Das Agentenmodell InteRRaP im Vergleich mit kognitiven Architekturen

Ralf Sesseler

Diplomarbeit im Fach Informatik

bei Professor J. Siekmann

25. November 1996

AUFGABENSTELLUNG UND ÜBERBLICK

Aufgabenstellung

Kognitive Architekturen integrieren verschiedene kognitive Fähigkeiten – wie beispielsweise Planen, Wissensrepräsentation und Lernen – zu einem Gesamtsystem, sowohl um einen möglichst großen Bereich an intelligenten Fähigkeiten abzudecken, als auch um eine gegenseitige Unterstützung der einzelnen Komponenten untereinander mit einer sich daraus ergebenden höheren Funktionalität zu erreichen. Die Motivation für die Gestaltung kognitiver Architekturen besteht in einem besseren Verständnis von Intelligenz im allgemeinen, in der Modellierung der menschlichen Kognition sowie in der Entwicklung intelligenter technischer Anwendungen. Das Ziel dieser Diplomarbeit ist eine Untersuchung der Stärken und Schwächen des am DFKI entwickelten Agentenmodells INTERRAP im Vergleich zu anderen kognitiven Architekturen. Dieser Vergleich besteht aus einer strukturellen Gegenüberstellung, bei der die einzelnen Komponenten der verschiedenen Architekturen sowie deren Integration in der Gesamtarchitektur betrachtet werden, und aus einer funktionalen Analyse, bei der untersucht wird, wie die Integration der einzelnen Komponenten die Funktionalität der Gesamtarchitektur unterstützt. Als Vergleichsarchitekturen zu INTERRAP werden ACT, TOK/PRODIGY und SOAR betrachtet.

Kapitelübersicht

Das erste Kapitel dient einer kurzen Einführung und Begriffsbestimmung. Im Rahmen des heute in der Kognitionswissenschaft vorherrschenden Informationsverarbeitungsansatzes werden zunächst Definitionen für grundlegende Begriffe wie Kognition und Intelligenz angegeben und diskutiert. Nach einem kurzen Überblick über kognitive Architekturen wird ein Vergleichsschema für integrierende Architekturen angegeben, das die strukturelle Integration einzelner Komponenten und die Funktionalität der Gesamtarchitektur berücksichtigt. Dieses Schema wird in den Kapiteln drei und vier auf INTERRAP und die Vergleichsarchitekturen ACT, TOK/PRODIGY und SOAR angewendet.

Zuvor werden im zweiten Kapitel diese Architekturen in ihrer Grundstruktur sowie die ihnen zugrundeliegenden Annahmen vorgestellt. ACT (Anderson 1983, 1993) ist ein empirisches Modell der menschlichen Kognition und stellt den Versuch dar, verschiedene Modelle der Kognitionspsychologie zu einer Gesamtarchitektur zu vereinen. Mit TOK (Bates et al. 1992) werden künstliche Charaktere modelliert, die autonom in simulierten Welten agieren

können. Das reaktive Planungsmodul HAP von TOK kann für deliberatives Planen mit dem Planer PRODIGY kombiniert werden (Blythe & Reilly 1993). PRODIGY (Veloso et al. 1995) entstand als Experimentierumgebung zur Integration von rationaler Planung mit verschiedenen Lernmechanismen. SOAR (Laird et al. 1987, Newell 1990) basiert auf einer theoretischen Definition von Intelligenz und verbindet heuristische Suche und Lernen miteinander. INTERRAP (Fischer et al. 1995, Müller 1996), der eigentliche Untersuchungsgegenstand dieser Arbeit, ist ein pragmatischer Ansatz zur Agentenmodellierung, der sich an BDI- und geschichteten Architekturen orientiert.

Das dritte Kapitel beginnt den Vergleich der Architekturen mit der strukturellen Gegenüberstellung, die sich in vier Teile gliedert: die Informationsarten, die Informationsspeicherung, die Informationsverarbeitungsprozesse und die Struktur der Gesamtarchitektur. Die verschiedenen Arten von Informationen und ihre Speicherung bilden die Grundlage für alle kognitiven Prozesse. Die Informationsverarbeitungsprozesse dienen der Transformation von Informationen beginnend bei der Wahrnehmung über interne kognitive Prozesse bis hin zur Handlungsausführung. Die Struktur der Gesamtarchitektur umfaßt die Integration der einzelnen Komponenten und die Kontrolle der Interaktion zwischen den Komponenten.

Die funktionale Analyse von INTERRAP im Vergleich mit den anderen Architekturen geschieht im vierten Kapitel. Die Funktionalität einer Architektur umfaßt zum einen deren Verhaltensspektrum – die verschiedenen Möglichkeiten der Verwendbarkeit einer Architektur – und zum anderen Aspekte der Intelligenz und Rationalität – Erfolg und Qualität des Verhaltens. An kognitivem Spektrum werden reaktives, deliberatives, adaptives und soziales Verhalten betrachtet. Die Zweckmäßigkeit dieses Verhaltens wird anhand von Rationalität, Robustheit, Flexibilität, Vielseitigkeit und Universalität der Handlungen beurteilt.

Der Vergleich der Architekturen endet im fünften Kapitel mit einer Zusammenfassung und einer abschließenden Bewertung von INTERRAP im Licht der Gegenüberstellung mit den anderen kognitiven Architekturen.

INHALT

1. KOGNITIVE ARCHITEKTUREN

1.1. Kognition

1.1.1. Informationsverarbeitung

Die Grundannahme der heutigen Kognitionswissenschaft[1] und zugleich das verbindende Glied zwischen den einzelnen kognitionswissenschaftlichen Disziplinen ist der Informationsverarbeitungsansatz: ”Das kognitive System von Menschen kann man am besten als ein Informationsverarbeitungssystem verstehen.“ (Wessells 1984, S. 371). Er dient zur Etablierung einer eigenständigen Ebene kognitiver Prozesse und Phänomene, auf der Kognition als Verarbeitung mentaler Information beschrieben werden kann und für die mentale Repräsentationen wie Symbole, Vorstellungen, Schemata und Regeln als theoretische Konstrukte postuliert werden (Gardner 1989, S. 50).

Der Ursprung dieses Ansatzes liegt in der Übertragung und Verwendung von Modellen und Begriffen der Informationstheorie und der Computerwissenschaft zur Beschreibung kognitiver Prozesse des Menschen durch Psychologen Mitte dieses Jahrhunderts[2]. Diese Theorien ermöglichten der Psychologie die Annahme interner mentaler Prozesse und Strukturen, die die zu dieser Zeit in Amerika vorherrschende behaviouristische Tradition als unwissenschaftlich ablehnte, da sie nicht wie das körperliche Verhalten objektiv von außen beobachtbar sind. Wenn jedoch die Informationstheorie eine formale Grundlage zum Umgang mit Information liefert und Computer eine maschinelle Verarbeitung symbolischer Information ermöglichen, so sollte es auch in der Psychologie wissenschaftlich akzeptierbar sein, interne Informationsverar-

[1] Die Kognitionswissenschaft entwickelt sich seit Mitte der siebziger Jahre aus interdisziplinären Bestrebungen von Psychologen, Linguisten, Informatikern, Neurophysiologen, Anthropologen und Philosophen. Einen geschichtlichen Überblick über die Kognitionswissenschaft bietet Gardner (1989).

[2] Eine ausführliche Darlegung des Einflusses der Informationstheorie auf die kognitive Psychologie bietet Neisser (1974, S. 23ff).

beitungsprozesse zwischen Wahrnehmung und Verhalten – Input und Output in der Sprache der Informationstheorie – zu postulieren, die über die einfachen behaviouristischen Reiz-Reflex-Bögen weit hinausgehen, und Kognition als Speicherung und Transformation mentaler Repräsentationen aufzufassen. Obwohl Information oder Programme nicht unbedingt beobachtbare Entitäten sind, besitzen sie dennoch genügend Realität, um Computer zu steuern, und sind damit als wissenschaftliche Konstrukte unbedenklich.

Ein weiterer Vorteil des Informationsverarbeitungsansatzes besteht darin, daß er die prinzipielle Unabhängigkeit der Psychologie von der materiellen Grundlage psychischer Prozesse annimmt und somit die Eigenständigkeit von psychologischen Theorien gegenüber neurophysiologischen begründet. Auch wenn alle kognitiven Prozesse beim Menschen letztendlich Gehirnprozesse sind, so ist doch die Art der Betrachtung eine grundsätzlich verschiedene: ”Psychologie ist ... eine Wissenschaft, die sich mit der wechselseitigen Abhängigkeit bestimmter Geschehnisse beschäftigt und nicht mit ihrer physikalischen Natur.“ (Neisser 1974, S. 22). Dies rechtfertigt auch den Einsatz von Computerprogrammen zur Simulation kognitiver Prozesse und die Möglichkeit von künstlicher Intelligenz auf Computern.

Viele Kognitionspsychologen sehen die Parallelen, die der Informationsverarbeitungsansatz zwischen menschlicher Kognition und Computern zieht, jedoch nicht als eine Gleichsetzung an, sondern nur als eine Anlehnung an das Vokabular der Informationstheorie und als fruchtbare Quelle für Analogien und Hypothesen. So ist Wessells der Meinung ”... daß, auch wenn dieser Ansatz der Informationsverarbeitung nützlich ist, damit nicht alle wichtigen Aspekte menschlicher Kognition angemessen bearbeitbar sind.“ (1984, S. 375). Gerade die Behandlung von Emotionen und Stimmungen auf der einen Seite sowie von Verständnis und Bewußtsein auf der anderen Seite scheint sich mit dem Informationsverarbeitungsparadigma nicht gut vereinbaren zu lassen (Haugeland 1978, 7. Kapitel; Neisser 1974, S. 25).

Andere Kognitionswissenschaftler nehmen die Computeranalogie wörtlich und versuchen zu zeigen, daß die menschliche Kognition oder sogar jedes intelligente System nicht nur einem Computer vergleichbar ist, sondern auf einer abstrakten Ebene tatsächlich ein Computer ist. Ausgehend von einer abstrakten Darstellung der Symbolverarbeitungsprozesse in Computern haben Newell und Simon (1975) ihre *Physical Symbol System Hypothesis* formuliert, während sprachphilosophische Überlegungen über die semantischen Eigenschaften der Syntax formaler Sprachen den verschiedenen Spielarten der *Computational Theory of Mind* zugrundeliegen.

The Physical Symbol System Hypothesis

Mit ihrer Charakterisierung eines Physical Symbol System (PSS) versuchen Newell und Simon (1975, S. 40ff) eine allgemeine Grundlage für die empirische Erforschung von Computersystemen in Form eines qualitativen Strukturgesetzes zu geben. Ein PSS ist definiert als ein System, das physisch ist, insofern es den Gesetzen der Physik gehorcht und auch technisch realisierbar ist, und das Symbolstrukturen manipulieren kann. Symbole sind dabei definiert als

physische Muster, die in größeren Symbolstrukturen, den Ausdrücken, organisiert sind. Die Manipulation dieser Symbolstrukturen durch das System bedeutet, daß sich die Menge an Symbolen, aus denen das PSS besteht, mit der Zeit selbst nach festen Regeln verändert. Symbolische Ausdrücke erfüllen dabei innerhalb eines PSS zwei grundlegende Funktionen:

- *Bezeichnung*: Ein Ausdruck bezeichnet ein Objekt, wenn das PSS mit Hilfe des Ausdrucks auf das Objekt zugreifen oder sich in einer dem Objekt angemessenen Weise verhalten kann.
- *Interpretation*: Das PSS kann einen Ausdruck interpretieren, wenn dieser einen Prozeß bezeichnet, den das System ausführen kann.

Ausgehend von dieser Definition eines PSS formulieren Newell und Simon ihre zentrale These des PSS als notwendiger und hinreichender Bedingung für Intelligenz:

> "*The Physical Symbol System Hypothesis*. A physical symbol system has the necessary and sufficient means for general intelligent action." (1975, S. 41)

Unter allgemeiner Intelligenz verstehen sie dabei ein dem menschlichen Handeln vergleichbares situationsabhängiges und zielgerichtetes Verhalten. Ein PSS ist notwendig für Intelligenz, insofern jedes System mit allgemeiner Intelligenz ein PSS sein muß, und es ist hinreichend, insofern jedes genügend umfassende PSS so organisiert werden kann, daß es allgemeine Intelligenz zeigt. Genügend umfassend sind all die PSS, die einerseits berechnungsvollständig sind – d.h. sie können zumindest prinzipiell jede überhaupt berechenbare mathematische Funktion berechnen – und andererseits genügend Speicherplatz für Symbole besitzen, um hinreichend komplexe Berechnungen auch tatsächlich ausführen zu können. Abgesehen von Unterschieden im Speicherplatz sind alle diese PSS gleich mächtig in dem Sinne, daß jedes PSS jedes andere vollständig emulieren kann.

Die PSS-Hypothese ist kein logisches Theorem, sondern eine empirische Annahme, die der empirischen Bestätigung oder Widerlegung bedarf: "The hypothesis could indeed be false." (Newell & Simon 1976, S. 42). Zur Stützung ihrer Hypothese geben Newell und Simon deshalb drei Arten von empirischer Evidenz an. Die Annahme von PSS als hinreichender Bedingung für Intelligenz ist ihrer Meinung nach durch die Programmierung intelligenter Computersysteme seitens der KI gerechtfertigt und die Annahme als notwendiger Bedingung durch die Modellierung menschlichen Verhaltens mit Computerprogrammen seitens der Kognitionspsychologie. Das heißt, PSS (Computer) können intelligentes Verhalten zeigen, und intelligentes Verhalten (von Menschen) läßt sich durch PSS beschreiben. Als dritten Beleg für ihre Hypothese sehen sie das Fehlen von Alternativen an, die sowohl klar formulierbar sind, als auch eindeutig nicht als PSS aufgefaßt werden können.

The Computational Theory of Mind

Die Computational Theory of Mind (CTM) ist weniger eine einzelne Theorie als eine ganze Familie von Theorien mit einer gemeinsamen Grundauffassung über formale Systeme,

Computer und Kognition. Alle Vertreter der CTM teilen die Ansicht, daß der menschliche Geist vom Prinzip her auf die gleiche Art funktioniert wie ein Computer und folglich auf der funktionalen Ebene ein Computer ist. Mentale Prozesse bestehen gemäß der CTM in formalen Operationen, die Symbole rein syntaktisch so manipulieren, daß ihre semantische Interpretation erhalten bleibt:

> "The basic idea of cognitive science is that *intelligent beings are semantic engines* – in other words, automatic formal systems with interpretations under which they consistently make sense." (Haugeland 1981a, S. 31)

Die Argumentation für die CTM gliedert sich gemäß der Darstellung von Haugeland (1981a) in mehrere Teilschritte ausgehend von formalen Systemen über Computer hin zu Repräsentationen und mentalen Prozessen[3]. Formale Systeme sind Notationen zur Manipulation von Symbolen nach festen syntaktischen Regeln. Computer sind besondere formale Systeme, da sie zum einen die Symbolmanipulation automatisch ausführen und zum anderen universell sind, d.h. sie können jedes beliebige formale System simulieren. Formale Systeme werden semantisch interpretiert, indem den Symbolen eine Bedeutung zugewiesen wird, und sie sind wahrheitserhaltend, wenn die syntaktischen Regeln nur Symbolmanipulationen zulassen, die gemäß dieser Interpretation semantisch korrekt sind. Besteht die Bedeutung der Symbole eines formalen Systems in einem Bezug zur Realität, so spricht man von mentalen Repräsentationen. Gemäß der CTM sind mentale Prozesse somit formale Prozesse, die über Repräsentationen definiert sind und diese durch rein syntaktische Operationen wahrheitserhaltend manipulieren.

Wenn Computer als semantische Maschinen sinnvoll mit bedeutungshaltigen Symbolen umgehen können, so der Grundgedanke der CTM, dann wäre es denkbar, daß auch die menschlichen Denkprozesse auf diese Art funktionieren und ihre Bedeutung erhalten und daß Bedeutung, Kognition und Intelligenz überhaupt nur so wissenschaftlich erklärt werden können. Damit wären Menschen nicht nur mit Computern vergleichbar, der menschliche Geist wäre tatsächlich ein Computer im Sinne einer semantischen Maschine. Und da Digital-Computer universelle semantische Maschinen sind, eignen sie sich nicht nur für die Simulation kognitiver Prozesse, in ihnen können genauso echte, sogar formal vollkommen äquivalente kognitive Prozesse stattfinden wie in einem menschlichen Gehirn. Weil formale Systeme rein syntaktisch definiert sind, können sie zudem vollkommen unabhängig von ihren physischen Eigenschaften untersucht werden – die offensichtlichen Unterschiede der physischen und technischen Struktur von Digital-Computer und menschlichem Gehirn wären auf dieser Ebene irrelevant.

[3] Alternative Darstellungen der CTM finden sich beispielsweise bei Fodor (1975), Dennett (1978) und Cummins (1994).

Symbole und Repräsentation

Mit der Physical Symbol System Hypothesis und der Computational Theory of Mind gibt es zwei theoretische Formulierungen des Informationsverarbeitungsansatzes, deren streng formalistischer Charakter eine klare Grundlage für eine wissenschaftliche Beschäftigung mit kognitiven Prozessen bietet. Auch wenn beide Theorien von verschiedenen Ansätzen ausgehen – die PSS-Hypothese von der Informationsverarbeitung durch Computer und die CTM von den semantischen Eigenschaften formaler Systeme –, ihre Kernaussage stimmt überein: Kognitive Prozesse sind Computeroperationen, die rein mechanisch Symbole manipulieren.

Der wesentliche Unterschied zwischen beiden Theorien liegt in den jeweiligen Annahmen über die Funktion von Symbolen. Die Funktion von Symbolen ist Repräsentation, für die CTM besteht diese jedoch in einer system*externen* Bezugnahme, für die PSS-Hypothese zunächst nur in einer system*internen* Bezugnahme. Die CTM faßt die Bedeutung von Symbolen gemäß der logisch-mathematischen Modelltheorie (s. z.B. Barwise & Etchemendy 1989) auf. Danach besteht die Interpretation eines Symbols in einer eindeutigen Abbildung zu einem realen Gegenstand der externen Welt. Das Hauptproblem dieser *Representational Theory of Mind* besteht darin, daß bei formalen logischen oder mathematischen Systemen (und i.a. auch bei Computerprogrammen) diese Interpretation durch den menschlichen Benutzer vorgenommen wird, bei einem kognitiven System dieses die Interpretation jedoch selbst gewährleisten muß[4].

Die Funktion eines Symbols besteht gemäß der PSS-Hypothese in der Steuerung interner Prozesse sowohl durch die Bezeichnung anderer interner Strukturen, als auch indem Symbole durch das System interpretiert werden als auszuführende Operationen, die vorhandene Symbolstrukturen manipulieren. Symbole repräsentieren innerhalb eines PSS Operationen oder andere Symbolstrukturen, aber nicht die externe Welt: "... a symbol system: *memory*, *symbols*, *operations*, and *interpretation*[5]. However, none of these functions (not even symbols) is the function of *representation* of the external world. Symbols do provide an internal representation function." (Newell et al. 1989, S. 104). Diese internen Prozesse können jedoch die externe Welt repräsentieren, indem sie dem Repräsentationsgesetz entsprechen, nach dem externe Objekte und Transformationen durch Kodierung und Dekodierung zu entsprechenden internen in Bezug gesetzt werden (Newell 1990, S. 54ff). Eine direkte Zuordnung von Symbolen zu externen Objekten und Transformationen wird dabei nicht vorausgesetzt, aber auch nicht ausgeschlossen.

[4] Eine Darstellung der Representational Theory of Mind sowie verschiedene Versuche zur Lösung des Problems der Bezugnahme enthält Stich & Warfield (1994). Eine Kritik der Modelltheorie als objektivistische Bedeutungstheorie findet sich bei Putnam (1981) und Lakoff (1987).

[5] Interpretation hier im Sinne der oben gegebenen Definition eines PSS als Bezeichnung für einen internen Prozeß, nicht im Sinne der CTM als Abbildung auf die externe Welt.

1.1.2. Kognition und Intelligenz

Der Informationsverarbeitungsansatz ist nicht nur die zentrale und verbindende Annahme der Kognitionswissenschaft, er dient auch zur Definition des Begriffs Kognition:

> ”*Kognition* ... betrifft die Arten von Informationen, die wir in unserem Gedächtnis haben, und die Vorgänge, die sich auf die Aufnahme, das Behalten und Verwenden solcher Informationen beziehen.“ (Wessells 1984, S. 14)

> ”Kognition ist die Aktivität des Wissens: der Erwerb, die Organisation und der Gebrauch von Wissen.“ (Neisser 1979, S. 13)

Die menschliche Kognition wird dabei als eine Abfolge geordneter Phasen von Informationsverarbeitungsschritten angesehen, die der Gewinnung, dem Abruf und der Manipulation kognitiver Information dienen (Anderson 1988, S. 15; Wessells 1984, S. 42).

Der zweite wichtige Aspekt von Kognition ist neben der Informationsverarbeitung deren Verwendung zur Verhaltenssteuerung, also die Umsetzung von sensorischer Information, die aus der Umgebung aufgenommen wird, in Körperbewegungen bzw. in Wissen, das für (körperliches) Verhalten benutzt werden kann (Neisser 1974, S. 19).

Da diese Steuerung des Verhaltens in Abhängigkeit von der Umgebung den verschiedenen Zielen des Handelnden dienen soll, ist der dritte Aspekt von Kognition Intelligenz. Denn nur wenn den Handlungen ein gewisser Sinn zugesprochen werden kann, ist es überhaupt möglich, von Kognition zu reden, weil andernfalls jeder beliebige Mechanismus als Steuerung seines Verhaltens durch Informationsverarbeitung aufgefaßt werden könnte[6].

Leider gibt es keine wirklich befriedigende und allgemein akzeptierte Definition für Intelligenz, so daß meist nur auf den intuitiven Begriff und auf das menschliche Verhalten als Beispiel zurückgegriffen wird:

> ”Although no really satisfactory definition of intelligence has been proposed, we are ordinarily willing to judge when intelligence is being exhibited by our fellow human beings.“ (Simon & Kaplan 1989, S. 1)

Ein wichtiges Charakteristikum von Intelligenz ist zielgerichtetes Verhalten, das längerfristige und abstraktere Zielsetzungen ermöglicht. Ebenso wichtig ist die Fähigkeit, diese Ziele auch erreichen zu können – möglichst unabhängig von der gegebenen Situation:

> ”We measure the intelligence of a system by its ability to achive stated ends in the face of variations, difficulties, and complexities posed by the task environment.“ (Newell & Simon 1976, S. 37)

[6] Dennett (1971) argumentiert dafür, daß einem System nur dann sinnvoll Wissen zugeschrieben werden kann, wenn ihm gleichzeitig auch ein gewisses Maß an Rationalität zugebilligt wird: ”A false belief system is a conceptual impossibility“ (S. 237). Denn nur wenn die Handlungen des Systems weitgehend konsistent mit seinem Wissen sind, ist es sinnvoll, das Wissen als Grund für die Handlung und damit die Handlungen als (kognitiv gesteuerte) Handlungen und nicht nur als zufällige Körperbewegungen aufzufassen.

Da Ziele immer in einer bestimmten gegebenen Situation erreicht werden müssen, in der sie mehr oder weniger schwer erreicht werden können, bedeutet dies die Fähigkeit, Probleme von verschiedenster Art und Komplexität lösen zu können. Des weiteren gibt es immer verschiedene Möglichkeiten, ein Problem zu lösen, so daß auch Rationalität ein Zeichen für Intelligenz ist, d.h. die Fähigkeit möglichst zweckmäßige und effiziente Problemlösungen zu wählen.

Probleme auch in unbekannten oder unvorhergesehenen Situationen lösen zu können, bedeutet die Unabhängigkeit von fest vorgegebenem Wissen und damit Universalität. Flexibilität ermöglicht neuartige Problemlösungen auch in bekannten Situationen und Lernfähigkeit die Verbesserung des Verhaltens anhand von Erfahrung. Diese Charakterisierung von Intelligenz durch zielgerichtetes Problemlösen zusammen mit Rationalität, Universalität, Flexibilität und Lernfähigkeit stellt selbstverständlich keinerlei Anspruch auf Vollständigkeit[7].

Als kurze, vereinfachte Definition für Kognition und Intelligenz läßt sich zusammenfassen: *Kognition* besteht in der (intelligenten) Verarbeitung von Information zur Steuerung des Verhaltens. *Intelligenz* besteht in zweckmäßiger und erfolgreicher Kognition in Bezug auf Wissen und Ziele[8].

1.2. Kognitive Architekturen

1.2.1. Architekturen

Wird ein kognitives System als System zur Informationsverarbeitung aufgefaßt, so muß zwischen dem eigentlichen kognitiven System, der Architektur, und der Information, die von diesem verarbeitetet wird, unterschieden werden. Unter einer Architektur versteht man dabei eine Menge von grundlegenden Mechanismen zur Gewinnung, Verarbeitung und Speicherung von Information:

> ”The fundamental design specifications of an information-processing system are called its *architecture*.“ (Simon & Kaplan 1989, S. 7)
>
> ”An architecture is a fixed set of mechanisms that enable the acquisition and use of content in a memory to guide behaviour in the pursuit of goals.“ (Newell 1992a, S. 27)

[7] Beispielsweise ermöglicht diese Charakterisierung von Intelligenz keine Unterscheidung zwischen Zielen, deren *Aufstellen* ein bestimmtes Maß an Intelligenz voraussetzt, oder die Berücksichtigung der Kohärenz und Rechtfertigung von Wissen.

[8] Diese Definition scheint zirkulär, da Kognition über Intelligenz und Intelligenz über Kognition definiert wird. Dies liegt jedoch daran, daß – wie bereits erwähnt – Kognition ohne Intelligenz (oder Rationalität) nicht sinnvoll ist und Intelligenz eine Eigenschaft von Kognition ist.

Die Architektur bestimmt dabei die feste Struktur eines kognitiven Systems, die Information seinen veränderlichen Gehalt. Die Mechanismen der Architektur sind meist sehr allgemein und weitgehend unabhängig von bestimmten kognitiven Bereichen, während die enthaltene Information mehr spezifisch und nur speziell für einzelne Bereiche verwendbar bzw. nützlich sein kann.

Entsprechend hängt das Verhalten eines kognitiven Systems sowohl von der Architektur, als auch von der enthaltenen Information ab. Die Architektur bestimmt das Verhalten anhand der aktuellen Information – Wissen und Ziele des Systems –, sie benötigt jedoch immer erst Information, um überhaupt ein Verhalten bestimmen zu können.

Die Architektur spezifiziert eine feste Menge von allgemeinen Mechanismen, die die Grundlage für alle kognitiven Prozesse bilden. Auch wenn manche Teile einer Architektur spezielle Funktionen ausüben, so muß doch die Gesamtarchitektur Prinzipien gehorchen, die allgemein genug sind, um der Universalität und Flexibilität allgemeiner Intelligenz gerecht zu werden. Für eine feste Menge allgemeiner kognitiver Prinzipien für unterschiedliche kognitive Vorgänge beim Menschen sprechen weiterhin seine große Lernfähigkeit, die Gemeinsamkeiten und Zusammenhänge zwischen verschiedenen kognitiven Bereichen sowie die enorme Zahl an kognitiven Fähigkeiten des Menschen, an die keine spezifische, sondern nur eine allgemeine Anpassung im Laufe der Evolution möglich war (Anderson 1983, S. 1ff).

Eine weithin angenommene Grundvoraussetzung für derart universelle kognitive Architekturen ist die Berechnungsvollständigkeit (wie sie die Physical Symbol System Hypothesis fordert, s. Kapitel 1.1.1.)[9], allerdings reicht diese allein zur Charakterisierung einer kognitiven Architektur noch nicht aus, da sie lediglich eine allgemeine Klasse von möglichen Informationsverarbeitungsprozessen festlegt, jedoch nichts über die tatsächliche Realisierung und Ausführung dieser Prozesse besagt. Jede berechnungsvollständige Architektur kann zwar jede andere exakt simulieren, dies betrifft jedoch nur das (abstrakte) Input-Output-Verhalten, nicht aber das genaue (Zeit-)Verhalten. Weiterhin ist der Aufwand zur Realisierung eines bestimmten Prozesses nicht in jeder Architektur gleich, ebenso wie der jeweilige Aufwand zur Ausführung. Ein weiterer Unterschied ist, wieviel an Information durch Vorbereitung und Programmierung bereitgestellt werden muß, um das gleiche Verhalten zu erreichen, d.h. wie autonom und lernfähig ein System ist.

Zur Beurteilung einer Architektur ist deshalb weniger die prinzipielle Realisierbarkeit bestimmter Informationsverarbeitungsprozesse von Interesse, sondern vielmehr, welche konkreten Möglichkeiten zur Realisierung die Architektur vorgibt. Dabei sind drei Fälle zu unterscheiden: kognitive Prozesse können entweder direkt ein integrierter Teil der Architektur sein, sie können relativ einfach und unkompliziert innerhalb des durch die Mechanismen der Architektur vorgegebenen Rahmens realisierbar sein oder sie können nur unter Ausnutzung der Berechnungsvollständigkeit verwirklicht werden, indem die Strukturen der Architektur

[9] Sämtliche in Kapitel 2. betrachteten Architekturen sind PSS im Sinne der gegebenen Definition.

entgegen ihrer vorgesehenen Funktion verwendet werden. Die beiden letzten Fälle lassen sich zwar nicht so eindeutig voneinander abgrenzen, wie sie sich von der ersten Möglichkeit unterscheiden, im konkreten Einzelfall ist jedoch meist eine klare Unterscheidung möglich.

1.2.2. Motivation für kognitive Architekturen

Kognitive Architekturen dienen neben einem besseren allgemeinen Verständnis von Intelligenz zwei Zwecken: der psychologischen Modellbildung und der Entwicklung intelligenter technischer Anwendungen.

Architekturen sind von psychologischem Interesse, da sie einerseits aussagekräftige Theorien über die menschliche Kognition als Gesamtheit ermöglichen und andererseits einen größeren theoretischen Rahmen zur Formulierung detaillierter kognitiver Modelle darstellen:

> ”Psychology has arrived at the possibility of unified theories of cognition – theories that gain their power by positing a single system of mechanisms that operate together to produce the full range of human cognition.“ (Newell 1990, S. 1)

Außerdem kann die Architektur der menschlichen Kognition als weitgehend identisch für alle Menschen angenommen werden, während individuelle Unterschiede auf Differenzen im Wissen und in den Zielen des einzelnen Menschen zurückgeführt werden können. Beispiele für kognitive Architekturen, die als psychologische Modelle zur Erklärung menschlichen Verhaltens entwickelt bzw. verwendet wurden, sind ACT und SOAR (s. Kapitel 2.2. und 2.4.).

Bei der Entwicklung intelligenter technischer Anwendungen dienen kognitive Architekturen als Rahmen zur Gestaltung von Agenten. Agenten sind Computersysteme, die intelligent und weitgehend eigenständig handeln können:

> ”Agents are autonomous or semi-autonomous hardware or software systems that perform tasks in complex, dynamically changing environments.“ (Müller 1996, S. 1)

Architekturen legen dabei die Grundcharakteristika eines Agenten fest und bilden zudem eine Grundlage zur Programmierung seines Verhaltens. Beispiele für Architekturen, die zur Gestaltung technischer Agenten entwickelt wurden, sind INTERRAP und TOK/PRODIGY (s. Kapitel 2.1. und 2.3.).

1.2.3. Arten kognitiver Architekturen

Die vielen vorgeschlagenen Modelle für kognitive Architekturen[10] lassen sich in vier Grundansätze unterteilen:

1. integrierende Architekturen,
2. geschichtete Architekturen,
3. modulare Architekturen und
4. konnektionistische Architekturen.

Viele Architekturen sind allerdings von mehr als nur einem dieser Ansätze beeinflußt. Beispielsweise sind alle in Kapitel 2. vorgestellten Architekturen integrierende Architekturen, INTERRAP ist jedoch zugleich eine geschichtete Architektur, PRODIGY verwendet mehrere voneinander unabhängige Lernmodule, und ACT berücksichtigt konnektionistische Prinzipien.

Integrierende Architekturen bestehen aus einzelnen funktionalen Komponenten, die in einer Gesamtarchitektur integriert sind. Jede Komponente erfüllt eine bestimmte Funktion innerhalb der Gesamtarchitektur – beispielsweise Wahrnehmung, Speichern oder Problemlösen –, und die Architektur kontrolliert die Interaktion und den Informationsaustausch zwischen den einzelnen Komponenten. Die integrierten Komponenten sind dabei eher unselbständig, die Kontrolle wird zentral durch die Architektur gesteuert, wobei meist eine strenge Reihenfolge ausgehend von der Wahrnehmung über die zentrale Kognition hin zur Handlung eingehalten wird.

Verschiedene Typen integrierender Architekturen sind beispielsweise hierarchische Architekturen, Kaskadenarchitekturen, Blackboard-Architekturen oder Produktionensysteme (Strube et al., Kapitel 4. in Görz 1995, S. 308). Bei hierarchischen Architekturen wird die gesamte Verarbeitung zentral durch ein ”Supervisor“-Modul gesteuert. Kaskadenarchitekturen verarbeiten eingehende Information schrittweise durch aufeinanderfolgende Komponenten, die nur mit den direkten Nachbarkomponenten interagieren, bis hin zur Ausgabe. Ein zentraler gemeinsamer Datenspeicher, auf dem weitgehend autonome Teilsysteme operieren, ist der Grundgedanke von Blackboard-Architekturen. Produktionensysteme besitzen einen Interpreter für Produktionen, der auf einem zentralen Arbeitsspeicher operiert.

Geschichtete Architekturen bestehen aus mehreren eigenständigen Ebenen, die jeweils ihre eigene funktionale Gliederung aufweisen. Die einzelnen Schichten können dabei entweder seriell hintereinander, parallel zueinander oder hierarchisch angeordnet sein[11]. Serielle Ebenenarchitekturen verbinden Ein- und Ausgabe durch aufeinanderfolgende Schichten, während in parallelen Architekturen sämtliche Schichten mit den Ein- und Ausgabekomponen-

[10] Laird 1991 gibt einen Überblick über verschiedene kognitive Architekturen, darunter auch SOAR und PRODIGY.

[11] Eine Diskussion verschiedener Arten von Ebenenarchitekturen einschließlich INTERRAP findet sich in Müller et al. (1995).

ten verbunden sind. Bei hierarchischen Ebenenarchitekturen ist eine Schicht direkt mit den Wahrnehmungs- und Ausführungseinheiten verbunden, über der hierarchisch weitere Schichten angeordnet sind, die die jeweils darunterliegenden kontrollieren. Die Interaktion zwischen den Ebenen geschieht durch Informationsaustausch mit den Nachbarebenen oder durch Mechanismen zur Unterdrückung der Informationsaufnahme und zur Hemmung der Ausführung durch andere Ebenen. Repräsentation und Informationsspeicherung können zentral oder für jede Ebene einzeln vorgenommen werden, wobei jede Ebene nur die für sie relevanten Aspekte der Welt zu verarbeiten braucht. Subsumptionsarchitekturen (Brooks 1986, 1991) verzichten völlig auf eine explizite Repräsentation der Welt, die Funktionalität des Gesamtsystems ergibt sich aus der Interaktion paralleler Ebenen, die Wahrnehmung und Ausführung direkt in Form einzelner Verhaltensweisen miteinander verbinden.

Modulare Architekturen – auch als Multi-Agenten-Modelle (Minsky 1986) bezeichnet – gehen von einer Vielzahl eigenständiger spezialisierter Module aus. Anders als bei integrierenden Architekturen führen diese nicht allgemeine, sondern sehr spezifische Funktionen aus, die höhere Funktionalität des Gesamtsystems ergibt sich erst aus der Interaktion der lokalen Teilsysteme. Die Aufgabe der Architektur besteht nicht mehr in einer zentralen Kontrolle der Module, sondern in deren (Selbst-)Organisation.

Konnektionistische Architekturen (Rumelhart 1989) orientieren sich am Aufbau menschlicher und tierischer Gehirne. Sie bestehen aus Netzstrukturen vieler gleichartiger Einheiten, den sogenannten Neuronen, die sehr einfach aufgebaut und auf keine bestimmte Funktion spezialisiert sind. Die kleinsten funktionalen Einheiten konnektionistischer Architekturen bestehen dann aus Systemen mit mehreren Neuronen, deren funktionale Organisation sich vor allem aus den Verbindungen zwischen den einzelnen Neuronen ergibt. Die einzelnen Teilsysteme und Repräsentationen sind verteilt und können sich auch überlagern. Das Verhalten des Gesamtsystems ergibt sich aus den parallelen Interaktionen der Neuronen.

1.3. Die Struktur integrierender Architekturen

Da ein detaillierter struktureller Vergleich zwischen Architekturen verschiedenen Typs nur schwer möglich ist, werden zum Vergleich mit INTERRAP nur integrierende Architekturen herangezogen. Dabei können einerseits die einzelnen Komponenten – soweit sie eine ähnliche Funktion ausüben – einander gegenübergestellt werden und andererseits die Gesamtstruktur der Architektur, die in der Integration der Komponenten und in der Kontrolle der Interaktionen zwischen diesen besteht.

Gemäß dem Informationsverarbeitungsansatz besteht die Funktion kognitiver Prozesse in der Transformation von Information. Eine Architektur ist deshalb charakterisiert durch die Arten an Information, die transformiert werden können, die Möglichkeiten der Speicherung von Information sowie die verschiedenen Informationsverarbeitungsprozesse zur Transforma-

tion der gespeicherten Information. Integrierende Architekturen verwenden eigene Komponenten sowohl zur Speicherung der einzelnen Informationsarten, als auch für verschiedene Informationsverarbeitungsprozesse. Nicht zuletzt ist zudem die Organisation der Komponenten durch die Architektur zu berücksichtigen. Diese Aspekte integrierender Architekturen werden nun eingehender diskutiert.

1.3.1. Informationsarten

Die durch eine kognitive Architektur verarbeiteten Informationen können sich in folgender Hinsicht unterscheiden:

1. in der Art, wie die Information verwendet werden kann,
2. in dem Gehalt, der durch eine Information repräsentiert wird,
3. in dem Grad der Abstraktion, den eine Information besitzt, und
4. in der Funktion, die eine Information ausübt.

Diese verschiedenen Aspekte können innerhalb einer Architektur direkt durch eigene Speicher- und Verarbeitungskomponenten oder indirekt durch verschiedene Speicherungs- und Transformationsprozesse unterstützt werden.

In der Verwendung von Information wird zwischen prozeduralem und deklarativem Wissen unterschieden. Prozedurale Information beinhaltet kognitive und motorische Fertigkeiten und kann von der Architektur direkt ausgeführt werden. Deklarative Information stellt Repräsentationen der Welt in Form von Fakten dar und kann nur anhand von prozeduraler Interpretation genutzt werden, die durch prozedurales Wissen oder die Architektur selbst vorgenommen werden kann. Deklarative Information ist episodisch, wenn sie von konkreten Einzelereignissen handelt, und semantisch, wenn sie allgemeine und mehr kontextunabhängige Tatsachen beschreibt.

Der semantische Gehalt einer Information besteht in der Repräsentation der Welt bei deklarativem Wissen und in kognitiven und motorischen Prozessen bei prozeduralem Wissen. Bei deklarativer Information wird je nach Inhalt zwischen verschiedenen Repräsentationsarten für zeitliche, räumliche und propositionale Information unterschieden (Anderson 1983, S. 45). Propositionale Information beschreibt Zustände und Eigenschaften von Objekten sowie Relationen zwischen Objekten.

Information kann auf verschiedenen Abstraktionsebenen repräsentiert werden. Analoge Repräsentationen entsprechen strukturell den sensorischen Daten der Wahrnehmung. Kategoriale Repräsentationen gliedern diese Information nach gegebenen Kategorien mit gemeinsamen Eigenschaften. Kategorien dienen der Generalisierung und der Differenzierung der aufgenommenen Information, da durch sie gleichartige Objekte zu Klassen zusammengefaßt und verschiedenartige Objekte voneinander abgegrenzt werden (Wessells 1984, S. 212). Kategorien sind um so abstrakter, je mehr individuelle Eigenschaften vernachlässigt werden

zugunsten einiger weniger, für die Kategorie wesentlicher Eigenschaften. Propositionale Repräsentationen beschreiben Zusammenhänge zwischen mehreren (kategorialen) Objekten.

Mentale Zustände unterscheiden sich nicht nur in ihrem Gehalt, sondern auch in ihrer Funktion innerhalb der Architektur, je nach ihrer Rolle in verschiedenen kognitiven Prozessen. Informationen werden anders verwendet, je nachdem ob sie tatsächliche, mögliche oder angestrebte Zustände der Welt repräsentieren. Überzeugungen, Wissen, Glaube oder Meinungen repräsentieren gegenwärtige oder vergangene Zustände, Vorstellungen und Fiktionen mögliche oder zukünftige Zustände der Welt. Information darüber, wie die Welt sein soll (oder nicht sein soll), wird durch Ziele, Wünsche, Intentionen, Absichten, Hoffnungen oder Pläne repräsentiert. Die Funktion all dieser Repräsentationen liegt in der Steuerung des Verhaltens – jedoch auf verschiedene Weise. Die Repräsentation angestrebter Weltzustände leitet kognitive Prozesse auf das Erreichen dieser Zustände hin, während die Repräsentation tatsächlicher Weltzustände der Bestimmung von Handlungsmöglichkeiten und -notwendigkeiten in der jeweiligen Situation dient. Mögliche Weltzustände dienen der Bewertung und dem Vergleich verschiedener Handlungsmöglichkeiten.

1.3.2. Informationsspeicherung

Damit Information verwendet werden kann, muß sie auch vorhanden und deshalb innerhalb des Systems gespeichert sein. Dabei ist die prinzipiell vorhandene Information von der für einen bestimmten Prozeß direkt verfügbaren zu unterscheiden. Langzeitspeicher enthalten Information, die längerfristig oder sogar unbegrenzt erhalten bleibt, ihr Fassungsvermögen wird meist als unbegrenzt oder zumindest als ausreichend groß angenommen. Da solche längerfristigen Speicher sehr viel mehr Information enthalten, als während eines bestimmten Informationsverarbeitungsprozesses benötigt wird und verwendet werden kann, wird in kurzfristigen Speichern die direkt verwendbare Information (zwischen)gespeichert. Diese Arbeitsspeicher sind in ihrer Kapazität oft stark beschränkt und enthalten Information nur, solange sie gebraucht wird. Da ”langfristig“ sich auf die Dauer der Speicherung, ”kurzfristig“ jedoch auf die Verfügbarkeit bezieht, kann auch ein einziges Speichermodul sowohl die langfristige, als auch die kurzfristige Speicherung von Information übernehmen.

Beide Arten der Informationsspeicherung können innerhalb einer Architektur zentral durch einen Hauptspeicher oder lokal für einzelne Komponenten organisiert sein. Des weiteren können getrennte Speichermodule mit eigenen Abruf- und Speichermechanismen für verschiedene Arten von Information verwendet werden.

Neben den einzelnen Speicherkomponenten ist die Organisation und Strukturierung der Information im Speicher von Bedeutung. Da der Langzeitspeicher große Mengen an Information enthält, muß diese so organisiert werden, daß ein schneller Abruf relevanter Informationen möglich ist. Durch Assoziation werden Speicherinhalte miteinander verbunden, die in einem Bezug zueinander stehen und oft zusammen benötigt werden. Dadurch kann nicht nur rele-

vante Information schnell im Speicher aufgefunden werden, die Geschwindigkeit des Abrufs dient gleichzeitig als Maß für die Relevanz einer Information in der gegebenen Situation (Anderson 1993, S. 51).

Kategoriale Information kann zusätzlich durch Begriffstaxonomien strukturiert werden. Dazu werden Kategorien anhand von Klasseninklusionen hierarchisch angeordnet. Eine Kategorie ist einer anderen übergeordnet, wenn sie sämtliche Elemente der untergeordneten Kategorie enthält. Je höher eine Kategorie in dieser Hierarchie steht, desto abstrakter und allgemeiner ist sie und desto weniger gemeinsame Eigenschaften besitzen deren Elemente. Besonders häufig verwendet werden die Kategorien der basalen Ebene, die sowohl einen hohen Informationsgehalt besitzen, da ihre Elemente viele gemeinsame Eigenschaften untereinander besitzen und nur wenige mit Elementen anderer Kategorien teilen, als auch abstrakt genug sind, um keine überflüssige Information zu enthalten (Wessells 1984, S. 222).

