

[Miller_76] Miller, G.A. and Johnson-Laird, P.N., Language and perception, Harvard Univ. Press (Cambridge Mass, 1976).

[Miller_78] Miller, L., Has artificial intelligence contributed to an understanding of the human mind?, Cognitive Science **2** (1978) pp. 111-127.

[Miller_80] Miller, J.R. and Kintsch, W., Readability and recall of short prose passages: A theoretical analysis, Journal of Experimental Psychology: Human Learning and Memory **6** (1980) pp. 335-354.

[Minsky_67] Minsky, M.L., Computation: Finite and infinite machines, Prentice Hall (Englewood Cliffs NJ, 1967).

[Mitchel_84] Mitchel, T.M., Towards combining empirical and analytical methods for inferring heuristics, Elithorn and Banerij (eds.) Artificial and human intelligence, North Holland (Amsterdam, 1984) pp. 81-103.

[Moran_81] Moran, T.P., The Command Language Grammar: a representation for the user interface of interactive computer systems, Int. J. Man-Machine Studies **15** (1981) pp. 3-50.

[Muthig_85] Muthig, K.P., "Gedächtnis" im Labor vs "Behalten" und "Erinnern" in natürlichen Umgebungen, Day, P., Fuhrer, U. und Laucken, U. (Hrsg.), Umwelt und Handeln: Ökologische Anforderungen und Handeln im Alltag. Attempto-Verlag (Tuebingen, 1985) pp. 334-348.

[Newell_56] Newell, A. and Simon, H.A., The logic theory machine: A complex information processing system, Transactions on information theory (Institute of Radio Engineers) **IT-2** (1956) pp. 61-79.

[Newell_57] Newell, A., Shaw, J.C., and Simon, H.A., Preliminary description of the general problem solving program I (GPS I), CIP working paper No 7, December (1957).

[Newell_72] Newell, A. and Simon, H.A., Human Problem Solving, Prentice Hall (Englewood Cliffs NJ, 1972).

[Nisbett_77] Nisbett, R.E. and Wilson, T.D., Telling more than we can know: verbal reports on mental processes, Psychological Review **84** (1977) pp. 231-259.

[Norman_75] Norman, D.A., Rumelhart, D.E., and the LNR Research Group, Explorations in cognition, Freeman (San Francisco, 1975).

[Polson_85] Polson, P.G., Vortrag am Fraunhofer-Institut IAO, (Stuttgart, August 1985).

[Polson_86] Polson, P.G., A quantitative theory of human computer interaction , erscheint in: Carrol, J.M. (ed.), Cognitive Aspects of Human-Computer-Interaction, .

[Popper_66] Popper, K.R., Logik der Forschung, Mohr (Tuebingen, 1966).

[Post_36] Post, E.L., Finite combinatory processes - Formulation I, Journal of Symbolic Logic **1** (1936) pp. 103-105.

[Ratcliff_78] Ratcliff,R. and McKoon,B., Priming in item recognition : Evidence for the propositional structure of sentences, Journal of verbal Learning and verbal Behavior **17** (1978) pp. 403-418.

[Reiser_85] Reiser, B.J., Anderson, J.R., and Farrell, R.G., Dynamic student modelling in an intelligent tutor for LISP programming, Proceedings of Ninth International Joint Conference on Artificial Intelligence (Los Angeles, 1985).

[Rosenbloom_85] Rosenbloom, P.S., Laird, J.E., McDermott, J., Newell, A., and Orciuch, E., R1-Soar: An experiment in knowledge-intensive programming in a problem-solving architecture, IEEE Transact. Pattern Analysis and Machine Intelligence **PAMI-7** (1985) pp. 561-569.

[Rumelhart_72] Rumelhart, D.E., Lindsay, P.H., and Norman. D.A., A process-model for human long-term memory, Tulving, E. and Donaldson, W. (eds.), Organisation of memory, Academic Press (New York, 1972).

[Schank_69] Schank, R.C. and Tesler, L., A conceptual parser for natural language, Proc. IJCAI Washington DC (1969).

[Schank_75] Schank, R.C., Conceptual Information Processing, North Holland (Amsterdam, 1975).

[Schank_77] Schank, R.C. and Abelson, R.P., Scripts, plans, goals, and understanding, Erlbaum (Hillsdale NJ, 1977).

[Schank_80] Schank, R.C., Language and Memory, Cognitive Science **4** (1980) pp. 243-284.