1.3.3. Informationsverarbeitungsprozesse

Abbildung 1 stellt schematisch die grundlegenden Informationsverarbeitungsprozesse und den Informationsfluß zwischen diesen dar. Information muß aufgenommen, gespeichert, aus dem Speicher abgerufen, transformiert und schließlich für Handlungen genutzt werden können. Daraus ergeben sich fünf fundamentale Informationsverarbeitungsprozesse zur Steuerung des Verhaltens anhand der gespeicherten Information ausgehend von der Sensorik hin zur Motorik:

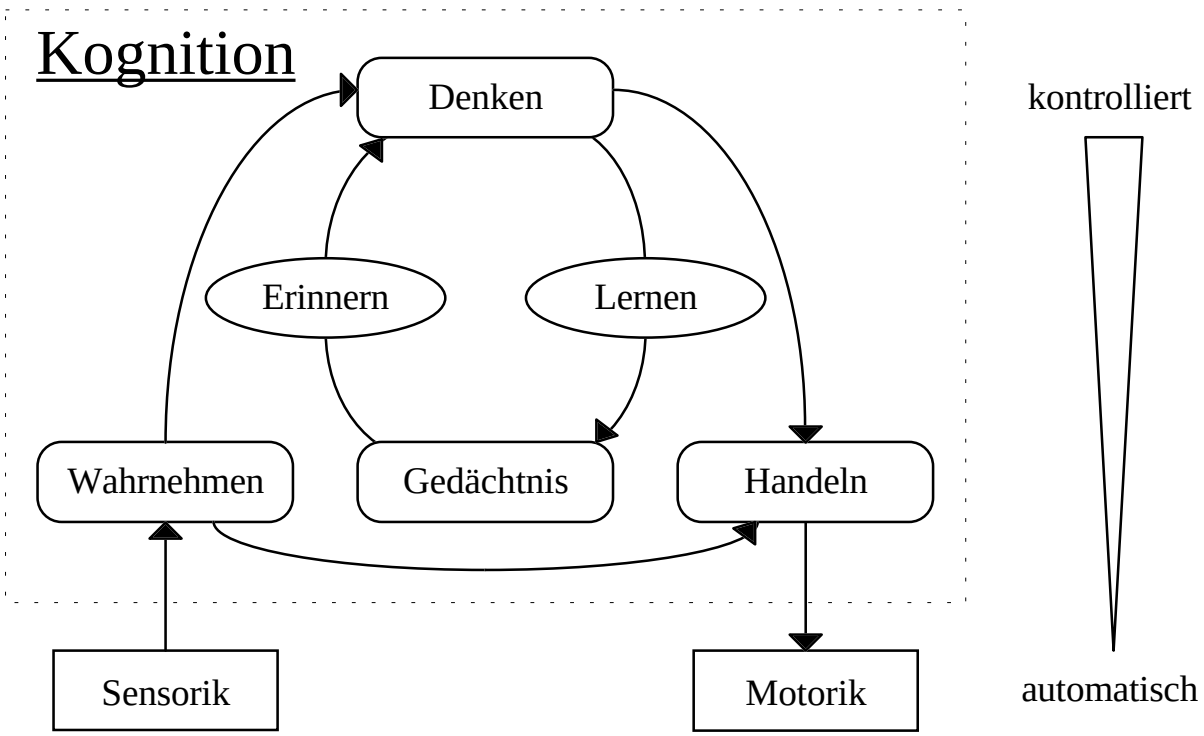

Abbildung 1: Schematische Darstellung der Grundstruktur kognitiver Architekturen

1. Wahrnehmen,
2. Handeln,
3. Lernen,
4. Erinnern und
5. (höhere) Denkprozesse.

In integrierenden Architekturen wird jede dieser Funktionen durch eine oder mehrere zusammengehörige Komponenten realisiert.

Wahrnehmung und Bewegung dienen der Interaktion mit der Umgebung. Diese kann durch Stimulierung der Sensoren bzw. durch physische Veränderungen der Welt geschehen, oder sie besteht – beispielsweise bei Software-Agenten – im Austausch von Information durch Empfangen und Senden von Nachrichten. Zur Wahrnehmung gehört nicht nur die reine Aufnahme von sensorischen Daten und kommunizierter Information, sie umfaßt auch deren Abstraktion, Selektion und Interpretation im aktuellen Kontext. Ebenso besteht die Ausführung von Handlungen nicht nur in einer rein motorischen Umsetzung, sondern auch in deren Koordinierung und Überwachung.

Derartige Prozesse werden vor allem durch die verarbeiteten Daten gesteuert und geschehen eher automatisch. Höhere Denkprozesse sind kontrolliert, da sie weniger von den aufgenommenen Daten als von den Zielen des Agenten abhängen. Zu den höheren Denkprozessen gehören das Treffen von Entscheidungen, induktive und deduktive Schlußfolgerungen sowie das Lösen von Problemen (Barsalou 1992, S. 277). Entscheiden bedeutet die zielgerichtete Auswahl aus Alternativen und Problemlösen die Auswahl von Handlungen zum Erreichen von Zielen, beispielsweise durch Aufstellen eines Plans. Deduktive Schlüsse sind sichere logische Konsequenzen des momentanen Wissens und induktive Schlüsse plausible Verallgemeinerungen aus Einzelfällen (Barsalou 1992, S. 293, S. 311).

Lernen und Erinnern dienen der Nutzung eines längerfristigen Informationsspeichers. Lernen beinhaltet neben der eigentlichen Aufnahme der Information in den Speicher auch deren Kodierung, Strukturierung und Eingliederung in das vorhandene Wissen, so daß die Erinnerung relevante Information schnell und zuverlässig aus dem Speicher wieder abrufen kann.

1.3.4. Die Architektur: Integration der Komponenten

Bei der Betrachtung der Gesamtarchitektur sind drei Aspekte zu berücksichtigen. Die eigentliche Architektur besteht zum einen aus ihren verschiedenen Teilsystemen, zum zweiten aus den Interaktionsmöglichkeiten zwischen diesen sowie zum dritten aus der Kontrolle dieser Interaktionen durch die Architektur.

Die Teilsysteme integrierender Architekturen sind ihre integrierten funktionalen Komponenten. Dies sind entweder Speicherelemente oder Module für bestimmte Informationsverarbeitungsprozesse, wie sie bereits besprochen wurden.

Die Struktur einer Architektur ergibt sich aus den verschiedenen Möglichkeiten der Interaktion zwischen den einzelnen Komponenten. Die Informationsverarbeitungsmodule können untereinander Information austauschen, andere Prozesse überwachen sowie sich gegenseitig kontrollieren, indem andere Prozesse aufgerufen, aktiviert, unterbrochen oder beendet werden. Des weiteren können sie Information in einem Speicher abrufen, speichern oder auch löschen.

Die Architektur kontrolliert den Ablauf der kognitiven Prozesse nicht nur anhand des Informationsflusses, indem sie die Struktur dieser Interaktionsmöglichkeiten vorgibt, sondern auch indem sie die Abfolge einzelner Informationsverarbeitungsschritte innerhalb eines Kontrollzyklus festlegt, der beständig durchlaufen wird.

1.3.5. Zusammenfassung

Das Verhalten einer integrierenden kognitiven Architektur hängt sowohl von den *integrierten Komponenten*, als auch von der *Integration der Komponenten* ab. Die integrierten Komponenten sind Speicher zur Aufbewahrung sowie Prozesse zur Verwendung verschiedener Arten von Information. Die Integration der Komponenten bestimmt die Kontrolle der Prozesse und den Informationsaustausch zwischen den Komponenten.

Die folgende Liste zeigt zusammenfassend ein Schema zur Analyse der Struktur integrierender Architekturen:

1) *Informationsarten*
   - Verwendung: deklarativ/prozedural
   - Repräsentationsarten: verschiedenartige Inhalte
   - Abstraktionsebenen: analog, kategorial, propositional
   - funktionale Rolle: Funktion in kognitiven Prozessen

2) *Informationsspeicher*
   - langfristiger Speicher: prinzipiell vorhandene Information
   - kurzfristiger Speicher: direkt verwendbare Information
   - Speicherstruktur: zentral/lokal, Speicher für bestimmte Informationsarten
   - Organisation und Strukturierung der Information im Speicher

3) *Informationsverarbeitungsprozesse*
   - Wahrnehmung: Kodieren von Information über Welt
   - Bewegung: Ausführen von Handlungsabsichten
   - Lernen: Speichern und Strukturieren von Information
   - Erinnern: Abruf von relevanter Information
   - höhere Denkprozesse: zielgerichtete Transformation von Repräsentationen

4) *Architektur*
   - Teilsysteme der Architektur: integrierte Komponenten
   - Struktur der Architektur: Interaktion der einzelnen Komponenten
   - Kontrolle: Organisation der kognitiven Prozesse und des Informationsflusses

Dieses Schema wird in Kapitel 3. als Grundlage für den strukturellen Vergleich von INTERRAP mit den Architekturen ACT, TOK/PRODIGY und SOAR verwendet, wobei sowohl die einzelnen Komponenten, als auch die Gesamtstruktur der Architektur berücksichtigt werden.

1.4. Die funktionale Analyse kognitiver Architekturen

Neben der internen Struktur ist das äußere Verhalten als Gesamtsystem, das sich aus dieser Struktur ergibt, charakterisierend für eine kognitive Architektur. An das Verhalten kognitiver Systeme werden zahlreiche funktionale Anforderungen gestellt, sowohl um technische Ansprüche zu erfüllen (Laird 1991, S. 12), als auch um als Modell für menschliche Kognition dienen zu können (Newell 1990, S. 19). Diese Anforderungen betreffen entweder das *kognitive Spektrum* – das Bereitstellen bestimmter kognitiver Fähigkeiten – oder die *Intelligenz* – Erfolg und Qualität des Verhaltens – des Systems. Da derartige Anforderungen weniger die im System vorhandene Information, als deren *Verwendung* durch das System betreffen, ist es Aufgabe der Architektur, Mechanismen bereitzustellen, die den verschiedenen Ansprüchen gerecht werden.

1.4.1. Das kognitive Spektrum

Die zentrale Grundanforderung an ein kognitives System ist, daß es sich in realen Situationen angemessen verhalten kann. Dazu muß es in Echtzeit in komplexen, sich dynamisch verändernden Umgebungen agieren können, die ebenso reichhaltig wie detailliert sind. Um unter solchen Bedingungen erfolgreich zu sein, bedarf es zum einen der Möglichkeit einer flexiblen und umfangreichen Interaktion mit der Umgebung, was durch ein detailliertes Wahrnehmungssytem zusammen mit einem Bewegungssystem mit großen Freiheitsgraden gewährleistet wird, sowie zum anderen einer effizienten und situationsgerechten Verarbeitung einer großen Menge an Information (Newell 1990, S. 19). Da die im folgenden betrachteten Architekturen recht abstrakt und auf zentralere Aspekte der Kognition konzentriert sind, sind ihre Wahrnehmungs- und Bewegungssysteme meist weitgehend unspezifiziert, so daß diese die Basis darstellen, mit der die Architekturen durch Informationsaustausch interagieren.

Für komplexe, dynamische Umgebungen sind nicht nur hoch entwickelte Interaktionsmöglichkeiten nötig, sondern auch verschiedenartige Verhaltensweisen, um Handlungen der gegebenen Situation entsprechend auswählen und ausführen zu können. Im einzelnen muß ein

kognitives Sytem reaktives, deliberatives, adaptives und soziales Verhalten zeigen können (Müller 1996, S. 2).

Um in unerwarteten und dringenden Situationen angemessen zu handeln, müssen jederzeit schnelle und dennoch situationsgerechte Reaktionen möglich sein. Außerdem sind derartige Situationen rechtzeitig und zuverlässig zu erkennen.

Deliberatives Verhalten beinhaltet die Fähigkeit zu ziel- und zweckgerichtetem Verhalten auch bei komplexen und schwierigen Aufgaben. Dazu müssen Pläne für Handlungsfolgen erstellt werden, indem Entscheidungen getroffen und Handlungen entsprechend den verfügbaren Ressourcen und den angestrebten Zwecken ausgewählt werden.

Die Anpassung des Verhaltens an die Umwelt geschieht zunächst bereits bei der Konstruktion eines kognitiven Systems. Gerade in dynamischen Umgebungen können jedoch nicht immer alle Möglichkeiten vorhergesehen werden, so daß auch spätere Adaptation durch das System selbst unverzichtbar ist. Ein kognitives System sollte deshalb sowohl aus Beobachtungen und Beispielen anderer, als auch aus eigener Erfahrung und eigenen Fehlern lernen und neues Wissen erwerben können. Weiterhin muß auch mit grundlegenden Veränderungen der Umgebung umgegangen werden können.

Ein besonderer Teil der Umgebung sind unter Umständen andere kognitive Agenten mit eigenem Wissen und eigenen Zielen. Die Interaktion mit diesen unterscheidet sich wesentlich von der mit anderen Teilen der Umgebung. Einerseits sind sie ein äußerst dynamischer Teil der Umgebung, dessen zukünftiges Verhalten oft nur schwer abzuschätzen und vorherzusehen ist, andererseits besteht durch Kommunikation die Möglichkeit des Informationsaustauschs und einer direkten Beeinflussung des Verhaltens anderer Agenten. So können Aufgaben auch gemeinsam durch Kooperation und Koordinierung von Handlungen ausgeführt werden, es müssen aber auch zusätzlich die Ziele anderer Agenten berücksichtigt werden.

1.4.2. Das Verhaltensspektrum

Bereits in Kapitel 1.1.2. wurde ein kurzer Überblick über die Intelligenz des Verhaltens im Sinne von erfolgreicher Kognition und zweckrationalem Handeln gegeben und Intelligenz durch zielgerichtetes Problemlösen in Verbindung mit Rationalität, Universalität, Flexibilität und Lernfähigkeit charakterisiert. Diese knappe Darstellung wird nun etwas systematischer vertieft. Dabei sind die verschiedenen Anforderungen nicht auf das deliberative Verhalten beschränkt – auch wenn dieses für Intelligenz zentral ist –, sondern betreffen auch reaktives, adaptives und soziales Verhalten bzw. das Verhalten des Systems als Ganzem.

Handlungen sind rational, wenn sie den Zwecken des Agenten möglichst optimal dienlich sind: ”*optimization* means maximizing the probability of achieving these goals while minimizing the cost of doing so”, oder kurz: “maximizing the expected *utility*“ (Anderson 1993, S. 47). Dazu müssen sie geeignet sein, das angestrebte Ziel auch tatsächlich erreichen zu können, und (zumindest weitgehend) vereinbar mit dem momentanen Wissen und den übrigen Zielen des

Agenten sein. Da ein geringer Aufwand an Zeit, Energie und anderen Ressourcen immer sinnvoll und erstrebenswert ist, sollten die Handlungen zudem möglichst optimal, effizient und effektiv sein. Um ein rationales Verhalten zu erreichen, müssen deshalb die Fähigkeiten des logischen Schlußfolgerns – für konsistente Handlungen – sowie des effizienten und zweckgerichteten Planens – für effiziente Handlungen – durch eine kognitive Architektur unterstützt werden.

Ein ebenso wichtiges Kriterium für die Qualität eines Systems ist neben der Rationalität seine Robustheit (Müller 1996, S. 1). Ein kognitiver Agent muß nicht nur Handlungen für seine Ziele rational auswählen, sondern sie auch erfolgreich umsetzen können. Dabei sind einerseits Fehler in der Planung und unvorhergesehene Ereignisse bei der Ausführung zu erkennen sowie schnell und der neuen Situation angemessen zu beheben. Andererseits muß mit der in komplexen und dynamischen Umgebungen unvermeidlichen Unvollständigkeit und Unsicherheit des momentanen Wissens umgegangen werden können.

Weiterhin sollte ein kognitives System nicht zu sehr auf eine bestimmte Umgebung und bestimmte (fest vorprogrammierte) Verhaltensweisen festgelegt sein, sondern flexibel und vielseitig sein, um eine möglichst hohe Universalität zu erreichen. Die Flexibilität eines Systems beinhaltet die Fähigkeit, für bekannte Arten von Aufgaben neuartige Lösungen finden zu können, wozu ein breites Spektrum an alternativen Repräsentationen und Verhaltensstrategien vorhanden sein muß. Vielseitigkeit bedeutet, daß ein System sich auch in völlig neuen Problembereichen zurechtfinden und Lösungen für neuartige Aufgabenstellungen finden kann. Dazu sind neben Flexibilität auch Kreativität und Lernfähigkeit nötig. Gerade Flexibilität, Vielseitigkeit und Universalität eines kognitiven Systems sind jedoch recht schwer zu beurteilen, da sie nicht nur von der Architektur, sondern auch stark von der Qualität wie auch der Quantität des vorhandenen Wissen eines Agenten abhängen.

1.4.3. Zusammenfassung

Die Tabellen 5 und 6 (s. Anhang) fassen die funktionalen Anforderungen an kognitive Architekturen – sowohl in Hinsicht auf deren kognitives Spektrum, als auch in Hinsicht auf deren Verhaltensspektrum – zusammen. Um ein breites kognitives Spektrum an Verwendbarkeit und Universalität abzudecken, muß ein kognitives System reaktives, deliberatives, adaptives und soziales Verhalten zeigen können. Damit dieses Verhalten erfolgreich seinen Zweck erfüllen kann, benötigt ein Agent Rationalität, Robustheit, Flexibilität, Vielseitigkeit und ein hohes Maß an Universalität.

Inwieweit und wodurch die Architekturen INTERRAP, ACT, TOK/PRODIGY und SOAR diese Anforderungen an die Funktionalität erfüllen, ist Thema des vierten Kapitels. Dabei liegt der Schwerpunkt in der Untersuchung der Funktionalität von INTERRAP und in der Gegenüberstellung der Mechanismen der einzelnen Architekturen, die die verschiedenen Arten von Funktionalität gewährleisten, jedoch nicht in einer vergleichenden Bewertung der Architekturen.

2. MODELLE KOGNITIVER ARCHITEKTUREN

Da die einzelnen Komponenten integrierender Architekturen sich gegenseitig ergänzen und bedingen, kann ein detaillierter Vergleich verschiedener Architekturen, wie er Gegenstand dieser Arbeit ist, nur durchgeführt werden auf Grundlage der umfassenden Rahmenstrukturen der Gesamtarchitekturen. Deshalb werden in diesem Kapitel zunächst der Aufbau und die Prinzipien der Architekturen INTERRAP, ACT, TOK/PRODIGY und SOAR in ihrer Grundstruktur vorgestellt, um dann während der Gegenüberstellung in den beiden folgenden Kapiteln noch eingehender vertieft zu werden.

In den Architekturen spiegeln sich auch die verschiedenen Ansätze und die Motivation zu ihrer Entwicklung wider. So ist ACT (Anderson 1983, 1993) als ein Modell der menschlichen Kognition stark den experimentellen Ergebnissen und den empirisch gestützten Theorien der Kognitionspsychologie verpflichtet. Deshalb wurden verschiedene kognitionspsychologische Modelle wie Produktionensysteme und assoziative Netze in ACT zu einer Gesamtarchitektur vereinigt, wobei zentrale Aspekte der menschlichen Kognition, insbesondere kognitive Fertigkeiten wie zielgerichtetes Denken und Problemlösen, den Schwerpunkt bilden.

INTERRAP (Fischer et al. 1995, Müller 1996) stellt einen mehr an praktischen technischen Anwendungen orientierten Ansatz zur Modellierung einzelner Agenten und von Agentengesellschaften dar. Um ein kohärentes Verhalten in realen Umgebungen erreichen zu können, werden reaktives, deliberatives und kooperatives Verhalten in einer geschichteten BDI-Architektur miteinander verbunden.

Ebenfalls pragmatisch ausgerichtete Ansätze liegen TOK und PRODIGY (Bates et al. 1992, Blythe & Reilly 1993, Veloso et al. 1995) zugrunde. TOK dient der Modellierung einfacher autonomer Agenten für simulierte Welten, die als glaubwürdige Charaktere auch mit menschlichen Besuchern dieser Welt interagieren sollen. Während HAP, der Planer von TOK, rein reaktiv ausgelegt ist, kann er für komplexere Planungen auf den deliberativen Planer PRODIGY zurückgreifen. PRODIGY ist ein klassisches KI-Planungssystem, das verschiedene Lernkomponenten zur Effizienzsteigerung benutzen kann.

SOAR (Laird et al. 1987, Newell 1990) basiert auf der Konzeption von Intelligenz als effizienter heuristischer Suche in einem Problemraum. Kern von SOAR ist dementsprechend ein Problemraumzustand, der durch in Produktionen gespeichertes Wissen mittels Anwendung von Suchoperatoren auf ein Ziel hin verändert wird. Die Effizienz dieses Suchprozesses wird durch permanentes automatisches Lernen verbessert.

2.1. INTERRAP: BDI und Ebenenmodell

INTERRAP (**Inte**gration of **R**eactive Behaviour and **Ra**tional **P**lanning) stellt einen pragmatisch orientierten Ansatz zur Gestaltung komplexer Agentengesellschaften dar, in denen mehrere Agenten in dynamischen Umgebungen unter realistischen Bedingungen miteinander interagieren können: "INTERRAP is a pragmatic approach to designing complex dynamic agent societies." (Fischer et al. 1995, Abstract). In realen Situationen steht neben der Rationalität eines Systems vor allem dessen Robustheit im Vordergrund (Müller 1996, S. 26). Die Robustheit von INTERRAP-Agenten wird durch eine vertikal geschichtete Architektur erreicht, ihre Rationalität basiert auf der Realisierung dieser Ebenen in Form einer BDI-Architektur (vgl. Rao & Georgeff 1991).

2.1.1. Das Ebenenmodell

Integration von reaktivem, deliberativem und kooperativem Verhalten

Der Grundgedanke von INTERRAP ist – wie auch schon der Name (s.o.) besagt – die Integration von reaktiven Verhaltensmustern mit rationalen Planungsvorgängen (Müller & Pischel 1993, S. vi). Fest vorgegebene Verhaltensmuster, die direkt ausführbar sind, ermöglichen effizientes Verhalten in bekannten Situationen und Flexibilität durch Reaktionen auf unvorhergesehene Veränderungen in der Umgebung. Andererseits dient Planung aufgrund einer expliziten symbolischen Repräsentation der Umwelt der intelligenten Lösung komplexer Problemstellungen, die durch direkt ausgeführte Verhaltensmuster nicht effizient handhabbar sind. Die Unflexibilität abstrakter Pläne, die nicht alle möglichen Fehler und Umweltänderungen detailliert berücksichtigen können, wird durch Verhaltensmuster ausgeglichen, die für Reaktionen und Standardhandlungen vordefiniert sind, für sich genommen jedoch nicht die Effizienz rationaler Planung erreichen können. Rationale Planung und Verhaltensmuster ergänzen sich somit gegenseitig.

Für Gesellschaften mit mehreren Agenten integriert INTERRAP zusätzlich Mechanismen für kooperatives Handeln. Die Interaktion mit anderen kognitiven Agenten unterscheidet sich wesentlich von der mit der übrigen Umgebung, so daß eine Erweiterung der Architektur um

Kommunikation zwischen Agenten und gemeinsames Erstellen und Ausführen von Plänen sinnvoll ist.

Geschichtete Architekturen

Diese drei Arten von Verhalten werden in INTERRAP durch miteinander interagierende, nebenläufige Ebenen realisiert. Die funktionale Gliederung einer Architektur in parallele Ebenen bringt für die Integration reaktiven, deliberativen und kooperativen Verhaltens mehrere Vorteile (Müller 1996, S. 38).

Zum einen ermöglicht die Modularisierung anhand einzelner Ebenen eine deutliche Trennung verschiedener Funktionen, die zudem über klar definierte Schnittstellen miteinander verbunden sind. Dies sorgt für mehr Übersichtlichkeit und vereinfacht die Gestaltung von Agenten für bestimmte Aufgaben.

Die getrennten Ebenen erhöhen des weiteren die Robustheit eines Agenten, da Reaktivität und Planung voneinander unabhängig sind. Die Überwachung der Umgebung nach Ereignissen, die direkte Reaktionen erfordern, wird nicht durch langwierige und rechenintensive Planungsvorgänge verlangsamt, und Planungen werden nicht durch plötzliche Reaktionen unterbrochen.

Schließlich kann auch das Wissen eines Agenten hierarchisch organisiert werden, so daß jede Ebene nur die Information verarbeiten muß, die für sie relevant ist.

Die Grundstruktur: Wissensbasis, Kontrolleinheit und Weltschnittstelle

Ein INTERRAP-Agent (Abbildung 2) besteht aus drei Modulen, einer Weltschnittstelle, einer hierarchischen Wissensbasis und einer geschichteten Kontrolleinheit:

> ”An INTERRAP agent consists of a knowledge base, a world interface, and three hierarchically organized control layers: (i) the behavior-based layer provides basic reactiv behavior and procedural knowledge for routine tasks; (ii) the local planning layer incorporates the agents capabilities to do means-ends reasoning to achieve its local goals; (iii) the cooperative planning layer holds facilities allowing an agent to coordinate its actions with other agents through the negotiation of plans.“ (Müller 1996, S. 128)

Die Weltschnittstelle (World Interface, WIF) dient der Interaktion mit der Umgebung. Sie stellt Funktionen zur Überwachung der Sensorik, zur Steuerung der Motorik und zur Kommunikation mit anderen Agenten bereit.

Wissensbasis und Kontrolleinheit sind entsprechend den drei Arten von Verhalten in Ebenen für reaktives, deliberatives und kooperatives Verhalten unterteilt. Die Kontrolleinheit (Control Unit, CU) dient der Steuerung des Verhaltens des Agenten anhand des in der Wissensbasis gespeicherten Wissens. Sie gliedert sich in eine verhaltensbasierte Ebene (Behavior-Based Level, BBL) für schnelle Reaktionen und Routineaufgaben, in eine lokale Planungsebene (Local Planning Level, LPL) für zielgerichtetes Planen und Entscheiden und in eine koopera-

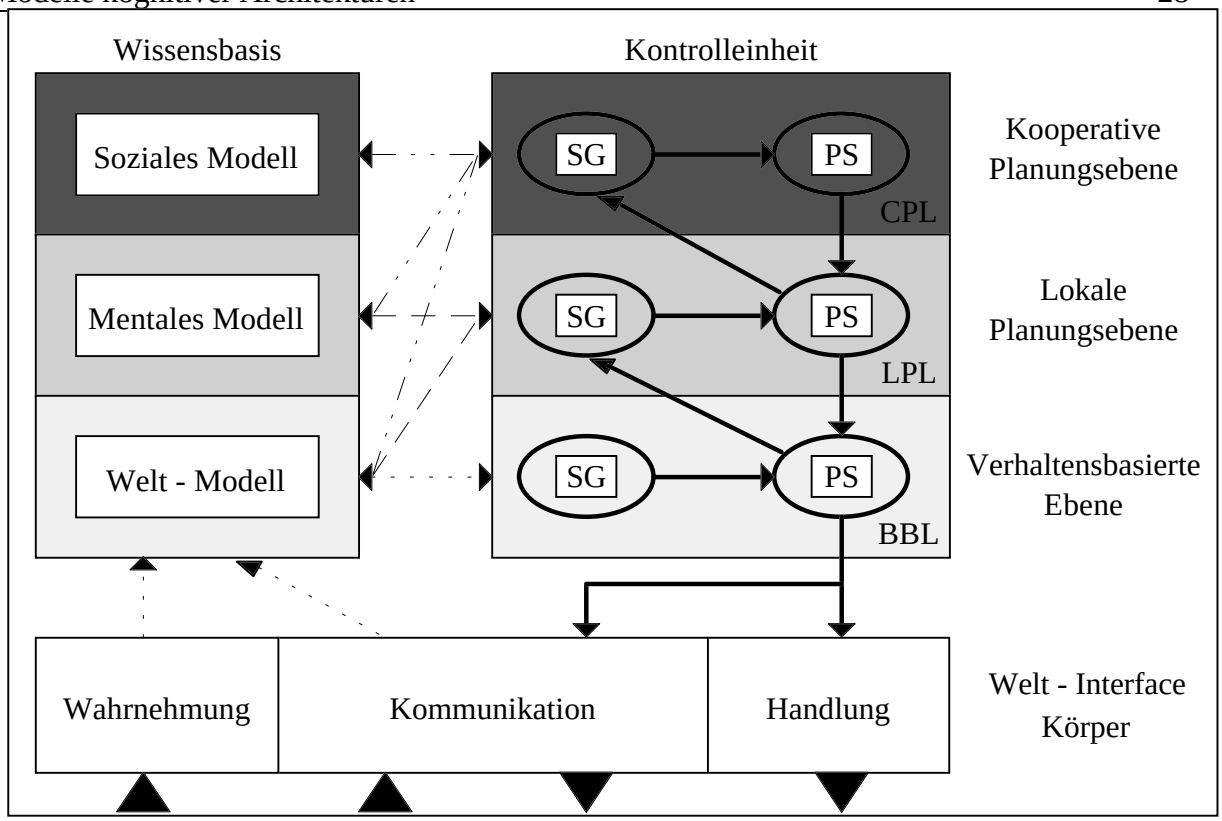

Abbildung 2: Die Architektur INTERRAP (Fischer et al. 1995, S. 3)

tive Planungsebene (Cooperative Planning Level, CPL) für die Koordinierung von Handlungen mit anderen Agenten. Die Interaktion dieser Ebenen untereinander geschieht durch eine einheitliche Kontrollstruktur (s.u.), wobei die Kontrolle über die Weltschnittstelle und damit über das (äußere) Verhalten des Agenten bei der BBL liegt.

Die Wissensbasis (Knowledge Base, KB) enthält das Wissen eines Agenten über die Welt. Es besteht aus einem Weltmodell (World Model, WM), das den momentanen Weltzustand repräsentiert, einem mentalen Modell (Mental Model, MM), das den mentalen Zustand des Agenten umfaßt, sowie einem sozialen Modell (Social Model, SM), das Wissen über andere Agenten enthält. Jede Ebene der Wissensbasis enthält dabei das Wissen, das von der zugehörigen Kontrollebene (und den höheren) benötigt wird bzw. verwendet werden kann. Zugleich bildet diese hierarchische Strukturierung verschiedene Abstraktions- und Komplexitätsebenen an Wissen.

Ein INTERRAP-Agent ist somit eine Menge paralleler funktionaler Ebenen, die durch eine aktivierungsbasierte Kontrollstruktur und eine gemeinsame hierarchische Wissensbasis miteinander verbunden sind (Müller 1995, S. 2). Die hierarchische Schichtung bildet verschiedene Ebenen an Abstraktion, Wissenskomplexität, Verantwortung und Funktionalität.

Kompetenzgesteuerte Kontrolle zwischen den Ebenen

Die Interaktion zwischen den einzelnen Kontrollebenen geschieht kompetenzgesteuert. Jede Ebene entscheidet, ob sie in einer gegebenen Situation bzw. bei einer gegebenen Aufgabe kompetent ist, um zu reagieren oder das Problem zu lösen. Ist sie es nicht, so wird die Problemstellung an die nächsthöhere Ebene weitergeleitet, so lange bis eine kompetente Ebene gefunden ist. (Fischer et al. 1995, S. 6; Müller 1996, S. 64f)

Dazu wird eingehende Information der Wahrnehmungskomponente der Weltschnittstelle an die unterste Ebene, die BBL, weitergeleitet. Diese analysiert die Situation anhand der aktualisierten Beschreibung der Welt sowie des eigenen internen Zustands und entscheidet über ihre Kompetenz, in dieser Situation aktiv werden zu können. Kann sie auf eine Aufgabe nicht angemessen reagieren, so sendet sie eine Problembeschreibung an die nächsthöhere Ebene mit der Aufforderung, die Aufgabe zu lösen. Erhält eine Ebene eine derartige Aktivierungsaufforderung (upward activation request), so wird die Problemstellung um Wissen erweitert, das auf der darunterliegenden Ebene nicht verfügbar war. Dann beurteilt die Ebene aufgrund dieser erweiterten Situationsbeschreibung ihre Kompetenz und wird entweder selbst aktiv oder leitet das Problem erneut weiter. Dieser Vorgang aus Erweiterung der Situationsbeschreibung, Kompetenzbeurteilung und Weitergabe von Information und Kontrolle nach oben wird wiederholt, bis eine Ebene sich für kompetent erachtet und die gegebene Problemstellung bearbeitet. Dabei braucht jede Ebene nur ihre eigene Kompetenz und nicht die der anderen Ebenen zu kennen, da höhere Ebenen prinzipiell auch eine höhere Problemlösekompetenz besitzen.

Hat eine Ebene ein Problem gelöst und sich für eine Handlung oder eine Folge mehrerer Handlungen entschieden, so wird diese an die nächsttiefere Ebene als Ausführungsverpflichtung weitergeleitet (downward commitment posting). Diese Ebene koordiniert die beabsichtigte Handlung der höheren Ebene mit ihren eigenen Absichten oder gibt sie, falls dies nicht möglich ist, an die darüberliegende Ebene zur Neuplanung zurück. Bei erfolgreicher Koordinierung wird die Handlung eine Verpflichtung dieser Ebene, die sie schließlich zur Ausführung an die Ebene unter ihr sendet. Auch dieser Vorgang wird wiederholt, so lange bis die unterste Ebene, die BBL, erreicht ist. Diese wählt nun ihrerseits Handlungen aus und übergibt sie an die Ausführungskomponente der Weltschnittstelle, wo sie in reale Handlungen umgesetzt werden.

Somit entstehen zwei Richtungen des Kontrollflusses. Problemstellungen werden von unten nach oben geleitet, bis eine kompetente Ebene sie lösen kann. Die Handlungsabsichten, die sich aus der Problemlösung ergeben, werden von oben nach unten weitergeleitet, sofern sie mit den Absichten der unteren Ebenen vereinbar sind. Diese Art der Interaktion zwischen den Ebenen ermöglicht ein dynamisches Verhalten und Problemlösen, indem für jede Problemstellung die effizienteste kompetente Ebene gefunden wird:

> ”competence-based control flow ... ensures that situations requiring quick reaction are handled by the behavior-based layer, whereas other situations that encode more

> complex planning problems are shifted upward until a layer has been reached which is competent for solving the problem.“ (Müller 1996, S. 64)

Die Reaktivität eines INTERRAP-Agenten wird dabei gewährleistet, indem die Kontrolle über die tatsächlich ausgeführten Handlungen letztendlich ausschießlich bei der reaktiven BBL liegt, da sie als einzige direkt mit der Weltschnittstelle interagiert.

Die Ebenenstruktur von INTERRAP

Die Ebenen-Architektur von INTERRAP beruht auf vier grundlegenden Gestaltungsprinzipien: (Müller 1996, S. 53)

- eine geschichtete Kontrolle (layered control),
- eine geschichtete Wissensbasis (layered knowledge base),
- die Aktivierung von unten nach oben (bottom-up activation) und
- die Ausführung von oben nach unten (top-down execution).

Die geschichtete Kontrolle ermöglicht die Integration von reaktivem, deliberativem und kooperativem Verhalten sowie die funktionale Gliederung der Architektur auf verschiedenen Abstraktions- und Komplexitätsebenen. Parallel dazu wird auch die Wissensbasis hierarchisch angeordnet, wodurch die Menge der auf jeder einzelnen Ebene verfügbaren und zu verarbeitenden Information verringert wird. Die Kontrolle über eine Situation wird von unten nach oben weitergereicht, um eine kompetente Ebene zu finden, die diese Problemstellung möglichst effizient verarbeiten kann. Die Ausführung von Handlungen geschieht von oben nach unten, so daß die abstrakten Handlungen höherer Ebenen durch die unteren Ebenen detailliert ausgeführt und überwacht werden können.

Durch diese Gestaltungsprinzipien soll die Konstruktion von Agenten ermöglicht werden, die unter realistischen Bedingungen sowohl intelligent Aufgaben ausführen können, als auch flexibel auf Änderungen der Umgebung reagieren können.

2.1.2. Realisierung von BDI innerhalb des Ebenenmodells

BDI-Architekturen

BDI-Architekturen (Bratman et al. 1987) charakterisieren einen Agenten anhand bestimmter mentaler Kategorien, namentlich *Überzeugung* (**B**elief), *Wunsch* (**D**esire) und *Absicht* (**I**ntention). Dieses Konzept wurde für praktische Zwecke um weitere Kategorien wie *Ziel* (Goal) und *Plan* erweitert. Die Verwendung dieser Kategorien dient einerseits zur Etablierung einer formalen Logik (Cohen & Levesque 1990), die als normative Rationalitätsanforderung an BDI-Agenten aufgefaßt werden kann. Andererseits dient sie – wie bei INTERRAP – als konzeptuelle Grundlage zur Entwicklung von Agenten-Architekturen: ”INTERRAP adopts the BDI-model rather in a conceptual than in a strictly theoretical sense.“ (Fischer et al. 1995, S. 2).

Die Überzeugungen beinhalten die Annahmen eines Agenten über den aktuellen Zustand der Welt, umfassen aber auch Annahmen über die möglichen Auswirkungen von Handlungen und deren Wahrscheinlichkeiten. Wünsche stellen Präferenzen für zukünftige Weltzustände und Ereignisse dar. Während diese Wünsche nicht immer erreichbar und auch nicht untereinander konsistent sein brauchen, sind die Ziele eine konsistente Teilmenge dieser Wünsche, von denen der Agent glaubt, daß er alle tatsächlich erreichen kann. Die Absichten sind eine weitere Einschränkung der Wünsche, sie umfassen ausgewählte Ziele mit der Verpflichtung zu Plänen (Handlungssequenzen), die diese Ziele erreichen. (Müller 1996, S. 18)

Eine Verpflichtung zu bestimmten Zielen oder Handlungen besagt, daß die entsprechenden Handlungen tatsächlich ausgeführt werden. Dabei kann zwischen verschiedenen Arten von Einstellungen gegenüber den Verpflichtungen unterschieden werden je nachdem, unter welchen Bedingungen eine Verpflichtung wieder aufgegeben werden darf (Müller 1996, S. 22). Die einfachste Möglichkeit besteht in ”fanatischen“ Agenten (blind commitment), die an einer Absicht festhalten, bis sie erfüllt ist. Etwas realistischer sind ”engstirnige“ Agenten (single-minded commitment), die auch dann eine Absicht beenden, wenn sie als nicht erfüllbar angesehen wird. Flexibler sind ”aufgeschlossene“ Agenten (open-minded commitment), die Verpflichtungen aufgeben, sobald sie kein Ziel des Agenten mehr darstellen.

Zu den mentalen Kategorien des BDI-Konzepts gesellen sich in INTERRAP noch weitere (Müller 1996, S. 47). Die *Wahrnehmungen* (perception) umfassen den aktuellen Zustand der Sensoren der Weltschnittstelle. *Situationen* sind strukturierte Ausschnitte aus der Gesamtinformation der Wissensbasis, die Klassen von relevanten Weltzuständen beschreiben. Sie dienen der Beurteilung der Kompetenz der einzelnen Ebenen und der Auswahl von Handlungsmöglichkeiten. Ziele werden in INTERRAP kontextunabhängig aufgefaßt, während *Optionen* eine situationsabhängige Teilmenge dieser Ziele umfassen und als Pläne repräsentiert werden. Die Absichten sind in INTERRAP diejenigen Optionen, die als auszuführende Handlungen ausgewählt werden und zu deren Ausführung der Agent damit verpflichtet ist. Des weiteren gibt es in INTERRAP *Handlungsprimitive* (operational primitives, OP), die als Planoperatoren verwendet werden und das Bindeglied zwischen den Motiven und den Entscheidungen des Agenten darstellen.

BDI in einer Ebenenarchitektur

Wie in Abbildung 3 ersichtlich ist, sind die mentalen Kategorien des BDI-Konzepts in INTERRAP geschichtet realisiert, wobei diese jeweils entsprechend der Funktionalität der Ebene instantiiert sind. (Müller 1996, S. 48).

Die Überzeugungen in der Wissensbasis sind unterteilt in das Wissen des Agenten über die Umgebung (Weltmodell), über sich selbst (mentales Modell) und über andere Agenten (soziales Modell). Die Situationen stellen auf jeder Ebene Teilmengen des auf dieser Ebene verfügbaren Teils der Wissensbasis dar. So beschreiben die Verhaltenssituationen (behavioral

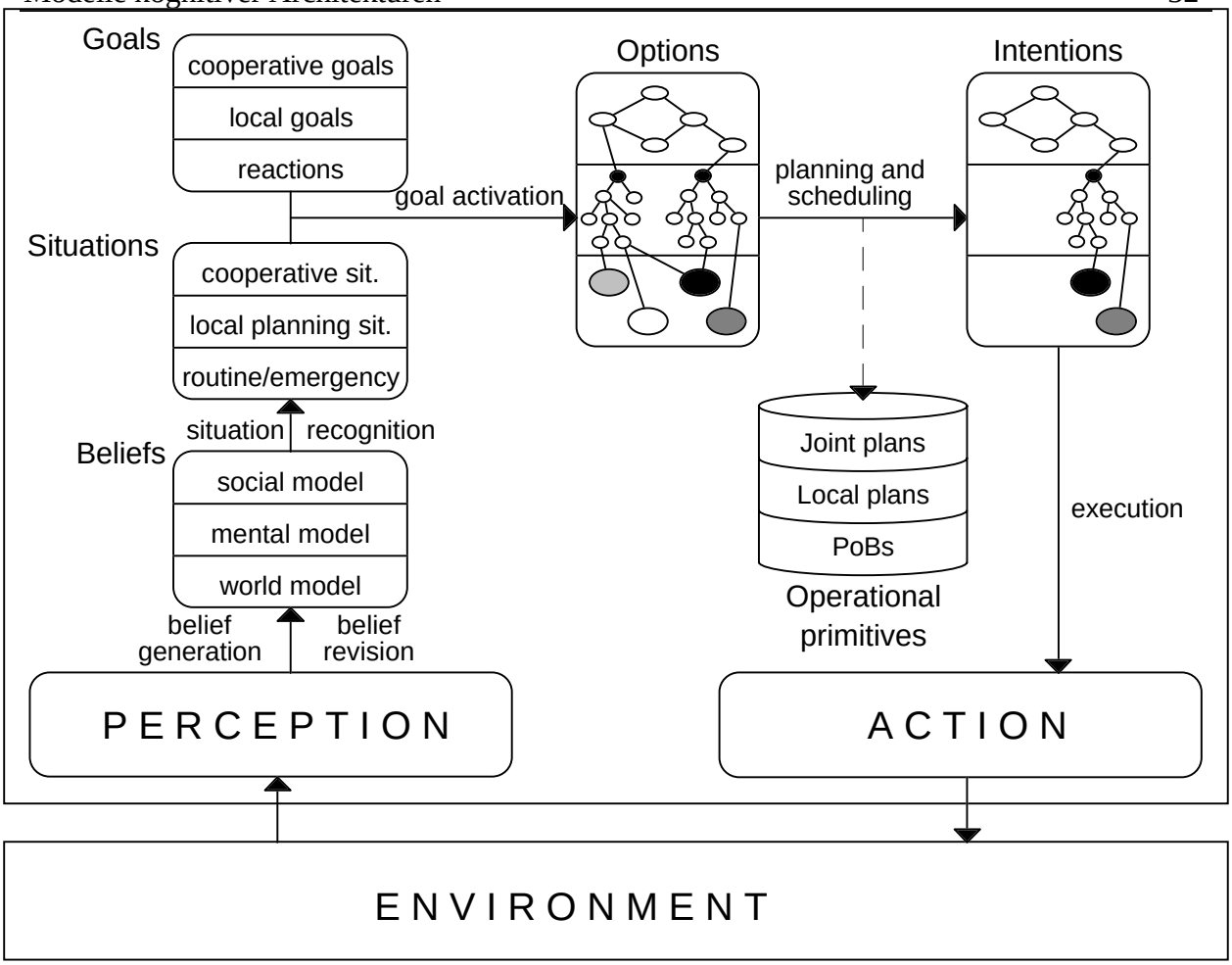

Abbildung 3: Das konzeptuelle Agentenmodell INTERRAP (Müller 1996, S. 48)

situations) der BBL mögliche Inhalte des Weltmodells zur Erkennung von Situationen, die auf dieser Ebene angemessen verarbeitet werden können, z.B. dringende Reaktionen auf äußere Umstände. Lokale Planungssituationen (local planning situations) umfassen zusätzlich Informationen des mentalen Modells und ermöglichen somit das Beurteilen der Kompetenz der LPL bzw. die Möglichkeit lokaler Planung als Antwort auf eine gegeben Situation. Auf der CPL treten zu Informationen des Weltmodells und des mentalen Modells noch die des sozialen Modells hinzu und beschreiben dann Situationen für kooperative Planung (cooperative situations).

Die Ziele der BBL sind kurzfristig und werden direkt durch externe Ereignisse ausgelöst. Sie umfassen beispielsweise Reaktionen, die die physische Unversehrtheit des Agenten in Gefahrensituationen erhalten sollen. Auf der LPL finden sich Ziele im eigentlichen Sinn der BDI-Konzeption, und die Ziele der CPL stellen gemeinsame Ziele mehrerer Agenten oder Zielkonflikte zwischen Agenten dar. Auch Handlungsprimitive und Absichten sind geschichtet organisiert in einfache Verhaltensmuster (pattern of behaviour, PoB) sowie lokale und gemeinsame Pläne.

Der Aufbau einer INTERRAP-Ebene

Die einzelnen INTERRAP-Ebenen besitzen einen gemeinsamen Aufbau (vgl. Abbildung 2 und Abbildung 4). Jede Ebene besteht aus drei Funktionen, die ähnliche Aufgaben innerhalb der Ebenen erfüllen, jedoch jeweils eine eigene Wissenskomplexität und Abstraktheit, sowie eine unterschiedliche Funktionalität für die Gesamtarchitektur besitzen:

> "The INTERRAP agent architecture implements three basic functions:
>
> - *BR*: $P \times B \rightarrow B$ is a belief revision and knowledge abstraction function, mapping an agent's current perception and its old beliefs into a set of new beliefs.
> - *SG*: $B \times G \rightarrow G$ is a situation recognition and goal activation function, deriving new goals from the agent's new beliefs and its current goals.
> - *PS*: $B \times G \times I \rightarrow I$ is a planning and scheduling function, deriving a set of new intentions (commitments to courses of action) based on the goals selected by *SG* and the current intentional structure of the agent." (Fischer et al. 1995, S. 4)

(Abkürzungen: *P*erception, *B*eliefs, *G*oals, *I*ntentions)

Die erste Funktion, *BR*, dient der Aktualisierung der Wissensbasis. Ausgehend vom bisherigen Wissen und der momentanen sensorischen Information wird das alte Wissen überprüft und gegebenenfalls aktualisiert sowie neues Wissen hinzugefügt. Des weiteren kann aus dem so erhaltenen Wissen durch logische Schlüsse zusätzlich abstrakteres und komplexeres Wissen hergeleitet werden.

Die zweite Funktion, *SG*, untersucht den durch *BR* aktualisierten Wissensstand auf vordefinierte Situationen hin, die die Kompetenzen der einzelnen Ebenen festlegen. Mit diesen Situationen sind Ziele verbunden, die eine angemessene Antwort auf die jeweilige Situation darstellen. Diese neuen Ziele werden mit den bereits vorhandenen zu den aktuellen Optionen (aktive Ziele) des Agenten vereinigt.

	BBL	LPL	CPL
BR	generation and revision of belief (world model)	abstraction of local beliefs (mental model)	maintaining models of other agents (social model)
SG	activation of reactor patterns	recognition of situations requiring local planning	recognition of situations requiring cooperative planning
PS	reactor PoB: direct link from situations to action sequences	modifying local intentions; local planning	modifying joint intentions; cooperative planning

Tabelle 1: Die Kontrollfunktionen der INTERRAP-Ebenen (Fischer et al. 1995, S. 5)

Die dritte Funktion, *PS*, leitet aus den so gewonnenen Optionen (und aus den bereits vorhandenen Intentionen) eine Menge neuer Absichten des Agenten her. Diese Funktion entscheidet also anhand der verschiedenen situationsbedingten Handlungsoptionen über das tatsächliche Verhalten des Agenten, indem sie für bestimmte Handlungen Verpflichtungen eingeht (Absichten sind gerade Ziele mit Verpflichtung, s.o.). Des weiteren organisiert *PS* auch die genaue Abfolge verschiedener Handlungen und sorgt für deren Ausführung.

Diese drei Funktionen bilden auf jeder Ebene die Verbindung zwischen der Wahrnehmung und dem internen mentalen Zustand eines Agenten auf der einen Seite und dessen aktivem Verhalten auf der anderen Seite. Sie sind von ihrer Grundkonzeption und in ihrer Rolle innerhalb der Ebene für jede Ebene gleich, sind aber jeweils entsprechend der Funktionalität der Ebene anders instantiiert (Tabelle 1). (Die Details dieser Instantiierung folgen weiter unten.)

Der Kontrollzyklus einer INTERRAP-Ebene

Die beschriebenen Funktionen bilden auch den Kontrollzyklus (Abbildung 4) innerhalb jeder Ebene. Dieser besteht aus vier Stufen: sensorische Wahrnehmung, Erkennen der Situation, Entscheiden und Ausführen (”sense-recognize-decide-act cycle“, Müller 1996, S. 61).

Zunächst wird die sensorische Wahrnehmung in die Wissensbasis aufgenommen und dadurch zu Überzeugungen des Agenten. Außerdem werden alte Überzeugungen anhand der neuen Wahrnehmungsinformation aktualisiert und gegebenenfalls revidiert.

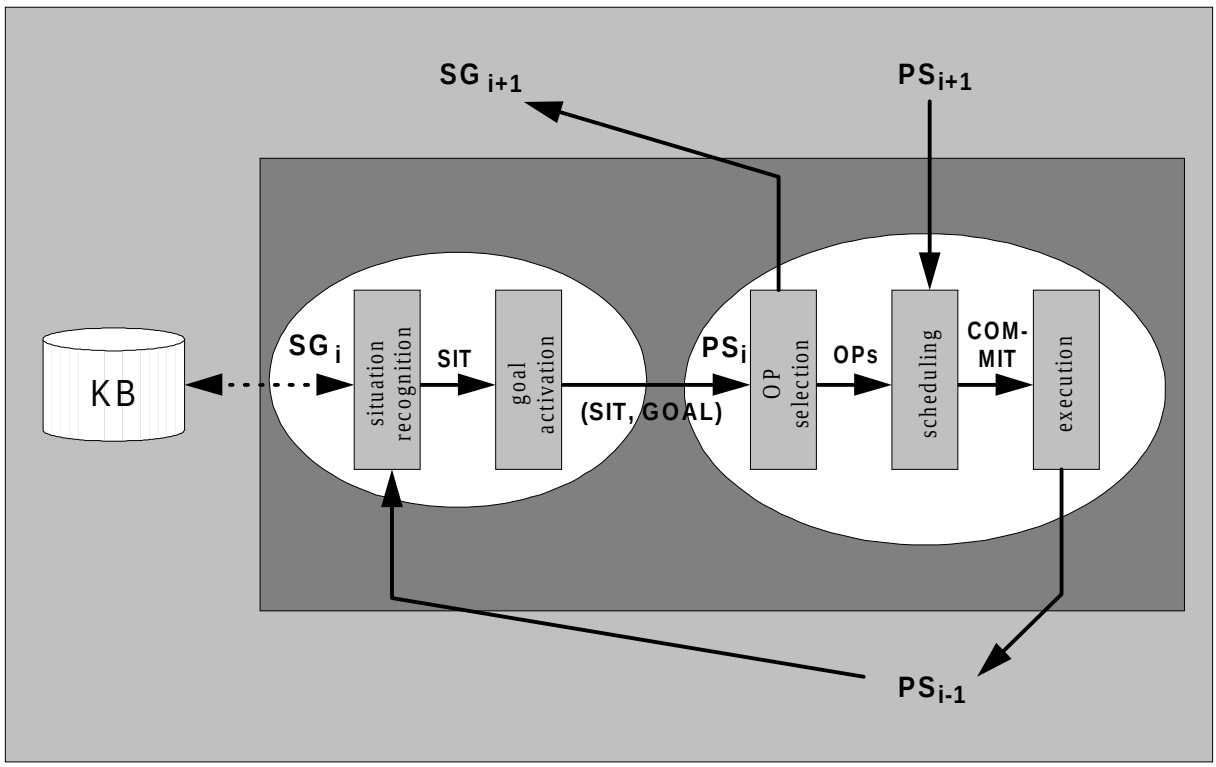

Abbildung 4: Der Kontrollzyklus innerhalb einer Ebene von INTERRAP (Müller 1996, S. 61)

Daraufhin wird die Wissensbasis auf neue relevante Situationen hin untersucht (situation recognition), wodurch die zunächst unstrukturierten Überzeugungen strukturiert werden. Diese Situationen ergeben zusammen mit den Nachrichten, die von der darunterliegende Ebene als Aktivierungsanforderungen gesendet – und gegebenenfalls um neue Information erweitert – wurden, die Menge der neuen Optionen (goal activation).

Im nächsten Schritt entscheidet der Planer, für welche Ziele die Ebene kompetent ist. Kann sie ein Ziel nicht bearbeiten, so wird es mittels einer Nachricht an die darüberliegende Ebene weitergeleitet. Andernfalls wird das Ziel durch den Planer bearbeitet, indem eine Handlungssequenz erstellt wird, deren Ausführung dieses Ziel erfüllen kann (OP selection). Dies ist auf der untersten Ebene, der BBL, ein fest vorgegebenes Verhaltensmuster, auf den Planungsebenen, LPL und CPL, wird entweder ein Plan aus einer Planbibliothek abgerufen oder ein neuer Plan durch Mittel-Zweck-Analyse erstellt.

Diese ausgewählten Handlungssequenzen und die Nachrichten von der nächsthöheren Ebene werden nun, wenn möglich, in die vorhandene Intentionsstruktur der Ebene integriert und der genaue Handlungsablauf wird festgelegt (scheduling). Andernfalls werden sie zur neuerlichen Bearbeitung an den Planer bzw. die höhere Ebene zurückgegeben. Schließlich werden die so entstandenen Absichten ausgeführt bzw. zur Ausführung an die nächsttiefere Ebene übergeben (execution).

Damit ist jede INTERRAP-Ebene und auch der gesamte Agent definiert über seine Fähigkeit, Situationen wahrzunehmen, daraus Ziele abzuleiten und diese durch Handlungen zu erreichen:

> ”The agent has the ability to recognize certain classes of situations, to derive goals from these situations, and to perform actions in order to achieve these goals.“ (Müller 1996, S. 12)

2.1.3. Die verhaltensbasierte Ebene

Die Aufgaben der BBL

Die verhaltensbasierte Ebene kontrolliert die Ausführung der Intentionen eines INTERRAP-Agenten, die dessen (motorisches) Verhalten in der Interaktion mit der Umgebung bestimmen. Dafür stellt sie Verhaltensmuster für sofortige Reaktionen und für Routineaufgaben bereit:

> ”The behavior-based layer serves two purposes: firstly, it incorporates the reactive abilities of the agent, which allow it to deal in real-time with a class of emergency situations. Secondly, it provides the agent’s procedural knowledge needed to perform routine tasks efficiently.“ (Müller 1996, S. 67)

Die BBL interagiert dazu mit drei Modulen: der Weltschnittstelle, der Wissensbasis und der LPL. Bei der Weltschnittstelle kann sensorische Information abgerufen und die Ausführung von Handlungen initiiert werden. In der Wissensbasis kann Information des Weltmodells

genutzt werden, um Situationen zu erkennen. Die Interaktion mit der LPL geschieht in zwei Richtungen. Zum einen werden Nachrichten von der BBL an die LPL gesendet, wenn diese Situationen erkennt, für die sie selbst nicht kompetent ist, damit die LPL einen Plan zum Erreichen des mit der Situation verknüpften Ziels erstellt. Zum anderen empfängt die BBL Handlungsverpflichtungen von der LPL, die sie wenn möglich ausführt oder zur Neuplanung zurücksendet.

Pattern of Behaviour

Die grundlegende (prozedurale) Datenstruktur der BBL sind Pattern of Behaviour (PoB)[12]. PoB bestehen aus zwei Teilen, einem Beschreibungsteil und einem Ausführungsteil (Müller 1996, S. 69). Der Beschreibungsteil enthält Information zur Aktivierung und Kontrolle der PoB, der Ausführungsteil besteht aus einem ausführbaren Programmcode.

Ein PoB kann drei verschiedene Zustände einnehmen: er kann inaktiv oder aktiv sein, und er kann ausgeführt werden. Ein PoB wird aktiviert, wenn die im Beschreibungsteil angegebenen Aktivierungsbedingungen erfüllt sind. Da aufgrund von Ressourcenkonflikten nicht alle in einem Schritt des Kontrollzyklus aktiven PoB gleichzeitig ausgeführt werden können, wird anhand von statischen Prioritäten[13] entschieden, welche aktiven PoB auch tatsächlich ausgeführt werden. Die Prioritäten stehen wie die benötigten Ressourcen im Beschreibungsteil des PoB. Die Aktivierung eines PoB bleibt so lange erhalten, bis die ebenfalls im Beschreibungsteil angegebenen Bedingungen für Erfolg oder Mißerfolg des PoB erfüllt sind.

Der Ausführungsteil eines PoB besteht aus einem direkt ausführbaren Programmcode, der in einzelne atomare Instruktionen unterteilt ist, die schrittweise abgearbeitet werden. Da in jedem Schritt des Kontrollzyklus nur eine atomare Instruktion pro ausgeführtem PoB abgearbeitet wird, können PoB jederzeit unterbrochen oder beendet werden. Zudem erhöht dies die Reaktivität des Agenten, da der Kontrollzyklus der BBL, die für die Reaktionen zuständig ist, nicht durch lange Programmausführungen blockiert wird. PoB können jedoch in sofern nur als Ganzes ausgeführt werden, als ihre interne Struktur nicht explizit repräsentiert ist und deshalb nicht bei Planungsvorgängen berücksichtigt werden kann.

Entsprechend der Funktionalität der BBL gibt es zwei Arten von PoB, reactor PoB und procedure PoB. Reactor PoB werden direkt durch externe Ereignisse bzw. deren Repräsentation im Weltmodell ausgelöst und ermöglichen so schnelle und flexible Reaktionen auf Änderungen der Umwelt. Um jederzeit Reaktivität zu gewährleisten, besitzen die reactor PoB höhere Prioritäten als die procedure PoB. Reactor PoB können auch dazu verwendet werden, planungsbedürftige Situationen zu erkennen und an die LPL zu übertragen (control modifier).

[12] Die Pattern of Behaviour in INTERRAP sind vom Prinzip her Produktionen ähnlich, wie sie in Produktionensystemen wie OPS5, aber auch in ACT und SOAR (s.u.) verwendet werden. PoB erweitern die einfachen Bedingung/Aktions-Paare der Produktionen jedoch um weitere Funktionen.

[13] Prinzipiell ist auch die Verwendung dynamischer Prioritäten möglich, die jedem PoB einen zeitabhängigen Erfülltheitsgrad zuordnen. (Müller et al. 1995, S. 10)

Zudem können reactor PoB den Zustand der Wissensbasis intern ändern und so für Deduktion und Wissensabstraktion genutzt werden (knowledge modifier).

Procedure PoB stellen die abstrakten Handlungen dar, die die höheren Ebenen bei der Planung als Operatoren verwenden. Sie umfassen nicht nur primitive Handlungen, sondern können auch Programme (kompilierte Pläne) für häufig durchgeführte Routineaufgaben bereitstellen. Procedure PoB werden aufgrund von Nachrichten der LPL aktiviert und können durch diese Ebene auch außerplanmäßig wieder deaktiviert werden.

Der Kontrollzyklus der BBL

Der allgemeine Kontrollzyklus der INTERRAP-Ebenen wird von der BBL folgendermaßen instantiiert (Müller 1996, S. 78). *BR* beschränkt sich auf das Aktualisieren der Wahrnehmung durch Aufruf von Wahrnehmungsmethoden der Weltschnittstelle. Dabei wird bei widersprüchlicher Information jeweils die neuere Information als zutreffend erachtet und die ältere gelöscht. Hinzu kommen noch die Nachrichten, die von der LPL empfangen wurden.

Die Situationserkennung beschränkt sich auf das Weltmodell und auf Situationen, die in den Aktivierungsbedingungen der PoB fest vordefiniert sind. Die Zielaktivierung besteht im Wechsel des Zustandes des PoB von inaktiv zu aktiv.

Die Planung geschieht ebenfalls in einem einzigen Schritt, da die Handlungssequenzen im Ausführungsteil der PoB direkt vorgegeben sind. Das Scheduling koordiniert die Ausführung der aktiven PoB, indem es aus diesen eine konsistente Menge gleichzeitig ausführbarer PoB erstellt. Dabei werden die statischen Prioritäten der PoB und dynamisch erkannte Ressourcenkonflikte berücksichtigt. Die Ausführung dieser konsistenten Menge von PoB kann direkt erfolgen, indem der Ausführungsteil der PoB abgearbeitet wird.

Die Instantiierung der drei Kontrollfunktionen in der BBL ist vor allem auf schnelle und effiziente Verarbeitung hin ausgerichtet, um jederzeit eine hohe Reaktivität zu gewährleisten. Deshalb ist der Wissenszugriff auf das Weltmodell der Wissensbasis beschränkt und die Zahl der zu erkennenden Situationen durch die Menge der Aktivierungsbedingungen der PoB begrenzt. Hierfür sind effiziente Algorithmen bekannt (z.B. RETE-Netze). Die direkte Verbindung von Situationsbeschreibungen mit Zielen und Intentionen in den PoB ermöglicht direktes Handeln ohne zeitaufwendiges Planen.

2.1.4. Die lokale Planungsebene

Der Aufbau der LPL

Die LPL übernimmt die Planung für Situationen, in denen die BBL keine fest vorgegebene Reaktion parat hat und die keine Interaktionen mit anderen Agenten verlangen. Das Verhalten der LPL wird durch ein Kontrollmodul bestimmt, das den Austausch von Nachrichten mit den benachbarten Ebenen übernimmt und den Kontrollzyklus ausführt (Müller 1996, S. 84). Die

weiteren Module dieser Ebene – *plan generator*, *plan library*, *plan evaluator*, *plan interpreter*, *plan scheduler* und *plan executor* – dienen der Generierung, der Auswahl und der Ausführung von lokalen Plänen (Pläne, die durch einen einzelnen Agenten ausgeführt werden).

Das Kontrollmodul der LPL interagiert mit der BBL, der CPL und der Wissensbasis. Von der BBL werden Planungsanforderungen empfangen für Situationen, die die BBL zwar erkannt, aber für die sie sich nicht als kompetent betrachtet hat, oder Planungsanforderungen werden wieder zurückgenommen. Bei Empfang einer Planungsanforderung wird ein neues Ziel für die Planung erstellt, oder sie wird an die CPL weitergeleitet, falls die LPL nicht kompetent ist. Von der CPL empfangene Ausführungsanweisungen werden in die intentionale Struktur der LPL aufgenommen oder, falls dies nicht möglich ist, zur Neuplanung an die CPL zurückgegeben. Zur Ausführung der eigenen Intentionen werden Verpflichtungen an die BBL gesendet und von dieser Meldungen bei Beendung oder Unterbrechung der Ausführung empfangen.

Die LPL hat Zugriff auf das Weltmodell und das mentale Modell der Wissensbasis. Das mentale Modell enthält Pläne, Ziele und Intentionen des Agenten, es gehört aber nur konzeptuell zur Wissensbasis und ist durch separate Speicher der Planungsmodule realisiert. (Müller 1996, S. 98)

Lokale Planung

INTERRAP besitzt keine fest vordefinierten Planungsmodule, sondern stellt nur eine Schnittstelle zu deren Einbindung bereit (Müller 1996, S. 84). Vorgegeben sind nur die Rolle der sechs oben genannten Module innerhalb des Kontrollzyklus und deren Interaktion, die genaue Realisierung kann für jede konkrete Agentenmodellierung innerhalb dieses Rahmens genau auf den jeweiligen Aufgabenbereich abgestimmt erfolgen.

Bei jeder Planungsanforderung wird ein neues Ziel aufgestellt. Ein Ziel besteht aus einer Beschreibung der gegebenen Ausgangssituation und einer Charakterisierung der möglichen Zielsituationen anhand derer Eigenschaften[14]. Aus dieser Problembeschreibung kann auf zwei Arten versucht werden, einen Plan zum Erreichen des Ziels zu erstellen. Bei *planning from first principles* (klassische KI-Planer wie z.B. STRIPS) wird mit einem Planungsalgorithmus (*plan generator*) durch (heuristisch gesteuerte) Suche eine Operatorfolge innerhalb des Problemraums aller möglichen Handlungsfolgen generiert. Bei *planning from second principles* wird anhand der Problembeschreibung ein Plan oder mehrere mögliche Pläne aus einer vorgegebenen Sammlung von Plänen, einer Planbibliothek (*plan library*) abgerufen. Ein Plan besteht in INTERRAP aus einer Hierarchie von Teilplänen und procedure PoB, die auch situationsbedingte Alternativen enthalten können.

Um die Qualität der erzeugten Pläne zu beurteilen, berechnet der *plan evaluator* deren Nützlichkeit in Abhängigkeit vom Wert des zu erreichenden Ziels und der Kosten für die

[14] Zusätzlich ist auch die Einstellung des Agenten zu dieser Zielsituation – z.B. Erreichen oder Erhalten der Zielsituation – möglich. (Müller 1996, S. 86)

Ausführung der einzelnen Planschritte. Dies kann dazu verwendet werden, aus mehreren alternativen Plänen der Planbibliothek für ein Ziel den besten auszuwählen, oder um bei Plänen mit schlechtem Kosten/Nutzen-Verhältnis einen besseren zu finden – entweder durch Neuplanung auf der gleichen Ebene oder durch Weitergabe an die CPL.

Zur Ausführung der fertigen Pläne werden diese durch den *plan interpreter* parallel interpretiert, indem für jeden Plan schrittweise entweder eine Handlung ausgewählt oder eine Entscheidung bei einer Verzweigung getroffen wird. Der *plan scheduler* koordiniert diese Planschritte zeitlich und der *plan executor* sorgt für deren rechtzeitige Ausführung, die er auch überwacht.

Der Kontrollzyklus der LPL

Der Kontrollzyklus der LPL instantiiert wiederum den allgemeinen Kontrollzyklus der INTERRAP-Ebenen. Zunächst wird durch *BR* das mentale Modell aktualisiert.

SG besteht im Erkennen lokaler Planungssituationen sowie in der Ergänzung der von BBL empfangenen Planungsanforderungen durch Informationen des mentalen Modells. Aus diesen Situationen wird ein Planungsproblem in Form einer Ausgangs- und einer Zielbeschreibung erstellt und an den Planer bzw. bei mangelnder Kompetenz an die CPL weitergegeben.

Ein Plan für eine gegebene Problemstellung wird entweder durch Mittel-Zweck-Analyse neu generiert oder aus der Planbibliothek abgerufen. Anschließend werden die einzelnen Planschritte der fertigen Pläne vom Scheduler in eine genaue zeitliche Ordnung gebracht. Gegebenenfalls können dabei Inkompatibilitäten entdeckt werden, die eine Neuplanung erfordern. Zur Ausführung werden die Pläne interpretiert, indem in jedem Zyklus von jedem Plan je ein primitiver Planschritt (der einem procedure PoB oder einer Nachricht entspricht) ausgewählt und zur vorgesehenen Zeit zur Aktivierung an die BBL gesendet wird. Wird das Ende der Ausführung eines PoB durch die BBL gemeldet, so wird deren Erfolg anhand der Wissensbasis überprüft.

2.1.5. Die kooperative Planungsebene

Der Aufbau der CPL

Der Aufbau der CPL entspricht in weiten Zügen dem der LPL. Der Unterschied besteht darin, daß die Module nicht zur Generierung und Ausführung von lokalen, sondern von kooperativen Plänen dienen. Des weiteren enthält die CPL die notwendigen Module zur Kommunikation und zur Durchführung von Verhandlungen mit anderen Agenten über gemeinsam durchzuführende Pläne und zur Umwandlung dieser gemeinsamen Pläne in die vom einzelnen Agenten auszuführenden Handlungssequenzen.

Die CPL interagiert mit der LPL und der Wissensbasis, von der sie zusätzlich das soziale Modell, das Informationen über den physischen und mentalen Zustand anderer Agenten enthält, verwenden kann. Der Austausch von Nachrichten zwischen LPL und CPL geschieht

analog der Kommuniktion zwischen BBL und LPL: Es werden Planungsanforderungen von der LPL an die CPL und Handlungsverpflichtungen wieder zurück gesendet. Des weiteren kann die CPL auch das Plan-Evaluierungsmodul der LPL aufrufen, um die Kosten des Anteils des Agenten an einem gemeinsamen Plan zu berechnen.

Verhandlung

Verhandlungen zwischen Agenten sind notwendig, um Konflikte zwischen Agenten aufzulösen oder Aufgaben für gemeinsame Ziele unter Agenten zuzuweisen (Müller 1996, S. 102). Eine Verhandlung beginnt mit dem Festlegen der Menge aller möglichen Abkommen (negotiation set), die eine Lösung der gemeinsamen Aufgabe ermöglichen, und unter denen eine Übereinkunft getroffen werden muß. Des weiteren wird ein Verhandlungsprotokoll ausgewählt, das die erlaubten Aktionen und Reaktionen der verhandelnden Agenten während des Verhandlungsprozesses festlegt. Schließlich wählt jeder Agent eine Verhandlungsstrategie zur Auswahl aus den Alternativen innerhalb des ausgewählten Protokolls, um zu einem für ihn möglichst günstigen Verhandlungsergebnis zu gelangen. Die eigentliche Verhandlung besteht im Befolgen des Verhandlungsprotokolls anhand der einzelnen Verhandlungsstrategien durch die beteiligten Agenten, bis diese zu einer Übereinkunft gelangen.

Ein Verhandlungsprozeß beginnt mit dem Austausch von Informationen zwischen den Agenten, die für die zu verhandelnde Situation relevant sind. Dabei wird davon ausgegangen, daß alle Agenten kooperativ sind und ihre eigenen Ziele und Informationen bereitwillig und wahr mitteilen, sowie daß keine Kommunikationshindernisse oder -probleme einer Verhandlung im Wege stehen. Anhand dieser Informationen wird die Notwendigkeit bzw. Möglichkeit einer Verhandlung und einer anschließenden Kooperation festgestellt und der Verhandlungsgegenstand festgelegt.

Anschließend wird ein Protokoll und ein Leiter für die Verhandlung ausgewählt sowie bei symmetrischen Protokollen die Rollen der einzelnen Agenten innerhalb des Protokolls verteilt. Um zu verhindern, daß über die Auswahl eines Protokolls und eines Verhandlungsleiters wiederum eine langwierige Verhandlung geführt wird (inklusive einer Verhandlung über Protokoll und Leiter, usw.), geschieht die Auswahl zufällig oder willkürlich (z.B. anhand der Prozessor-ID).

Der Verhandlungsleiter übernimmt die Aufgabe, die Verhandlungsmenge der möglichen Abkommen zu berechnen, und teilt diese dann den anderen Verhandlungsteilnehmern mit. Die Verhandlungsmenge besteht aus einer Menge möglicher gemeinsamer, jedoch nicht unbedingt vollständig instantiierter Pläne, die eine Lösung für den Verhandlungsgegenstand darstellen.

Daraufhin bestimmen die einzelnen Agenten ihre Verhandlungsstrategien, und die eigentliche Verhandlung beginnt. Dabei werden so lange entsprechend des Protokolls und der einzelnen Verhandlungsstrategien Angebote ausgetauscht, bis eine Übereinkunft erlangt ist

(d.h. alle beteiligten Agenten akzeptieren den vorgeschlagenen Plan und keiner weist ihn zurück).

Kooperative Planung

Der Gegenstand der Verhandlungen sind gemeinsame Pläne mehrerer Agenten, die entweder ein globales Ziel der Agentengemeinschaft erfüllen oder Konflikte zwischen Agenten auflösen sollen[15]. Ein Multiagentenplanungsproblem (MAPP) umfaßt neben einer Beschreibung der Ausgangssituation nicht nur die Ziele eines einzelnen, sondern die (lokalen) Ziele mehrerer Agenten. Die Bearbeitung eines MAPP ist wie bei der lokalen Planung durch heuristische Suche wie auch anhand einer Planbibliothek möglich. Seine Lösung besteht aus den Handlungen der beteiligten Agenten sowie deren Koordinierung in Form von Randbedingungen, die angeben, welche Handlungen (anderer Agenten) vor Beginn eines Planschrittes vollendet sein müssen.

Zur Ausführung muß ein Multiagentenplan in Einzelagentenpläne umgewandelt werden. Jeder Agent berechnet dazu seinen Anteil des gemeinsamen Plans, der zur Synchronisierung der Einzelagentenpläne um Nachrichten erweitert wird, die anderen Agenten die Vollendung eines Planschrittes mitteilen, sowie um Haltepunkte, an denen auf entsprechende Nachrichten anderer Agenten gewartet werden muß[16].

Der Kontrollzyklus der CPL

Auch der Kontrollzyklus der CPL instantiiert das gemeinsame Schema der INTERRAP-Ebenen. Die Aktualisierung des sozialen Modells im Zuge der Funktion *BR* der CPL kann entweder durch Beobachtung anderer Agenten oder durch Kommunikation erfolgen.

Die Situationserkennung geschieht analog zur LPL, nur daß die (erweiterten) Planungsanforderungen der LPL und die neu erkannten Situationen zusätzlich Wissen über Zustand und Ziele anderer Agenten, also Informationen des sozialen Modells umfassen. Die erkannten Situationen führen zudem nicht direkt zum Erstellen eines Ziels für einen Planungsprozeß, sondern sie initiieren Protokolle für Verhandlungen.

Die Funktion *PS* der CPL ist auf zwei Ebenen realisiert, einer Objektebene und einer Metaebene (Müller 1996, S. 101). Auf der Metaebene werden die Protokolle gemäß einer gewählten Strategie ausgeführt, während auf der Objektebene aus dem Verhandlungsprotokoll heraus eine Plangenerierung aufgerufen werden kann, falls der Agent die Rolle des Leiters übernimmt und die Verhandlungsmenge berechnet.

Die Durchführung von Verhandlungen durch die CPL geschieht analog der Ausführung von Plänen durch die LPL mit den entsprechenden Modulen für Protokolle statt für lokale Pläne:

[15] Die Auflösung eines Konflikts zwischen den Zielen einzelner Agenten beinhaltet noch nicht unbedingt auch die Erfüllung der einzelnen Ziele der Agenten.

[16] Bei zu langen Wartezeiten kann auch der gemeinsame Plan als gescheitert angesehen werden, so daß keine ewigen Wartezeiten entstehen.

protocol library, *protocol selector*, *negotiation strategy library*, *protocol interpreter*, *protocol scheduler* und *protocol executor*. Zunächst wählt der *protocol selector* ein der Situation angemessenes Protokoll aus der *protocol library*. Zur Ausführung der Verhandlung wählt der *protocol interpreter* nach der Verteilung der Rollen des Protokolls eine Strategie aus der *negotiation strategy library* aus und führt die Verhandlung gemäß dieser Strategie. Schließlich koordiniert der *protocol scheduler* die einzelnen Schritte verschiedener Verhandlungen und der *protocol executor* sorgt für deren termingerechte Ausführung.

Mit Hilfe des *joint plan generator* und der *joint plan library* werden durch den Verhandlungsleiter gemeinsame Pläne als Verhandlungsmenge generiert. Während der Verhandlung können zur Verhandlung stehende Pläne global durch den *joint plan evaluator* der CPL oder lokal durch den *plan evaluator* der LPL bewertet und anhand dieser Bewertung akzeptiert oder abgelehnt werden.

Bei erfolgreichem Abschluß einer Verhandlung wird der gemeinsame Plan wie oben beschrieben durch den *plan transformator* in einen Einzelagentenplan umgewandelt und zur Ausführung an die LPL übergeben.

2.2. ACT: Aktivierungsausbreitung und Produktionen

Die kognitive Architektur ACT (**A**daptive **C**ontrol of **T**hought) ist das Ergebnis der Bemühungen von John R. Anderson, die verschiedenen voneinander weitgehend unabhängigen experimentellen Resultate und Modelle der Kognitionspsychologie in einer Gesamtarchitektur zu integrieren und so einen größeren Rahmen zur Erklärung der menschlichen Kognition zu erhalten. Ausgehend von der Kombination der HAM-Theorie (**H**uman **A**ssociative **M**emory, Anderson & Bower 1973) mit Produktionensystemen wurde ACT über mehrere Versionen bis hin zu ACT* (Anderson 1983) weiterentwickelt. Dieses Modell wurde aufgrund der rationalen Analyse von Anderson (1990), die die Anpassung der Kognition an die statistische Struktur der Umgebung berücksichtigt, modifiziert zur ACT-R Theorie (Anderson 1993). Weiterhin konzentriert sich die Theorie nunmehr vor allem auf die Erklärung kognitiver Fertigkeiten mit Hilfe von aktivierungsbasierten Produktionensystemen: ”The basic claim of the ACT-R theory is that a cognitive skill is composed of production rules.“ (Anderson 1993, S. 4).

2.2.1. Die Grundarchitektur

Deklaratives und prozedurales Wissen

Die Architektur von ACT basiert auf der Unterscheidung zwischen deklarativem und prozeduralem Wissen (vgl. Kapitel 1.3.1.):

> ”The most fundamental distinction in ACT is the distinction between declarative and procedural knowledge. ... declarative knowledge is factual knowledge that people can report or describe, whereas procedural knowledge is knowledge people can only manifest in their performance.“ (Anderson 1993, S. 18)

Diese Unterscheidung geschieht im Rahmen der Verwendung von Produktionensystemen, die eine klare Trennung vornehmen zwischen dem deklarativen Arbeitsspeicher und den Produktionen, die auf diesen angewandt werden, indem sie Information aus dem Arbeitsspeicher lesen, verarbeiten, verändern und hinzufügen (Anderson 1993, S. 19). Das deklarative Wissen entspricht dem Faktenwissen über die Umgebung sowie den Annahmen und Zielen eines Menschen und wird als zumindest prinzipiell mitteilbar angenommen, während sich prozedurales Wissen nur in der (kognitiven) Tätigkeit eines Menschen zeigt.

Das gleiche abstrakte Wissen kann dabei sowohl in deklarativer, als auch in prozeduraler Form vorliegen. Beide Wissensrepräsentationsarten sind jedoch für einen effizienten und flexiblen Umgang mit Wissen erforderlich, da jede ihre Stärken und Schwächen hat. Deklaratives Wissen kann schnell und einfach aufgenommen und mitgeteilt werden, eignet sich jedoch nicht so sehr für eine direkte Verwendung, da es zuvor durch allgemeine Produktionen

interpretiert werden muß. Prozedurales Wissen ist für eine schnelle und effiziente Nutzung des Wissens optimiert, wozu es jedoch in seiner Verwendbarkeit auf bestimmte Zwecke und Kontexte beschränkt werden muß. Deklarative und prozedurale Repräsentation ergänzen sich somit gegenseitig:

> ”Declarative representational capacity allows the system to acquire knowledge rapidly and in a flexible form that is not committed to a particular use. A procedural representational capacity allows the system the ability to optimize the application of that knowledge for a specific use.“ (Anderson 1993, S. 20)

Gemäß der ACT-R Theorie liegt alles Wissen zunächst nur in deklarativer Form vor und wird durch prozedurale Interpretation verwendet. Dabei kann das deklarative Wissen auch in prozedurales Wissen umgeformt und dann direkt ausgeführt werden.

Die Architektur

Entsprechen der deklarativ/prozedural-Unterscheidung besteht die ACT-Architektur (Abbildung 5) aus zwei Langzeitspeichern, einem für deklaratives und einem für prozedurales Wissen. Den deklarativen Speicher bildet ein assoziatives Netz aus sogenannten Working Memory Elements (WME) oder Chunks, den prozeduralen ein Produktionensystem (Anderson

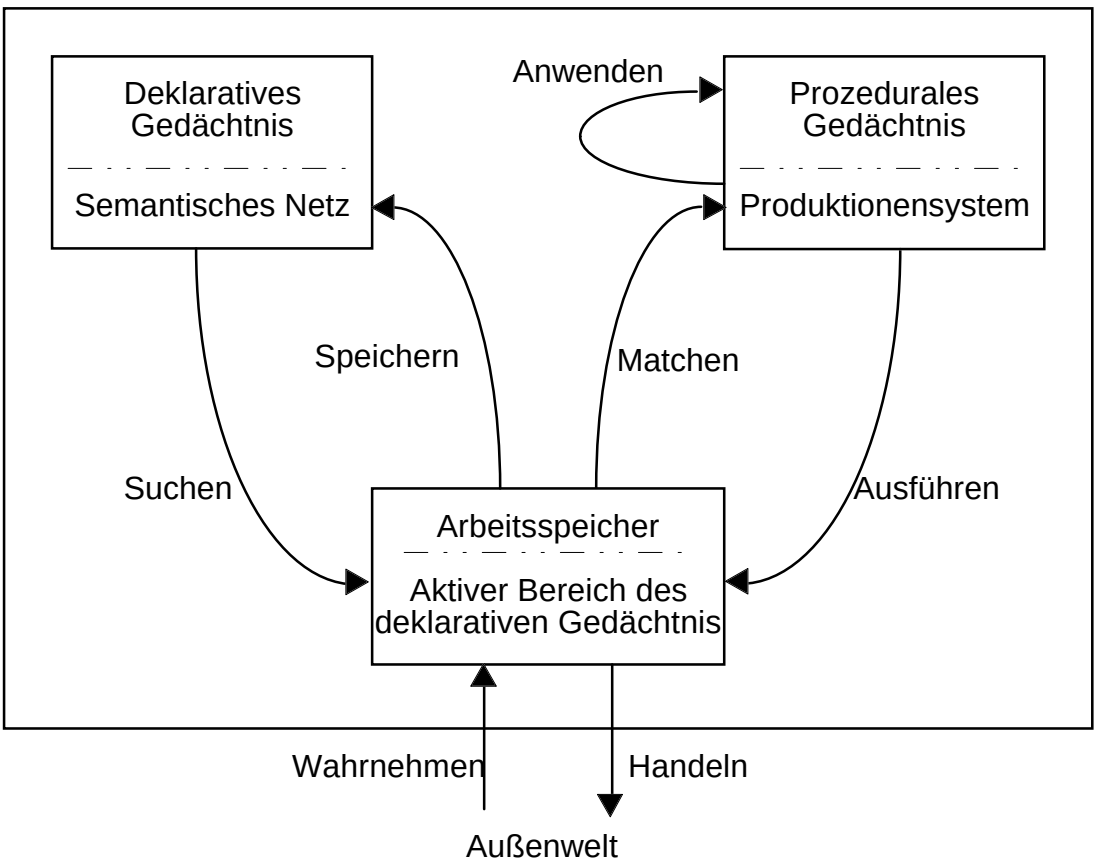

Abbildung 5: Die ACT-Architektur (Anderson 1983, S. 12)

1993, S. 17). Als dritter Speicher dient der Arbeitsspeicher, der aus dem aktuell verwendbaren (aktiven) Teil des deklarativen Langzeitspeichers besteht.

Die Aktivitäten der Architektur zerfallen in drei Bereiche: die Interaktion mit der Außenwelt, die Aktivierung des deklarativen Speichers und die Anwendung von Produktionen (Anderson 1983, S. 20). Die Interaktion mit der Außenwelt über die Körperorgane besteht beim *Wahrnehmen* im Einkodieren von Information aus der Welt in den Arbeitsspeicher und beim *Handeln* im Ausführen von motorischen Befehlen. Die im Arbeitsspeicher enthaltene temporäre Wahrnehmungsinformation wird durch *Speichern* im deklarativen Langzeitgedächtnis permanent. Die *Suche* nach Information im deklarativen Speicher geschieht durch Aktivierung der Chunks. Das Ausführen der Produktionen geschieht in zwei Schritten. Zunächst wird die Anwendbarkeit der Produktionen im Arbeitsspeicher durch *Matchen* überprüft, dann wird eine Produktion durch Konfliktauflösung *ausgewählt* und schließlich *ausgeführt*, indem der Inhalt des Arbeitsspeichers oder des Produktionenspeichers verändert wird.

2.2.2. Der deklarative Speicher: Aktivierung und Assoziation

Working Memory Elements

Die Datenstruktur des deklarativen Speichers sind Working Memory Elements (WME, Chunk). Ein WME besteht aus einer Liste von Slots, denen jeweils ein Wert zugewiesen ist. Der erste Slot jedes WME legt immer seinen Typ (ISA-Slot) fest und damit auch die Anzahl und Rolle der übrigen Slots. Ein Slot kann linear angeordnete, räumlich strukturierte oder abstrakte propositionale Information enthalten (Anderson 1993, S. 27). Lineare Information besteht in einer Liste von Elementen, räumliche Information beschreibt die räumliche Anordnung von Objekten und propositionale Information enthält semantische Information über Eigenschaften und Relationen von Objekten. Diese verschiedenen Informationsarten werden jedoch alle symbolisch repräsentiert.

Chunks sind hierarchisch organisiert, indem sie aus anderen zusammengesetzt werden. Dabei kann ein WME auch in mehreren anderen vorkommen, wodurch sich eine netzartige Stuktur aus miteinander verbundenen Chunks ergibt. Nimmt man eine zahlenmäßige Begrenzung auf etwa drei Elemente pro Chunk an, so wird die Effizienz der Speicherung und des Abrufs optimiert[17] (Anderson 1993, S. 26f).

Ein neues WME entsteht entweder durch die Kodierung externer Ereignisse in der Wahrnehmung oder durch den Ausführungsteil von Produktionen. Neben der Repräsentation der externen Welt werden auch allgemeine Tatsachen und Ziele im deklarativen Speicher aufbewahrt. Die Ziele sind in einem Stapel (stack) organisiert, wobei jedoch nur das oberste Ziel

[17] Die Begrenzung auf drei Elemente pro WME ergibt sich aus psychologischen Beobachtungen, ist jedoch kein Teil der ACT-R-Implementierung.

aktiv ist und dadurch die Auswahl von Produktionen steuert. (Teil-)Ziele werden ausschließlich explizit durch Produktionen hinzugefügt und wieder entfernt.

Aktivierung von Chunks

Jedes WME besitzt einen Grad an Aktivierung, der angibt, wie schnell es durch eine Produktion verwendet werden kann:

> ”Level of activation of a declarative chunk should reflect an estimate of the probability that the chunk will match to a production in the next cycle.“ (Anderson 1993, S. 50)

Indem die Aktivierung so als Maß für die Wahrscheinlichkeit (odds) dient, daß eine bestimmte Information in einer gegebenen Situation benötigt wird bzw. nützlich oder hilfreich ist, ist relevante Information sofort verfügbar und braucht nicht langwierig im meist sehr umfangreichen deklarativen Langzeitspeicher gesucht werden[18] (Anderson 1993, S. 50; Anderson 1983, S. 87).

Dazu hängt die Aktivierung von zwei Komponenten ab, einer Basisaktivierung und einer durch assoziierte Chunks weitergeleiteten Aktivierung, deren Summe die Gesamtaktivierung eines WME ergibt (für die genaue numerische Berechnung der Aktivierung und ihrer Teilkomponenten s. Tabelle 2 sowie Abschnitt über deklaratives Lernen). Die Basisaktivierung hängt von der Häufigkeit des Gebrauchs eines Chunks ab sowie von der Zeitspanne seit dem Gebrauch, und gibt somit eine Abschätzung der *situationsunabhängigen* Wahrscheinlichkeit für eine erneute Verwendung ab.

Für eine *kontextbezogene* Aktivierung von Chunks sorgt die assoziative Aktivierungsausbreitung. Chunks, deren Information häufig zusammen mit anderen Chunks benötigt wird, sind assoziativ miteinander verbunden. Ist ein WME das aktuelle Ziel-Chunk oder wird es durch die Wahrnehmung aktiviert, so wird es zu einer Quelle von Aktivierung und leitet diese gemäß einer erfahrungsabhängigen Gewichtung an alle assoziativ verbundenen Chunks weiter. Während in ACT* die Aktivierung sich noch entlang von Pfaden miteinander assoziierter Knoten ausbreitete, ist in ACT-R nur noch eine Aktivierungsweitergabe an direkt mit Aktivierungsquellen verbundene Knoten vorgesehen.

Durch diesen Aktivierungsmechanismus – dessen Effizienz am ehesten durch massiv parallele Verarbeitung erreichbar ist – wird in jeder Situation die Information aktiv und damit zur Verarbeitung verfügbar, die am wahrscheinlichsten im gegebenen Kontext benötigt wird (Anderson 1993, S. 51). Die Basisaktivierung entspricht dabei einer a priori-Wahrscheinlichkeit und die Assoziation sorgt für den Kontexteinfluß. Mehr noch, die Aktivierung kann

[18] Genaugenommen findet in ACT überhaupt keine wirkliche Suche nach Information im deklarativen Speicher statt. Das Wissen wird bei der Produktioneninstantiierung nicht im Langzeitspeicher gesucht und dann in einen separaten Arbeitsspeicher geschrieben, sondern es löst die Instantiierung der Produktionen aus und beeinflußt deren Geschwindigkeit und damit deren Erfolg anhand der Aktivierung der Chunks (s. u., Abschnitt über Arbeits- und Langzeitspeicher, Abschnitt über Produktionenauswahl).

gleichzeitig als ein heuristisches Maß für die Relevanz der aktiven Information für eine bestimmte Situation dienen (Anderson 1983, S. 87), da die Stärke der assoziativen Verknüpfungen erfahrungsabhängig ist.

Deklaratives Lernen

Das Lernen deklarativen Wissens besteht gemäß der ACT-Theorie nicht im reinen Speichern von Information, sondern in erster Linie in der Organisation dieser Speicherung (Anderson 1993, S. 71). Wurde in ACT* noch erstmalig im Arbeitsspeicher enthaltene Information nur mit einer bestimmten Wahrscheinlichkeit permanent und damit im Langzeitspeicher gespeichert, gibt es in ACT-R kein Vergessen durch nichtvollzogene Speicherung, alle Information wird permanent. Da in ACT im Langzeitspeicher vorhandene Information nur genutzt werden kann, wenn ihr ein genügend hohes Maß an Aktivierung zukommt, kann nicht alles Wissen jederzeit genutzt werden. Das deklarative Lernen besteht folglich im Lernen der Relevanz (der Wahrscheinlichkeit der Verwendung) von Information für gegebene Situationen. Dieses Lernen der Relevanz eines Chunks geschieht über die Veränderung der Basisaktivierung eines WME sowie der Stärken der assoziativen Verbindungen zwischen den einzelnen Chunks.

Die Basisaktivierung steigt bei jedem Gebrauch eines WME und fällt (logarithmisch) mit der Zeit ab (Anderson 1993, S. 71). Sie ergibt sich als die logarithmisierte Summe der Inversen der Zeitabstände seit den einzelnen Verwendungen (s. Tabelle 2). Damit ist die Basisaktivierung

Aktivierung A_i:	$A_i = B_i + \sum_j W_j S_{ji}$ (W_j: Gewichtung der assoziativen Verbindung)
Basisaktivierung B_i:	$B_i = \log (\sum_{k=1...n} t_k^{-d}) + B$ (n: Anzahl der Verwendungen von i) (t_k: Zeitspanne seit der k. Verwendung) (d, B: konstant, $0 < d < 1$)
assoziative Stärke S_{ji}:	$S_{ji} = \log R_{ji}$ mit: $R_{ji} = (aR_{ji}^* + F(C_j)\, E_{ji})/(a + F(C_j))$ ($F(C_j)$: Anzahl der Vorkommnisse von j als Kontext) (E_{ji}: Wahrscheinlichkeit, daß i bei Kontext j relevant ist) (a: konstant)
Ausgangsassoziation R_{ji}^*:	$R_{ji}^* = m/n$ (für alle mit i verbundenen Chunks j) $R_{ji}^* = 1$ (sonst) (m: Anzahl aller Chunks) (n: Anzahl der mit j verbundenen Chunks)

Tabelle 2: Berechnung der Aktivierung A_i eines WME i (Anderson 1993, S. 51; S. 71; S. 76f)

ein statistisches Maß für die Häufigkeit der Nutzung eines Chunks in der (näheren) Vergangenheit und zugleich für die (logarithmisierte) Wahrscheinlichkeit einer Verwendung in der Zukunft – unter der heuristischen Annahme, daß in der Vergangenheit häufig genutztes Wissen auch in Zukunft oft hilfreich sein wird.