[Schmalhofer_83] Schmalhofer,F., Text processing with and without Prior Domain Knowledge: Knowledge-versus Heuristic-Dependent Representations, Bericht aus dem Psychologischen Institut der Universität Heidelberg, Diskussionspapier Nr. 32 (1983).

[Schmalhofer_86a] Schmalhofer, F. and Glavanov, D., Three components of understanding a programmer's manual: Verbatim, propositional, and situational representations, Journal of Memory and Language **25** (1986) pp. 279-294.

[Schmalhofer_86b] Schmalhofer,F. and Schäfer,I., Lautes Denken bei der Wahl zwischen benannt und beschrieben dargebotenen Wahlalternativen, Sprache und Kognition **2** (1986) pp. 73-81.

[Schmalhofer_86c] Schmalhofer,F., The construction of programming knowledge from system explorations and explanatory text : a cognitive model , in : Rollinger,C. und Horn, W. GWAI-86 und 2. Oesterreichische Artificial Intelligence Tagung, Springer-Verlag (Heidelberg, 1986) pp. 152-163.

[Siekmann_83] Siekmann, J.H., Einfuehrung in die kuenstliche Intelligenz, Bibel, W. & Siekmann, J.H.: Kuenstliche Intelligenz Fruehjahrsschule, Teisendorf, Maerz 1982 (Berlin, 1982) pp. 1-60.

[Sleeman_82] Sleeman, D. and Brown, J.S., Intelligent tutoring systems, Academic Press (London, 1982).

[Spada_85] Spada, H. and Opwis, K., Intelligente Tutorielle Systeme aus psychologischer Sicht, H. Mandl and P.M. Fischer (eds.), Lernen im Dialog mit dem Computer, Urban & Schwarzenberg (Muenchen, 1985) pp. 13-23.

[Sternberg_69] Sternberg, S., Memory Scanning: Mental Processes Revealed by Reaction-Time Experiments, American Scientist **57** (1969).

[Swinney_79] Swinney, D.A., Lexical access during sentence comprehension: (re)consideration of context effects, Journal of Verbal Learning and Verbal Behaviour **18** (1979) pp. 645-659.

[Swinney_84] Swinney, D.A., Theoretical and methodological issues in cognitive science: A psycholinguistic perspective, Kintsch, W., Miller, J.R., and Polson, P.G. (eds.), Method and tactics in cognitive science, Erlbaum (Hillsdale, N.J., 1984) pp. 217-233.

[Turing_36] Turing, A.M., On computable numbers, with an application to the Entscheidungsproblem, Proceedings of the London Mathematic Society (Series 2) **42** (London, 1936) pp. 230-265.

[Turing_63] Turing, A.M., Computing machinery and intelligence, Feigenbaum and Feldman (eds.), Computers and thought, McGraw-Hill (New York, 1963) pp. 1-35.

[Turner_78] Turner,A. and Greene,E., The construction and the use of a propositional text base, Technical Report No.63, April 1977, Institute for the Study of Intellectual Behavior. University of Colorado (1977).

[Vorberg_86] Vorberg, D., Programmierkonzepte als Werkzeuge zum Problemlösen, Vortrag beim Workshop "Modellierung von Programmierwissen" (Marburg, 22. - 24.5.1986).

[Wahlster_82] Wahlster, W., Natuerlichsprachliche Systeme: Eine Einfuehrung in die sprachorientierte KI-Forschung, Bibel, W. & Siekmann, J.H.: Kuenstliche Intelligenz Fruehjahrsschule, Teisendorf, Maerz 1982 (Berlin, 1982) pp. 203-283.

[Weber_86] Weber, G. and Wender, K., Modellierung episodischen Wissens in einem LISP-Tutor, Vortrag beim Workshop "Modellierung von Programmierwissen" (Marburg, 22. - 24.5.1986).

[Wetter_85] A framework for modelling the knowledge and behaviour of occasional users of application software, 3rd Symposium on Empirical Foundations of Information and Software Sciences (EFISS), Risoe, October 1985, Proceedings to appear (Plenum Press) (1985).

[Winston_84] Winston, P.H., Artificial intelligence, Addison-Wesley (Reading, Mass., 1984).

[Ziegler_86] Ziegler, J.E., Hoppe, H.U., and Faehnrich, K.P., Learning and transfer for text and graphics editing with a direct manipulation interface, In: Proc. CHI '86, Human factors in computing systems (Boston, 1986) pp. 72-77.