Das Lernen der assoziativen Stärke zwischen zwei Chunks bedeutet zu lernen, mit welcher Wahrscheinlichkeit die Information eines Chunks gebraucht wird, wenn der assoziierte bereits aktiv ist (Anderson 1993, S. 76). Dazu wird bei jedem neu entstehenden WME die assoziative Stärke zu allen anderen Chunks zunächst auf Null gesetzt bis auf diejenigen, die direkt (über einen Slot) mit ihm verbunden sind. Bei diesen berechnet sich die Stärke der assoziativen Verbindung als logarithmisierter Quotient aus der Anzahl aller vorhandenen Chunks dividiert durch die Anzahl der mit *j* verbundenen Chunks[19] (s. Tabelle 2). Durch Erfahrung ändern sich die Stärken der assoziativen Verbindungen, wenn zwei Chunks gemeinsam aktiv sind, in Abhängigkeit von der Häufigkeit und der Relevanz der gemeinsamen Aktivierung (s. Tabelle 2). Damit ergibt die assoziative Stärke zwischen Chunks ein statistisches Maß für deren gemeinsame Nutzung in der Vergangenheit und zugleich für die Wahrscheinlichkeit einer gemeinsamen Verwendung in der Zukunft – wiederum unter der heuristischen Annahme, daß in der Vergangenheit häufig gemeinsam genutztes Wissen auch in der Zukunft oft gemeinsam hilfreich sein wird.

Arbeitsspeicher und Langzeitspeicher

Die Aktivierung eines Elementes des deklarativen Langzeitgedächtnisses ist nicht nur ein Maß für dessen Nutzen, sie dient gleichzeitig zur Etablierung des Arbeitsspeichers, also der Information, die durch die Produktionen genutzt werden kann. Der Arbeitsspeicher ist in ACT definiert als der aktive Teil des deklarativen Langzeitspeichers. Er ist somit kein physisch eigenständiger Speicher, sondern nur graduell je nach Aktivierung vom Langzeitspeicher unterschieden.

Ein WME kann aus drei Gründen aktiviert sein (Anderson 1993, S. 55). Zum einen kann es selbst eine Quelle an Aktivierung sein, wenn es durch die Wahrnehmung oder durch Produktionenausführung aktiviert wurde oder das aktuelle Ziel ist. Eine andere Möglichkeit ist, daß es seine Aktivierung durch Aktivierungsausbreitung von einer Quelle erhält. Des weiteren gelten Chunks für eine Produktion als aktiviert, sobald sie vollständig gematcht sind, und können dadurch bei einem zweiten Vorkommen in der gleichen Produktion schneller verwendet werden.

Da nur Chunks zum Arbeitsspeicher gehören, die auf eine dieser Weisen aktiviert wurden, und da nur diese Chunks von den Produktionen verarbeitet werden können, kann es passieren, daß vorhandene Information nicht genutzt werden kann, da sie nicht aktiviert werden kann[20].

[19] Für die genaue wahrscheinlichkeitstheoretische Herleitung dieser Werte s. Anderson 1993, S. 76/77.

[20] Einige experimentelle Ergebnisse über das menschliche Gedächtnis lassen sich auf diese Weise erklären.

Des weiteren werden die üblicherweise vorhandenen (und sinnvollen) Kapazitätsbeschränkungen des Arbeitsspeicher durch die Beschränkung des Zugriffs auf aktivierte Teile des Langzeitgedächtnisses (und evtl. eine Obergrenze an Gesamtaktivierung) erreicht: ”The limitation on working memory capacity in ACT concerns *access* to declarative knowledge, not the *capacity* of declarative knowledge.“ (Anderson 1993, S. 20).

2.2.3. Der prozedurale Speicher: Produktionen

Produktionen

Ein Produktionensystem ist ein System aus Regeln, sogenannten Produktionen, die prozedurales Wissen kodieren. Jede Produktion besteht aus zwei Teilen, einer Bedingung und einer Aktion:

> ”Production rules are if-then or *condition-action* pairs. The *if*, or *condition*, part specifies the circumstances under which the rule will apply. The *then*, or *action*, part of the rule specifies what to do in that circumstances.“ (Anderson 1993, S. 4)

Der Bedingungsteil beschreibt die Situationen, in denen die Produktion angewendet werden kann, in Form einer Liste von Chunks, die aktiviert sein müssen bzw. nicht aktiviert sein dürfen, falls sie negiert sind. Die erste Bedingung in dieser Liste ist immer ein Ziel, das zuoberst im Zielstapel liegen muß. Dadurch werden einzelne Produktionen zu Gruppen zusammengefaßt, die jeweils für bestimmte Arten von Problemen zuständig sind und gemeinsam an einer Aufgabe arbeiten. Durch den Zielstapel ergibt sich somit ein einheitliches, zweckgerichtetes Verhalten in der Abfolge mehrerer Produktionenausführungen.

Der Aktionsteil einer Produktion besteht wiederum aus einer Liste von Chunks, die angibt, wie der Arbeitsspeicher bei Anwendung der Produktion verändert werden soll. Dies geschieht, indem die Slots vorhandener Chunks abgeändert oder neue Chunks hinzugefügt werden. Weiterhin kann die Zielhierarchie verändert werden, indem ein neues Teilziel generiert wird oder ein Ziel, das durch die Produktion erreicht wird, aus dem Stapel wieder entfernt wird.

Die Chunks können auch Variablen enthalten, die beim Überprüfen der Aktivierung des Bedingungsteils (Matchen) erst instantiiert werden. Auf diese Weise werden die Produktionen allgemeiner und abstrakter, da sie auf viele ähnliche Situationen angewandt werden können. Zudem kann eine einzige Produktion nun u.U. auf mehrere Arten auf eine gegebene Situation angewandt werden, wenn mehrere Instantiierungen der Variablen möglich sind.

Die Produktionsregeln eines Produktionensystems bilden eine modulare Organisation prozeduralen Wissens. Einerseits sind Produktionen durch die Zielbedingungen voneinander abhängig, andererseits können sie unabhängig voneinander gelernt und verwendet werden. Dadurch wird sowohl ein hoher Grad an Linearität und Zielgerichtetheit der Verarbeitung erreicht, als auch der Transfer zwischen verschiedenen Problembereichen und somit eine hohe

Flexibilität ermöglicht (Anderson 1993, S. 32). Jede einzelne Produktion stellt einen klar definierten Schritt innerhalb eines kognitiven Prozesses dar. Komplexere kognitive Prozesse bestehen aus einer Folge von Produktionsregeln, die durch die Generierung von (Teil-)Zielen und durch die Verwendung und Modifizierung des Arbeitsspeichers aneinandergereiht werden.

Im Gegensatz zu deklarativ gespeichertem Wissen sind Produktionen prinzipiell irreversibel. Wegen der Asymmetrie zwischen Bedingung und Aktion können auch dem Inhalt nach reversible Operationen nur ausgeführt werden, wenn die Bedingung erfüllt ist. Für die entgegengesetzte Richtung muß – falls sie möglich und sinnvoll ist – eine eigene Produktion vorhanden sein.

Der Ausführungszyklus

Der Ausführungszyklus eines Produktionensystems besteht aus drei Schritten (Anderson 1993, S. 7). Zunächst wird durch Pattern-Matching festgestellt, welche Regeln anwendbar sind, weil ihre Bedingungen im Arbeitsspeicher erfüllt sind. Aus allen anwendbaren Regeln wird im Zuge der Konfliktauflösung eine zur Ausführung ausgewählt. Schließlich wird der Aktionsteil der ausgewählten Produktion ausgeführt. Zentral für das Verhalten eines Produktionensystems ist dabei die Konfliktauflösung, da hier die eigentliche Entscheidung über das Verhalten getroffen wird:

> "*Conflict resolution* is ... the process by which a produciton system decides which of the possible matching productions will fire." (Anderson 1993, S. 45)

Das Pattern-Matching legt die Menge der Möglichkeiten fest, die Konfliktauflösung wählt aus diesen aus, und die Ausführung realisiert diese Entscheidung.

Produktionenauswahl

In traditionellen Produktionensystemen wie OPS5 und auch noch in ACT* geschieht die Konfliktauflösung in der durch Pattern-Matching berechneten Konfliktmenge nach einigen pragmatisch motivierten Prinzipien (Anderson 1993, S. 46; Anderson 1983, S. 132). Zum ersten wird eine Regel nach Möglichkeit nicht wiederholt direkt hintereinander völlig gleich ausgeführt (principle of refractoriness), da dies zu Endlosschleifen führen könnte und selten ein neues Ergebnis bringt. Des weiteren werden Produktionen bevorzugt, deren Vorbedingungen mit möglichst neuen Arbeitsspeicherinhalten übereinstimmen (principle of recency) und die möglichst speziell sind (principle of specifity), d.h. die viele Bedingungen enthalten. In ACT* dienten die erfahrungsabhängige Stärke und der Grad an Übereinstimmung zwischen Bedingung und Arbeitsspeicher als zusätzliche Auswahlkriterien.

Da diese Konfliktauflösungsstrategien jedoch recht willkürlich und nicht immer praktikabel sind, versucht die Produktionenauswahl in ACT-R nach zweckrationalen Gesichtspunkten zu treffen, d.h. die Kosten zur Berechnung der auszuführenden Produktion möglichst zu minimieren und zugleich deren statistische Wahrscheinlichkeit für einen Erfolg zu maximieren

(Anderson 1993, S. 46). Dazu erhalten die einzelnen Produktioneninstantiierungen während des Pattern-Matching Ressourcen in Abhängigkeit von ihrer Erfolgswahrscheinlichkeit, so daß die Geschwindigkeit des Match-Prozesses als ein Maß für die Erfolgswahrscheinlichkeit dient. Dadurch sind die Produktionen, deren Bedingungen zuerst im Arbeitsspeicher vollständig überprüft worden sind, mit höherer Wahrscheinlichkeit in der gegebenen Situation sinnvoll und nützlich als diejenigen, deren Überprüfung mehr Zeit beansprucht.

Die Konfliktauflösung geschieht in ACT-R also während und durch das Pattern-Matching. Dabei werden alle möglichen Produktionen (und auch alle verschiedenen Variableninstantiierungen einer Produktion) gleichzeitig parallel gematcht, indem die einzelnen Chunks nacheinander im Arbeitsspeicher überprüft werden.

Die Ressourcenzuweisung und damit die Match-Geschwindigkeit hängt von der Aktivierung des zu überprüfenden Chunks im Arbeitsspeicher und von der Stärke der Produktion ab (für die genaue numerische Berechnung der Match-Zeiten s. Tabelle 3). Die Stärke einer Produktion wird wie die Basisaktivierung eines Chunks in Abhängigkeit von der Häufigkeit und der Zeitspanne seit des Gebrauchs berechnet. Die Dauer für das Matchen einer Produktion ergibt

Match-Zeit für Produktion p:	$T_p = \sum_{k=1...m} T_{pi}$
	(m: Anzahl der Chunks)
Match-Zeit für Chunk i:	$T_{pi} = B\, e^{-b\,(A_i+S_p)}$
	(A_i: Aktivierung des Chunks i)
	(B, b: konstant)
Stärke der Produktion p:	$S_p = \log\,(\sum_{k=1...n} t_k^{-d}) + B$
	(n: Anzahl der Verwendungen von p)
	(t_k: Zeit seit der k. Verwendung)
	(d, B: konstant, $0 < d < 1$)
Wert einer Produktionsinstantiierung:	$V = P\,G - C$
	(G: Wert des Ziels)
Erfolgswahrscheinlichkeit einer Produktion:	$P = q\,r/(1 - (1 - q)\,f)$
	(q: Wahrscheinlichkeit für erwarteten Effekt)
	(r: Wahrscheinlichkeit für Erreichen des Gesamtziels)
	(f: erwarteter Schaden bei einem Fehlschlag)
Kosten einer Produktionsinstantiierung:	$C = a + b$
	(a: Kosten der Produktion)
	(b: Kosten der nachfolgenden Produktionen)

Tabelle 3: Produktionenauswahl in ACT-R

sich als die Summe der Zeiten für das Matchen der einzelnen Vorbedingungen und hängt damit auch von der Komplexität des Bedingungsteils der Produktionsregel ab. Damit wird eine Regel um so schneller gematcht, je höher der Grad an Aktivierung der einzelnen Vorbedingungs-Chunks im Arbeitsspeicher und je höher die Stärke einer Produktion ist, und um so langsamer, je komplexer die Bedingung ist. Durch den Grad der Aktivierung der Chunks ergibt sich eine *kontextabhängige* Abschätzung der Wahrscheinlichkeit des Nutzens einer Regel, durch die Produktionenstärke eine *situationsunabhängige* Abschätzung. Die zeitliche Reihenfolge der Instantiierung der Produktionen entspricht somit gerade deren Plausibilität für eine erfolgreiche Anwendung: ”This will tend to deliver more plausible instantiations first, because it favors more probable productions matching to more probable (i.e. active) data structures.“ (Anderson 1993, S. 54)

Da wegen der Komplexität der Vorbedingungen speziellere und damit oft nützlichere (weil kontextbezogenere) Produktionen mehr Zeit zum Matchen benötigen, wird nicht einfach die zuerst vollständig gematchte Regel angewendet, sondern zunächst ihr Wert in der gegebenen Situation abgeschätzt. Der Wert einer Produktionsinstantiierung ergibt sich als Differenz des Nutzens – das Produkt aus Erfolgswahrscheinlichkeit und Wert des Ziels – und der zu erwartenden Kosten für die Ausführung der Produktion (einschließlich der abgeschätzten Folgekosten zum Erreichen des Gesamtziels). Die genaue Berechnung dieser Werte hängt von verschiedenen durch Erfahrung statistisch abzuschätzenden Parametern ab (s. Tabelle 3). Es werden nun so lange Produktionen fertig instantiiert und bewertet, bis die zu erwartende Verbesserung des Wertes weiterer Produktionsinstantiierungen den Aufwand für weiteres Matchen nicht mehr rechtfertigt. Dadurch wird zwar nicht unbedingt die bestmögliche Wahl getroffen, jedoch i.a. ein gutes Verhältnis zwischen Aufwand und Ergebnis erreicht (Anderson 1993, S. 57).

Die Produktionenstärke legt fest, ob und wie schnell eine Produktion gematcht wird, der Wert einer Produktion, ob sie angewendet wird (Anderson 1993, S. 59). Beide Werte hängen aufgrund von statistischen Parametern in komplexer Weise vom Verhalten und der Erfahrung des Systems in der Vergangenheit ab und ermöglichen so ein prozedurales Lernen. Sie spiegeln die Wahrscheinlichkeit einer erfolgreichen Anwendung einer Produktion in einer gegebenen Situation wider.

Die Auswahl einer Produktion hängt somit von folgenden neun Faktoren ab (Anderson 1993, S. 63):

”1. The goal that is currently active.
2. The past history of use of various declarative chunks.
3. The elements in the current context.
4. The complexity of the rule.
5. The past frequency of use of the production rule.
6. The past history of success of the production rule.

7. The amount of effort put into solving the problem so far.
8. The similarity between the goal state and the state resulting from applying the production rule.
9. What other options for behavior are available.“

Die Auswahl berücksichtigt im Arbeitsspeicher das aktive Ziel (1) und die Aktivierung der Chunks, sowohl die Basisaktivierung (2), als auch die assoziativen Verbindungen (3). Im prozeduralen Speicher spielen die Komplexität des Bedingungsteils (4) und die Stärke der Produktion eine Rolle (5) sowie ihr statistisch erfaßter Wert (6, 7, 8). Schließlich beeinflussen auch die Unterschiede dieser Werte bei den einzelnen Produktionen die Konfliktauflösung (9). Dadurch wird das Gesamtverhalten des Systems zielorientiert (1), kontextsensitiv (1, 3) und erfahrungsabhängig (2, 5, 6) und beruht auf einem Kompromiß zwischen Effizienz (4, 7) und Rationalität (6, 8, 9).

Analogie

Normalerweise dienen Produktionenregeln der effizienten Verwendung von Wissen in wohlbekannten Situationen. Wenn keine (zufriedenstellend) anwendbaren Produktionen für ein vorliegendes Problem gefunden werden, die das aktuelle Ziel erreichen oder auf es hin arbeiten, so wird in ACT-R interpretierendes Problemlösen durch Analogie verwendet (Anderson 1993, S. 79). Dabei wird ein deklarativ gespeichertes Beispiel der Lösung eines ähnlichen Problems als Vorlage verwendet.

Das interpretierende Problemlösen geschieht in vier Schritten (Anderson 1993, S. 80). Zunächst muß ein Beispiel mit möglichst ähnlicher Zielstruktur und vergleichbarer Situation gefunden werden. Dies geschieht über die Aktivierung gemeinsamer Elemente der Ausgangssituation und möglicher Beispielsituationen. Als nächstes wird die aktuelle Zielstruktur auf das Ergebnis des Beispiels abgebildet, indem die einzelnen Chunks einander zugeordnet werden, wobei deren Typ (ISA-Slot) übereinstimmen muß. Dann wird die Lösungsstruktur des Beispiels auf die gegebene Situation angewendet. Schließlich müssen noch die Vorbedingungen des Beispiels überprüft und als neue Teilziele aufgestellt werden, falls sie in der vorgegebenen Situation nicht erfüllt sind.

Wird ein Beispiel als Analogie verwendet, so wird es verallgemeinert, indem in den bei der Abbildung verwendeten Chunks Konstanten durch Variablen ersetzt werden. Dazu werden neue Produktionen erstellt, deren Bedingungen sich aus der Situationsbeschreibung und der Zielstruktur ergeben und deren Aktionsteil aus den verallgemeinerten Lösungsschritten besteht. Die neuen Produktionen erhalten zunächst nur eine geringe Stärke, die durch erfolgreiche Verwendung oder wiederholte gleiche Analogiebildung verstärkt wird. Auf diese Weise können auch häufig hintereinander ausgeführte Produktionen zu einer komplexeren Regel zusammengefaßt werden, so daß dies in ACT-R anders als in ACT* kein eigenständiger

0. Technical Time Assumption
1. Procedural-Declarative Distinction
2. Declarative Representation
3. Procedural Representation
4. Goal-Directed Processing
5. Sources of Activation
6. Activation in Declarative Memory
7. Production Pattern Matching
8. Production Selection
9. Strength in Declarative Memory
10. Production Strength
11. Interpretive Application of Declarative Knowledge
12. Knowledge Compilation
13. Learning Production Utility

Liste 1: Die 14 Grundannahmen von ACT-R (Anderson 1993, S. 284)

Mechanismus mehr ist. Das Compilieren von Analogien ist der einzige Mechanismus in ACT-R, bei dem neue Produktionen entstehen[21].

Diese generalisierten Analogien sind induktive Inferenzen, die von einem Einzelfall auf den Allgemeinfall schließen, so daß ihnen keine absolute Sicherheit zukommt. Deshalb dürfen sie nicht zu beliebig und zu häufig vorgenommen werden, sie sind jedoch weit effizienter als die aufwendige Interpretation deklarativer Beispiele (Anderson 1993, S. 86).

2.2.4. Zusammenfassung: Die 14 Grundannahmen von ACT-R

ACT-R beruht auf 14 Grundannahmen (Anderson 1993, S. 284ff).

Die Annahme 0. besagt, daß die Zeit als kontinuierlich angesehen wird, d.h. es wird kein zentral gesteuerter Takt vorausgesetzt. Die Zeit für einen Produktionenzyklus ergibt sich aus der Zeit für die Auswahl einer Produktion durch Matchen und Bewerten sowie der Zeit zur Ausführung des Aktionsteils der ausgewählten Produktion.

Die zentrale Annahme 1. ist die Unterscheidung zwischen deklarativem und prozeduralem Wissen auf der Basis eines Produktionensystems, das auf einem deklarativen Speicher arbeitet. Eine deklarative Repräsentationsform ermöglicht eine schnelle Aufnahme von Information aus der Umgebung und die Weitergabe durch Kommunikation. Während deklaratives Wissen der direkten Erfahrung entstammt, wird prozedurales Wissen durch wiederholte Ausführung, durch

[21] Menschen lernen meist anhand von Beispielen. Als mögliche Erklärung hierfür nimmt Anderson (1993, S. 88) an, daß direkte Anweisungen im Verlauf der menschlichen Evolution nur eine sehr untergeordnete Rolle spielten.

Übung gewonnen. Dabei wird es für einen spezifischen Gebrauch optimiert und zugleich eingeschränkt. Prozedurales Wissen unterliegt anders als deklaratives einer prinzipiellen Asymmetrie zwischen der Bedingung und dem Aktionsteil.

Annahme 2. betrifft die Form der deklarativen Repräsentation. Diese besteht aus einer Menge hierarchisch organisierter und vernetzter Chunks. Jedes Chunk besitzt einen bestimmten Typ und besteht aus einer Eigenschaftsliste mit einer begrenzten Anzahl von Einträgen.

Prozedurales Wissen besteht gemäß der Annahme 3. aus Produktionen in Form von Bedingung/Aktion-Paaren. Die Bedingung beschreibt die möglichen Anwendungssituationen und die Aktion die Veränderung des Arbeitsspeichers bei der Anwendung. Das Lernen komplexer kognitiver Fertigkeiten besteht im Lernen vieler relativ einfacher Produktionen, die ausgerichtet an einer Zielstruktur miteinander kooperieren.

Diese Zielorientiertheit der Verarbeitung ist die 4. Annahme. Die hierarchische Zielstruktur besteht aus einem Last-in-first-out-Stapel, dessen oberstes Ziel eine Quelle von Aktivierung ist und dadurch die Produktionenauswahl auf das Ziel ausrichtet. Die Zielhierarchie reflektiert die Abhängigkeit der einzelnen Ziele voneinander.

Annahme 5. beinhaltet die Quellen an Aktivierung, die den Kontext für die kognitiven Prozesse abgeben. Neben dem obersten Zielelement gehören hierzu fokussierte Elemente der Wahrnehmung und durch Wiederholung aktiv gehaltene Knoten.

Nach Annahme 6. besitzen die Elemente des deklarativen Speichers eine Aktivierung, die der (logarithmisierten) Wahrscheinlichkeit eine Produktion zu matchen entspricht und so eine schnelle Informationsverarbeitung bei großen Datenbeständen ermöglicht. Die Aktivierung setzt sich aus einer gebrauchsabhängigen Basisaktivierung und einer situationsabhängigen Aktivierung aufgrund assoziativer Verbindungen zusammen.

Die Geschwindigkeit zum Matchen der Bedingungen der Produktionen im Arbeitsspeicher hängt laut Annahme 7. durch die Aktivierung der Chunks vom gegebenen Kontext und über die Produktionenstärke vom Gebrauch ab. Dadurch werden die Kosten für den Instantiierungsvorgang minimiert, da dieser von der Wahrscheinlichkeit für eine Anwendung der instantiierten Produktionen abhängen.

Die Auswahl aus den instantiierten Produktionen geschieht gemäß Annahme 8. anhand ihres erfahrungsabhängig abgeschätzten Wertes, wodurch der zu erwartende Nutzen einer Produktionenausführung relativ zu den Ausführungskosten maximiert wird.

Annahme 9. beschreibt das Lernen der Stärken der Chunks des deklarativen Speichers. Die Basisaktivierung hängt von der Häufigkeit und der Zeit seit dem Gebrauch ab, die Stärke der assoziativen Bindungen zwischen zwei Chunks von der Häufigkeit einer gemeinsamen Verwendung.

Die Stärke der Produktionen hängt nach Annahme 10. ebenfalls von der Häufigkeit der Verwendung ab und beschreibt die a priori-Wahrscheinlichkeit für deren Anwendung.

Annahme 11. besagt, daß deklaratives Wissen per Analogie zum Problemlösen verwendet wird. Dabei wird die gegebene Problemsituation auf das Beispiel abgebildet und dann dessen Lösungsschritte analog ausgeführt.

Diese Generalisierung von Beispielen wird gemäß Annahme 12. zur Erzeugung neuer Produktionen genutzt.

Annahme 13. betrifft die Parameter für die Bewertung einer Produktioneninstantiierung und deren statistisches Lernen aus Erfahrung (vgl. Tabelle 3).

2.3. TOK/PRODIGY: Agieren, Planen und Lernen

Die Architekturen TOK und PRODIGY sind zwei eigenständige Systeme, die unabhängig voneinander entstanden sind. PRODIGY wurde als Experimentierumgebung für die Verwendung von Lernverfahren in Kombination mit klassischer Planung entwickelt, während TOK als reaktive Architektur für autonome Agenten in der simulierten Welt OZ konzipiert wurde. Im Vergleich mit INTERRAP ist jedoch vor allem die Kombination beider Ansätze (Blythe & Reilly 1993) interessant, da sich hier – wie bei INTERRAP – eine Integration von reaktivem und deliberativem Planen ergibt. Auch die Motivation für die Integration – die Gestaltung zugleich flexibler und intelligenter Agenten für komplexe, dynamische Umgebungen – stimmt überein:

> ”An agent should be able to combine timely responses to well-understood situations with the ability to synthesize appropriate responses to novel situations.“ (Blythe & Reilly 1993, S. 1)

Durch diese Kombination können die Vorteile beider Systeme die Nachteile des jeweils anderen kompensieren. Reaktive Systeme sind deliberativen Planern in komplexen, dynamischen Umgebungen überlegen, da sie auf viele verschiedene Situationen vorbereitet sind und ohne aufwendige Planung direkt (re)agieren können. Ihre Schwäche ist ihre Unflexibilität in Bezug auf neuartige, bei der Gestaltung nicht vorhergesehene Situationen und der hohe Aufwand bei der Programmierung, bei der möglichst viele Situationen sehr detailliert berücksichtigt werden müssen. Gerade hier liegt der Vorteil deliberativer Planer, die durch die Kombination einfacher Handlungsprimitive lange, zielgerichtete Handlungssequenzen durch Planung neu erstellen können. Andererseits ist dieser Planungsprozeß zu zeitaufwendig für genügend schnelle Reaktionen in zeitkritischen Situationen.

Durch die Integration dieser beiden Ansätze erhält man ein hybrides System, das sowohl in bekannten Situationen schnell und effizient handeln, als auch bei unbekannten Situationen durch rationale Planung eine Lösung finden kann:

> ”HAP [die Planungskomponente von TOK] is designed to react quickly and intelligently in a dynamic environment by using stored behaviors when possible. PRODIGY is designed to plan for sets of goals that may interact, and to learn to plan more efficiently. We integrate these two systems so as to retain the strengths of each by giving primary control of the agent to HAP.“ (Blythe & Reilly 1993, S. 4)

Dabei übernimmt der reaktive Planer HAP die Kontrolle über die Interaktion mit der Umgebung und agiert in ihm bekannten Situationen gemäß fest vorprogrammierter Pläne. Ist kein Plan auf der reaktiven Ebene verfügbar, so wird die Planung an PRODIGY übertragen, das den fertigen Plan zur Ausführung wieder an HAP übergibt.

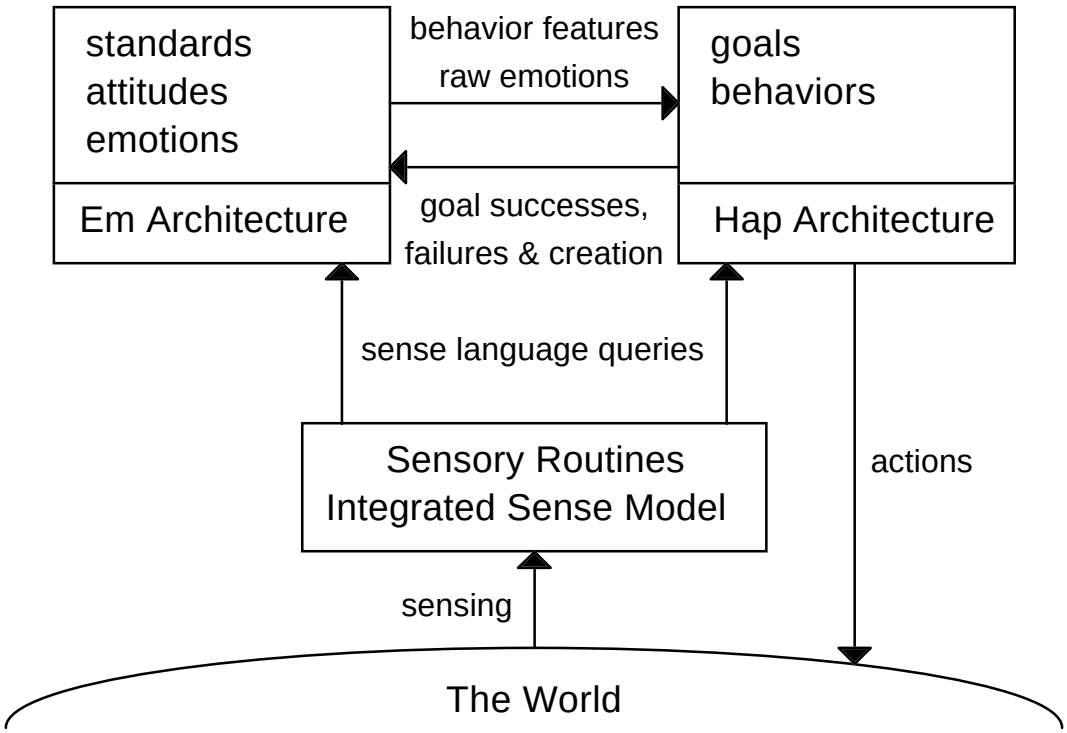

Abbildung 6: Die Architektur TOK (Bates et al. 1992, S. 2)

2.3.1. Die Grundarchitektur: TOK

Mit der Architektur TOK (Abbildung 6) werden autonome Agenten für die Simulation OZ modelliert. OZ ist eine virtuelle Welt, in der menschliche Besucher und künstliche Agenten handeln und miteinander interagieren können. Damit die TOK-Agenten dabei als eigenständige Charaktere für Menschen überzeugend wirken, muß ihr Verhalten nicht unbedingt vollkommen realistisch, aber doch glaubwürdig sein, vergleichbar den animierten Charakteren in einem Zeichentrickfilm. Für eigenständiges Handeln benötigen sie Fähigkeiten zur Wahrnehmung sowie für reaktives und zielgerichtetes Verhalten. Um mit künstlichen und menschlichen Bewohnern der virtuellen Welt interagieren zu können, müssen sie Sprache verstehen und erzeugen können sowie ein soziales Verhalten und Emotionen zeigen (Bates et al. 1992, S. 2).

Dazu verwendet TOK drei Module (s. Abbildung 6). Sensorische Routinen werten sensorische Daten aus und aktualisieren ein Wahrnehmungsmodell der Welt (Integrated Sense Model, ISM). Dieses wird von dem reaktiven Planer HAP verwendet, um Reaktionen und zielgerichtete Handlungen hervorzubringen. Für ein emotionales und soziales Verhalten sorgt EM, das die Emotionen des Agenten verändert und mit diesen die Handlungsauswahl von HAP beeinflußt.

Für die Verwendung natürlicher Sprache wird das System GLINDA benutzt (Loyall & Bates 1996), das sprachliche Ausdrücke durch hierarchische Expansion erzeugt. Dabei werden situativ sprachliche Strukturen detatillierter ausgearbeitet, so daß ausgehend von einer groben Struktur gleichzeitig der Satz konstruiert und die ersten Teile bereits ausgegeben werden

können. GLINDA wird als eigenständiges Modul von HAP aufgerufen, es kann aber auch wegen der strukturellen Ähnlichkeit mit der situativen Planung von HAP durch Pläne in HAP realisiert werden.

Der Kontrollzyklus von TOK geschieht in drei Schritten: Wahrnehmen, Denken und Handeln (Bates et al. 1992, S. 4). Zunächst wird durch die sensorischen Routinen das ISM aktualisiert. Dazu werden nicht nur die rein sensorischen Daten, sondern auch Inferenzen verwendet. Anschließend greifen HAP und EM auf das aktualisierte Modell der Welt zu. EM aktualisiert den emotionalen Zustand des Agenten, und HAP wählt anhand des internen Zustands des Agenten, der das Wissen über den wahrgenommenen Zustand der Welt, die Emotionen und die Ziele des Agenten umfaßt, Handlungen aus, die schließlich ausgeführt werden.

2.3.2. Der reaktive Planer: HAP

Die Grundarchitektur

Die HAP-Architektur (Abbildung 7) besteht aus zwei Komponenten: dem Planbaum (Active Plan Tree, APT) und dem Planspeicher. Der Planspeicher enthält die vorgegebenen Verhal-

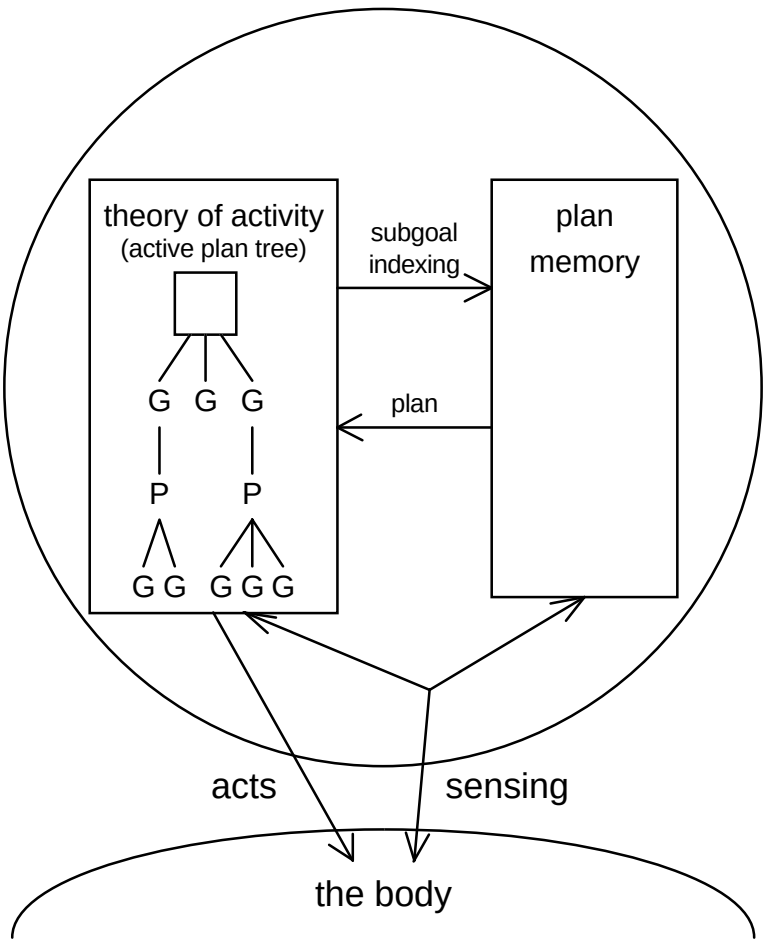

Abbildung 7: Die Architektur von HAP (Loyall & Bates 1991, S. 5)

tensmuster, die das reaktive Verhalten des Agenten bestimmen, der Planbaum die aktuelle Ziel- und Handlungsstruktur des Agenten.

Die Zielstruktur des APT ist hierarchisch organisiert. An oberster Stelle stehen die permanenten Ziele des Agenten. Auf jedes Ziel folgt ein Plan aus dem Planspeicher, um dieses zu erreichen. Da Pläne wiederum aus mehreren Zielen bestehen, ergibt sich eine Struktur aus abwechselnden Schichten von Zielen und Plänen (Loyall & Bates 1991, S. 6). Die Teilziele für einen Plan können dabei sowohl parallel, als auch sequentiell angeordnet sein, je nachdem, ob die genaue Reihenfolge eine Rolle spielt oder nicht. Jedes Ziel enthält einen Erfolgstest, mit dem überprüft werden kann, ob das Ziel erreicht ist, und eine Priorität, zur Auswahl aus mehreren aktiven Zielen.

Ein Ziel ist in HAP keine Beschreibung eines angestrebten Zustands, sondern ein Index für einen Plan im Planspeicher, der aus einer Menge von Produktionen besteht (Loyall & Bates 1991, S. 8). Der Bedingungsteil einer Produktion beinhaltet das Ziel und weitere Vorbedingungen zur Anwendung des Plans. Der Aktionsteil besteht aus einem Plan – einer Menge von neuen Teilzielen – und einer Kontextbedingung, die während der Planausführung erfüllt bleiben muß. Zudem gibt er eine (numerische) Spezifität an, anhand der mehr oder weniger spezifische Pläne unterschieden werden, sowie die Wichtigkeit eines Planes.

Die Erfolgstests der Ziele und die Kontextbedingungen der Pläne sind als Dämonen realisiert, die eigenständig tätig werden (Blythe & Reilly 1993, S. 3). Andere Dämonen werden dazu verwendet, dynamisch in vorgegebenen Situationen neue Ziele aufzustellen – beispielsweise für dringende Reaktionen.

Der Kontrollzyklus

Der Kontrollzyklus von HAP beginnt mit der Aktualisierung des APT aufgrund der wahrgenommenen Änderungen der Umgebung (Loyall & Bates 1991, S. 4). Dazu werden plötzlich erreichte Ziele, deren Erfolgsbedingungen erfüllt sind, und nicht mehr ausführbare Pläne, deren Kontextbedingungen nicht mehr erfüllt sind, aus dem Baum entfernt.

Anschließend wird anhand der Priorität ein Ziel ausgewählt und ausgeführt. Bei gleicher Priorität wird die Kontinuität der Handlungen erhalten, indem entlang des zuletzt ausgewählten Ziels der Baum expandiert wird. In HAP gibt es drei Arten von Zielen: acts, mental acts und subgoals (Blythe & Reilly 1993, S. 2). *Acts* sind physische Handlungen und *mental acts* interne Handlungen (Aufruf einer LISP-Funktion). Diese werden direkt ausgeführt. Beschreibt ein Ziel ein neues Teilziel (*subgoal*), so wird aus dem Speicher ein möglichst spezifischer Plan ausgewählt und an das Ziel angehängt. Pläne, die bereits fehlgeschlagen sind, werden bei der Auswahl nicht berücksichtigt. Indem die Pläne zunächst abstrakt aus Teilzielen bestehen, können sie so situationsabhängig in Teilziele und primitive Handlungen zerlegt werden (Loyall & Bates 1991, S. 3).

Interaktion mit PRODIGY

HAP ist in seinem Verhalten rein reaktiv, da es ausschließlich für bekannte Situationen vordefinierte Pläne verwendet (Blythe & Reilly 1993, S. 3). In Kombination mit PRODIGY können jedoch auch Pläne für neuartige Problemstellungen erstellt und verwendet werden. Dazu wird eine Teilmenge der aktiven Ziele von HAP explizit repräsentiert und zusammen mit einer abstrakten Situationsbeschreibung an PRODIGY übergeben. Dieser Aufruf von PRODIGY ist eine normale mentale Handlung in HAP, die durch ein aktives Ziel im Planbaum ausgelöst wird.

PRODIGY erhält neben den Zielen mit ihren Prioritäten auch ein Zeitlimit, nach dem ein Plan für möglichst viele Ziele mit möglichst hoher Priorität ausgearbeitet sein soll (Blythe & Reilly 1993, S. 5). Die Kommunikation zwischen HAP und PRODIGY sollte dabei auf einer so hohen Abstraktionsebene stattfinden, daß dynamische Aspekte der Umgebung weitgehend vernachlässigbar werden und somit der statische Plan, den PRODIGY an HAP zurückgibt, auch nach längerer Planungsdauer noch anwendbar ist. Der zurückgegebene Plan stellt nicht unbedingt eine reine Folge von physischen Handlungen dar, sondern er besteht aus einer Sequenz von Teilzielen, die HAP situationsabhängig instantiieren und ausführen kann.

2.3.3. Die emotionale Komponente: EM

Unter den Auswahl- und Kontextbedingungen der Pläne von HAP können auch Emotionen sein, die auf diese Weise oder durch das Aktivieren von Dämonen das Handeln des Agenten beeinflussen (Bates et al. 1992, S. 7). Neben den einfachen Emotionen gibt es noch Verhaltenstendenzen (behavioral features), die eine abstraktere Beschreibung des emotionalen Zustands darstellen und durch EM oder HAP geändert werden können. Verhaltenstendenzen bestimmen die Art, wie der Agent seine Ziele verfolgt.

Die Emotionen eines Agenten hängen vom Zustand der Welt und dem Erfolg bzw. Mißerfolg von Zielen ab, worüber EM von HAP benachrichtigt wird. Einmal entstanden, werden Emotionen und damit ihr Einfluß auf das Verhalten mit der Zeit schwächer.

Das emotionale Spektrum

Das emotionale Spektrum eines TOK-Agenten gliedert sich in verschiedene Kategorien. Freude und Trauer hängen davon ab, ob die Ziele erreicht werden oder nicht, ihre Stärke von der Wichtigkeit dieser Ziele. Kann die Wahrscheinlichkeit des Erfolges oder Mißerfolges eines Ziels abgesehen werden, so entstehen dadurch Hoffnung oder Furcht.

Andere Emotionen basieren auf Verhaltensstandards (standards), mit denen der Agent eigene und fremde Handlungen auf ihren moralischen Wert hin beurteilt. Eigene Handlungen oder Unterlassungen rufen Stolz oder Scham hervor, die anderer Agenten Bewunderung oder Ablehnung.

Durch das gleichzeitige Auftreten von Emotionen über Erfolg und über Verhaltensstandards entstehen neue Emotionen. Ärger auf einen anderen Agenten ergibt sich aus Trauer über ein nicht erreichtes Ziel und Ablehnung gegenüber dem anderen, Dankbarkeit aus Freude und Bewunderung. Bei eigenen Handlungen entstehen Befriedigung aus Stolz und Freude über eine Handlung und ein schlechtes Gewissen aus Trauer und Scham.

Schließlich gibt es noch Emotionen, die aus einer Einstellung (attitude) gegenüber anderen Agenten entstehen, sobald dieser in der näheren Umgebung wahrgenommen wird. Bei einer positiven Einstellung ergibt sich Liebe, bei einer negativen Haß.

2.3.4. Der deliberative Planer: PRODIGY

PRODIGY (Abbildung 10) integriert einen klassischen KI-Planungsalgorithmus mit verschiedenen Lernmodulen, um durch Erfahrung die Planungseffizienz und die Qualität der erstellten Pläne zu verbessern bzw. um die Möglichkeiten der Verbesserung klassischen Planens durch Lernmodule zu erforschen: ”PRODIGY’s architecture integrates planning and different learning

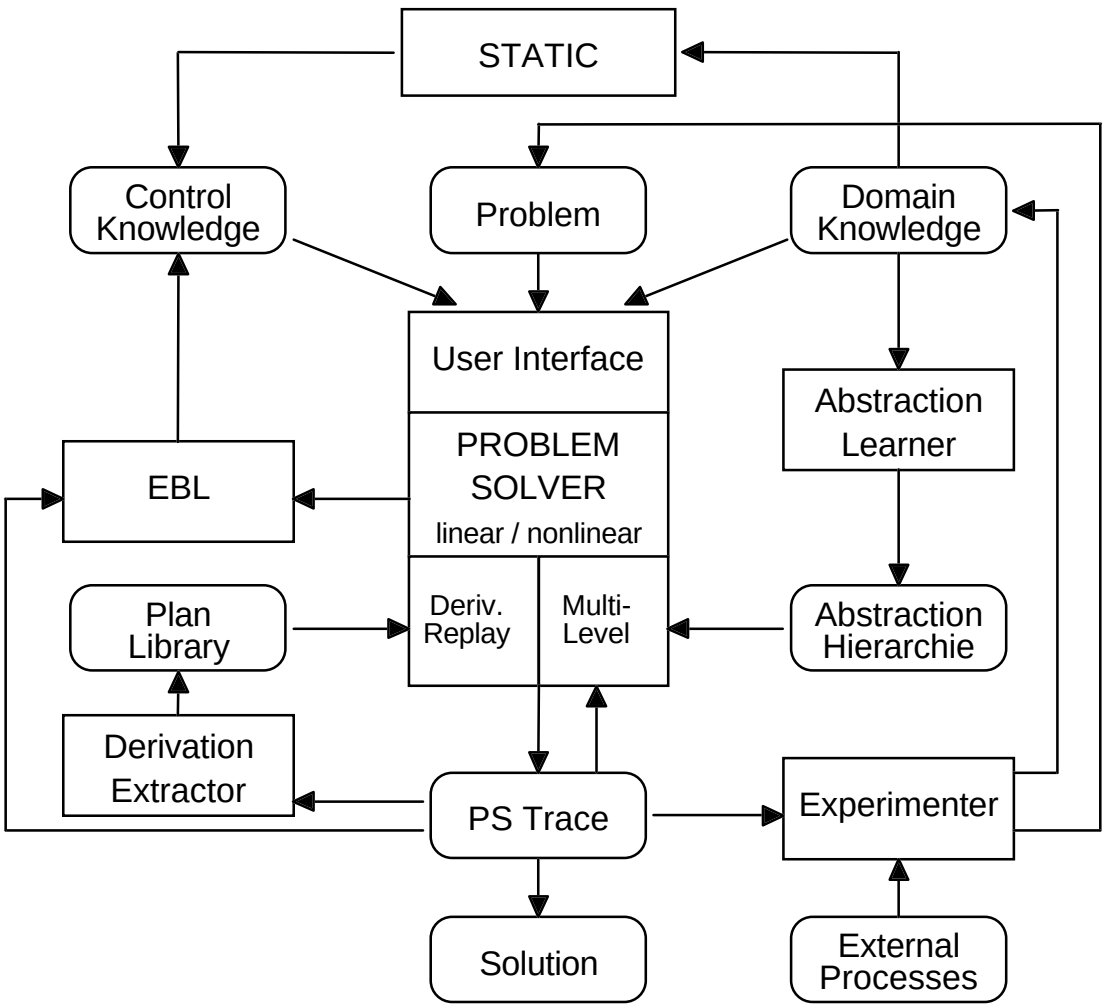

Abbildung 8: Die PRODIGY Architektur (Carbonell et al. 1991, S. 53)

mechanisms in a robust reasoning framework.“ (Veloso et al. 1995, S. 33). Dabei ist der zentrale Mechanismus der Planungsalgorithmus, die Lernmodule werden einzeln an diesen angehängt und erweitern anhand des Planungsvorgangs das Planungswissen des Agenten.

Problemlösen durch Planen

PRODIGY verwendet als klassischer KI-Planer heuristisch geleitete Suche innerhalb eines durch Planoperatoren definierten Problemraums zur Lösung von Problemen, die in Form von expliziten, formalen Ausgangs- und Zielbeschreibungen vorliegen:

> ”PRODIGY is a classical deliberative planner that uses means-ends search to create plans from descriptions of operators, given initial and goal state descriptions.“ (Blythe & Reilly 1993, S. 4)

Die Planung findet jeweils innerhalb einer Planungsdomäne statt, die über eine Menge von Objektklassen sowie durch Operatoren und Inferenzregeln über diesen Objekten definiert ist (Veloso et al. 1995, S. 2). Die Operatoren formalisieren mögliche Handlungen, die an den Objekten vorgenommen werden können, die Inferenzen mögliche Schlußfolgerungen für bestimmte Situationen. Als Problemlösung liefert der Planer bei Erfolg einen Plan, der aus einer Folge von Operatoren besteht, deren Ausführung angefangen beim Ausgangszustand schließlich das Ziel erreicht.

Die Suche wird durch Kontrollwissen gesteuert, das an den Entscheindungspunkten (*decision point*, vgl. Abbildung 10) des Suchalgorithmus Alternativen in Abhängigkeit vom aktuellen Planungszustand auswählt, bevorzugt oder ausschließt. Ist kein Kontrollwissen verfügbar, so wird eine beliebige Wahl unter den Alternativen der tiefsten Suchknoten getroffen (Carbonell et al. 1991, S. 51). Dadurch entsteht eine Tiefensuche bei fehlendem Kontrollwissen, andernfalls können andere Suchstrategien wie Breitensuche oder hierarchisches Planen durch entsprechende Kontrollregeln erreicht werden.

Domänenwissen und Kontrollwissen

Zur Planung verwendet PRODIGY zwei Arten von Wissen: Domänenwissen und Kontrollwissen (Veloso et al. 1995, S. 30). Das Domänenwissen umfaßt Wissen über die Welt und mögliche (physische) Handlungen, das Kontrollwissen dient der erfahrungsabhängigen Steuerung des Suchprozesses. Beide Wissensarten sind modular aufgebaut, so daß Erweiterungen und Änderungen des Wissens einfach möglich sind.

Das Domänenwissen besteht aus den Planungsoperatoren und den Inferenzregeln, die formal wie Operatoren aufgebaut sind, denen aber keine physische Handlung entspricht. Ein Operator beschreibt in seiner Vorbedingung die Situationen, in denen eine Handlung möglich ist, sowie deren Auswirkungen in den Effekten. Die Vorbedingungen bestehen aus wohldefinierten Formeln der Prädikatenlogik erster Stufe, die auch Variablen enthalten können, die bei jeder Anwendung durch Objekte des gleichen Typs neu instantiiert werden. Die Effekte geben

an, wie die Situationsbeschreibung bei Anwendung des Operators geändert wird, indem einzelne Fakten (atomare PL1-Formeln) hinzugefügt oder entfernt werden. Dabei können auch bedingte Effekte angegeben werden, die von der gegebenen Situation abhängen. Die Operatoren können auf verschiedenen Abstraktionsebenen definiert werden, so daß auch hierarchisches Planen, bei dem in mehreren Schritten abstrakte Pläne detaillierter ausgearbeitet werden, möglich ist.

Das Kontrollwissen besteht aus Kontrollregeln, die wie Produktionen aufgebaut sind. Eine Anwendungsbedingung legt die Menge aller Situationen fest, in denen das Kontrollwissen benutzt werden kann, und die Auswahlkriterien geben an, welche Wahl zu treffen ist. Anders als bei der Verwendung von Produktionen werden jedoch alle anwendbaren Regeln zugleich berücksichtigt. Es gibt drei Arten von Kontrollregeln: SELECT, REJECT und PREFER (Veloso et al. 1995, S. 8). SELECT-Regeln geben Alternativen an, die ausschließlich betrachtet werden sollen, während PREFER Alternativen nur bevorzugt, ohne andere auszuschließen. Durch REJECT können einzelne Möglichkeiten explizit ausgeschlossen werden. Durch SELECT- und REJECT-Regeln wird die Auswahl bestimmter Alternativen verhindert, so daß die Suche unvollständig wird. In jedem Entscheidungspunkt wird zuerst anhand der SELECT-Regeln eine Menge möglicher Kandidaten bestimmt. Ist diese Menge leer, so werden alle Alternativen als Kandidaten berücksichtigt außer denen, die durch REJECT ausgenommen werden. Schließlich wird die Menge der Kandidaten anhand der PREFER-Regeln geordnet und die verschiedenen Möglichkeiten gemäß dieser Ordnung durchprobiert.

Durch die Kontrollregeln kann sowohl die Effizienz der Suche, als auch die Qualität des Suchergebnisses verbessert werden. Die einzelnen Kontrollregeln sind entweder vorprogrammiert oder durch eines der Lernmodule gelernt worden. Sie können nicht nur für bestimmte Suchdomänen gelten, sondern auch allgemeine Suchstrategien und bereichsunabhängige Heuristiken beinhalten. Durch Lernen kann allgemeines Kontrollwissen zu bereichsspezifischem werden. Die Regeln ermöglichen eine flexible Steuerung der Suche, indem situations- und erfahrungsabhängig eine definitive Auswahl oder eine heuristische Präferenz für bestimmte Alternativen getroffen werden kann.

Der Planungsalgorithmus

Der Planungsalgorithmus von PRODIGY ist bidirektional. Ausgehend vom Zielzustand werden Operatoren ausgewählt, deren Anwendung das Ziel erreicht. Ausgehend vom Anfangszustand werden so ausgewählte Operatoren angewendet, bis eine Folge von Operatoren gefunden ist, deren Anwendung angefangen beim Ausgangszustand das Ziel erreicht.

Dazu bestehen die noch unvollständigen Pläne (Abbildung 9) während des Planvorgangs aus zwei Teilplänen einem *head-plan* und einem *tail-plan* (Veloso et al. 1995, S. 4). Der head-plan ist ein gültiger total geordneter Plan, der die Ausführung von Operatoren simuliert und beim Anfangszustand beginnt. Der tail-plan ist ein partiell geordneter Plan in Form eines Baumes aus

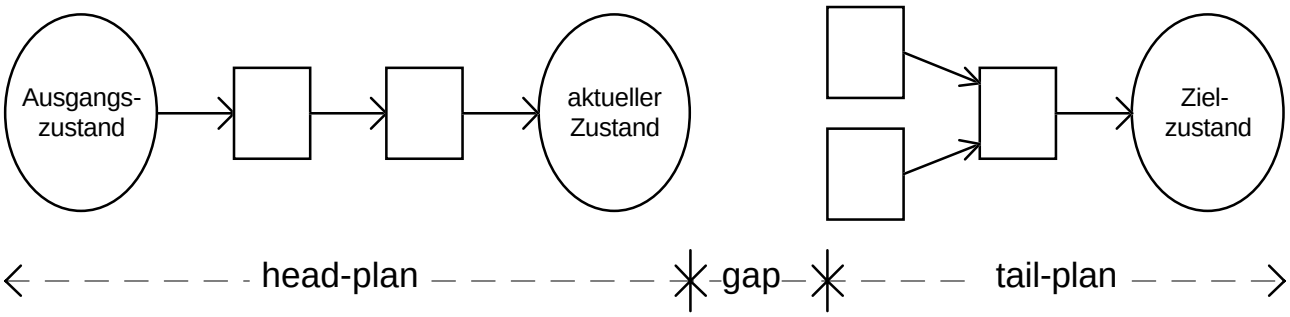

Abbildung 9: Unvollständige Pläne in PRODIGY (Veloso et al. 1995, S. 4)

Operatoren, dessen Wurzel der Zielzustand ist. Zu Beginn des Planungsvorgangs besteht der unvollständige Plan nur aus dem Anfangs- und dem Zielzustand, die den Anfang des head-plan und die Wurzel des tail-plan darstellen. Die Durchführung der Planung besteht nun in der Überbrückung des Unterschiedes zwischen diesen beiden Zuständen bzw. der wiederholten Verringerung des Abstandes zwischen dem Ende des head-plan und dem Anfang des tail-plan (Veloso et al. 1995, S. 4).

Der Planungsprozeß von PRODIGY (s. Abbildung 10) erweitert in jedem Schritt entweder durch Rückwärtsverkettung den tail-plan um einen neuen Operator, oder er simuliert die Anwendung eines Operators am Anfang des tail-plan auf das Ende des head-plan. Bei der Rückwärtsverkettung wird zunächst ein noch nicht erfülltes (Teil-)Ziel des tail-plan ausgewählt. Ein nicht erfülltes Ziel ist ein Literal, das in einer der Vorbedingungen der Operatoren des tail-plan vorkommt und weder im aktuellen Zustand (dem letzten Zustand des head-plan) erfüllt, noch mit einem anderen Operator des tail-plan verbunden ist. Dann wird ein Operator ausgewählt, durch dessen Anwendung das Literal erfüllt wird, und instantiiert. Dieser Operator wird mit dem Vorbedingungsliteral verbunden, so daß dieses nicht weiter geplant wird.

Ein Operator kann angewendet – d.h. vom tail-plan zum head-plan verschoben – werden, wenn sämtliche Vorbedingungen im aktuellen Zustand des tail-plan erfüllt sind und wenn es keinen Operator gibt, der im tail-plan vor diesem steht. Bei der Anwendung wird der aktuelle Zustand des Planvorgangs aktualisiert, indem dieser den Effekten des angewandten Operators entsprechend verändert wird. Dabei wird der Operator aus dem tail-plan entfernt und an den head-plan angehängt. Weiterhin werden mögliche Inferenzregeln auf den aktuellen Zustand angewandt und so implizite Information explizit verwendbar gemacht. Dieser Vorgang wird so lange wiederholt, bis der aktuelle Zustand des tail-plan das Ziel erfüllt oder kein Operator mehr anwendbar ist. Falls notwendig, können bereits angewandte Operatoren auch wieder aus dem head-plan entfernt werden – beispielsweise bei Backtracking. Wird ein Plan gefunden, so wird dieser bewertet und entweder ausgeführt oder ein neuer gesucht (Veloso et al. 1995, S. 5).

PRODIGY

1. If the goal statement *G* is satisfied in the current state *C*, then return *Head-Plan*.
2. Either
   - (A) *Back-Chainer* adds an Operator to the *Tail-Plan*, or
   - (B) *Operator-Application* moves an operator from *Tail-Plan* to *Head-Plan*.

   Decision Point: Decide whether to apply an operator or to add an operator to the tail.
3. Recursivly call *PRODIGY* on the resulting plan.

Operator-Application

1. Pick an operator *op* in *Tail-Plan* which is an *applicable operator*, that is
   - (A) there is no operator in *Tail-Plan* ordered before *op*, and
   - (B) the preconditions of *op* are satisfied in the current state *C*.

   Decision point: Choose an applicable operator to apply.
2. Move *op* to the end of *Head-Plan* and update the current state *C*.

Back-Chainer

1. Pick an unachieved goal or precondition literal *l*.
   Decision Point: Choose an unachieved literal.
2. Pick an operator *op* that achieves *l*.
   Decision point: Choose an operator that achieves this literal.
3. Add *op* to the plan and establish a link from *op* to *l*.
4. Instantiate the free variables of *op*.
   Decision point: Choose an instantiation for the variables of the operator.

Abbildung 10: Der Planungsalgorithmus von PRODIGY (Veloso et al. 1995, S. 6, 7)

2.3.5. Die Lernmodule von PRODIGY

Die Lernmodule von PRODIGY[22] können den in einem Baum protokollierten Suchprozeß nutzen, der alle Entscheidungen und Sackgassen enthält. Dabei profitieren sie von der ”glass box-philosophy“ von PRODIGY, nach der alle Entscheidungen explizit und möglichst durchsichtig repräsentiert werden: ”all decisions are deliberate, explicit, and all comprise opportunities to learn“ (Veloso et al. 1995, S. 32). Möglichkeiten zum Lernen neuen Kontrollwissens bieten alle Entscheidungspunkte des Planalgorithmus (s. Abbildung 10). Dieses Lernen kann sowohl die Effizienz des Planprozesses, als auch die Qualität der erstellten Pläne betreffen. Eine dritte Lernmöglichkeit bieten die verschiedenen Plandomänenen, für die neue oder verfeinerte Operatoren gelernt werden können.

[22] Da Lernen nicht im Mittelpunkt dieser Arbeit steht, werden die Lernmodule von PRODIGY nur knapp vorgestellt und der Vollständigkeit halber erwähnt.

Die Lernmodule arbeiten dabei weitgehend unabhängig und isoliert voneinander. Sie sind nur über den gemeinsamen Problemlöser und über das gemeinsame Domänen- und Kontrollwissen miteinander verbunden.

Lernen von Domänenwissen

Zum Lernen von neuem Wissen über eine Planungsdomäne dienen drei Module: APPRENTICE, EXPERIMENT und OBSERVE (Veloso et al. 1995, S. 11). APPRENTICE ist eine graphbasierte Benutzerschnittstelle, mit dem der Benutzer den Problemlösevorgang unterstützen und bewerten sowie neues Domänenwissen hinzufügen kann. Mit EXPERIMENT kann PRODIGY bei unvollständigem oder unkorrektem Domänenwissen aus Erfahrung lernen. Dabei werden Abweichungen zwischen der Planausführung und den erwarteten Effekten der Operatoren korrigiert. Durch OBSERVE lernt PRODIGY aus der Beobachtung des Verhaltens von Experten und aus der eigenen Praxis. Eine Beobachtung besteht dabei aus einer Folge von Handlungen und den Zuständen vor oder nach jeder Handlung. Aus diesen Beobachtungen werden durch induktive Generalisierungen neue Planoperatoren gelernt.

Effizienzsteigerung durch Lernen von Kontrollwissen

Für die Effizienzsteigerung durch neues Kontrollwissen wurden bislang fünf Module verwendet: EBL, STATIC, DYNAMIC, ALPINE und ANALOGY (Veloso et al. 1995, S. 9/10). EBL (Explanation Based Learning) analysiert die Entscheidungen in vorgegebenen Problemlösungen anhand einer axiomatisierten Theorie der Domäne und relevanter Teile der Problemlösearchitektur. Aus den Erklärungen für die Entscheidungen werden neue Kontrollregeln erstellt. Anhand der Domänenaxiomatisierung wird mit STATIC Kontrollwissen durch Analyse der Planungsdomäne unabhängig von Beispielen erstellt. DYNAMIC stellt eine Kombination von EBL und STATIC dar, bei der Beispiele nur für die Lerngelegenheiten, aber nicht für die Erklärung der Entscheidungen verwendet werden. Durch ALPINE wird das axiomatisierte Domänenwissen in verschiedene Abstraktionebenen aufgeteilt, die für hierarchisches Planen verwendet werden. ANALOGY verwendet gespeicherte Problemlösebeispiele, um bei ähnlichen Problemstellungen die entsprechenden Entscheidungen zu treffen.

Qualitätssteigerung durch Lernen von Kontrollwissen

Die Qualität des Planungsergebnisses kann bereits während des Suchprozesses erhöht werden, indem das Wissen der Planbewertungsfunktion in Kontrollregeln übertragen wird. (Die Qualität eines Plans hängt von den summierten Kosten der verwendeten Operatoren ab.) Dazu stehen in PRODIGY zwei Module zur Verfügung: QUALITY und HAMLET (Veloso et al. 1995, S. 12). QUALITY vergleicht alternative Pläne für ein Problem und analysiert die Unterschiede zwischen den Folgen verschiedener Entscheidungen für die Qualität des Ergebnisses. HAMLET verwendet vollständige Suchbäume einfacher Pläne zur Erklärung der Bedingungen für qualitativ bessere Pläne.

2.4. SOAR: Wissenssuche, Problemraumsuche und Chunking

Ziel der Entwickung von SOAR (**S**tate, **O**perator, **a**nd **R**esult) ist eine kognitive Architektur, die das gesamte Spektrum an Intelligenz umfaßt: ”SOAR is an architecture for a system that is to be capable of general intelligence.“ (Laird et al. 1987, S. 2). Auch wenn Aspekte der menschlichen Kognition nur bedingten Einfluß auf die Gestaltung von SOAR hatten, versteht sich SOAR zudem als eine mögliche Theorie der menschlichen Kognition (Newell 1990).

Intelligenz wird dabei als das optimale Erreichen von Zielen angesehen, indem vorhandenes Wissen genutzt wird, um eine Lösung für ein Problem innerhalb des Problemraums aller möglichen Handlungen zu finden. Entsprechend ist bei SOAR der zentrale Prozeß eine zielgesteuerte heuristische Suche in einem Problemraum. Dieser Suchprozeß arbeitet auf einem zentralen Arbeitsspeicher und verwendet dabei Wissen, das durch Assoziation aus einem Langzeitspeicher abgerufen wird. Das Langzeitwissen wird durch permanentes Lernen aus Erfahrung verbessert, indem die Resultate aller Problemlöseprozesse in neues Wissen umgewandelt werden.

2.4.1. Intelligenz als heuristische Suche in einem Problemraum

Der Knowledge-Level

Newell (1990, S. 48) beschreibt Computer und kognitive Architekturen auf verschiedenen Abstraktionsebenen, die jede auf eigene Weise eine Erklärung und Vorhersage des Systems erlauben. Die höheren Ebenen abstrahieren von den Details der darunterliegenden und ermöglichen so eigene Gesetzmäßigkeiten, die auf denen der unteren basieren. Newell unterscheidet physikalische Ebenen, Ebenen der Schaltlogik, die Symbolverarbeitungsebene und eine Ebene des Wissens (Knowledge-Level). Die physikalischen Ebenen beschreiben die Hardware und darauf ablaufenden Prozesse mit den Mitteln der Physik (Elektronen, Ladungen, ...). Die Ebene der Schaltkreislogik umfaßt die elementaren Prozesse des Computers auf der Maschinenebene (Bits, Register, CPU, ...). Unter Symbolverarbeitung wird die programmgesteuerte Manipulation von informationstragenden Daten verstanden.

Auf dem Knowledge-Level wird das Verhalten des Systems durch Wissen und Ziele charakterisiert (Newell 1990, S. 363). Das System agiert durch Handlungen in seiner Umgebung, indem es aufgrund seines Wissens, wie die Welt ist, und seiner Ziele, wie die Welt sein soll, gemäß dem Rationalitätsprinzip Handlungen auswählt, um seine Ziele zu erreichen: ”behavior is determined by a principle of rationality that knowledge is used in the service of the agents goals.“ (Newell 1992, S. 426). Dabei wird angenommen, daß es in einer bestimmten Situation

immer eine (oder mehrere gleichwertige) optimale Reaktion des Systems gibt, die sich rein rational aus dem Wissen und den Zielen des Systems ergeben. (Newell 1990, 150).

Intelligenz

Unter der Intelligenz eines Systems versteht Newell nun die Annäherung an den Knowledge-Level, also die Fähigkeit, abhängig von Umgebung und Wissen optimale Handlungen zu wählen, um seine Ziele zu erreichen:

> *"A system is intelligent to the degree that it approximates a knowledge-level system*. ... If a system does not have some knowledge, failure to use it cannot be a failure of intelligence, which can only work on the knowledge the system has. If a system uses *all* the knowledge it has and nothing improves its performance, then there is no role left for intelligence. Thus intelligence is the ability to use the knowledge the system has in the service of the system's goals." (Newell 1992, S. 428)

Je mehr ein System sich dieser Wissensebene annähert, desto weniger wird sein Verhalten durch interne Einschränkungen bestimmt, die beispielsweise aus Zeitgründen suboptimale Reaktionen gegenüber der rationalsten begünstigen. Nutzt ein System alles Wissen, das ihm zur Verfügung steht, so erreicht es vollkommene Intelligenz[23]: "If a system uses *all* knowledge that it has, it must be perfectly intelligent." (Newell 1990, S. 90). Wissen umfaßt dabei nicht nur die Repräsentationen eines Systems, sondern auch dessen logische Konsequenzen. Nicht der bloße Mangel an Wissen, sondern nur dessen Nichtverwendung ist für Newell ein Zeichen für fehlende Intelligenz. Intelligenz besteht somit in der Fähigkeit, das vorhandene Wissen zum Erreichen seiner Ziele zu benutzen:

> "Intelligence is the ability to bring to bear all the knowledge that one has in the service of one's goals. To describe a system at the knowledge level is to presume that it will use the knowledge it has to reach its goal. Pure knowledge-level creatures cannot be graded by intelligence – they do what they know and they can do no more, that is, no better." (Newell 1990, S. 90)

[23] Diese Definition von Intelligenz vernachlässigt einige Aspekte von Intelligenz. Beispielsweise ist ein System ohne jegliches Wissen vollkommen intelligent, denn es nutzt tatsächlich alles Wissen, das es besitzt. Aber schlimmer noch, Systeme mit wenigem und sehr einfachem Wissen tendieren nach dieser Definition dazu, als intelligenter angesehen zu werden als Systeme mit reichhaltigem und detailliertem Wissen. Denn es ist offensichtlich eher möglich, aus wenigem sehr oberflächlichem Wissen die *relativ zu diesem Wissen* optimale Lösung auf ein Problem zu finden, als wenn sich aufgrund detaillierteren Wissens die Problemstellung und die Lösungsmöglichkeiten differenzierter darstellen und damit das Auffinden einer Lösung komplizierter wird. Mehr Wissen führt nach der Definition von Newell fast zwangsläufig zu weniger Intelligenz. Diese Definition berücksichtigt nicht die verschiedenen Fähigkeiten zum Erwerb neuen Wissens, sei es durch aktive Erkundung der Umgebung oder durch Neustrukturierung und Abstraktion des vorhandenen Wissens.

Die Problemraumhypothese

Ein Knowledge-Level-System befindet sich in einer Problemsituation, wenn es in einer gegebenen Situation keine angemessenen Reaktionen parat hat. Besteht eine derartige Handlungsungewißheit, so besitzt das System nach Newell nur einen Ausweg: es muß in einem Problemraum nach einer Lösung suchen (Problemraumhypothese; Newell 1990, S. 96). Der Problemraum beschreibt eine Menge möglicher Handlungsalternativen, die für die Ziele des Systems relevant und nützlich sein könnten:

> ”An intelligent system is allways operating in a problem space, the space of the system’s own creation that attempts to restrict the arena of action to what is relevant.“ (Newell 1992, S. 428)

Da ein Problemraum meist in einer impliziten Form repräsentiert ist, in der lediglich die Generierung möglicher Handlungssequenzen beschrieben ist, bedeutet dies nicht nur die Auswahl aus Alternativen, sondern die Suche nach einer Lösung bzw. deren Konstruktion. Suche ist damit zentral für Intelligenz: ”Search is fundamental for intelligence. It is not just another method or cognitive mechanism, but a fundamental process.“ (Newell 1990, S. 96).

Suche wird notwendig, da beim Erstellen einer möglichen Lösung Fehler auftreten (und erkannt werden) können, die durch neuerliche Versuche korrigiert werden müssen. Suche ist zudem immer eine sinnvolle Möglichkeit, da immer größere Problemräume durch Generieren und Testen durchsucht werden können, bis sämtliche Handlungsmöglichkeiten eines Systems vollständig ausgeschöpft sind. Um zu große Suchräume und damit all zu hohen Suchaufwand zu vermeiden, muß die Suche durch Wissen gesteuert werden. Dies beginnt bei der Auswahl eines Problemraums und kann bei jedem Suchschritt, der eine Auswahl aus mehreren Alternativen zuläßt, genutzt werden. Bei genügend Wissen entfällt sogar die Suche, weil die Lösung direkt konstruiert werden kann. Intelligenz besteht somit nicht allein darin, daß gesucht wird, sondern vor allem in der Art, wie gesucht wird: ”Intelligent behaviour will be deeply involved with the process that determine what spaces are searched and control search in those spaces.“ (Newell 1990, S. 97).

2.4.2. Die Gesamtarchitektur

Die Architektur von SOAR (Abbildung 11) basiert auf diesen Überlegungen zur Intelligenz. Sämtliche Aufgaben in SOAR sind als Problemräume repräsentiert, die durch einen Anfangs- und einen Zielzustand definiert sind sowie durch eine Menge von Operatoren, die abstrakt mögliche Zustandsänderungen durch Handlungen beschreiben. Die Suche im Problemraum besteht in einer sukzessiven Veränderung des Anfangszustandes durch die Anwendung von Operatoren, bis der Zielzustand erreicht ist. Die Problemraumsuche geschieht während eines Entscheidungszyklus (decision).

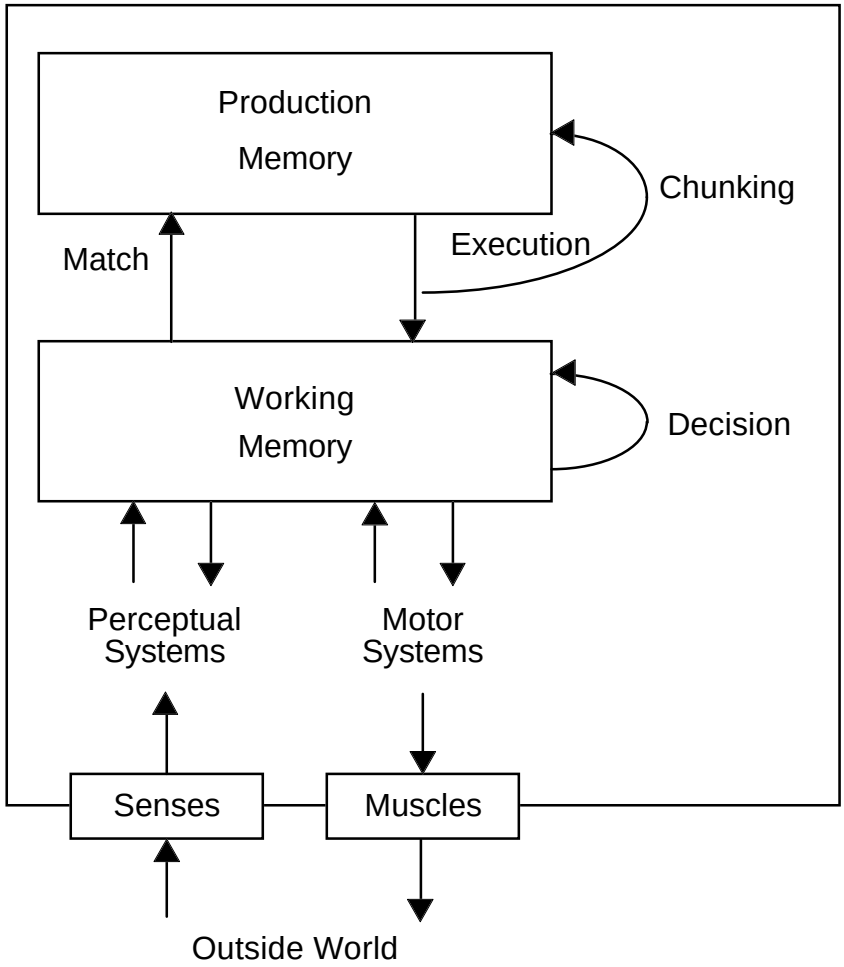

Abbildung 11: Die Architektur von SOAR (Newell et al. 1989, S. 111)

Die Problemraumsuche findet in einem zentralen Arbeitsspeicher (working memory) statt. Dort befindet sich neben dem aktuellen Zustand der Suche auch alles Wissen, das zur Suche verwendet werden kann, sowie ein Zielstapel, der die Suche leitet. Damit Wissen verwendet werden kann, muß es zuvor aus dem Langzeitspeicher in den Arbeitsspeicher abgerufen werden. Das Wissen im Langzeitspeicher (production memory) ist assoziativ durch Produktionen[24] gespeichert, die Wissenseinheiten in den Arbeitsspeicher schreiben (execute), wann immer ihre Vorbedingung erfüllt ist (match). Weiterhin sind mit dem Arbeitsspeicher Systeme für Wahrnehmung (perceptual systems) und Motorik (motor systems) verbunden, die ihre Information dort eintragen bzw. Befehle von dort auslesen. Da alle Prozesse zentral über den

[24] Die Verwendung von Produktionen in SOAR unterscheidet sich von der in ACT. Während in ACT Produktionen prozedurales Wissen darstellen, aus denen die kognitiven Prozesse bestehen, werden Produktionen in SOAR – wie die Kontrollregeln in PRODIGY, die ebenfalls wie Produktionen aufgebaut sind – als Mechanismus für einen kontextsensitiven Speicherabruf verwendet: ”In a traditional production-system architecture, each production is a problem-solving operator. ... SOAR’s productions are neither operators nor implications. Instead, SOAR’s productions perform (parallel) memory retrieval.“ (Rosenbloom et al. 1991, S. 294)

gemeinsamen Arbeitsspeicher ablaufen, können prinzipiell auch weitere Module in SOAR integriert werden, die mit den bereits vorhandenen Prozessen interagieren, indem sie Information in den Arbeitsspeicher einfügen und aus diesem entnehmen.

Das Wissen im Langzeitspeicher entsteht durch permanentes Lernen aus Erfahrung (chunking). Dabei werden Suchpfade in Produktionen umgewandelt, die in vergleichbaren Situationen das neue Wissen zum Arbeitsspeicher hinzufügen und so den Suchprozeß verkürzen.

2.4.3. Wahrnehmung und Bewegung

Sensorik und Motorik sind von der Kognition unabhängige, parallel arbeitende Module, die nur über den Arbeitsspeicher mit dem zentralen Problemlöseprozeß verbunden sind (Newell 1992, S. 430). Durch die Sensorik werden permanent neue Elemente im Arbeitsspeicher erzeugt und die Motorik führt selbständig die im Arbeitsspeicher enthaltenen motorischen Befehle aus:

> ”Sensors autonomously generate working memory structures representing what is being sensed, and motor systems autonomously take commands from working memory and execute them.“ (Rosenbloom et al. 1991, S. 312)

Da Sensorik und Motorik sehr detaillierte Repräsentationen und Befehle verwenden, die Problemraumsuche jedoch auf einer abstrakten Ebene stattfindet, gibt es Produktionen, die zwischen diesen Beschreibungsebenen übersetzen. Kodierungsproduktionen erzeugen aus den sensorischen Daten der Wahrnehmung eine abstrakte Repräsentation der Welt, und Dekodierungsproduktionen interpretieren die abstrakten Befehle, die durch die Planungsoperatoren entstehen, und zerlegen sie in primitive motorische Befehle. Dabei sind auch komplexere Prozesse möglich wie die sensorische Überwachung motorischer Aktivitäten oder die Unterstützung der sensorischen Wahrnehmung durch motorische Veränderung der Sensoren. Die Kodierungs- und Dekodierungsproduktionen sind gewöhnliche Produktionen des Langzeitspeichers, die jedoch unabhängig von Kontrollzyklus parallel auf dem Arbeitsspeicher operieren (Rosenbloom et al. 1991, S. 299).

2.4.4. Wissensspeicher und Wissenssuche

Wissensrepräsentation

Sämtliches Wissen wird in SOAR durch Attribut/Wert-Listen repräsentiert und in Form von Produktionen permanent gespeichert. Ein Objekt besteht aus einem Namen und einer Liste von Attribut/Wert-Paaren, die seine Eigenschaften angibt und es dadurch definiert. Es gibt zwar keine explizite Strukturierung des Wissens, durch den assoziativen Zugriff anhand der Bedingungen der Produktionen des Langzeitspeichers ergibt sich jedoch eine implizite Strukturie-

rung, wodurch auch Objekthierarchien mit Vererbung realisiert werden können (Rosenbloom et al. 1991, S. 310).

Anders als in anderen Produktionensystemen (z.B. ACT) sind in SOAR nicht nur prozedurale Repräsentationen, sondern sämtliche Repräsentationsarten durch Produktionen realisiert, sie beinhalten deklaratives, prozedurales und episodisches Wissen (Rosenbloom et al. 1991, S. 293). Das deklarative Wissen umfaßt Fakten über den Zustand der Welt und über mögliche Handlungen. An prozeduralem Wissen sind Fakten über die Ausführung von Handlungen (Operatoren) und Kontrollwissen zur Auswahl von Handlungen (Präferenzen) kodiert. Episodisches Wissen speichert Fakten über Handlungen, die das System ausgeführt hat.

Prozedurales Wissen kann in den Produktionen deklarativ oder prozedural kodiert sein (Rosenbloom et al. 1991, S. 295). Ist es prozedural kodiert, so besteht dessen Anwendung direkt im Hinzufügen von Befehlen in den Arbeitsspeicher, sobald die Bedingung erfüllt ist. Andernfalls wird die Information als deklaratives Wissen in den Arbeitsspeicher geschrieben und dann durch interpretierende Operatoren ausgeführt.

Arbeitsspeicher und Produktionenspeicher

Die Repräsentationen von SOAR sind in zwei Speichern enthalten: einem Arbeitsspeicher und einem Langzeitspeicher. Das Wissen im Arbeitsspeicher kann direkt von allen Modulen verwendet werden, die auf den Arbeitsspeicher zugreifen können. Der Inhalt des Arbeitsspeichers ist temporär, seine Elemente werden entfernt, sobald sie nicht mehr benötigt werden. Dazu sind sie mit dem Ziel und dem Problemraum verbunden, durch den sie aus dem Langzeitspeicher abgerufen wurden und mit deren Beendung sie wieder gelöscht werden können. Der Arbeitsspeicher ist unterteilt in verschiedene Bereiche für Wahrnehmung, Motorik und Problemlösen, auf die nur die entsprechenden Module zugreifen können, und dient als zentraler Datenbus für die Interaktion zwischen den einzelnen Modulen.

Das Wissen im Langzeitspeicher ist permanent, es wird nicht entfernt oder verändert, sondern nur durch Chunking erweitert. Es besteht aus Produktionen, die ihr Wissen dem Arbeitsspeicher hinzufügen, wann immer ihre Vorbedingungen im Arbeitsspeicher erfüllt sind. Dadurch entsteht ein mustergesteuerter, assoziativer Zugriff auf das Langzeitwissen, das Wissen im Langzeitspeicher ist durch seine Bedeutung und seine Relevanz für bestimmte Situationen adressiert. Während der Wissenssuche des Entscheidungszyklus werden alle Produktionen parallel im Arbeitsspeicher überprüft und vorhandene Variablen instantiiert sowie gegebenenfalls ihr Inhalt in den Arbeitsspeicher geschrieben. Da mehrere Variableninstantiierungen einer Produktion möglich sein können, werden solange Produktionen angewendet, bis keine neue Instantiierung mehr möglich ist.

2.4.5. Entscheiden und Problemraumsuche

Der Entscheidungszyklus

Der Entscheidungszyklus von SOAR besteht aus drei Phasen. Der erste Schritt ist die Wissenssuche (Elaborationsphase). Dabei werden die Vorbedingungen der Produktionen, die den Langzeitspeicher bilden, im Arbeitsspeicher überprüft und deren Wissen, falls die Bedingung erfüllt ist, zum Arbeitsspeicher hinzugefügt. Dies geschieht solange, bis keine Vorbedingung einer noch nicht verwendeten Produktion mehr erfüllt ist.

Als nächstes folgt die eigentliche Entscheidung anhand des im Arbeitsspeicher enthaltenen Wissens. Dabei werden alle Entscheidungen – wie die Auswahl eines Problemraums oder eines Operators – im aktuellen Zielkontext – bestehend aus Problemraum, Zustand im Problemraum und oberstem Ziel – anhand von Präferenzen getroffen. Präferenzen können Möglichkeiten entweder akzeptieren oder ablehnen sowie verschiedene Alternativen untereinander vergleichen (besser/indifferent/schlechter). Sie sind wie das übrige Wissen auch in Produktionen gespeichert und werden so kontextabhängig verfügbar.

Zuletzt wird die Entscheidung umgesetzt. Ist keine eindeutige Entscheidung aufgrund der vorhandenen Präferenzen möglich, so gerät SOAR in eine Sackgasse. Dies geschieht, wenn es keine Alternativen oder mehrere gleichwertige gibt, wenn keine Veränderung durch die getroffene Entscheidung erreicht wird sowie bei einem Konflikt zwischen verschiedenen Präferenzen. Bei jeder Sackgasse (und nur dann) wird automatisch ein neues Teilziel aufgestellt und zum Zielstapel hinzugefügt, um diese aufzulösen: ”Whenever an impasse occurs, the architecture generates the goal to resolve the impasse.“ (Rosenbloom et al. 1991, S. 296). Diese Teilziele (und alle von ihnen abhängigen) werden wieder aus dem Zielstapel entfernt, sobald die Sackgasse aufgelöst wird, d.h. sobald eine Lösung für die Ursache der Sackgasse gefunden ist.

Tritt eine Sackgasse auf, so wird ein neuer Kontext gesetzt, da in dem alten keine Lösung möglich war. Ein Kontext besteht aus einem Ziel, einem Problemraum, einem aktuellen Zustand sowie möglichen Operatoren. Kann anhand des Wissens im Arbeitsspeicher kein neuer Kontext gesetzt werden, so werden allgemeine Operatoren verwendet, die von dem verwendeten Problemraum unabhängig sind, beispielsweise für verschiedene Suchmethoden wie Operator-Subgoaling oder Lookahead-Search (Rosenbloom et al. 1991, S. 300).

Wird eine Sackgasse aufgelöst, so wird dieses neue Wissen durch Chunking in eine neue Produktion umgewandelt. Die Vorbedingung beschreibt dabei den relevanten Zielkontext und als neues Wissen dient das Resultat der Auflösung der Sackgasse. Dadurch kann die gleiche Sackgasse nicht noch einmal entstehen, da nun das Wissen zu ihrer Vermeidung in einer Produktion enthalt ist, die dieses in der gegebenen Situation in den Arbeitsspeicher schreibt.

Problemraumsuche in SOAR

Die Lösung eines Problems besteht in einer Folge von Entscheidungszyklen. Zunächst wird ein Ziel formuliert, das die Suche leitet. Anschließend wird ein Problemraum ausgewählt, nachdem durch Wissenssuche für die Aufgabe relevantes Wissen aus dem Langzeitspeicher in den Arbeitsspeicher abgerufen wurde. Ein Problemraum besteht aus einer Menge von Zuständen, die Situationen beschreiben, und einer Menge von Operatoren, die Handlungen in Form von Zustandsänderungen beschreiben. Innerhalb dieses Problemraums werden der Anfangszustand und der Endzustand ausgewählt. Auch dies geschieht während eigener Entscheidungszyklen, also in einer Folge von Wissenssuche und Auswahl durch Präferenzen.

Daraufhin wird der Anfangszustand durch sukzessive Entscheidungszyklen durch Operatoren so lange verändert, bis der Endzustand erreicht ist. Die Operatoren werden wie alles andere Wissen durch Produktionen gespeichert und zur Verwendung in den Arbeitsspeicher abgerufen. Sind keine oder mehrere Operatoren anwendbar oder gibt es keine eindeutigen Präferenzen für einen einzelnen Operator, so entstehen Sackgassen und dadurch die für das Suchen in Problemräumen typische Teilzielstruktur.

Die Lösung von Problemen kann dabei in SOAR auf zwei Arten geschehen. Bei der eigentlichen Problemraumsuche wird durch Erzeugen und Testen verschiedener Alternativen eine Lösung gesucht. Dies geschieht immer dann, wenn kein Wissen durch Produktionen verfügbar ist, das eine direkte Lösung ermöglicht. Andernfalls wird durch das in den Produktionen gespeicherte Wissen eine Lösung bereitgestellt, so daß diese bereits bei der Wissenssuche gefunden wird, Problemraumsuche und Entscheidungen zwischen Alternativen sind nicht notwendig. Durch den Chunking-Mechanismus wird bei der Problemraumsuche neu entstandenes Wissen so in Produktionen kodiert, daß es in Zukunft bereits bei der Wissenssuche verfügbar wird, und somit eine aufwendige Problemraumsuche vermieden wird.

2.4.6. Lernen durch Chunking

Chunking ist der einzige Lernmechanismus in SOAR, durch den neue Produktionen (Chunks) und damit langfristig gespeicherte Informationen entstehen. Das Lernen durch Chunking geschieht permanent während des Problemlösevorgangs immer dann, wenn eine Sackgasse aufgelöst wird. Eine Sackgasse entsteht, wenn im Arbeitsspeicher nicht genügend Information für eine eindeutige Entscheidung vorhanden ist, und sie wird aufgelöst, sobald durch Suche eine Entscheidung gefunden worden ist. Durch Chunking wird dann eine neue Produktion erstellt, die situationsabhängig das Wissen zum Treffen dieser Entscheidung in den Arbeitsspeicher schreibt.

Dazu wird der für die Auflösung der Sackgasse relevante Teil des Arbeitsspeichers, also alles Wissen, das zum Erreichen des Teilziels verwendet wurde, zur Vorbedingung der Produktion, wobei Objektbezeichnungen durch Variablen ersetzt werden. Anhand dieser Vorbedingungen kann in allen Situationen, in denen der gleiche Problemlösevorgang möglich

ist, die Produktion angewendet werden. Als Inhalt der Produktion wird das Ergebnis der Auflösung der Sackgasse verwendet, das alles durch die Auflösung neu im Arbeitsspeicher enthaltene Wissen umfaßt. Dadurch kann dieses Ergebnis in Zukunft ohne *Problemraumsuche* erreicht werden, da es in entsprechenden Situationen durch die neue Produktion bereits während der *Wissenssuche* in den Arbeitsspeicher geschrieben wird. Der Chunking-Prozeß wandelt somit implizit vorhandenes Problemlösewissen in explizites um und verändert das Verhalten von SOAR von zeitaufwendiger Suche hin zu vorbereiteten Lösungen: "Chunking converts SOAR's behavior from search into plan (or procedure) following." (Rosenbloom et al. 1991, S. 307).

SOAR lernt ausschließlich anhand von Erfahrung, von tatsächlich durchgeführten, erfolgreichen Problemlösevorgängen. Dabei werden alle Arten von Sackgassen als Anlaß zum Lernen verwendet und alle Arten von Information gelernt: Fakten über Operatoren, Suchkontrolle und Problemräume sowie deklaratives Wissen[25]. An Lernquellen stehen neben den internen Prozessen auch externe Quellen wie Beispiele und Ratschläge zur Verfügung, insofern bei deren Verwendung Sackgassen auftreten.

Durch Chunking werden auch andere Lernprozesse außer dem reinen Übertragen von Problemlöseprozessen in Langzeitwissen erfaßt. Da beim Erstellen neuer Produktionen nicht der gesamte, sondern nur der relevante Teil des Arbeitsspeichers in die Vorbedingungen einfließt, stellen diese implizite Generalisierungen der Problemlösung dar. Auf diese Weise wird auch der Transfer von Wissen zur Verwendung für andere Situationen und Probleme möglich. Bei der Verwendung allgemeiner Problemlöseverfahren werden diese für bestimmte Problemräume spezialisiert, und bei der Verwendung interpretierender Prozeduren wird Wissen prozeduralisiert. Mit Hilfe von Chunking sind in SOAR somit fünf Lernarten realisiert: Optimierung der Suche, Generalisierung, Transfer, Spezialisierung und Prozeduralisierung.

[25] Das Lernen deklarativen Wissens mit Hilfe von Chunking ist insofern problematisch, als die zu lernende Information zwangsläufig zum relevanten Teil des Arbeitsspeichers und damit zu den Vorbedingungen der Produktion gehört (Data-Chunking-Problem; Newell 1990, S. 327). Deshalb kann das gelernte deklarative Wissen nur dann in den Arbeitsspeicher geschrieben werden, wenn es bereits in ihm enthalten ist (eine Produktion kann nur angewendet werden, wenn *alle* Vorbedingungen erfüllt sind). Dieses Problem wird gelöst, indem das deklarative Speichern in zwei Phasen geschieht. Zunächst entsteht eine Testproduktion, die das Ergebnis bereits in der Vorbedingung enthält. Anschließend wird durch allgemeine Produktionen das deklarative Wissen generiert, die das deklarative Wissen nunmehr lediglich innerhalb der Testproduktion, aber nicht innerhalb des Arbeitsspeichers verwenden. Dadurch entsteht eine Produktion, in der die Information nicht mehr in den Vorbedingungen, sondern nur noch als Inhalt vorkommt. (Newell 1990, S. 332)

3. STRUKTURELLE GEGENÜBERSTELLUNG

Die strukturelle Gegenüberstellung der in Kapitel 2. vorgestellten Architekturen INTERRAP, ACT, TOK/PRODIGY und SOAR richtet sich nach dem in Kapitel 1.3. ausgeführten Schema. Sie beginnt mit den von den Architekturen verwendeten Informationsarten. Es folgen die Komponenten zur Speicherung und die Prozesse zur Verarbeitung der Information. Schließlich werden die Architekturen hinsichtlich ihrer Gesamtstruktur, die deren einzelne Komponenten verbindet und kontrolliert, miteinander verglichen.

Da sich die Architekturen in vielen Punkten ähneln, beginnen die vier folgenden Unterkapitel jeweils mit einer Tabelle (s. Anhang), die die in Kapitel 2. bereits ausgeführten Details der Architekturen kurz zusammenfassen und gegenüberstellen. Anschließend werden vor allem die Unterschiede der übrigen Architekturen im Vergleich zu INTERRAP – soweit vorhanden – ausführlicher diskutiert.

3.1. Informationsarten

Tabelle 7 (s. Anhang) zeigt die zentralen Annahmen der Architekturen in Bezug auf die verschiedenen Arten an Informationen, die diese verarbeiten können. Im einzelnen sind die verwendeten Datenstrukturen für deklaratives und prozedurales Wissen, die Konzepte zur Realisierung verschiedener Abstraktionsebenen an Information sowie die funktionalen Rollen, die Informationen innerhalb der Architektur einnehmen können, aufgeführt.

3.1.1. Deklaratives Wissen

In der Repräsentation deklarativer Information stimmen die Architekturen weitgehend überein. Sie wird in allen Architekturen durch symbolische Ausdrücke vorgenommen, wie sie in der formalen mathematischen Logik verwendet werden. Von dieser entstammt auch die zugrunde gelegte Ontologie, nach der die Welt aus eindeutig individuierbaren Objekten mit Eigenschaften und aus Relationen zwischen diesen Objekten besteht.

In den meisten Architekturen werden Objekte und Fakten über Objekte durch Eigenschaftslisten repräsentiert, die aus einer Menge von Attribut/Wert-Paaren bestehen, die jeweils Eigenschaften eines Objekts in Form einer Eigenschaftskategorie samt ihres Wertes angeben. Nur PRODIGY verwendet eine geringfügig abgewandelte Form der Prädikatenlogik, die PDL[26] (PRODIGY Description Language; Carbonell et al. 1992, S. 16), so daß Objekte in PRODIGY durch isolierte Fakten über sie charakterisiert werden.

Zusätzlich können in der PDL Objekte durch Typen in verschiedene Objektklassen eingeteilt werden, mit denen der Objektbereich von Quantoren eingeschränkt werden kann. Auch INTERRAP und ACT verwenden Objektklassen. Die Objektklassen von ACT werden durch das erste Attribut jedes Chunks, das IS-A-Attribut, festgelegt. Dies legt nicht nur den Objekttyp, sondern auch die Anzahl und die Rolle der übrigen Attribute fest. INTERRAP verwendet im Rahmen der Datenbanksprache AKB (Assertional Knowledge Base; Müller 1996, S. 58; Weiser 1995) für die Objektklassen Konzepte, die neben dem Konzeptnamen aus einer Menge von Attributen mit einem zugehörigen Typ, der die möglichen Werte für das Attribut einschränkt, besteht. Die Attribute können feste Eigenschaften aller Objekte eines Konzepts bezeichnen, die in der Konzeptdefinition festgelegt sind, oder je nach Objekt veränderliche, für die im Konzept ein Default-Wert angegeben werden kann. Auch wenn in SOAR keine Objektklassen ausdrücklich Teil der deklarativen Repräsentation sind, so lassen sich diese jedoch recht einfach in SOAR durch spezielle Attribute – vergleichbar dem IS-A-Attribut in ACT – realisieren.

Eine Besonderheit besitzen die Eigenschaftslisten von ACT, da sie auf eine maximale Anzahl von Elementen beschränkt sind[27]. Eine größere Anzahl von Eigenschaften für ein Objekt wird erreicht, indem Chunks hierarchisch aufgebaut und aus anderen Chunks zusammengesetzt sind. Dadurch werden die einzelnen Eigenschaften eines Objektes zusätzlich inhaltlich gegliedert.

Während in ACT und SOAR Objekte ausschließlich anhand ihrer Eigenschaften identifiziert und unterschieden werden, besitzen sie in INTERRAP eine interne, eindeutige Identifikationsnummer, so daß auch unterschiedliche Objekte mit übereinstimmenden Eigenschaften möglich sind. Dies ist sinnvoll, vor allem wenn Eigenschaften – wie in INTERRAP möglich – auch als unbekannt angenommen werden können. Allerdings kann auch ohne architekturbedingte eindeutige Identifikation von Objekten diese durch ein spezielles Attribut (Name) in der Eigenschaftsliste realisiert werden.

[26] PDL-Formeln unterscheiden sich hinsichtlich der Notation von PL1-Formeln in zwei Punkten. Zum einen können Konjunktion und Disjunktion nicht nur auf zwei, sondern auf eine beliebige Anzahl von Formeln angewendet werden. Zum anderen ist die Negation auf atomare Formeln und Existenzaussagen eingeschränkt. Diese Unterschiede beeinträchtigen nicht die Ausdrucksstärke der Formeln.

[27] Die Begrenzung auf etwa drei Elemente pro WME ist nur ein Teil der ACT-Theorie, jedoch nicht der momentanen Implementierung.

Da die Struktur von Attribut/Wert-Paaren sehr allgemein ist, läßt sie sich nicht nur für reine Beschreibungen von Objekten und Situationen, die aus der Konfiguration mehrerer Objekte bestehen, verwenden, sondern auch für andere Inhalte wie episodisches, semantisches oder auch prozedurales Wissen wie Planoperatoren (die dann durch anderes prozedurales Wissen erst interpretiert werden).

In INTERRAP und PRODIGY gibt es dank eigener formaler Sprachen (AKB bzw. PDL) eine klare Syntax für das verwendete deklarative Wissen. Dies ermöglicht auch die Definition einer eindeutigen Semantik für die Ausdrücke der Sprache. ACT und SOAR verzichten hingegen auf eine explizit festgelegte Interpretation deklarativer Strukturen und favorisieren statt dessen eine operationale Semantik, die allein auf der Verwendung des Wissens basiert. Dadurch wird das Repräsentationsschema allgemeiner und offener für andere Arten deklarativer Information. Diese erhöhte Flexibilität geht jedoch auf Kosten eines einheitlichen Gebrauchs des Wissens und der Möglichkeit einer eindeutigen Semantik.

Die Unterschiede zwischen den Formalismen der Architekturen zur Repräsentation deklarativer Information liegen lediglich im Detail, gemeinsam ist allen eine symbolische Repräsentation aus Objekten mit Eigenschaften. Während SOAR – wie auch in vielen anderen Punkten – eine möglichst einfache und offene Struktur verwendet, die den möglichen Inhalten der Repräsentation nur wenige Grenzen setzt, legen die anderen Architekturen mehr Wert auf eine klar definierte Struktur, die eine eindeutigere Semantik ermöglicht. In ACT gehören sämtliche Chunks zu einem bestimmten Typ (IS-A), der die möglichen Attribute festlegt. Auch PRODIGY verwendet Typen für Objekte, die jedoch nur die Objektklassen für Quantifizierungen, nicht aber gemeinsame Eigenheiten der Objekte bestimmen. Den detailliertesten Repräsentationsformalismus verwendet INTERRAP, bei dem Konzeptdefinitionen mit festen, veränderlichen und üblichen Eigenschaften verschiedenen Typs vorgesehen sind.

3.1.2. Prozedurales Wissen

Deklaratives Wissen kann nur verwendet werden, indem es prozedural interpretiert wird. Deshalb ist es explizit repräsentiert und abstrakt und dementsprechend unabhängig von der Architektur. Dies ist auch der Grund, warum es bei der im vorigen Unterkapitel betrachteten deklarativen Repräsentation nur geringfügige Unterschiede zwischen den Formalismen der einzelnen Architekturen gab. Prozedurales Wissen hingegen kann durch die Architektur direkt ausgeführt werden und erhält seine Interpretation somit direkt von den Mechanismen der Architektur, so daß hier auch die Unterschiede zwischen den Architekturen deutlicher werden.

Produktionen

Da die Prozeßkontrolle der Architekturen (s. Kapitel 3.4.2) nicht in einer sequentiellen Abarbeitung eines festen Programmes besteht, sondern durch die gegebene Situation und die vorhandene Information gesteuert wird, besitzen alle Architekturen prozedurale Strukturen, die

zunächst den Speicher auf eine Vorbedingung hin überprüfen und – falls diese erfüllt ist – eine mit dieser verbundene Aktion ausführen. Dies ist das Grundprinzip klassischer Produktionensysteme, bei denen die Aktion in einer Änderung des Arbeitsspeicherinhalts besteht. Allgemein dienen Produktionen einer situationsgesteuerten Kontrolle von Prozessen oder einem kontextabhängigen Speicherzugriff.

In ACT bestimmen Produktionen der klassischen "Bedingung → Aktion"-Form den Ablauf aller kognitiven Prozesse. Sie stellen einzelne Schritte der Kognition dar, die bei komplexeren Prozessen sich modular anhand der Zielstruktur zu längeren Produktionenfolgen aneinanderreihen. Die Produktionen in ACT sind zwar isolierte Wissenseinheiten, sie sind aber nicht völlig unabhängig voneinander, da sie sich durch gemeinsame Ziele und das Erzeugen und Beenden von Teilzielen sowie anhand gemeinsamer Vorbedingungen zu Gruppen von Produktionen organisieren, die nur zusammen umfassendere Aufgaben und Probleme bewältigen können (Anderson 1993, S. 32).

Anders als in ACT bilden die reactor PoB der BBL in INTERRAP – die wie Produktionen direkte Verbindungen zwischen Situationserkennung und Handlungsausführung sind – keine modularen, sondern voneinander unabhängige Wissenseinheiten, die nur für einfache kognitive Prozesse zuständig sind. Sie erfüllen drei Arten von Aufgaben (Müller et al. 1995, S. 7). Zum einen überwachen *reactor PoB* die von der Wahrnehmung eingehende Information auf Situationen, die eine direkte Reaktion erfordern, und führen diese sofort aus. *Knowledge modifier* ändern kontextabhängig den Inhalt der Wissensbasis. Sie können zum Erkennen und Klassifizieren von Situationen sowie zur Abstraktion von Wissen und für deduktive Folgerungen verwendet werden. Schließlich gibt es noch *control modifier*, die Situationen erkennen, die eine Planung auf einer der höheren Ebenen von INTERRAP verlangen, und diese zur Planung an die LPL übergeben. Außerdem werden reactor PoB dazu benutzt, die Terminierungs- und Abbruchbedingungen der procedure PoB für Erfolg bzw. Mißerfolg zu überwachen (Müller 1996, S. 75).

Eine ähnliche Funktion wie die reactor PoB besitzen die Dämonen in HAP (Blythe & Reilly 1993, S. 3), dem reaktiven Planer von TOK. Sie überprüfen, ob ein Ziel bereits erreicht ist oder nicht mehr erreichbar ist, weil seine Randbedingungen nicht mehr erfüllt sind. Zudem können sie situationsabhängig neue Ziele – beispielsweise für dringende Reaktionen – aufstellen. Produktionenartige Strukturen werden in HAP nicht nur für reaktive Prozesse, sondern auch für den Zugriff auf den Planspeicher benutzt. Die Vorbedingungen dieser Produktionen enthalten – wie die in ACT – außer den faktischen Bedingungen immer ein Ziel, ihr Aktionsteil gibt neben dem Plan, der im Planbaum das Ziel ersetzt, die Kontextbedingungen zur Ausführung des Plans und dessen Spezifizität zur Auswahl an. Schließlich ähneln auch die Pläne in HAP Produktionen, da sie Ziele direkt mit einer Folge von Teilzielen verbinden.

Die Produktionen von SOAR dienen einem kontextsensitiven Speicherzugriff. Sie schreiben ihren Inhalt in den Arbeitsspeicher, sobald ihre Bedingungen erfüllt sind. Dabei können sie

entweder nur neues Wissen hinzufügen, aber auch bereits die eigentliche Aktion ausführen, wie es die Produktionen zur Kodierung und Dekodierung ausschließlich tun. In diesem Fall entspricht die Verwendung der Produktionen in SOAR der in klassischen Produktionensystemen.

PRODIGY benutzt Produktionen nur zur Kontrolle des Planungsvorgangs. Sie geben an Entscheidungspunkten abhängig vom momentanen Stand der Planung an, welche Wahl zu treffen, zu bevorzugen bzw. zu unterlassen ist.

Planungswissen

In allen Architekturen spielt Problemlösen durch Planung eine zentrale Rolle (s. Kapitel 3.3.3.). Dabei wird entweder aus Planoperatoren ein neuer Plan erstellt oder ein fertiger Plan aus einer Bibliothek abgerufen. Ein Planoperator beschreibt in einer abstrakten Form mögliche Handlungen eines Agenten, und ein Plan besteht aus einer Folge von Operatoren, die eine Handlungssequenz zum Erreichen eines Ziels angeben.

Planoperatoren ähneln Produktionen insofern, als sie ebenfalls aus zwei Teilen bestehen, einer Anwendungsbedingung und den Folgen der Anwendung. Der zweite Teil beschreibt jedoch nicht wie bei Produktionen die auszuführende Handlung, sondern deren Folgen. Dadurch kann geplant werden, ohne die Handlungen direkt auszuführen.

In ACT werden Planoperatoren einfach als Produktionen angesehen, deren Aktion keine direkte Handlung, sondern nur deren Effekte beschreibt. Auf die gleiche Weise können Planoperatoren auch in SOAR als Produktionen realisiert werden, es besteht jedoch auch die Möglichkeit, Planoperatoren in deklarativer Form zu speichern, die durch den Entscheidungszyklus erst interpretiert werden (Rosenbloom et al. 1991, S. 295). PRODIGY verwendet klassische KI-Planoperatoren, die aus einer Vorbedingung aus PL1-Formeln und einer Menge von Effekten, die Fakten zu einem Planzustand hinzufügen oder entfernen, bestehen. Zusätzlich sind auch bedingte Effekte möglich, die vom Planungszustand abhängen (Carbonell et al. 1991, S. 51). Zur Steuerung der Planung verwenden SOAR und PRODIGY zusätzliches Kontrollwissen in Form von Präferenzen (s. Kapitel 3.3.5.). HAP verwendet keine Operatoren, da es nicht zielgerichtet, sondern situativ plant.

Die Planoperatoren von INTERRAP beschreiben die procedure PoB der BBL als abstrakte Handlungen (Müller & Pischel 1993, S. 19). Die procedure PoB sind fest vorprogrammierte Programme für Routinehandlungen, die ihre Aufgabe möglichst flexibel und eigenständig verrichten können sollen. Dazu können sie nicht nur die Handlungsprimitive der Weltschnittstelle, sondern auch andere PoB aufrufen sowie die Ausführung anhand der Wissensbasis überwachen und situationsabhängig steuern.

Pläne bestehen in PRODIGY, ACT und SOAR aus partiell geordneten Folgen von Planoperatoren. INTERRAP erlaubt zusätzlich verschiedene Kontrollstrukturen, wie sie in Programmiersprachen üblich sind, wie bedingte Verzweigungen, Iterationen und die Komposition von

Teilplänen (Müller 1996, S. 88). Dadurch entsteht eine bessere Strukturierung der Pläne sowie die Möglichkeit, erst während der Ausführung situationsabhängig aus verschiedenen Alternativen auszuwählen oder einen abstrakten Planschritt detaillierter auszuarbeiten. Ein Plan in HAP besteht aus einer Menge von ausführbaren Handlungen und Teilzielen, die situationsabhängig durch Pläne aus dem Planspeicher ersetzt werden.

Fertige Pläne können in INTERRAP, HAP und PRODIGY aus Planbibliotheken abgerufen werden. Diese Möglichkeit besteht auch in ACT und SOAR, indem Pläne deklarativ gespeichert werden, dies ist jedoch kein fester Bestandteil dieser Architekturen. In ACT und PRODIGY gibt es zudem Mechanismen, um Beispiele von Planausführungen als Analogien zum Problemlösen zu verwenden.

INTERRAP spezifiziert neben Einzelagentenplänen auch Pläne für mehrere Agenten, die aus jeweils einem Einzelagentenplan pro beteiligtem Agenten und einer Menge von Synchronisationsbedingungen bestehen. Zudem gibt es an prozeduralem Kooperationswissen noch Verhandlungsprotokolle, die den Ablauf von Kommunikationen zwischen Agenten steuern, sowie Verhandlungsstrategien, mit denen ein Agent ein Verhandlungsprotokoll ausführt.

Prozedurales Wissen

Das prozedurale Wissen spiegelt viele Eigenheiten der zugehörigen Architekturen wider. Auch wenn Produktionen und Planoperatoren in allen Architekturen vorkommen, so werden sie doch jeweils recht verschieden verwendet. Zudem besitzen die meisten Architekturen spezielle Formen prozeduralen Wissens für bestimmte Aufgaben und architekturbedingte Funktionen.

ACT verwendet an prozeduralen Repräsentationen ausschließlich Produktionen. Allerdings stellen auch die verschiedenen Parameter zur Aktivierungsausbreitung und Produktionenauswahl wie die assoziativen Verbindungen und die Produktionenstärken sehr architekturnahe Formen prozeduralen Wissens dar. Sie erfüllen dieselbe Funktion wie das Kontrollwissen in SOAR und PRODIGY oder die Prioritäten in INTERRAP und HAP, sie dienen der Entscheidungsfindung durch die Architektur (s. Kapitel 3.3.5.). Auch SOAR beschränkt sich auf Produktionen zur Repräsentation prozeduralen Wissens, deren deklarativ gespeicherter Inhalt kann jedoch wiederum prozedural interpretiert und benutzt werden, beispielsweise bei Planoperatoren oder Präferenzen.

PRODIGY unterscheidet bei prozeduraler Information zwischen dem Domänenwissen und dem Kontrollwissen. Ersteres repräsentiert in Form von Planoperatoren die in einer Planungsdomäne möglichen Handlungen, letzteres in Form von Produktionen Wissen zur internen Steuerung der Suche im Problemraum. Das prozedurale Wissen in TOK besteht in den vorprogrammierten Plänen von HAP, die jeweils fest mit vordefinierten Zielen verbunden sind.

Nur INTERRAP unterscheidet ausdrücklich zwischen verschiedenen prozeduralen Kodierungen von Handlungswissen, jeweils auf den einzelnen Ebenen der Architektur. Die BBL stellt Verhaltensmuster für Reaktionen (reactor PoB) und Routinehandlungen (procedure PoB)

bereit. Letztere werden auf den Planungsebenen abstrakt durch Planoperatoren repräsentiert. Die CPL unterstützt zudem mehrere Arten von Wissen, die für die Kommunikation und Kooperation mit anderen Agenten benötigt werden, wie Multiagentenpläne, Verhandlungsprotokolle und Verhandlungsstrategien.

Wie bei deklarativem Wissen unterscheiden sich die Architekturen dadurch, ob sie eher explizite oder mehr offene Strukturen verwenden. Außer INTERRAP verwenden alle Architekturen lediglich einen einzigen Formalismus zur internen Repräsentation von Handlungen, die Kombination von HAP und PRODIGY ergibt jedoch ebenfalls eine Aufteilung in situative und deliberative Formen prozeduralen Wissens. Der Vorteil einer einheitlichen Repräsentationsform liegt vor allem in der Kompatibilität des Wissens. Einzelne Wissenseinheiten können sich gegenseitig ergänzen, bei genügend elementaren Einheiten besteht auch die Möglichkeit des Wissenstransfers. Auf der anderen Seite ermöglicht die Verwendung verschiedenartiger Repräsentationsformen eine Spezialisierung und Optimierung des Wissens für bestimmte Aufgaben und Arten der Ausführung. INTERRAP kombiniert die Vorteile von einheitlicher und spezialisierter Repräsentation, indem auf jeder Ebene eine einheitliche, für die Funktionalität der Ebene optimierte prozedurale Datenstruktur verwendet wird. Die Strukturen der höheren Ebenen stellen dabei Abstraktionen der darunterliegenden dar und sind aus diesen aufgebaut, so daß auch die Einheitlichkeit der Gesamtarchitektur gewahrt bleibt. Entsprechendes gilt auch für die Kombination von situativen und deliberativen Plänen in HAP/PRODIGY.

3.1.3. Abstraktionsstufen

Jede Information – deklarative wie semantische – kann auf verschiedenen Abstraktionsstufen repräsentiert werden, die mehr oder weniger Details enthalten und ähnliche oder verwandte Fälle zusammenfassen. Zur Strukturierung deklarativer Information werden Objekte in Klassen eingeteilt. Dies geschieht in INTERRAP und in PRODIGY durch eine Begriffstaxonomie, in der durch Inklusion eine Hierarchie aus Ober- und Unterbegriffen definiert wird. Auch wenn in ACT und SOAR keine Objekthierarchien Teil der Architektur sind, können sie hier durch Eigenschaftslisten oder durch den assoziativen Speicherzugriff realisiert werden. In ACT stellt zusätzlich die Verschachtelung der Chunks eine hierarchische Organisation dar, bei der eine deklarative Information abstrakt durch mehrere Chunks repräsentiert wird, die jeweils detailliertere Eigenschaften angeben.

Verschiedene Abstraktionsstufen prozeduralen Wissens werden vor allem bei der Planung benutzt. PRODIGY verwendet das Modul ALPINE, um das Domänenwissen in unterschiedliche Abstraktionsebenen einzuteilen, so daß der Planungsalgorithmus zunächst einen abstrakten Plan erstellen kann, der anschließend detaillierter ausgearbeitet wird. Auch die situative Planung in HAP geschieht durch detailliertere Ausarbeitung abstrakter Pläne, indem sukzessiv abstrakte Planschritte (Ziele) situationsabhängig durch detailliertere Pläne (Folgen von Teilzielen) ersetzt werden. Zudem erstellt PRODIGY für HAP Pläne aus abstrakten Zielen, die

durch HAP vor der Ausführung weiter geplant werden müssen. Die Planung in INTERRAP geschieht auf jeder Ebene auf einem anderen Abstraktionsniveau. Auf den oberen Ebenen werden Pläne abstrakt erstellt und von der darunterliegenden Ebene bei der Ausführung situationsabhängig weiter ausgearbeitet. Dabei wird das prozedurale Wissen der unteren Ebenen auf den höheren durch abstrakte Handlungen repräsentiert. Auch in SOAR und ACT werden Pläne insofern als abstrakt aufgefaßt, als sie nicht direkt, sondern erst durch weitere Zerlegung in motorische Befehle ausgeführt werden.

Wie bei der Repräsentation deklarativer und prozeduraler Information kann auch die Organisation dieser Information auf den verschiedenen Abstraktionsstufen explizit oder auch nur implizit vorgenommen werden. Und wiederum sind es INTERRAP und auch TOK/PRODIGY, die zugunsten einer klaren Struktur eine möglichst explizite Organisation des Wissens und der Kontrolle benutzen, während ACT und vor allem SOAR zugunsten einer hohen Flexibilität die hierarchische Strukturierung eher implizit lassen. INTERRAP verwendet eine hierarchische Anordnung von Wissens- und Kontrollebenen um verschiedene Grade an Abstraktion, Verantwortung und Wissenskomplexität voneinander zu trennen (Müller 1996, S. 1). Die Kontrolleinheit gliedert sich in eine Ebene für die reaktiven und situativen Fähigkeiten des Agenten, die BBL, und je eine Ebene für lokale und kooperative Planung, LPL und CPL. Dem entsprechend ist die Wissensbasis in drei Schichten unterteilt. Sie besteht aus dem Weltmodell, das Fakten über die Welt und die PoB enthält, dem mentalen Modell, in dem der interne Zustand des Agenten repräsentiert wird, sowie dem sozialen Modell mit Annahmen über den mentalen Zustand anderer Agenten und anderem sozialen Wissen. Wenn auch keine Ebenenarchitektur im strengen Sinne, so besteht auch in TOK/PRODIGY eine klare funktionale Trennung zwischen den beiden Abstraktionsebenen einer detaillierten Planung durch HAP und einer abstrakten Planung durch PRODIGY. ACT und SOAR hingegen verwenden Abstraktionsebenen nur insofern, als sie sich als Modelle der zentralen Kognition verstehen, die ohnehin auf einer abstrakten Ebene stattfindet.

3.1.4. Funktionale Rolle

Die funktionale Rolle von Information umfaßt zum einen die (deklarativen) Repräsentationen von tatsächlichen, möglichen und angestrebten Zuständen der Welt und zum anderen das (prozedurale) Wissen über kognitive und physische Handlungsmöglichkeiten des Agenten.

Repräsentation von Weltzuständen

Die Repräsentation vergangener und gegenwärtiger Zustände geschieht in allen Architekturen durch Faktenwissen, wobei zusätzlich zwischen den momentanen Wahrnehmungen und den gespeicherten Erinnerungen unterschieden wird. In ACT bilden Wahrnehmungen und aktivierte Erinnerungen den Arbeitsspeicher, der für kognitive Prozesse verwendet werden kann. Während in ACT alle Wahrnehmungen im Langzeitspeicher gespeichert werden, benutzen TOK

und INTERRAP diese direkt zur Aktualisierung des internen Modells von der Welt, vergangene Zustände werden überschrieben. Dieses interne Modell dient dann als Grundlage für die weiteren kognitiven Vorgänge, beide Architekturen erlauben jedoch auch einen expliziten Zugriff auf sensorische Daten. Durch die direkte Verwendung der Wahrnehmungsdaten zur Aktualisierung des Weltmodells wird in TOK und INTERRAP gewährleistet, daß dieses in sich widerspruchsfrei und immer im Einklang mit der Wahrnehmung ist, während in ACT Widersprüche erst durch Produktionen entdeckt und aufgelöst werden müssen. In SOAR besitzt die Wahrnehmung eigene Bereiche des Arbeitsspeichers, inwieweit diese jedoch genutzt oder auch nur längerfristig gespeichert werden, ist nicht durch die Architektur vorgegeben, sondern hängt ausschließlich von der Wissens- und Problemraumsuche ab. In der Verwendung der Information gibt es dabei keinen Unterschied zwischen Wahrnehmungen und Erinnerungen. PRODIGY berücksichtigt als zentraler Planungsalgorithmus während des Planungsvorgangs keine Änderungen in der Welt, sondern plant für den von TOK übergebenen Weltzustand. Eine Unterscheidung des Faktenwissens anhand der Zuverlässigkeit, Genauigkeit oder Wahrscheinlichkeit wird in keiner Architektur ausdrücklich vorgenommen.

Neben tatsächlichen Zuständen der Welt können auch mögliche und fiktive Zustände repräsentiert werden. Dies geschieht bei den betrachteten Architekturen jedoch nicht im Langzeitspeicher, sondern lediglich in temporären Speichern während der Planung durch die Planzustände, die bei der deliberativen Planung durch die Anwendung von Planoperatoren entstehen. Ansonsten sind hypothetische Situationen in keiner Architektur explizit vorgesehen.

Eine wichtige Rolle spielt die Repräsentation von Zuständen der Welt, die durch den Agenten angestrebt werden, da diese das deliberative Verhalten bestimmen und leiten und dadurch für ein einheitliches Verhalten sorgen. Im einfachsten Fall besteht dies in Zielen, die den Endzustand für die deliberative Planung angeben, wie dies in SOAR und PRODIGY wie auch in den Planungsebenen von INTERRAP geschieht. Statt einer derartigen expliziten Repräsentation von Zielen, benutzt HAP ebenso wie INTERRAP auf der BBL eine implizite Repräsentation, die keinen Zielzustand beschreibt, sondern direkt auf einen oder mehrere mögliche Pläne verweist, mit denen das Ziel erreicht werden kann. Der Vorteil einer impliziten Repräsentation von Zielen besteht in einer schnellen Auswahl von geeigneten Handlungen, da diese direkt durch das Ziel indiziert werden, sie ist jedoch beschränkt auf vordefinierte Ziele. Andererseits ermöglicht eine explizite Repräsentation eine flexible zielgerichtete Planung durch Suche, falls kein vorgegebener Plan vorhanden ist. INTERRAP und TOK/PRODIGY nutzen die Vorteile beider Alternativen, indem Ziele zunächst implizit und nur zur deliberativen Planung expliziert repräsentiert werden. Auch in SOAR können fertige Pläne gespeichert sein, diese werden dann jedoch nicht durch das Ziel, sondern durch die gegebene Situation bei der Wissenssuche anhand des Matchen der Produktionen aufgefunden. Die Ziele in ACT stellen weder explizite noch implizite Zielbeschreibungen dar, sondern sie leiten lediglich die Auswahl von Produktionen, die dem Erreichen des Ziels dienlich sein können. In allen Architekturen werden Ziele

hierarchisch in einem Zielstapel organisiert, bei dem ein oberstes Ziel so lange sukzessiv in Teilziele zerlegt wird, bis Handlungen zum Erreichen der Teilziele bekannt sind, so daß durch deren Ausführung schließlich das Gesamtziel erreicht wird. HAP und INTERRAP lassen auch parallel mehrere solcher Zielstapel zu.

Fertigkeiten und Handlungen

Bei der funktionalen Rolle prozeduralen Wissens läßt sich zwischen den ausführbaren prozeduralen Strukturen, dem Wissen über diese Strukturen, zwischen alternativen und beabsichtigten Handlungen sowie dem Entscheidungswissen zur Kontrolle kognitiver Prozesse unterscheiden.

Die direkt durch die Architektur ausführbaren Strukturen unterteilen sich in interne kognitive Fertigkeiten und in physische Handlungen, mit denen der Zustand der externen Welt verändert werden kann. Physische Handlungen bestehen in allen betrachteten Architekturen aus einzelnen primitiven Handlungseinheiten, die zur Planung abstrakt repräsentiert werden. Kognitive Fertigkeiten sind alle Prozesse, die den internen Zustand – zumeist den Inhalt eines Arbeitsspeichers – ändern. Sie werden in allen Architekturen durch produktionenartige Datenstrukturen repräsentiert, wie sie bereits oben diskutiert wurden.

Bevor Handlungen ausgeführt werden, muß anhand des Wissens über mögliche Handlungen eine Handlung ausgewählt werden. Dazu werden die Handlungen entweder wie bei den Produktionen in ACT, den Plänen von HAP und den reactor PoB in INTERRAP durch Anwendungsbedingungen indiziert oder durch Operatoren abstrakt zur Planung repräsentiert, wie es in PRODIGY, SOAR und den Planungsebenen von INTERRAP geschieht (s. Kapitel 3.1.2.).

Die Auswahl von Handlungen wird durch Kontrollwissen vorgenommen. SOAR und PRODIGY verwenden dabei explizite Regeln zur Auswahl, während ACT, INTERRAP und HAP verschiedene numerische Mechanismen benutzen (s. Kapitel 3.3.5.). Ausgewählte Handlungen werden in HAP und auf der reaktiven Komponente der BBL von INTERRAP sofort ausgeführt, ansonsten werden sie zunächst zu beabsichtigten Handlungen (Pläne), deren Ausführung weiter kontrolliert werden kann. In SOAR und ACT geschieht dies, indem abstrakte Befehle durch Produktionen in primitive Handlungen zerlegt werden, INTERRAP besitzt hierfür auf jeder Ebene einen Scheduler, der verschiedene Handlungsabsichten miteinander integriert und deren Ausführung kontrolliert.

Funktionale Rolle

In allen Architekturen findet sich weitgehend das gleiche Spektrum an Funktionen für Wissen: Repräsentation tatsächlicher, möglicher und angestrebter Weltzustände, Repräsentation von Handlungen, Handlungsalternativen und Absichten sowie Wissen zur internen Kontrolle und Entscheidungsfindung. Diese Funktionen ergeben sich bereits aus dem Anspruch der Architekturen, einen Rahmen für die kognitive Kontrolle von Handlungen, die sowohl situationsabhängig, als auch zielgerichtet sind, abgeben zu können.

Auf eine detaillierte Unterscheidung dieser Funktionen der Information wird jedoch meist verzichtet, auch wenn sie vorhanden sind. Nur INTERRAP entlehnt der BDI-Theorie die umfassende Einteilung in die mentalen Kategorien Wissen, Ziel, Absicht, Wahrnehmung, Situation, Option und Plan (s. Kapitel 2.1.2.). SOAR beschränkt sich hingegen auf Wissen und Ziele, ACT unterscheidet zusätzlich zwischen deklarativem Wissen und prozeduralen Fertigkeiten, PRODIGY zwischen Domänen- und Kontrollwissen, hinzu kommen noch Pläne. TOK besitzt als Besonderheit das Modul EM, das die Entscheidungsfindung von HAP durch eine emotionale Komponente erweitert, die sich aus dem Erfolg von Handlungen sowie aus moralischen Standards und Einstellungen gegenüber anderen Agenten ergibt.

3.2. Informationsspeicher

Tabelle 8 (s. Anhang) zeigt die Organisation der Informationsspeicher der Architekturen im Überblick. Sie enthält die Speichermodule für langfristige Speicherung und für kurzfristigen Abruf sowie die Konzepte einerseits zur Organisation und Strukturierung der Information im Speicher, andererseits für den Speicherzugriff.

3.2.1. Speichermodule

Die Unterscheidung verschiedener Speichermodule innerhalb einer Architektur betrifft weniger physisch getrennte Einheiten, sondern die konzeptuelle Einteilung, die sich aus der Art der Speicherung und des Zugriffs oder aus der Art der gespeicherten Information ergibt.

Gerade bei großen Datenmengen ist es sinnvoll, einen längerfristigen Speicher, in dem alle Information erhalten bleibt, von einem Arbeitsspeicher zu trennen, dessen Inhalte die momentanen kognitiven Prozesse direkt nutzen können, die jedoch entfernt werden, sobald sie nicht mehr gebraucht werden. Eine derartige Trennung wird in SOAR vorgenommen, wo die zentrale Problemraumsuche nur Information des Arbeitsspeichers berücksichtigen kann und braucht, während der Großteil der Information in einem Produktionenspeicher dauerhaft untergebracht ist und nur bei Bedarf in den separaten Arbeitsspeicher übertragen wird. Die Unterscheidung zwischen (deklarativem) Arbeits- und Langzeitspeicher geschieht in ACT nur konzeptuell anhand der Aktivierung der Chunks des Langzeitspeichers. Ob und wie schnell ein Chunk mit einer Vorbedingung einer Produktion gematcht werden kann, hängt von der Aktivierung des Chunks ab, die der Wahrscheinlichkeit eines sinnvollen Gebrauchs in der gegebenen Situation entsprechen soll. Dadurch ergibt sich ein graduelles Konzept eines Arbeitsspeichers[28], in dem Information in dem Maße nutzbar ist, in dem sie aktiviert und somit nützlich ist. Ein separater Arbeitsspeicher ist nur dann sinnvoll, wenn der Umfang der in ihm enthaltenen Information deutlich geringer ist, als der des Langzeitspeichers. Weder in ACT, noch in SOAR gibt es zwar ausdrückliche Kapazitätsbeschränkungen des Arbeitsspeichers, durch die Art der Verwaltung des Arbeitsspeicherinhalts wird dessen Umfang jedoch begrenzt. Die Inhalte des Arbeitsspeichers von SOAR sind mit dem zugehörigen Problemraum verbunden und werden entfernt, sobald der Problemraum verlassen wird. Die Arbeitsspeicherinhalte von ACT verlieren ihre Aktivierung mit der Zeit, sofern sie nicht durch Verwendung oder durch Aktivierungsquellen aufrecht erhalten wird. Die übrigen Architekturen nehmen keine Trennung des Speichers für den kurzfristigen Abruf von den Langzeitspeichern vor.

[28] In der aktuellen Implementierung ist der Arbeitsspeicher eindeutig abgegrenzt, indem er auf aktivierte Chunks beschränkt ist, die einen bestimmten Aktivierungsgrad übersteigen. Dadurch braucht der Match-Algorithmus nur auf einen klar definierten Teil des deklarativen Langzeitspeichers angewendet werden.

Während SOAR nur einen einzigen Langzeitspeicher benutzt, in dem alle Arten von Information gleich gespeichert – wenn auch unterschiedlich kodiert – und aus ihm abgerufen werden, unterscheidet ACT zwischen einem deklarativen Langzeitspeicher, der Chunks enthält, und einem prozeduralen für Produktionen. Der Zugriff auf den deklarativen Speicher besteht in der Aktivierung eines Chunks durch Aktivierungsausbreitung, der Zugriff auf den prozeduralen Speicher im aktivierungsgesteuerten Matchen der Vorbedingungen. TOK unterscheidet ebenfalls zwischen einem deklarativen Speicher, dem ISM, das ein Modell der Welt enthält, und einem prozeduralen Planspeicher. PRODIGY verwendet lediglich einen zentralen Speicher, der wie der Arbeitsspeicher von SOAR als Schnittstelle zwischen den einzelnen Komponenten des Systems dient. Der Speicher von INTERRAP ist hingegen nicht inhaltlich, sondern nach Abstraktionsebenen gegliedert. Das Weltmodell umfaßt Annahmen über den Zustand der Welt, die Begriffstaxonomie und die PoB, das mentale Modell die Ziele, Pläne und Intentionen des Agenten und das soziale Modell Wissen über die Ziele und Absichten anderer Agenten sowie das Kooperationswissen. Jede der drei Kontrollebenen kann dabei nur auf die korrespondierende Schicht der Wissensbasis und auf tiefere, aber nicht auf höhere Schichten zugreifen, da nur deren Informationen von der jeweiligen Ebene auch genutzt werden können. Dies erfüllt den Zweck der Verringerung des aktuell verfügbaren Datenumfangs im Vergleich zu einem separaten Arbeitsspeicher jedoch nur geringfügig.

3.2.2. Speicherorganisation und Speicherzugriff

Neben der Aufteilung in verschiedene Speichermodule ist auch die Organisation der Information innerhalb eines Moduls wichtig, um einen möglichst effizienten Zugriff zu erreichen. Hierfür verwenden ACT und SOAR Assoziationsmechanismen, die Information aus dem Langzeitspeicher in den Arbeitsspeicher übertragen, wenn diese in der gegebenen Situation nützlich sein könnte. SOAR verwendet eine explizite Beschreibung der Situationen, indem die Erfülltheit der Vorbedingungen der Produktionen, aus denen der Langzeitspeicher besteht, im Arbeitsspeicher überprüft wird. In ACT stellt ebenfalls der Arbeitsspeicher (genauer die Aktivierungsquellen) den Kontext dar, anhand dessen neue Information aktiviert wird. Dies geschieht entlang der assoziativen Verbindungen, deren Stärke die statistische Wahrscheinlichkeit angibt, daß die assoziierte Information benötigt wird. Zusätzlich besitzen die Chunks von ACT eine Basisaktivierung, die eine situationsunabhängige Abschätzung der Nützlichkeit darstellen. In den übrigen Architekturen geschieht der Speicherzugriff entweder durch explizite Suche nach Information oder durch architekturbedingte Prozesse, wie das Matchen von Produktionenbedingungen.

Der Vorteil eines assoziativen Speicherzugriffs, wie ihn ACT und SOAR verwenden, besteht in einem bedeutungsadressierten Zugriff auf den Speicher (Newell 1990, S. 165), bei dem keine aufwendige (vollständige) Suche nach Information durchgeführt werden braucht. Statt dessen wird Information direkt durch ihren Inhalt bzw. durch die Situationen, in denen sie

benötigt wird, adressiert und aufgefunden. Zugleich dient Assoziation als eine heuristische Bewertung der Relevanz von Information (Anderson 1993, S. 51). Durch einen Assoziationsmechanismus gelangt immer nur die Information in den Arbeitsspeicher und ist somit verfügbar, die mit hoher Wahrscheinlichkeit auch nützlich ist und verwendet wird. Dies gilt vor allem für die statistisch berechneten assoziativen Verbindungen in ACT, aber durch den Chunking-Mechanismus entsteht auch in SOAR ein erfahrungsabhängiger Zusammenhang zwischen den Abrufbedingungen und dem Inhalt einer Produktion. Der situationsabhängige Speicherzugriff durch Assoziation ermöglicht jedoch nur dann eine wirkliche Effizienzsteigerung gegenüber anderen Suchverfahren, wenn der Vorteil des kontextgesteuerten Zugriffs nicht durch den hohen Aufwand des Assoziationsmechanismus wieder aufgewogen wird, was am ehesten durch massiv parallele Verarbeitung erreicht werden kann. Ein anderer Nachteil des assoziativen Speicherzugriffs ergibt sich aus der fehlenden Möglichkeit einer direkten Suche, wodurch vorhandene Information nicht genutzt werden kann, wenn keine assoziative Verknüpfung zwischen der gegebenen Situation und der Information besteht.

Eine andere Art eines aktiven Speicherzugriffs als die Assoziation ermöglichen INTERRAP und HAP. Diese Architekturen stellen Mechanismen zur Überwachung des Weltmodells bereit, die bei Veränderungen oder bei erfüllten Bedingungen den Kontrollprozeß benachrichtigen oder Aktionen ausführen.

3.3. Informationsverarbeitungsprozesse

Die einzelnen Informationsverarbeitungsprozesse integrierender Architekturen, wie sie Gegenstand dieser Arbeit sind, werden meist durch eigene Module realisiert. TOK besteht aus einer sensorischen Komponente, die ein Modell der Welt (ISM) erzeugt, einem emotionalen Modul (EM) und einem Modul zur reaktiven Planung (HAP), das zudem zur deliberativen Planung PRODIGY und zur Spracherkennung und -erzeugung GLINDA aufrufen kann. PRODIGY selbst besteht aus einer zentralen Planungskomponente, die mit mehreren Lernmodulen interagiert. Ebenfalls Komponenten für Planen durch Problemraumsuche und Lernen (Chunking) kombiniert SOAR, hinzu kommt noch die Wissenssuche durch das Produktionensystem. Die Architektur ACT beschränkt sich auf ein zentrales Produktionensystem, das auf einem deklarativen assoziativen Netz operiert. INTERRAP ist in drei Ebenen für reaktives, deliberatives und kooperatives Verhalten unterteilt, die jeweils mehrere Komponenten zur Situationserkennung und Zielauswahl (SG) sowie zum Planen und Ausführen (PS) umfassen.

Wie die Architekturen verschiedene Informationsverarbeitungsprozesse durch ihre Komponenten realisieren, ist in Tabelle 9 (s. Anhang) zusammengefaßt. Im einzelnen sind die Mechanismen für Wahrnehmung und Handeln, für deklaratives und prozedurales Lernen, für Problemlösen und Planen, für logische Schlußfolgerungen sowie für Entscheidungen aufgeführt.

3.3.1. Wahrnehmung und Handeln

Wahrnehmung und Handeln stellen die Schnittstelle zwischen dem kognitiven Agenten und seiner Umwelt dar. Aus der Wahrnehmung gewinnt er sein Wissen über die Welt, durch das Handeln kann er diese verändern. Obwohl Wahrnehmung und Handeln somit ein unverzichtbarer Bestandteil einer Agentenarchitektur sind, spielen sie in den Architekturen nur eine untergeordnete Rolle. Dies hat mehrere Gründe. Zum einen sind Architekturen recht abstrakte Modelle kognitiver Prozesse, während Wahrnehmung und Handlung stark von den Details des Anwendungsbereichs abhängen. Des weiteren werden die Agenten oft in simulierten Welten verwendet, die deren interner symbolischen Repräsentation angepaßt sind, so daß die Interaktion mit der Welt unproblematisch ist, da sie auf einem hohen Abstraktionsniveau direkt stattfinden kann. Deshalb gehen alle Architekturen davon aus, daß durch Sensoren bereits recht abstrakte, symbolische Information bereitgestellt wird, und daß die Motorik abstrakte, symbolische Befehle weitgehend eigenständig in physische Handlungen umsetzen kann (vgl. Anderson 1993, S. 78, Müller 1996, S. 50).

ACT, das sich als psychologisches Modell vor allem mit der zentralen Kognition beschäftigt, beschränkt Wahrnehmung und Handeln auf diese Annahmen. Dabei werden

Wahrnehmungen dem Arbeitsspeicher hinzugefügt und bleiben aktiv, so lange sie wahrgenommen werden. Motorische Befehle werden ausgeführt, wenn sie Teil des Aktionsteil einer Produktion sind. Ebenso verzichtet PRODIGY als reiner Planungsalgorithmus auf eigene Spezifikationen auf diesem Bereich, im Zusammenspiel mit HAP liefert dieses die notwendige Information über den Zustand der Welt und ist in der Lage, die von PRODIGY ausgearbeiteten Pläne auszuführen.

Die anderen Architekturen, die zur Gestaltung künstlicher Agenten oder Robotern gedacht sind, integrieren auch Module für die Interaktion mit einer virtuellen oder realen Umgebung. INTERRAP besitzt eine Weltschnittstelle, die für die Sensorik, die Ausführung von Handlungen und für die physischen Aspekte der Kommunikation zuständig ist. Sensoren besitzen Mechanismen zur Kalibrierung, zur Aktivierung und Deaktivierung sowie zum Lesen des aktuellen Wertes. Die sensorischen Daten werden automatisch zur Aktualisierung des Weltmodells benutzt, sie können aber auch gezielt durch Prozesse der Kontrolleinheit abgefragt werden. Handlungsprimitive stellen Mechanismen zur Kalibrierung und zur Ausführung einer Handlung bzw. zur Aktivierung, Unterbrechung oder Deaktivierung eines kontinuierlichen Vorgangs bereit. Sie erhalten ihre Anweisungen von der BBL, die die Abarbeitung der aktiven reactor und procedure PoB durchführt. Die Kommunikation besteht aus speziellen Sensoren und Handlungen, die den Austausch von Nachrichten mit anderen Agenten ermöglichen. TOK verwendet die sensorischen Daten ebenfalls sowohl zur Aktualisierung eines Weltmodells (ISM), als auch für direkte Abfragen durch die anderen Komponenten der Architektur. Die Änderungen des ISM werden von HAP dann zur Aktualisierung des Planbaums durch Entfernen erfüllter und unerfüllbarer Ziele verwendet. Handlungen werden direkt ausgeführt, sobald das entsprechende Ziel innerhalb eines Plans ausgewählt wird. SOAR nutzt den Arbeitsspeicher als zentralen Bus zwischen Wahrnehmung, Kognition und Handlung. Durch sensorische Mechanismen werden neue Elemente im Arbeitsspeicher erzeugt. Diese werden durch spezielle Kodierungsproduktionen zu abstrakteren Repräsentationen ausgearbeitet, die für eine Problemraumsuche verwendet werden können. Handlungen werden durch motorische Befehle ausgelöst, die entweder durch Wissenssuche oder durch Problemraumsuche entstanden sind. Diese Befehle werden durch Dekodierungsproduktionen interpretiert und in direkt ausführbare Teilschritte zerlegt.

3.3.2. Deklaratives und prozedurales Lernen

Da INTERRAP bislang keine Lernmechanismen integriert[29] werden in diesem Kapitel gegebenenfalls die Möglichkeiten der Erweiterung von INTERRAP durch die Lernmechanismen der anderen Architekturen diskutiert.

[29] Möglichkeiten zur Erweiterung von INTERRAP um Lernen werden in Müller (1996, S. 198f) kurz umrissen.

SOAR verwendet einen einzigen zentralen Lernmechanismus, das Chunking. Chunking geschieht permanent während der Problemraumsuche. Immer, wenn dabei eine Sackgasse erfolgreich aufgelöst wird, wird das Ergebnis der Auflösung – der neu entstandene Arbeitsspeicherinhalt – zu einer neuen Produktion. Als Abrufbedingungen erhält die Produktion alle die Inhalte des Arbeitsspeicher, die bei der Auflösung der Sackgasse verwendet wurden. Dadurch entsteht in der gleichen Situation nicht wieder eine Sackgasse, sondern die neu erstellte Produktion fügt direkt Wissen dem Arbeitsspeicher hinzu, das einen sinnvollen Planschritt darstellt oder eine sofortige Entscheidung ermöglicht. Zudem kann die Produktion auch in anderen, ähnlichen Situationen ihr Wissen bereit stellen, da nur der relevante Teil des Arbeitsspeichers als Abrufbedingung verwendet wird, wodurch eine induktive Verallgemeinerung und der Transfer von Wissen zwischen verschiedenen Aufgabenbereichen ermöglicht wird. Chunking verkürzt die Suche im Problemraum, indem es implizites Wissen, das erst durch Problemraumsuche gefunden werden kann, in explizites transferiert, das bereits während der Wissenssuche gefunden wird. Auch wenn Chunking primär ein rein prozedurales Lernen von ”Abkürzungen“ im Problemraum ist, so lassen sich auch andere Lernverfahren bis hin zu deklarativem Lernen durch ihn realisieren – wenn auch nicht immer ganz unproblematisch (vgl. das Data-Chunking-Problem, Kapitel 2.4.6.). Da der Chunking-Mechanismus von SOAR sehr von dessen Konzeption als zentrale Problemlösearchitektur in Verbindung mit einem produktionenbasierten Assoziationsmechanismus abhängt, ist eine Übertragung auf INTERRAP höchstens im Rahmen der – in INTERRAP bislang ohnehin nicht genauer spezifizierten – Planungsalgorithmen möglich.

ACT unterscheidet vier Lernarten, ausgehend vom Lernen deklarativen Wissens, der prozeduralen Interpretation und der Prozeduralisierung dieses Wissens, bis hin zur Abstimmung der Anwendung des prozeduralen Wissens. Das deklarative Lernen besteht dabei nicht im bloßen Abspeichern der Information – schließlich wird alle aufgenommene Information permanent –, sondern im Lernen der Relevanz der Information, im Lernen der Umstände, in denen das Wissen benötigt wird und durch den Assoziationsmechanismus bereitgestellt werden soll (Anderson 1993, S. 71). Dies geschieht durch die statistische Justierung der Basisaktivierung und der assoziativen Verbindungen der Chunks in Abhängigkeit von deren Verwendung (s. Tabelle 2, Kapitel 2.2.2.). Da INTERRAP keinen Assoziationsmechanismus verwendet, läßt sich diese Art des Lernens oder andere Arten deklarativen Lernens, die über ein reines Aufbewahren der Information hinausgehen, nicht in INTERRAP realisieren.

Die zweite Lernmethode von ACT ist die Verwendung deklarativ gespeicherter Beispiele als Analogie beim Problemlösen. Dabei wird die Struktur des Beispiels auf das aktuelle Problem abgebildet und der Lösungsweg des Beispiels auf dieses übertragen. Diese Analogien können in einem weiteren Schritt auch zu neuen Produktionen generalisiert werden, worin die einzige Möglichkeit des Erstellens neuer Produktionen in ACT besteht. Diese Art des Lernens kann in INTERRAP auf den Planungsebenen realisiert werden, indem neu gefundene Problem-

lösungen in der Planbibliothek gespeichert und gegebenenfalls auch generalisiert werden. Auf der BBL ist ein dem Lernen in ACT vergleichbares Verfahren jedoch nicht möglich, da die einzelnen PoB in sich geschlossene Einheiten bilden, während sich die Produktionen in ACT modular ergänzen und dadurch Transfer und Kombination verschiedener Verhaltensweisen ermöglichen.

Die letzte Lernart von ACT besteht in der statistischen Anpassung der Parameter zur Produktionenauswahl. Dazu werden die Stärke der Produktionen sowie deren Bewertung in Abhängigkeit von der Häufigkeit des Gebrauchs und des Erfolgs justiert. Eine ähnliche Funktion wie diese Parameter besitzen die Prioritäten der PoB in INTERRAP, die auf eine ähnliche statistische Art ihrer Nützlichkeit entsprechend verändert werden könnten.

Die Lernmodule von PRODIGY erfüllen drei Funktionen. Zum einen gibt es Module zur Verbesserung des Domänenwissens, indem abhängig von Beobachtungen und Fehlern Planoperatoren abgeändert werden. Des weiteren kann die Effizienz der Problemraumsuche durch Lernen neuer Kontrollregeln gesteigert werden, indem durch Analyse von Problemlösungen neue bereichsspezifische Regeln erstellt werden. Dabei werden alle Entscheidungspunkte des Planungsalgorithmus als Lernmöglichkeiten angesehen. Schließlich besteht noch die Möglichkeit, die Qualität der gefundenen Pläne zu steigern, indem die Planbewertungsfunktion in Kontrollwissen umgewandelt wird. Alle diese Lernverfahren beziehen sich auf einen Planungsalgorithmus, der jedoch bislang kein fester Bestandteil von INTERRAP ist, so daß eine Integration in INTERRAP von der Auswahl eines Planers abhängt.

3.3.3. Problemlösen und Planen

Unter Problemlösen versteht man alle kognitiven Tätigkeiten, mit denen Handlungen zum Verändern der Welt auf ein bestimmtes Ziel oder einen Zweck hin erzeugt werden (Barsalou 1992, S. 322). Da das Handeln kognitiver Agenten immer zweckgerichtet ist (vgl. Kapitel 1.1.2.), läßt sich alle kognitive Aktivität, sofern sie sich mit Handlungen befaßt, als das Lösen von Problemen auffassen (Anderson 1988, S. 188).

An Problemarten lassen sich Transformationsprobleme, Gruppierungsprobleme und Verstehensprobleme unterscheiden (Barsalou 1992, S. 323). Bei Transformationsproblemen wird eine Handlung oder eine Folge von Handlungen gesucht, um die Welt von einem gegebenen Ausgangszustand in einen klar definierten Zielzustand zu überführen. Dagegen soll bei Gruppierungsproblemen eine Anordnung so verändert werden, daß sie bestimmten Voraussetzungen genügt, ohne daß das Ziel jedoch eindeutig festgelegt zu sein braucht. Verstehensprobleme betreffen nicht die Ausführung von Handlungen, sondern das Erfassen eines Gegenstandbereichs, indem dessen Verhalten durch Regeln zur Vorhersage beschrieben wird, und stellen somit eine Form von Lernen dar. Die betrachteten Architekturen beschränken sich vor allem auf Transformationsprobleme, die durch Planen zu lösen versucht werden. Lediglich

PRODIGY besitzt mit EBL und STATIC Lernmodule, die sich mit dem Verstehen eines Gegenstandsbereiches befassen.

Problemlösen durch Planen kann auf verschiedenen Ebenen durchgeführt werden. Bei reaktiver Planung wird durch wahrgenommene Situationen direkt eine angemessene Reaktion ausgelöst, die einem permanenten Ziel des Agenten dient. Situative Planung berücksichtigt jeweils die gegenwärtigen Situationen, um neue Handlungen zum Erreichen eines Ziels auszuwählen, wird dabei jedoch primär von dem Ziel oder einem abstrakten Plan für dieses Ziel geleitet. Deliberative Planung schließlich geschieht weitgehend unabhängig von äußeren Veränderungen. Dabei wird durch Problemraumsuche eine Folge von Handlungen gesucht, um von einem festen Ausgangszustand zum Ziel zu gelangen, das meist ebenfalls durch einen eindeutigen Zustand der Welt charakterisiert wird.

Diese drei Ebenen des Planens ergeben sich daraus, inwieweit bei der Planung äußere Umstände und inwieweit interne Zielsetzungen des Agenten berücksichtigt werden. Während reaktive Planung fast ausschließlich auf der externen Situation beruht und Ziele lediglich implizit in den ausgelösten Reaktionen enthalten sind, wird deliberative Planung durch das interne Ziel gesteuert, während die Welt abgesehen von eigenen Handlungen als unverändert gegenüber dem Ausgangszustand angenommen wird. Situative Planung orientiert sich an den ständigen Veränderungen der Welt, wird dabei aber auf ein spezielles Ziel hin gesteuert. Diese drei Arten des Planens schließen einander nicht aus, sie sollten sogar innerhalb einer einzigen Architektur kombiniert werden, da jede für verschiedene Zwecke dienlich ist. Reaktive Planung vernachlässigt zwar temporäre Absichten und Ziele eines Agenten, dafür kann sie sehr schnell und direkt wirksam werden. Deliberative Planung hingegen ermöglicht auch das Erstellen komplexer Handlungssequenzen auf ein Ziel hin, während durch reaktive Planung nur einfache Ziele erreicht werden können. Situative Planung stellt einen Kompromiß zwischen reaktiver und deliberativer Planung dar, indem zwar längere Handlungssequenzen auf ein Ziel hin möglich sind, diese aber zugleich direkt und situationsabhängig ausgeführt werden können, ohne daß ein vollständiger Plan für das Ziel vorzuliegen braucht. Die Kombination von deliberativer Planung und situativer Ausführung der deliberativ erstellten Pläne ermöglicht zudem, die für Änderungen der Umgebung anfällige deliberative Planung auf einem hohen Abstraktionsniveau durchzuführen, auf dem keine Änderungen zu erwarten sind, und situativ geringfügigere Abweichungen bei der Ausführung zu korrigieren, ohne neu planen zu brauchen.

Reaktive Planung

Da reaktive Planung im direkten Auslösen von Handlungen durch externe Ereignisse besteht, findet dabei keine Planung im Sinne der Konstruktion einer Handlungssequenz als Lösung für ein Problem statt. Sie beschränkt sich statt dessen auf das schnelle Erkennen von Situationen,

die einer sofortigen Reaktion bedürfen, sowie auf die Entscheidung, ob und wie reagiert werden soll, also auf die reine Auswahl einer Handlung.

INTERRAP stellt hierfür reactor PoB bereit, die erkannte Situationen direkt mit Handlungen verbinden. Sobald die Bedingung eines reactor PoB in der Wissensbasis erfüllt ist, wird dieser aktiviert. Aus der Menge aller aktiven PoB wird anhand derer Priorität, die der Wichtigkeit einer Reaktion entspricht, eine Menge gleichzeitig ausführbarer PoB ausgewählt und direkt ausgeführt (vgl. Kapitel 2.1.3.).

In den übrigen Architekturen ergibt sich reaktive Planung nur als Spezialfall der situativen bzw. deliberativen Planung. Die reaktive Planung in TOK wird durch das Modul HAP vorgenommen. Dabei werden durch Dämonen externe Situationen erkannt und durch diese neue Ziele zu dem APT an der Spitze hinzugefügt. Besitzen diese Ziele eine genügend hohe Priorität, so werden sie direkt ausgeführt, was zu einer sofortigen Handlung führt, falls der für das Ziel ausgewählte Plan eine ausführbare Handlung darstellt. Dieses Vorgehen ähnelt dem von INTERRAP, insofern durch produktionenartige Strukturen Ziele durch wahrgenommene Situationen aktiviert werden und dann anhand einer Priorität eine Handlung zur direkten Ausführung ausgewählt wird.

Reaktive Planung in SOAR bedeutet Planung ohne Problemraumsuche. Statt dessen werden während der Wissenssuche durch Produktionen direkt ein Befehl für die auszuführende Handlung sowie Präferenzen, diese sofort auszuführen, in den Arbeitsspeicher geschrieben.

Auch wenn der Theorie ACT gemäß reaktive Planungsvorgänge nicht unbedingt durch die zentrale Kognition, die allein von ACT modelliert wird, ausgeführt werden, so ist doch eine Realisierung durch Produktionen, die Situationen direkt mit Handlungen verbinden, auch hier möglich[30].

Situative Planung

Eine situative Planung, die zwar zielgerichtet, aber dennoch situationsgesteuert ist, besitzen nur TOK und INTERRAP, wenn auch auf recht verschiedene Weise. Um bei der Ausführung der von einer der Planungsebenen erstellten deliberativen Pläne Änderungen der Umgebung dynamisch handhaben zu können, sind die procedure PoB – aus denen deliberative Pläne in INTERRAP bestehen – so programmiert, daß sie – beispielsweise durch situationsabhängige Verzweigungen – auch bei Änderungen der Umgebung ihre Aufgabe erfüllen können.

Statt isolierter Verhaltensmuster verwendet HAP für die situative Planung ein modulares System. Die Pläne der Planbibliothek bestehen jeweils aus Mengen von Handlungsprimitiven und Zielen, die sequentiell oder parallel auszuführen sind. Für jedes Ziel gibt es einen oder auch mehrere Pläne, aus denen anhand der Situation und einer Priorität ausgewählt wird. Da der Plan für ein Ziel weitere Ziele enthalten kann, besteht so die Möglichkeit, die weiteren Schritte

[30] Werden in ACT Produktionen für Reaktionen verwendet, ergibt sich das Problem, daß Produktionen zielabhängig sind, während die Reaktionen möglichst unabhängig von temporären Zielen sein sollten.

auf ein Ziel hin erst während der Ausführung des Plans situativ auszuwählen. Ein Ziel stellt somit eine abstrakte Handlungsanweisung dar, die bei Bedarf durch einen oder mehrere in der Planhierarchie des APT aufeinanderfolgende Pläne detaillierter ausgearbeitet wird. Dadurch werden immer nur die als nächstes auszuführenden Planschritte geplant, die übrigen bleiben zunächst abstrakt. Anders als bei abstrakter deliberativer Planung wird so der detailliertere Plan erst situationsabhängig während und abhängig von der Ausführung erstellt. Dadurch kann die Planung zugleich flexibel und dynamisch und dennoch zielgerichtet erfolgen.

Deliberative Planung

Deliberative Planung besteht in der Suche innerhalb eines Problemraums nach einer Lösung für ein Problem. Ein Problemraum besteht aus einer Menge von möglichen Weltzuständen, die ausgehend von einem Anfangszustand durch sukzessive Anwendung von Operatoren, die Handlungen in Form von Zustandsänderungen beschreiben, erreicht werden können. Ein Problem ist ein Zielzustand, der in diesem Problemraum gefunden werden muß, und die Lösung ist ein Plan, der aus der Folge von Operatoren besteht, mit denen vom Anfangszustand aus der Zielzustand erreicht wird. Befindet sich die Welt im Ausgangszustand der Planung, so kann das Ziel erreicht und damit das Problem gelöst werden, indem nacheinander für jeden Operator eine entsprechende Handlung ausgeführt wird – allerdings nur unter der Voraussetzung, daß keine für die Planausführung relevanten Teile der Umgebung sich in unvorhergesehener Weise ändern und jeder einzelne Schritt erfolgreich ausgeführt werden kann.

SOAR und PRODIGY sind reine Problemlösearchitekturen, da ihr zentraler Kontrollzyklus aus einer Problemraumsuche besteht. In ACT kann Problemraumsuche durch die Konstruktion einer Zielstruktur realisiert werden, wobei Produktionen als Planoperatoren fungieren. Für die Planungsebenen von INTERRAP ist bislang kein fester Planungsalgorithmus definiert. Eine Alternative zur Problemraumsuche stellt der Abruf eines fertigen Plans aus einer Planbibliothek dar. Diese Möglichkeit besteht in INTERRAP und PRODIGY, aber auch in ACT und SOAR können Pläne als deklarative Datenstruktur gespeichert und assoziativ abgerufen werden. Eine dritte Art, Pobleme zu Lösen, ist Analogie. In ACT und PRODIGY können Beispiele erfolgreicher Problemlösungen dazu verwendet werden, bei ähnlichen Problemen als Anleitung zur Konstruktion einer Lösung zu dienen, indem die Struktur des Problems auf den bekannten Lösungsweg abgebildet wird.

Um ein Problem durch Planung in Form von Problemraumsuche zu lösen, muß zunächst ein Problemraum ausgewählt werden, indem der Anfangszustand und der Endzustand eindeutig definiert werden und eine Menge von Planoperatoren bestimmt wird. Anschließend wird der Problemraum durchsucht, bis ein Zielzustand gefunden ist oder aber der Problemraum vollständig durchsucht ist. Da der Problemraum im allgemeinen sehr groß ist und die Dauer der Suche durch ungeeignete Operatorauswahl exponentiell steigen kann, ist die Kontrolle der Suche für eine effiziente Suche unabdingbar. Dazu können sowohl allgemeine Suchstrategien,

als auch problemspezifisches Kontrollwissen verwendet werden. In PRODIGY und SOAR wird beides durch explizite Kontrollregeln bzw. Präferenzen erreicht, die Entscheidungen auswählen, bevorzugen oder ausschließen (vgl. Kapitel 3.3.5.) und durch Erfahrung entstehen. Da durch die Suche im Problemraum nicht zwangsläufig eine optimale Lösung gefunden wird, ermöglichen INTERRAP und PRODIGY die Bewertung von Plänen anhand einer Bewertungsfunktion, die den Aufwand für einen Plan aufgrund der vorkommenden Operatoren berechnet.

Planung

Nur INTERRAP berücksichtigt ausdrücklich alle drei Arten des Planens. Die reactor PoB der BBL übernehmen die reaktive Planung und die Planungsebenen, LPL und CPL, erstellen deliberativ abstrakte Pläne, die situativ durch die procedure PoB der BBL ausgeführt werden. Zudem berücksichtigt INTERRAP als einziges das Erstellen und Verwenden von Plänen, an deren Ausführung mehrere Agenten beteiligt sind. Prinzipiell lassen sich diese Pläne jedoch durch jeden hinreichend allgemeinen Planungsalgorithmus realisieren, in sofern andere Agenten einfach als besondere, sehr komplizierte Objekte aufgefaßt werden. Faßt man reaktive Planung als Sonderfall der situativen Planung in HAP auf, bei der durch ein permanentes Ziel direkt eine ausführbare Handlung als Plan aus der Bibliothek abgerufen wird, so ergibt sich zusammen mit dem deliberativen Planer PRODIGY auch in TOK das gleiche Spektrum an Planung. ACT und SOAR hingegen sind rein deliberative Planer, durch deren Produktionen sich jedoch auch eine reaktive Planung realisieren läßt.

3.3.4. Logisches Schließen

An logischen Schlußfolgerungen wird zwischen der Deduktion, die nach logischen Regeln sichere Konsequenzen aus dem vorhandenen Wissen zieht, und der Induktion unterschieden, bei der Einzelfälle verallgemeinert werden.

Deduktion

Deduktive Schlüsse können als ”wenn ... dann“-Regeln aufgefaßt werden und sind insofern vergleichbar mit Produktionen. Während Produktionen jedoch bedingte Aktionen sind, die ausgeführt werden können, sobald die Vorbedingungen erfüllt sind, müssen deduktive Schlüsse ausschließlich wahre Folgerungen haben, die sich aus den Bedingungen rein logisch ergeben. Deduktive Schlüsse sind im Gegensatz zu Produktionen immer semantisch wahrheitserhaltend.

Da alle Architekturen produktionenartige Strukturen enthalten, können sie diese auch für deduktive Schlüsse verwenden. In ACT sind dies die Produktionen des prozeduralen Speichers, in SOAR die des assoziativen Langzeitspeichers. INTERRAP verwendet reactor PoB für Deduktionen, die bei erkannter Situation neues Wissen zur Wissensbasis hinzufügen. In entsprechender Weise können in TOK Dämonen oder mentale Handlungen von HAP genutzt werden. Da auch die Struktur von Planoperatoren denen von deduktiven Folgerungen ähneln,

gebraucht PRODIGY für Deduktionen Inferenzregeln, die Planoperatoren entsprechen, mit denen keine physische Handlung verbunden ist.

In allen Architekturen stellen deduktive Schlußfolgerungen somit lediglich einen Spezialfall prozeduralen Wissens dar, ihr semantischer Gehalt als garantiert wahrheitserhaltender Schluß bleibt implizit. Für die Wissensbasis von INTERRAP ist zumindest eine Erweiterung der AKB um deduktive Schlüsse vorgesehen (Weiser 1995), in der deduktive Ableitungen nicht mehr als Wissen unter anderem, sondern gesondert behandelt werden und gegebenenfalls zusammen mit den Fakten, von denen sie abhängen, wieder entfernt werden.

Induktion

Im Gegensatz zu deduktiven Schlüssen sind induktive Folgerungen unsicher, da von einzelnen Fällen auf den allgemeinen Fall generalisiert wird, wodurch Annahmen über unbekannte Fälle gemacht werden, die sich nachträglich als ungerechtfertigt herausstellen können. Explizite induktive Schlüsse kommen in keiner Architektur vor, durch Lernen entsteht jedoch oft eine Generalisierung, die einer impliziten Induktion gleichkommt.

So wird beim Lernen durch Analogie in ACT die Problemlösestruktur von Beispielen generalisiert, indem konstante Werte durch Variablen ersetzt werden. Dabei genügt es, daß das Beispiel erfolgreich auf einen einzigen anderen Fall übertragen wurde, die Variablen umfassen jedoch meist noch weit mehr Fälle. Generalisierungen in SOAR geschehen durch den Chunking-Prozeß, da die neu erstellte Produktion in ihren Vorbedingungen nicht die gesamte Situation, sondern nur den für die Auflösung der Sackgasse relevanten Teil des Arbeitsspeichers berücksichtigt. Dadurch wird die Produktion auch in Fällen anwendbar, in denen durch die Teile des Arbeitsspeichers, die im Bedingungsteil der Produktion nicht vorkommen, die Auflösung der Sackgasse zu einem anderen Ergebnis gekommen wäre. Auch das Lernen von Kontrollregeln in PRODIGY führt zu Verallgemeinerungen und damit zu induktiven Schlüssen.

Diese Formen der Induktion sind jedoch nicht ganz unproblematisch. Die Gefahr eines Fehlschlusses besteht zwar bei Induktion immer, diese Induktionen geschehen jedoch ohne weitere Rechtfertigung und in SOAR sogar rein zufällig. Des weiteren genügt bereits ein einziger Fall, damit die Generalisierung durchgeführt wird, eine recht dürftige Grundlage für einen induktiven Schluß. Überprüft wird die Korrektheit der Induktion nur insofern, als daß die neuen Regeln den üblichen Auswahlverfahren unterliegen und bei Mißerfolg durch Verringerung der Produktionenstärke (ACT) oder durch neue Präferenzen (SOAR) wieder ausgeschlossen werden können.

3.3.5. Entscheiden

Die Entscheidungsfindung – die Auswahl bei mehreren für eine Situation sinnvollen Alternativen – betrifft sowohl die interne Steuerung kognitiver Prozesse, als auch die Auswahl von Handlungen. Die Auswahl aus verschiedenen Möglichkeiten geschieht statisch, wenn dabei

ausschließlich Eigenschaften der Alternativen selbst berücksichtigt werden, und ist um so dynamischer, je mehr andere Faktoren einen Einfluß auf die Entscheidung ausüben. Diese können dabei von dem Wissen des Agenten – insbesondere über die gegenwärtige Situation – sowie seinen Zielen und Absichten, in manchen Architekturen auch von dessen emotionalen Zustand oder seinen Erfahrungen abhängen. Die zur Entscheidungsfindung verwendete Information kann explizit repräsentiert werden, indem sie in symbolischer Form Regeln zur Auswahl angibt, oder auch implizit, indem anhand numerischer Werte Alternativen bewertet werden.

Die Auswahl von Handlungen geschieht in allen Architekturen bei Planungsvorgängen (s. Kapitel 3.3.3.) oder bei der direkten Aktivierung von Handlungen durch externe Ereignisse. Letzteres ist in INTERRAP die Aufgabe der reactor PoB, kann aber auch in den anderen Architekturen mit Hilfe von Produktionen realisiert werden.

Entscheidungen in SOAR werden ausschließlich durch Präferenzen getroffen, die explizit Alternativen befürworten bzw. ausschließen oder untereinander vergleichen, indem sie durch ”besser“, ”indifferent“ oder ”schlechter“ miteinander in Relation gesetzt werden. Ähnlich verwendet auch PRODIGY Kontrollregeln, die mit Hilfe von ”SELECT“, ”REJECT“ und ”PREFER“ (vgl. Kapitel 2.3.4.) Möglichkeiten bewerten und vergleichen. Diese Entscheidungen von SOAR und PRODIGY sind insofern dynamisch, als sie durch Produktionen kontextabhängig aktiviert werden und lediglich Richtlinien darstellen, die durch andere Regeln in manchen Situationen wieder abgeschwächt werden. Des weiteren lassen sich derartige explizite Regeln leicht durch neue Regeln erweitern, die aus der Erfahrung gewonnen werden.

Im Gegensatz dazu verwenden die übrigen Architekturen numerische Werte zur Auswahl, die direkt mit den prozeduralen Datenstrukturen verknüpft sind. In HAP, dem reaktiven Planer von TOK, und auf der BBL von INTERRAP geschieht dies anhand von festen Prioritäten, die die Pläne von HAP bzw. die PoB von INTERRAP besitzen. Ausgewählt wird jeweils die Alternative mit der höheren Priorität unabhängig vom Kontext (der selbstverständlich beim Aufstellen der Alternativen bereits relevant ist). Durch diese festen Prioritäten wird zwar garantiert, daß wichtigere Aktivitäten immer Vorrang haben – in INTERRAP besitzen die reactor PoB höhere Prioritäten als die procedure PoB –, die Entscheidungen werden jedoch recht unflexibel und statisch, da weder die aktuelle Situation, noch Erfahrungen berücksichtigt werden.

Daß auch ein implizites numerisches Entscheidungsverfahren dynamisch sein kann, beweist ACT. Die Konfliktauflösung – die Auswahl einer Produktion aus allen möglichen Instantiierungen – geschieht hier anhand der numerischen Werte der Basisaktivierungen und der assoziativen Verbindungen der Chunks sowie der Stärke und des Wertes der Produktionen (vgl. Kapitel 2.2.3.). Da diese Werte alle statistisch gelernt werden, spiegeln sie die Erfahrung des Agenten wider. Der Kontext wird durch die Aktivierung der Chunks berücksichtigt, die um so schneller als Vorbedingung instantiiert werden können, je aktiver sie sind. Neben der

Geschwindigkeit der Produktioneninstantiierung ist der Wert einer Produktion bei der Auswahl entscheidend, bei dem statistisch abgeschätzt wird, ob die Erfolgswahrscheinlichkeit einer bereits vollständig instantiierten Produktion und ihr Nutzen so groß sind, daß der Aufwand zur Instantiierung weiterer und eventuell besserer Produktionen sich nicht mehr lohnt (Anderson 1993, S. 57). Dadurch wird zwar nicht zwangsläufig das beste Ergebnis erreicht, aber das Verhältnis von Aufwand und Nutzen optimiert.

Eine ähnliche Bewertung anhand von Aufwand und Nutzen von Handlungen ist in INTERRAP und PRODIGY möglich, wo vollständige Pläne anhand der Kosten für die einzelnen, durch Operatoren repräsentierten Planschritte bewertet werden können. Dies dient einerseits der Auswahl aus einer Menge von mehreren alternativen Plänen der Planbibliothek für ein Ziel, andererseits der Entscheidung, ob ein erstellter Plan verwendet oder nach besseren Alternativen gesucht werden soll. Die Suche nach einem besseren Plan kann in INTERRAP auch die Übergabe an eine höhere Planungsebene bedeuten, statt einfacher lokaler Planung wird dann ein kooperativer Plan unter Mitwirkung anderer Agenten zu erstellen versucht.

3.4. Die Gesamtstruktur der Architektur

Die Struktur einer integrierenden Architektur ergibt sich zum einen aus den integrierten Komponenten und deren Interaktionsmöglichkeiten, zum andern aus den verschiedenen Kontrollprozessen, die die Interaktionen der Komponenten steuern und organisieren. Tabelle 10 (s. Anhang) stellt die Konzepte zur Gesamtstruktur der Architekturen einander gegenüber, wobei TOK und PRODIGY teils getrennt, teils gemeinsam behandelt werden. Zum Aufbau der Architekturen werden die einzelnen integrierten Komponenten und die Mechanismen der Interaktion zwischen diesen aufgeführt. Zur Kontrollstruktur sind das Grundprinzip der Kontrolle, der Kontrollfluß zwischen den Komponenten sowie der Kontrollzyklus der Architekturen einander gegenübergestellt.

3.4.1. Integration der Komponenten

Vom Aufbau her die einfachste Architektur ist ACT, sie besteht lediglich aus einem assoziativen Netz als deklarativem Speicher und einem Produktionensystem mit prozeduralem Speicher. Diese beiden Komponenten interagieren durch Aktivierungsausbreitung miteinander. Die Elemente des deklarativen Speichers besitzen einen Grad an Aktivierung, der bestimmt, wie schnell die Produktionen diese beim Matchen der Bedingungen instantiieren können. In der anderen Richtung verändert der Aktionsteil einer ausgewählten Produktion den deklarativen Speicher, indem Chunks neu hinzugefügt oder aktiviert werden. ACT besteht somit aus zwei Komponenten, einem deklarativen Speicher und einem Produktionenspeicher, die durch Produktionen-Matchen und -Ausführen miteinander interagieren. Die Interaktion mit der Umgebung durch Wahrnehmung und Handeln geschieht über den deklarativen Speicher, ist jedoch kein Teil der ACT-Architektur.

SOAR verwendet einen zentralen Arbeitsspeicher als Datenbus zwischen der Wahrnehmung, dem assoziativen Langzeitspeicher, dem Problemlöser und dem Handeln. Neue Information gelangt entweder von außen durch die Wahrnehmung oder aus dem Langzeitspeicher durch dessen Produktionen in den Arbeitsspeicher. Diese Information verwendet der Problemlöser für die Problemraumsuche, bei der durch Operatoranwendung neue Planzustände in den Arbeitsspeicher geschrieben werden. Die Handlungskomponente entnimmt dem Arbeitsspeicher Befehle, die sie ausführt. Der einzige Informationsaustausch in SOAR, der nicht über den Arbeitsspeicher verläuft, ist die Generierung neuer Produktionen durch das Chunking, was zugleich die einzige Möglichkeit des Speicherns von Information im Langzeitspeicher darstellt. Der Arbeitsspeicher von SOAR dient damit als zentrale Stelle für die Integration und Interaktion der Komponenten, wodurch auch eine einfache Erweiterbarkeit der Architektur durch

weitere Module ermöglicht wird, die dem Arbeitsspeicher Wissen entnehmen und neues hinzufügen, wo es für alle anderen Komponenten erreichbar ist.

An Komponenten integriert TOK die Sensorik, das Wahrnehmungsmodell ISM, den reaktiven Planer HAP und für die Emotionen EM, hinzu kommt noch der deliberative Planer PRODIGY. Die Integration geschieht nicht nach einem bestimmten Schema, sondern rein funktional, d.h. es sind jeweils die Komponenten verbunden, für die eine Interaktion sinnvoll ist. Die Sensorik aktualisiert das ISM, dessen Informationen EM und HAP jeweils zur Aktualisierung ihres Zustandes verwenden. EM und HAP interagieren, indem HAP EM über das Aufstellen von Zielen sowie deren Erfolg oder Mißerfolg unterrichtet, während der emotionale Zustand von EM die Entscheidungen von HAP beeinflußt. Schließlich kann HAP noch den deliberativen Planer PRODIGY aufrufen. Dies geschieht als gewöhnliche Handlung von HAP, die als Teil eines Plans für ein Ziel zum APT hinzugefügt wurde. PRODIGY wird eine Menge von Zielen mit Prioritäten sowie ein Zeitlimit übergeben, nach dem es einen Plan für möglichst viele Ziele mit möglichst hoher Priorität zurückgeben soll. Der zurückgegebene Plan besteht wie alle Pläne von HAP aus einer Menge von Handlungen oder Teilzielen. Die Interaktion zwischen HAP und PRODIGY soll dabei auf einer Abstraktionsebene stattfinden, auf der die Welt während der Dauer der Planung und während der Ausführung des Plans als statisch angesehen werden kann. Entsprechend besteht der zurückgegebene Plan nicht aus einer Menge von Handlungen, sondern aus Zielen für HAP, die bei der Ausführung situativ erweitert werden, so daß auch die dynamischen Aspekte der Umgebung berücksichtigt werden.

PRODIGY selbst integriert Planen und Lernen miteinander. Auch hier geschieht die Integration nach funktionalen Aspekten, wobei der Planungsalgorithmus jedoch eine zentrale Position einnimmt. Die Lernkomponenten nutzen den Planungsverlauf und externe Eingaben um neues Domänen- oder Kontrollwissen zu erstellen, das wiederum der Planer zur Planung verwendet.

Anders als bei den übrigen Architekturen geschieht die Integration in der geschichteten Architektur INTERRAP auf drei verschiedene Arten, einmal durch die Module, zweitens durch die einzelnen Ebenen in den Modulen und zum dritten innerhalb jeder Ebene. Zunächst läßt sich INTERRAP in die Weltschnittstelle, die Wissensbasis und die Kontrolleinheit unterteilen. Die Weltschnittstelle dient der Interaktion mit der Umgebung, die Wissensbasis enthält das Wissen des Agenten, und die Kontrolleinheit leitet aus dem Wissen und den Zielen des Agenten Handlungen her. Die Weltschnittstelle umfaßt Wahrnehmung und Handeln, wozu auch die physischen Aspekte der Kommunikation gehören. Wissensbasis und Kontrolleinheit sind jeweils in drei korrespondierende Ebenen aufgeteilt. Die Kontrolleinheit gliedert sich in die verhaltensbasierte Ebene (BBL), die lokale Planungsebene (LPL) und die kooperative Planungsebene (CPL), die zugehörigen Ebenen der Wissensbasis sind das Weltmodell (WM), das mentale Modell (MM) und das soziale Modell (SM) (vgl. Abbildung 2, Kapitel 2.1.1.). Jede Ebene der Kontrolleinheit kann dabei auf die Informationen der zugeordneten Ebene der

Wissensbasis und auf die darunterliegenden, jedoch nicht auf höhere zugreifen, da nur diese Informationen auf der jeweiligen Ebene sinnvoll verarbeitet werden können.

Die drei Kontrollebenen sind hierarchisch angeordnet. Nur die unterste, die BBL, ist mit der Weltschnittstelle verbunden. Ansonsten kommuniziert jede Ebene mit der darüber- und der darunterliegenden – soweit vorhanden – durch den Austausch von Nachrichten. Die wichtigsten Nachrichten sind die nach oben gesandten Aufforderungen zur Planung und die nach unten übergebenen Verpflichtungen zur Ausführung. Durch diese geschichtete Struktur wird eine wahrgenommene Problemsituation so lange in der Ebenenhierarchie nach oben gereicht, bis eine kompetente Ebene zur Lösung gefunden ist. Einfache Reaktionen werden dabei von der BBL vorgenommen, zielgerichtete Planungen, die nur den Agenten selbst betreffen, übernimmt die LPL und Planungen für mehrere Agenten die CPL. Dadurch übernimmt jede Ebene eine spezifische Funktion in der Architektur, zugleich steigt auch die Abstraktion und Komplexität des Wissens und der Probleme. Zur Ausführung werden Handlungsanweisungen nach unten bis zur BBL weitergereicht, wo sie situativ durch procedure PoB ausgeführt werden.

Die dritte Integrationsstufe in INTERRAP besteht innerhalb der einzelnen Ebenen. Auch wenn jede Ebene eine andere Funktion hat, so besteht doch die Aufgabe jeder Ebene im Aufstellen von Handlungen für zuvor erkannte Situationen und Probleme, wenn auch auf verschiedenen Abstraktionsstufen. Deshalb ist es möglich und auch sinnvoll, den Aufbau aller Ebenen von der Struktur her gleich zu gestalten. Jede INTERRAP-Ebene besteht aus fünf aufeinanderfolgende Grundfunktionen, die auf jeder Ebene jeweils anders instantiiert werden. Die *Situationserkennung* strukturiert das Wissen des Agenten zu bekannten Situationen, die die *Zielaktivierung* zum Aufstellen neuer Ziele verwendet. Für diese Ziele erstellt ein *Planer* Pläne, die der *Scheduler* zeitlich miteinander koordiniert, dessen Ergebnis schließlich durch ein *Ausführungsmodul* in Handlungen umgesetzt wird.

In den integrierten Komponenten stimmen die betrachteten Architekturen noch weitgehend überein, diese ergeben sich jedoch bereits aus der in Kapitel 1.1.2. gegebenen Charakterisierung von Kognition als zweckgerichteter Informationsverarbeitung zur Verhaltenssteuerung. Für ein der Umgebung angemessenes Verhalten sind allein zur Interaktion mit der Umgebung Module zur Wahrnehmung und für (physische) Handlungen notwendig. Diese Komponenten sind in allen Architekturen als weitgehend eigenständige Module realisiert, die der zentralen Kognition Informationen über die Umgebung bereitstellen und deren Anweisungen für Handlungen umsetzen. Diese Module sind in ihrer Verarbeitungskapazität und -geschwindigkeit von der zentralen Kognition unabhängig, so daß die Wahrnehmungs- und Handlungsfähigkeit nicht durch aufwendige zentrale Prozesse negativ beeinflußt werden kann.

Für die zweckgerichtete Informationsverarbeitung schließlich werden Komponenten zur Informationsspeicherung sowie zur eigentlichen Verarbeitung der Information benötigt. ACT und SOAR beschränken sich dabei auf zentrale Speicher und Problemlöser, hinzu kommen noch die Lernmechanismen, die jedoch nicht durch eigene Module realisiert sind. Auch in TOK

geschieht die Integration weitgehend zentral. Um den reaktiven Planer HAP herum sind das ISM, die Emotionskomponente EM und der deliberative Planer PRODIGY – selbst eine zentral um den Problemlöser organisierte Architektur – gruppiert. In INTERRAP sind hingegen Informationsspeicherung und -verarbeitung geschichtet angeordnet, die Integration geschieht hierarchisch. Jede Ebene besitzt die Verantwortung für eine bestimmte Funktionalität, im einzelnen für reaktive, deliberative und kooperative Planung. Dadurch wird eine gewisse Eigenständigkeit der Ebenen erreicht, wie sie auch in den anderen Architekturen zwischen der zentralen Kognition und den Modulen für Wahrnehmung und Handeln besteht.

3.4.2. Kontrollstruktur

Die Kontrollstruktur bestimmt den Ablauf von Informationsverarbeitungsprozessen innerhalb einer Architektur. Wie beim strukturellen Aufbau der Architekturen ergibt sich auch dabei ein allen Architekturen gemeinsames Grundschema, das sich aus der Charakterisierung von Kognition ergibt. Dieses Schema besteht aus folgenden fünf Schritten:

1. Wissen aktualisieren,
2. Situation erkennen,
3. Handlungsmöglichkeiten finden,
4. Handlung auswählen,
5. Handlung ausführen.

Im ersten Schritt wird das verfügbare Wissen anhand von Wahrnehmung und Erinnerung aktualisiert und dann zu bekannten Situationen zusammengefaßt und strukturiert. Aus diesen Situationen wird als drittes eine Menge möglicher Handlungsalternativen bestimmt, aus denen eine oder mehrere zur Ausführung ausgewählt werden. Schließlich werden diese Entscheidungen in Handlungen umgesetzt. Diese Schritte werden sequentiell durchgeführt, wobei Wahrnehmen und Handeln jedoch gleichzeitig mit und parallel zu der zentralen Kognition stattfinden. In INTERRAP gilt dieses Schema sowohl für die Architektur als Ganzes, als auch innerhalb jeder einzelnen Ebene. Zusätzlich gibt es hier eine kompetenzgesteuerte Kontrolle zur Auswahl einer für eine gegebene Situation geeigneten Ebene.

1. Wissen aktualisieren

Die Aktualisierung des Wissens beginnt mit der Berücksichtigung der momentanen Wahrnehmung. In INTERRAP und TOK werden die aktuellen sensorischen Daten dazu verwendet, das Weltmodell bzw. das ISM zu aktualisieren, indem neue Fakten hinzugefügt und gegebenenfalls alte entfernt werden, falls diese mit der Wahrnehmung in Widerspruch stehen. In den Architekturen ACT und SOAR, die zwischen einem Arbeits- und einem Langzeitspeicher unterscheiden, werden Wahrnehmungsdaten einfach dem Arbeitsspeicher hinzugefügt, Widersprüche werden – zumindest in dieser Phase – nicht automatisch entdeckt und beseitigt. Die Trennung von Arbeits- und Langzeitspeicher macht in diesen beiden Architekturen noch einen zusätzlichen

Schritt der Wissenssuche notwendig. Dabei wird durch Assoziation neue Information in den Arbeitsspeicher transferiert, die in dem geänderten Kontext relevant ist.

2. Situation erkennen

Nachdem das verfügbare Wissen aktualisiert und gegebenenfalls um relevante Information erweitert wurde, wird es auf Situationen hin überprüft, für die Handlungsmöglichkeiten bekannt sind. In allen Architekturen geschieht dies durch die Vorbedingungen von Produktionen oder vergleichbaren Strukturen, die Situationen durch Aufzählung von Fakten beschreiben, die im (Arbeits-) Speicher erfüllt sein müssen. Dadurch werden die verfügbaren Speicherinhalte strukturiert und analysiert. In INTERRAP gehört zu dieser Phase auch die Berücksichtigung von Nachrichten, die eine Ebene als Aktivierungsaufforderung von einer darunterliegenden erhält, die bereits eine Situationsbeschreibung zur Planung enthält, die jedoch noch durch ebenenspezifisches Wissen erweitert werden kann.

3. Handlungsmöglichkeiten finden

Mit den Situationen, die erkannt werden können, sind entweder Handlungsmöglichkeiten oder Zielsetzungen, für die dann Handlungen zu planen sind, verbunden. Das Auffinden von Handlungsmöglichkeiten besteht somit im Überprüfen der Vorbedingungen, die die Situationen beschreiben. In ACT geschieht dies durch das parallele Matchen der Bedingungen im Arbeitsspeicher, wobei der Grad an Aktivierung der Chunks, das aktive Ziel, die Komplexität der Bedingungen und die Produktionenstärke sich auf die Geschwindigkeit der Instantiierung einer einzelnen Produktion auswirken. Auch SOAR bestimmt die Handlungsmöglichkeiten durch Produktionen-Matchen, während in ACT jedoch der Aktionsteil der Produktionen direkt ausführbare Handlungen angibt, werden in SOAR lediglich neue Informationen zum Arbeitsspeicher hinzugefügt, die unter anderem auch Handlungsmöglichkeiten umfassen können. Die Handlungsalternativen in TOK bestehen in den Zielen des APT von HAP, die durch Dämonen hinzugefügt werden, sofern deren Vorbedingung erfüllt ist.

In INTERRAP werden durch die erkannten Situationen Ziele aktiviert, für die durch Planen Handlungen erstellt werden, die das Ziel erreichen. Der Planer kann dabei auch das Ziel zusammen mit der Situationsbeschreibung an die nächsthöhere Ebene weitergeben, falls er nicht kompetent ist, das Problem zu lösen. Auf jeder Ebene ist die Planung für ein Ziel anders realisiert. Auf der BBL sind Ziele direkt mit Handlungsoptionen verbunden. Auf den Planungsebenen LPL und CPL werden Pläne entweder aus der Planbibliothek abgerufen oder durch Problemraumsuche neu erstellt.

4. Handlung auswählen

Im allgemeinen werden auf diese Weise mehrere mögliche Handlungen gefunden, die nicht alle gleichzeitig ausgeführt werden können und auch nicht alle gleich sinnvoll sind, so daß unter

ihnen eine Auswahl getroffen werden muß. Dies bedeutet die Entscheidung für eine bestimmte Handlung.

Unter allen im Arbeitsspeicher enthaltenen Alternativen wird in SOAR durch explizite Präferenzen eine ausgewählt. Ist dies nicht möglich, so entsteht eine Sackgasse und es wird als neues Ziel aufgestellt, diese aufzulösen. Die Entscheidung in TOK besteht in der Auswahl eines Ziels des APT von HAP, die anhand einer numerischen Priorität getroffen wird.

Den aufwendigsten Entscheidungsprozeß besitzt ACT-R. Während der Produktioneninstantiierung hängt deren Geschwindigkeit von mehreren Faktoren ab, die zusammen die Wahrscheinlichkeit für eine sinnvolle Anwendung der Produktion ergeben. Nach einer bestimmten Zeit werden die bis dahin vollständig instantiierten Produktionen auf ihren Wert hin überprüft und die beste ausgewählt, falls nicht zu erwarten ist, daß durch fortgesetztes Matchen von Produktionen eine noch bessere Instantiierung gefunden wird. Die Auswahl einer Produktion in ACT-R hängt somit von der Geschwindigkeit der Instantiierung, die wiederum von den oben genannten Faktoren abhängt, sowie von deren Wert ab.

INTERRAP besitzt wiederum auf jeder Ebene einen eigenen Entscheidungsprozeß. Auf der BBL wird abhand von Prioritäten eine Menge gleichzeitig ausführbarer Handlungen aus allen aktiven PoB ausgewählt. Auf den Planungsebenen werden durch einen Scheduler alle vorhandenen Pläne miteinander koordiniert und unter Umständen zur Neuplanung an den Planer zurückgegeben.

5. Handlung ausführen

Die so ausgewählten Handlungen können nun direkt ausgeführt werden. In ACT geschieht dies durch die Veränderung des Arbeitsspeichers entsprechend des Aktionsteils der ausgewählten Produktionen. In SOAR wird entweder ein Planoperator angewendet oder ein motorischer Befehl in den Arbeitsspeicher geschrieben. Eine Handlung in TOK ist entweder ein neues Ziel, das dem APT hinzugefügt und für das ein Plan aus der Planbibliothek abgerufen wird, oder eine ausführbare Handlung, die direkt umgesetzt wird.

Die BBL von INTERRAP führt die ausgewählten Handlungen aus, indem sie entsprechende Befehle an die Handlungskomponente sendet. Die LPL aktiviert für jeden Planschritt procedure PoB, für deren Ausführung die BBL zuständig ist. Die CPL übergibt den Anteil des Agenten an einem Multiagentenplan zur Ausführung an die LPL, die diesen in ihre Intentionsstruktur einfügt und für die weitere Umsetzung zuständig ist.

Der Kontrollzyklus

Tabelle 4 stellt die einzelnen Schritte des Kontrollzyklus der verschiedenen Architekturen einander gegenüber. Der Kontrollzyklus von INTERRAP besteht auf jeder Ebene aus den drei Kontrollfunktionen BR, SG und PS, die nacheinander ausgeführt werden (vgl. Kapitel 2.1.2.). BR (belief revision and abstraction) aktualisiert die Wissensbasis durch Wahrnehmung und Abstraktion. SG (situation recognition and goal activation) strukturiert das aktuelle Wissen zu

Architektur	INTERRAP	ACT	TOK	SOAR
Wissen aktualisieren	Weltmodell aktualisieren	Aktivierungs-ausbreitung	ISM aktualisieren	Wissenssuche
Situation erkennen	Situations-erkennung	Vorbedingungen	Vorbedingungen	Vorbedingungen
Handlungs-möglichkeiten finden	Zielaktivierung, Planung	Pattern-Matching	Aufstellen von Zielen, Auswahl aus Plänen	Wissenssuche
Handlung auswählen	Scheduling	Konfliktauflösung	Auswahl eines Ziels aus APT	Auswahl durch Präferenzen
Handlung ausführen	Aktivierung/ Ausführung von PoB	Aktion ausführen	Ausführung des Ziels	Ausführung oder neues Teilziel

Tabelle 4: Der Kontrollzyklus der Architekturen

Situationen und aktiviert zugehörige Ziele als Handlungsoptionen. PS (planing and scheduling) erstellt Pläne zum Erreichen dieser Ziele, koordiniert diese, stellt sie als Handlungsabsichten auf und führt sie aus.

TOK beginnt den Kontrollzyklus ebenfalls mit der Aktualisierung des Modells von der Welt, dem ISM. Anschließend wird der APT von HAP aktualisiert, indem plötzlich erreichte Ziele und nicht mehr ausführbare Pläne entfernt werden. Als nächstes wird ein Ziel aus dem APT ausgewählt und umgesetzt, indem entweder der Planbaum an dieser Stelle erweitert oder eine Handlung ausgeführt wird.

Der erste Schritt des Kontrollzyklus von ACT ist die assoziative Aktivierungsausbreitung. Anschließend wird eine Produktion durch die Geschwindigkeit des Instantiierungsprozeß der Vorbedingungen und durch die Bewertung des Nutzens ausgewählt. Schließlich wird die Aktion der ausgewählten Produktion ausgeführt.

Der Kontrollzyklus von SOAR besteht aus drei Schritten. Zunächst werden bei der Wissenssuche so lange Produktionen auf den Arbeitsspeicher angewendet, bis keine Information mehr aus dem Langzeitspeicher hinzugefügt wird, da keine neue Produktioneninstanz mehr möglich ist. Anschließend wird anhand der Präferenzen eine der bei der Wissenssuche gefundenen Handlungsmöglichkeiten ausgewählt. Kann eine Handlung eindeutig ausgewählt werden, so wird sie ausgeführt, andernfalls entsteht eine Sackgasse mit dem Ziel, eine eindeutige Entscheidung zu finden.

Kontrolle

Die Kontrolle über die Informationsverarbeitungsprozesse hängt nicht nur von dem Kontrollzyklus der Architektur ab, sie kann auch durch Information vorgenommen werden, die erst durch Erfahrung gewonnen wird. Dadurch wird die Kontrolle flexibler und offener. Zudem

kontrollieren in allen Architekturen selbstverständlich auch der situative Kontext und die internen Ziele des Agenten den Ablauf der Informationsverarbeitung.

Die Informationsverarbeitungsprozesse in ACT werden vor allem durch parallele Prozesse der Aktivierungsausbreitung kontrolliert. Auch das Instantiieren der Produktionen läßt sich als Aktivierungsausbreitung auffassen, bei der die Chunks um so schneller die Vorbedingungen der Produktionen instantiieren, je mehr Aktivierung sie an diese weiterleiten. Aktivierung bildet dabei eine Art kognitiver Ressource oder "Energie", die vor allem den Prozessen zukommt, die eine hohe Erfolgswahrscheinlichkeit besitzen (Anderson 1993, S. 46). Der Kontrollzyklus von ACT-R ist darauf ausgerichtet, den Verbrauch kognitiver Ressourcen möglichst zu minimieren und die Erfolgswahrscheinlichkeit zu maximieren. Deshalb hängt die Aktivierung der Chunks und die Stärke der Produktionen und damit deren Instanstiierungsgeschwindigkeiten vom statistisch ermittelten bisherigen Erfolg in ähnlichen Situationen ab (vgl. Kapitel 2.2.2. und 2.2.3.). Dadurch werden die Produktionen mit der höchsten Erfolgswahrscheinlichkeit am schnellsten instantiiert und somit der Aufwand für das Produktionen-Matchen minimiert. Um auch Produktionen mit komplexeren Vorbedingungen – die deshalb langsamer instantiiert werden – anwenden zu können, wird erst dann eine der vollständig gematchten Produktionen ausgeführt, wenn deren zu erwartender Wert so hoch ist, daß der Aufwand zur Instantiierung weiterer Produktionen mit eventuell noch höherem Wert sich nicht mehr lohnt. Dadurch wird ein Kompromiß zwischen dem gering zu haltenden Aufwand für die Auswahl einer Produktion und deren möglichst hoher Nützlichkeit erreicht. Dies entspricht der in Kapitel 1.4.2. zitierten Definition von Rationalität, die Anderson als Maximierung des Nutzens bei gleichzeitiger Minimierung der Kosten angibt. Die eigentliche Kontrolle über die kognitiven Prozesse liegt damit nicht so sehr bei der Architektur, die lediglich die Abfolge von assoziativer Aktivierungsausbreitung, Produktionenauswahl durch Pattern-Matching und Bewertung sowie Produktionenausführung vorgibt, sondern bei der statistischen Erfahrung, die in den Basisaktivierungen und den assoziativen Verbindungen der Chunks sowie in der Stärke und dem Wert der Produktionen enthalten ist.

Während ACT die Kontrolle statistisch ermittelter, impliziter Erfahrung überläßt, werden die Entscheidungen in SOAR anhand von explizitem Kontrollwissen getroffen, das während der Wissenssuche durch Produktionen bereit gestellt wird und ebenfalls der Erfahrung entstammt. Auch in SOAR gibt also die Architektur nur den Zyklus von Wissenssuche, Entscheidung durch Präferenzen und Ausführung vor, die eigentliche Kontrolle wird durch erfahrungsabhängiges Wissen vorgenommen, das bei der Auflösung von Sackgassen entsteht. Eine ähnliche Kontrolle durch explizite, erfahrungsabhängige Kontrollregeln ermöglicht auch PRODIGY.

INTERRAP und TOK beschränken die Kontrolle auf die Architektur. In TOK bedeutet dies einfach die situative Erweiterung des APT von HAP bis hin zu Handlungen. INTERRAP besitzt einen zusätzlichen Kontrollfluß zwischen den Ebenen, durch den eine kompetente Ebene zur Bearbeitung eines Problems bestimmt wird (vgl. Kapitel 2.1.1.). Dabei ergeben sich zwei

Kontrollrichtungen. Von unten her werden erkannte Situationen so lange nach oben weitergereicht, bis eine Ebene sich für kompetent erachtet, das damit verbundene Problem zu lösen. Ist eine Lösung gefunden, so werden von oben her nach unten Verpflichtungen für Handlungen weitergeleitet und mit den Verpflichtungen der unteren Ebenen koordiniert, bis sie die BBL erreichen, die die Verpflichtungen in Handlungen umsetzt.

4. FUNKTIONALE ANALYSE

Die funktionale Analyse von INTERRAP beruht auf dem in Kapitel 1.4. vorgestellten Schema. Sie orientiert sich vergleichend an den Architekturen ACT, TOK/PRODIGY und SOAR, ohne jedoch eine vollständige Gegenüberstellung anzustreben. Deshalb wird nicht jede Architektur zu jedem Thema herangezogen, sondern jeweils nur, soweit ein Vergleich interessant ist. Die funktionale Analyse umfaßt zum einen das kognitive Spektrum aus reaktivem, deliberativem, adaptivem und sozialem Verhalten, zum anderen die Mechanismen zum Erreichen intelligenten Verhaltens wie Rationalität, Robustheit, Flexibilität und Vielseitigkeit.

4.1. Das kognitive Spektrum

Auch wenn alle hier betrachteten Architekturen für sich beanspruchen, kognitiv im Sinne einer intelligenten Verhaltenssteuerung zu sein, stimmt das intendierte kognitive Spektrum nicht völlig überein. Ziel von INTERRAP ist die Integration von reaktivem, deliberativem und kooperativem Verhalten, um autonome Agenten in dynamischen Multiagentenumgebungen zu konzipieren. Für jede dieser Funktionen ist dabei eine eigene Ebene verantwortlich. Um glaubwürdige Charaktere für simulierte Welten zu schaffen, kombiniert TOK Module für reaktives und deliberatives Planen mit sozialen Komponenten wie Emotionen und Sprache.

Schwerpunkt der anderen Architekturen ist hingegen das Verständnis für zentrale kognitive Prozesse und allgemeine Intelligenz. ACT dient als empirisches Modell für kognitive Aspekte der menschlichen Psyche, insbesondere der Realisierung kognitiver Fertigkeiten. SOAR und PRODIGY verstehen sich als allgemeine Problemlöser, mit denen prinzipiell jede Art von Problemen durch Problemraumsuche gelöst werden können soll. Tabelle 11 (s. Anhang) stellt das kognitve Spektrum der Architekturen sowie deren Konzepte für reaktives, deliberatives, adaptives und soziales Verhalten gegenüber.

4.1.1. Reaktives Verhalten

In dynamischen Umgebungen muß ein Agent jederzeit auf Veränderungen reagieren können. Dies betrifft sowohl plötzlich eintretende Situationen, die neue Zielsetzungen erfordern, als auch unvorhergesehene Ereignisse und Effekte bei der Ausführung eines Planes sowie die sofortige Berücksichtigung von Änderungen beim Aufstellen von Plänen.

Reaktivität

Um eine hohe Reaktivität zu erreichen, bedarf es eines Kontrollzyklus, dessen zeitliche Auflösung der Dynamik der Umgebung angepaßt ist, d.h. der unter allen Umständen möglichst schnell ist. Dazu müssen Situtionen schnell erkannt und angemessene Handlungen schnell gefunden werden. Außerdem muß die Ausführung begonnener Handlungsabläufe jederzeit unterbrechbar sein.

Die BBL von INTERRAP ist speziell auf diese Anforderungen hin konzipiert. Für Reaktionen werden reaktive Verhaltensmuster (reactor PoB) verwendet, die durch die Wahrnehmung externer Ereignisse ausgelöst werden und direkt mit Handlungen verbunden sind. Für eine schnelle Situationserkennung ist die Anzahl der zu überprüfenden Situationen auf die expliziten Bedingungen der reactor PoB beschränkt, die zusätzlich deaktiviert werden können, falls bestimmte Reaktionen in einer gegebenen Situation nicht sinnvoll möglich sind. Zudem kommen in den Vorbedingungen ausschließlich atomare Grundfakten vor, die durch reines Pattern-Matching überprüft werden. Um eine schnelle Auswahl geeigneter Handlungen zu ermöglichen, sind die erkannten Situationen bei den reactor PoB direkt mit Handlungen verbunden. Falls für mehrere nicht miteinander vereinbare Handlungen Verpflichtungen bestehen, wird dieser Konflikt über Prioritäten effizient aufgelöst. Dabei besitzen reactor PoB prinzipiell eine höhere Priorität als procedure PoB, so daß dringende Reaktionen immer der Ausführung von anderen Handlungen vorgezogen werden. Die Unterbrechbarkeit von Handlungsabläufen wird erreicht, indem alle PoB in atomare Instruktionen unterteilt sind und so schrittweise über mehrere Kontrollzyklen hin ausgeführt werden. In jedem Kontrollzyklus wird für jeden aktiven PoB jeweils neu entschieden, ob die nächste atomare Instruktion ausgeführt wird oder ob andere Handlungen momentan wichtiger sind. Durch die hierarchische Einteilung in Ebenen wird in INTERRAP zudem die Unabhängigkeit des reaktiven Kontrollzyklus von zeitaufwendigen Planungsprozessen erreicht. Diese verlaufen parallel zu den reaktiven Fähigkeiten und werden durch sie nicht behindert oder unterbrochen. Die Ausführung der auf den höheren Ebenen erstellten Pläne besteht in der Aktivierung von procedure PoB, für deren Ausführung allein die reaktive BBL zuständig ist.

TOK und SOAR versuchen Reaktivität nicht durch eigene Mechanismen oder Module, sondern als Spezialfall des normalen Kontrollzyklus zu erreichen. In TOK überwachen Dämonen das ISM auf relevante Änderungen und stellen gegebenenfalls neue Ziele im APT von HAP auf. Besitzt dieses Ziel eine hohe Priorität und besteht der aus der Planbibliothek für es ausgewählte

Plan in einer physischen Handlung, so findet sofort eine angemessene Reaktion statt. Da HAP ohnehin darauf ausgelegt ist, situationsgerecht zu handeln, geschieht eine Reaktion entsprechend schnell. Wie bei den reactor PoB von INTERRAP können die Vorbedingungen der Dämonen effizient im Modell der Welt überprüft werden, so daß eine schnelle Situationserkennung gewährleistet ist. Die Auswahl eines Ziels durch Prioritäten und der Planabruf durch Produktionen ermöglichen eine schnelle Entscheidung für Handlungen. Die Unterbrechbarkeit hängt – wie auch in INTERRAP – von der Durchführungsdauer der Handlungsprimitiven ab, die deshalb bei der Konstruktion möglichst niedrig gehalten werden sollte. Ebenfalls wie in INTERRAP verläuft auch in TOK die deliberative Planung, die hier durch PRODIGY durchgeführt wird, parallel zu und damit unbeeinflußt vom reaktiven Kontrollzyklus.

Da in SOAR alle Wahrnehmungen in den zentralen Arbeitsspeicher geschrieben werden, haben alle Veränderungen in der Umgebung potentiell einen Einfluß auf die Entscheidungen des Systems. Werden während der Wissenssuche mögliche Reaktionen und zugleich Präferenzen für diese in den Arbeitsspeicher geschrieben, so kann innerhalb eines einzigen Durchlaufs des Kontrollzyklus eine Situation erkannt und angemessen reagiert werden. Anders als TOK ist SOAR jedoch nicht bereits auf eine schnelle Durchführung des Kontrollzyklus ausgerichtet. Gerade die Wissenssuche, bei der nicht nur mögliche Reaktionen, sondern der gesamte Langzeitspeicher überprüft wird, und bei der zudem so lange Produktionen instantiiert werden, bis keine neuen Instantiierungen mehr möglich sind, verlangsamt die Situationserkennung gegenüber TOK und INTERRAP, auch wenn deren Grundprinzip der Verwendung von Produktionen in allen drei Architekturen dasselbe ist. Da in SOAR auch keine Trennung von reaktiven Fähigkeiten und der zentralen Problemraumsuche stattfindet, besteht die Gefahr, daß diese durch Reaktionen beeinträchtigt und schlimmstenfalls sogar ergebnislos abgebrochen wird.

Eine Trennung von Reaktion und zielgerichtetem Planen wie in INTERRAP und TOK ist somit ratsam, sowohl um den reaktiven Kontrollzyklus möglichst effizient und reaktionsschnell zu halten, als auch um Störungen der zeitaufwendigen Planungsvorgänge durch dafür irrelevante Änderungen der Umgebung zu verhindern.

Situatives Handeln

Bei der Ausführung von Plänen muß auf im Plan unvorhergesehene Ereignisse und auch auf Effekte, die nicht den Erwartungen des Plans entsprechen, reagiert werden, sofern diese die Fortführung des Plans beeinflussen. In INTERRAP wird dies auf drei Arten erreicht. Zum einen werden Pläne auf einem so hohen Abstraktionsniveau aufgestellt, daß die meisten Veränderungen in der Umgebung keinen Einfluß auf den Plan haben. Zweitens werden die einzelnen abstrakten Planschritte dieser abstrakten Pläne durch procedure PoB durchgeführt, die flexibel und intelligent genug sind, um ihre Aufgabe auch in dynamischen Umgebungen zu erfüllen, indem mehrere Handlungsalternativen bereitstehen. Schließlich kann ein procedure PoB auch spezielle reactor PoB zur Überwachung der Ausführung aufstellen. Diese Überwachungsfunk-

tionen können einen procedure PoB bei Erfolg oder Mißerfolg beenden und Probleme bei der Ausführung des procedure PoB erkennen und beheben.

Ähnliche Überwachungsfunktionen verwendet auch TOK. Dabei entfernen Dämonen Ziele aus dem APT von HAP, falls ein Ziel zufällig erfüllt ist oder falls ein Plan nicht mehr ausführbar ist. Eine weitere Übereinstimmung zu INTERRAP ist die Verwendung von abstrakter deliberativer Planung, die in TOK von PRODIGY durchgeführt wird. Ein abstrakter Plan besteht aus einer Folge von Zielen, für die HAP situativ Pläne erstellt und ausführt. Die situative Ausführung dieser abstrakten Planschritte ist in TOK flexibler gelöst als in INTERRAP. Während in INTERRAP ein procedure PoB eine in sich geschlossene Prozedur ist, in der bestimmte mögliche Änderungen in der Umgebung berücksichtigt und gegebenenfalls ausgeglichen werden, findet bei der Ausführung durch HAP eine situative Planung statt.

Situatives Planen

Nur in TOK gibt es eine wirklich situative Planung, bei der Pläne erst während der Ausführung erstellt werden, wodurch Umgebungsveränderungen sofort und flexibel berücksichtigt werden können. Andererseits ist die deliberative Planung in TOK durch PRODIGY vollkommen unabhängig von Änderungen der Umgebung, da sie für die einmal übergebenen Situationsbeschreibung durchgeführt wird. Entsprechendes gilt für die Planungsebenen in INTERRAP, die gleichfalls für eine einmal übergebene Aufgabe einen Plan erstellen. Nur falls während der Planung explizit auf das Weltmodell zugegriffen wird, können Veränderungen der Umgebung einen Einfluß auf die Planung haben. Da in ACT und SOAR alle Wahrnehmungsdaten samt zugehörigen Assoziationen in den zentralen Arbeitsspeicher gelangen, in dem auch die Planung stattfindet, können Änderungen hier sofort einen Einfluß auf die Planung haben. Eine automatische Überprüfung des bisherigen Plans gibt es dabei jedoch nicht, so daß eventuell von einem Zustand aus weiter geplant wird, der mit dem bisherigen Plan gar nicht zu erreichen ist.

Einzig die situative Erweiterung des Planbaums in HAP stellt eine Form der Planung dar, die zugleich zielgerichtet ist und dennoch flexibel die jeweilige Situation berücksichtigt. Dazu wird ein Ziel zunächst abstrakt aufgestellt. Anschließend wird das Ziel rekursiv durch Pläne situationsabhängig in Teilziele und primitive Handlungen zerlegt. Für ein Ziel entsteht so ein Baum aus Teilzielen als Plan, wobei jede Ebene innerhalb des Baumes eine eigene Abstraktionsstufe der Planung darstellt. Durch die situative Erweiterung des Baumes entlang des vordersten Pfades bleibt der Plan auf einer hohen Abstraktionsstufe, lediglich die Teilziele, die als nächstes erreicht werden sollen, werden detaillierter geplant, wobei direkt die jeweilige Situation berücksichtigt werden kann. So werden konkrete Handlungen erst direkt vor ihrer Ausführung geplant und entsprechen so immer der gegebenen Situation, während die späteren Teile des Plans um so abstrakter bleiben, je später sie realisiert werden sollen und je mehr unvorhergesehene Veränderungen der Umgebung deshalb möglich sind.

4.1.2. Deliberatives Verhalten

Deliberation bedeutet die Steuerung des Verhaltens durch interne Zielsetzungen. Dies wird in allen betrachteten Architekturen durch zielgerichtete Planung erreicht, bei der durch Problemraumsuche eine Handlungssequenz zwischen einem gegebenen Anfangszustand und dem angestrebten Endzustand konstruiert wird, sowie durch zielabhängige Entscheidungen. Die Unterschiede zwischen den Architekturen bei der Planung bzw. bei der Entscheidungsfindung wurden bereits in den Kapiteln 3.3.3. bzw. 3.3.5. eingehend diskutiert.

Aus den bereits genannten Gründen (s. Kapitel 4.1.1.) trennen INTERRAP und TOK das deliberative vom reaktiven Verhalten. In INTERRAP geschieht dies durch eine separate Kontrollebene, TOK verwendet ein eigenes Planmodul. In beiden Architekturen werden Probleme von dem reaktiven Modul erkannt und zur Planung an das Planungsmodul übergeben. Während in TOK ein Plan zur Ausführung an HAP zurückgegeben wird, koordiniert und kontrolliert die Planungsebene von INTERRAP auch die Ausführung der Pläne und aktiviert nur für einzelne Planschritte die procedure PoB der BBL. In beiden Architekturen ist zudem die gleichzeitige Planung für mehrere Ziele vorgesehen.

4.1.3. Adaptives Verhalten

Da für komplexe, dynamische Umgebungen eine optimale Vorprogrammierung des Verhaltens nur schwer möglich ist, ist für eine intelligente Architektur eine Anpassung des Verhaltens an die Umgebung durch Lernen unverzichtbar. In INTERRAP sind bislang keine Lernmechanismen integriert, eine entsprechende Erweiterung ist jedoch vorgesehen (vgl. Müller 1996, S. 198f).

Es gibt verschiedene Möglichkeiten, Lernen in INTERRAP zu integrieren. Zum einen können spezielle Lernmechanismen für einzelne Ebenen in diesen integriert werden. Des weiteren besteht die Möglichkeit, ebenenspezifisches Wissen einer Ebene auf eine Nachbarebene zu transferieren. Dabei können sowohl von oben nach unten abstrakte Datenstrukturen zur Ausführung durch die tiefere Ebene optimalisiert werden, als auch von unten nach oben Datenstrukturen zur Verwendung bei der Planung abstrahiert werden. Schließlich kann eine eigene adaptive Ebene verschiedene Prozesse auf und zwischen den Ebenen überwachen und hieraus neues Wissen ableiten.

An Lerngelegenheiten und -verfahren bieten die Vergleichsarchitekturen viele Beispiele. Gelernt werden kann deklaratives und prozedurales Wissen sowie Wissen zur Kontrolle und zur Entscheidungsfindung.

Deklaratives Lernen beinhaltet weniger das bloße Abspeichern durch die Wahrnehmung aufgenommener Informationen, sondern vor allem deren Organisation und Strukturierung. Bei einer Trennung von Arbeits- und Langzeitspeicher, wie er bei großen Datenbeständen sinnvoll ist, wird ein kontextsensitiver Abrufmechanismus notwendig, dessen Effizienz und die Relevanz der abgerufenen Information durch Lernen verbessert werden kann. Beispiele hierfür

sind die Assoziationsmechanismen von ACT und SOAR. Während das Chunking von SOAR jedoch nur neue Assoziationen hinzufügt, werden diese in ACT zusätzlich statistisch nach ihrer Relevanz gewichtet, so daß der (wiederholte) Abruf irrevelanter Information sowie ein Übermaß an abgerufener Information vermieden wird.

Prozedurales Lernen bedeutet den Erwerb neuer Fertigkeiten. Da jede INTERRAP-Ebene eigene Arten prozeduralen Wissens verwendet, sind jeweils eigene Lernmethoden sinnvoll. Auf der BBL könnten PoB neu gelernt oder aufgrund von Erfahrung geändert werden. Eine andere Möglichkeit wäre die Kompilierung von Plänen der LPL zu procedure PoB, was durch die Ähnlichkeiten von Plansprache und PoB-Sprache vereinfacht wird (Müller 1996, S. 198). Auf der LPL könnten fertige Pläne in der Planbibliothek gespeichert und so wieder verwendet werden. Die Möglichkeit der Verbesserung des Planungsvorgangs durch Lernen hängt von dem gewählten Planungsalgorithmus ab. Auf der CPL bestehen die entsprechenden Lernmöglichkeiten für Multiagentenplanung, hinzu kommt die Verbesserung von Kommunikation und Kooperation durch Lernen von Verhandlungsprotokollen und -strategien.

Das Lernen von Kontroll- und Entscheidungswissen ist an den Stellen möglich, an denen die Kontrolle nicht allein durch die Architektur, sondern durch zusätzliche Information gesteuert wird. In INTERRAP wäre beispielsweise eine statistische Justierung der Prioritäten möglich, die auf Erfog und Mißerfolg von PoB beruht. Auch die Kompetenzentscheidungen der Ebenen, über die eine Ebene zur Problemlösung ausgewählt wird, könnten abhängig von Erfolg und Mißerfolg verändert und dadurch die Ebenenauswahl verbessert werden.

4.1.4. Sozialverhalten

Das Sozialverhalten umfaßt alle Verhaltensweisen, die der Interaktion mit anderen kognitiven Agenten dienen. Dazu zählen Fähigkeiten zur Kommunikation und Kooperation, das Wissen über andere Agenten und soziale Normen sowie Einstellungen gegenüber anderen Agenten. Die einzigen Architekturen, die soziales Verhalten explizit berücksichtigen, sind INTERRAP und TOK.

INTERRAP besitzt eine eigene Ebene, die CPL, für soziale Fähigkeiten. Zum einen führt die CPL Verhandlungen mit anderen Agenten durch, um zu Übereinkünften bei kooperativen Handlungen zu gelangen, zum anderen erstellt sie Pläne, die durch mehrere Agenten gemeinsam ausgeführt werden. Wissen über andere Agenten entstammt in INTERRAP entweder der Kommunikation oder aus Beobachtungen und ist im sozialen Modell der Wissensbasis enthalten, auf das nur die CPL zugreifen kann.

Verhandlungen werden anhand von Protokollen durchgeführt, die abhängig von zugewiesenen Rollen für die teilnehmenden Agenten den möglichen Verlauf der Verhandlung festlegen. Jeder Agent benutzt eine Verhandlungsstrategie für ein Protokoll, die dessen individuelle Durchführung bestimmt, um zu einem erfolgreichen und für ihn möglichst günstigen Ergebnis zu gelangen. Durch Verhandlungen können Konflikte zwischen Agenten aufgelöst oder Aufga-

ben für ein gemeinsames Ziel zugeteilt werden. Gegenstand und Ergebnis von Verhandlungen sind Multiagentenpläne, die vom Leiter der Verhandlung aufgestellt werden. Ein Multiagentenplan besteht aus einer partiell geordneten Folge von Handlungen mehrerer Agenten, die zur Ausführung jeweils in Einzelagentenpläne mit Synchronisationsmeldungen umgewandelt werden (vgl. Kapitel 2.1.5.).

Kommunikation mit anderen Agenten wird in TOK als gewöhnliche Handlung aufgefaßt, die wie andere Aktionen geplant wird. Zur Spracherzeugung – anders als INTERRAP verwendet TOK zur Kommunikation natürliche Sprache – wird das Modul GLINDA verwendet, dessen Funktionalität jedoch der von HAP so weit ähnelt, daß diese auch von HAP durchgeführt werden könnte (Loyall & Bates 1996). Eine Besonderheit von TOK ist die Verwendung der emotionalen Komponente EM (vgl. Kapitel 2.3.3.). Die Emotionen hängen dabei auch von sozialen Aspekten wie moralischen Standards, anhand derer eigene und fremde Handlungen beurteilt werden, und Einstellungen gegenüber anderen Agenten ab.

4.2. Das Verhaltensspektrum

Das Verhalten eines kognitiven Agenten ist intelligent, wenn es möglichst zweckoptimal und erfolgreich ist. Dazu gehört neben der Rationalität der Handlungen – dem Bestreben, Ziele mit möglichst geringem Aufwand bei gleichzeitiger Maximierung des Nutzens zu erreichen – auch die Robustheit des Systems gegenüber Fehlern, unvorhergesehenen Ereignissen und zeitkritischen Situationen. Weitere Merkmale für Intelligenz sind die Flexibilität, die situationsgerechtes und Veränderungen der Umgebung angepaßtes Verhalten ermöglicht, sowie die Vielseitigkeit, die bestimmt, für welche Arten von Aufgaben und Umgebungen eine Architektur geeignet ist.

Mit welchen Konzepten ein möglichst hohes Maß an Intelligenz in den einzelnen Architekturen zu erreichen versucht wird, ist in Tabelle 12 (s. Anhang) zusammengefaßt.

4.2.1. Rationalität

Die Rationalität eines Systems zeigt sich vor allem im zielgerichteten deliberativen Verhalten, sie betrifft aber auch das Gesamtverhalten der Architektur, das sich aus der Vereinbarkeit von Wissen und Handlungen sowie aus der Kohärenz zwischen reaktivem und deliberativem Verhalten ergibt.

Rationalität des zielgerichteten Verhaltens

Da deliberatives Verhalten im wesentlichen zielgerichtetes Planen durch Problemraumsuche bedeutet (s. Kapitel 4.1.2.), richtet sich die Rationalität des zielgerichteten Verhaltens nach der Effizienz des Suchalgorithmus und der Qualität der gefundenen Pläne.

Die Effizienz der Problemraumsuche läßt sich in INTERRAP nicht bewerten, da bislang kein Algorithmus fest integriert ist. SOAR und PRODIGY zeigen die Möglichkeit der Effizienzsteigerung durch explizite Kontrollregeln auf, die mit verschiedenen Suchverfahren, allgemeinen Heuristiken und auch bereichsspezifischem Kontrollwissen die Suche steuern können. Allerdings können diese Kontrollregeln auch die Vollständigkeit der Suche einschränken und so unter Umständen eine erfolgreiche Problemlösung verhindern. Der Konfliktauflösungsmechanismus von ACT-R ist statistisch auf Rationalität ausgelegt, indem er zugleich den Nutzen zu maximieren und die Kosten zu minimieren versucht (vgl. Kapitel 3.4.2.).

Die Qualität von Plänen kann in INTERRAP wie in PRODIGY durch eine Planbewertungsfunktion bewertet werden, die die Kosten für einzelne Planschritte und den Nutzen bei Erreichen des Ziels berücksichtigt. Dadurch können Pläne miteinander verglichen werden oder bei einem zu schlechten Suchergebnis nach einem besseren Plan gesucht werden. INTERRAP bietet zusätzlich die Möglichkeit, falls auf der LPL kein zufriedenstellender Plan gefunden

wird, statt eines Einzelagentenplans einen Multiagentenplan zu erstellen. Die Produktionenbewertung von ACT-R sorgt lediglich bei der Auswahl jedes einzelnen Planschritts für ein optimales Kosten/Nutzen-Verhältnis zum Erreichen des Gesamtziels, kann jedoch nicht für den Vergleich kompletter Pläne verwendet werden.

Vereinbarkeit von Wissen und Handlungen

Die Vereinbarkeit der beabsichtigten Handlungen mit dem prozeduralen Wissen eines Agenten wird erreicht, indem die verwendeten Planungsalgorithmen korrekt sind, was bei der üblichen Verwendung von Planoperatoren, die symbolisch Zustandsänderungen beschreiben, unproblematisch ist.

Entsprechendes gilt auch für das deklarative Wissen, das den Ausgangszustand der Planung beschreibt. Veränderungen des Wissens über die Welt, die während des Planungsvorgangs wahrgenommen werden, beeinflussen in SOAR und ACT zwar den weiteren Verlauf der Planung, der bereits erstellte Teilplan wird jedoch nicht entsprechend überprüft, so daß eventuell von einem nicht mehr mit diesem erreichbaren Planzustand aus weitergeplant wird. INTERRAP und TOK überprüfen während der Planausführung durch reactor PoB bzw. durch Dämonen, ob das Planziel bereits erreicht oder nicht mehr erreichbar ist und der Plan deshalb beendet bzw. aufgegeben werden kann. INTERRAP ermöglicht außerdem die Definition spezieller reactor PoB, die bei der Ausführung des zugehörigen procedure PoB spezifische Ausnahmesituationen entdecken und gegebenenfalls beheben.

Da TOK und INTERRAP nicht alles in der Wahrnehmung aufgenommene Wissen speichern, sondern dieses zur Aktualisierung eines Modells der Welt verwenden, ist dieses – zumindest für die im Weltmodell enthaltenen atomaren Wissenseinheiten – automatisch konsistent, da bei widersprechenden Überzeugungen einfach die ältere entfernt wird. Damit ist das als Ausgangspunkt für die Planung verwendete Wissen konsistent, dafür werden alte Weltzustände einfach vergessen. In SOAR und ACT besteht nicht nur das Problem, das Inkonsistenzen des Wissens erst entdeckt werden müssen, wegen der Assoziationsmechanismen steht der Planung nicht einmal alles vorhandene Wissen zur Verfügung, dafür ist stattdessen die Wahrscheinlichkeit, daß das verfügbare Wissen auch relevant ist, hoch. In allen Architekturen fehlen jedoch Möglichkeiten zum Umgang mit inkonsistentem Wissen.

Kohärenz des Gesamtverhaltens

Die Kohärenz des Gesamtverhaltens ist bei zentralen Architekturen einfacher zu gewährleisten als bei geschichteten oder modularen Architekturen, da diese alle Intentionen zentral planen. Durch einen zentralen Zielstapel wird in ACT und SOAR die Kontinuität des Verhaltens auf ein Ziel hin organisiert und koordiniert. Da in TOK alle Planungsaufträge von HAP an PRODIGY vergeben werden, ist hier HAP für die Kohärenz des Verhaltens verantwortlich. Es kann hier jedoch zu Konflikten kommen, weil der APT mehrere parallele Ziele mit Planbäumen enthält.

Da die Ebenen von INTERRAP parallel zueinander ihre Handlungsintentionen planen, besteht hier zusätzlich die Gefahr der Inkonsistenz zwischen den Ebenen, die durch die Kontrollstruktur aus nach oben gesandten Planungsanforderungen und nach unten weitergeleiteten Handlungsverpflichtungen vermieden wird. Die kompetenzgesteuerte Auswahl einer Ebene von unten nach oben sorgt dafür, daß sich für jede erkannte Situation nur eine Ebene mit der Planung beschäftigt. Die Handlungsabsichten jeder einzelnen Ebene werden jeweils durch deren Scheduler koordiniert. Die Intentionen der höheren Ebenen werden von oben nach unten ebenenweise weitergereicht und dabei jeweils vom Scheduler der unteren Ebene auf die Vereinbarkeit mit deren Handlungsabsichten überprüft und gegebenenfalls zur Neuplanung zurückgewiesen. So sind die Intentionen jeder Ebene zu denen der darüberliegenden kohärent und somit auf der BBL, die für die Umsetzung der Handlungen zuständig ist, alle Absichten miteinander vereinbar.

Allerdings sind Konflikte zwischen sofortigen Reaktionen und langfristigen Zielen möglich, da die Reaktionen meist dringend sind und eine Abstimmung auf bereits bestehende längerfristige Handlungsabsichten zu aufwendig sein kann, um eine genügend schnelle Reaktion zu ermöglichen. Schließlich sind Reaktionen gerade als Antworten auf unvorhergesehene Ereignisse gedacht. Dieses Problem ist aber nicht spezifisch für Ebenenarchitekturen, sondern betrifft alle Architekturen, soweit sie sofortige Reaktionen erlauben. Allerdings besteht in zentralen Architekturen wie SOAR bei Reaktionen eher eine Möglichkeit der Berücksichtigung des intentionalen Gesamtzustandes, da dieser vollständig im Arbeitsspeicher enthalten ist. Um Reaktionen zu vermeiden, die den momentanen längerfristigen Handlungsabsichten widersprechen, können die Planungsebenen von INTERRAP auch vorbeugend einzelne reactor PoB deaktivieren und so deren Anwendung verhindern. Dabei ist jedoch Vorsicht geboten, da Reaktionen oft für sehr wichtige Aufgaben wie die Unversehrtheit des Agenten vorgesehen sind, die bei Konflikten zu langfristigen Intentionen immer Vorrang haben sollten.

4.2.2. Robustheit

Das Verhalten einer Architektur ist robust, wenn es auch unter widrigen Umständen gute Erfolgsaussichten besitzt. Dazu muß zum einen die Möglichkeit der Berücksichtigung von relevanten Veränderungen in der Umgebung vorhanden sein. Kurzfristig wird dies durch reaktives und situatives Verhalten (s. Kapitel 4.1.1.) gewährleistet, längerfristig durch Adaptationen an die Umgebung (s. Kapitel 4.1.3.). Zum anderen muß das System in der Lage sein, mit unvollständigem, unsicherem und auch falschem Wissen umzugehen.

Unvollständiges und falsches Wissen führt zu Fehlern bei der Planung oder Handlungsausführung. Diese Fehler müssen erkannt und behoben werden, wenn möglich sollte auch aus ihnen gelernt werden, um erneute Fehler der gleichen Art zu vermeiden. Fehlererkennung und -behebung wird in INTERRAP entweder durch eine entsprechende Programmierung der procedure PoB oder anhand von durch diese aktivierte reactor PoB zur Ausnahmebehandlung

erreicht. Ein Lernen aus Fehlern bei der Ausführung enthält als einziges PRODIGY mit dem Modul EXPIREMENT, das bei unerwarteten Effekten bei der Ausführung von Planoperatoren diese modifizieren kann.

Die Unsicherheit von Wissen wird in keiner Architketur explizit berücksichtigt, sie ließe sich jedoch prinzipiell durch ein zusätzliches Attribut innerhalb von Eigenschaftslisten realisieren.

4.2.3. Flexibilität und Vielseitigkeit

Neben dem Erfolg der Handlungen ist auch deren Reichhaltigkeit ein Zeichen von Intelligenz, zumal ein reichhaltiges Verhalten längerfristig auch die Erfolgswahrscheinlichkeit von Handlungen steigert, da viele Alternativen zur Auswahl bereitstehen und viele Gelegenheiten für Erfahrungen entstehen. Ein reichhaltiges Verhalten ermöglicht durch Flexibilität und Vielseitigkeit ein weites Spektrum an Aufgaben und Umgebungen. Flexibilität umfaßt einerseits ein situationsgerechtes Verhalten, andererseits die Möglichkeit neuartiger, besserer Lösungen auch in bekannten Situationen. Vielseitigkeit wird einerseits durch ein hohes Maß an verschiedenen vorhandenen Informationen und Fähigkeiten gewährleistet, andererseits durch die Möglichkeit, auch mit völlig neuartigen Situationen umgehen zu können.

Die Flexibilität hängt von vielen bereits behandelten Fähigkeiten ab, sie betrifft eigentlich das gesamte kognitive Spektrum von reaktivem, situativem, deliberativem und adaptivem Verhalten (vgl. Kapitel 4.1.1. - 4.1.3.). Die hohe Reaktivität und die situative Handlungsausführung von INTERRAP ermöglichen eine flexible Anpassung des Verhaltens an die gegebene Situation und deren Dynamik, eine längerfristige Anpassung durch Lernen aus Erfolg und Fehlern fehlt jedoch. Die Flexibilität der deliberativen Planung hängt vom gewählten Planungsalgorithmus ab. Durch eine flexible, wissensgesteuerte Problemraumsuche wie in PRODIGY und SOAR können dabei verschiedene Suchmethoden realisiert und die Suchmethode abhängig vom Problem gewählt werden. Überhaupt muß für eine hohe Flexibilität das Verhalten möglichst von veränderbarer Information und nicht zu sehr von festen Strukturen der Architektur abhängen. INTERRAP fehlen flexible Entscheidungsmechanismen, die das Verhalten durch veränderbare Information kontrollieren und es somit sowohl situations- als auch erfahrungsabhängig bestimmen. Dagegen verwenden SOAR und PRODIGY explizites Kontrollwissen, das durch Erfahrung erworben wird. Die Produktionenauswahl von ACT-R hängt von vielen erfahrungsabhängig statistisch variierten Parametern ab (s. Kapitel 3.3.5.).

Damit ein System vielseitig und flexibel agieren kann, muß nicht nur ein hohes Maß an Information bereitstehen, es muß auch sinnvoll und rechtzeitig genutzt werden können. Dafür sind Assoziationsmechanismen wie in ACT oder SOAR sinnvoll, da sie die vorhandene Information anhand der gegebenen Situation auf die relevante Information beschränken und so die Informationsverarbeitungskapazität fokussieren.

Durch die Aufteilung in eine reaktive, eine deliberative und eine kooperative Ebene ermöglicht INTERRAP die Gestaltung von Agenten für verschiedenste Aufgabenbereiche. Dies

ermöglicht eine strukturiertere und übersichtlichere Gestaltung als bei zentralen Architekturen wie SOAR.

5. KONKLUSION

5.1. Zentrale versus geschichtete Architekturen

Die auffallendste strukturelle Besonderheit von INTERRAP ist die Aufteilung der Kontrolleinheit und der Wissensbasis in jeweils korrespondierende parallele Ebenen. In jeder kognitiven Architektur ergeben sich zwangsläufig verschiedene Ebenen an Funktionalität, Kontrolle, Abstraktion und Wissenskompexität. Zentrale Architekturen wie SOAR berücksichtigen diese Ebenen lediglich implizit durch verschiedenartige Informationsverarbeitungsprozesse, während hierarchisch geschichtete Architekturen wie INTERRAP sie explizit in ihrer Struktur widerspiegeln. Auch wenn sich zentrale und geschichtete Architekturen nicht in ihrer prinzipiellen Leistungsfähigkeit unterscheiden, so unterscheiden sie sich doch in architekturspezifischen Charakteristika wie der Effizienz verschiedener Prozesse oder der Kohärenz und Robustheit des Gesamtverhaltens.

Die Gliederung einer Architktur kann sich entweder an der Funktionalität und Kontrolle einzelner Prozesse oder an der Abstraktheit und Komplexität des Wissens orientieren. Die Funktionalität kann von Prozessen, die stark von der Umgebung abhängen, bis hin zu solchen, die mehr von internen Zielsetzungen bestimmt sind, variieren. Somit ergeben sich Schichten für reaktives, situatives und deliberatives Verhalten und schließlich eine kooperative Ebene, auf der globale Ziele mehrerer Agenten bearbeitet werden. Auch die Abstraktionsebenen beginnen bei Repräsentationen, die stark von der Umgebung abhängen, und reichen hin zu Beschreibungen, die durch interne Strukturierungen vorgegeben werden. In der Wahrnehmung ergibt sich ein Informationsfluß ausgehend von sensorischen Daten über Merkmale und Objekte hin zu Situationen, bei Handlungen in der entgegensetzten Richtung von Plänen über einzelne Handlungen und Bewegungen zur Motorik. Diese beiden Arten der Ebenenaufteilung schließen einander nicht aus, im Gegenteil, es ergibt sich eine parallele Gliederung, die in der Gemeinsamkeit des Spektrums von der Abhängigkeit von der Umgebung hin zur internen Strukturie-

rung besteht. Entsprechend sollten bei der Gestaltung einer Ebenenarchitektur beide Aspekte beachtet und aufeinander abgestimmt werden. Da dieses Spektrum jedoch weitgehend kontinuierlich verläuft, stellt sich die Frage, ob es sinnvoller ist, die Aufteilung in verschiedene Ebenen explizit durchzuführen oder implizit zu belassen, kurz, ob eine hierarchisch geschichtete oder eine zentral organisierte Architektur mehr Vorteile bietet.

Eine Ebenenarchitektur besitzt eine explizitere Kontrollstruktur, so daß ein größerer Teil der Kontrolle durch die Architektur und weniger durch spezielle Kontrollinformation vorgegeben ist. Dadurch wird die Prozeßkontrolle übersichtlicher, kann aber auch an Flexibilität einbüßen. Eine explizite Einteilung in Ebenen stellt eine Modularisierung dar, wodurch eine Spezialisierung auf bestimmte Aufgaben und Funktionen und somit eine entsprechend effizientere Verarbeitung möglich wird. Gegenüber anderen modularen Architekturen erlauben Ebenenarchitekturen eine bessere Strukturierung und Kontrolle. Andererseits beeinträchtigt eine Modularisierung die Einheitlichkeit der Architektur und dadurch eventuell die Effizienz und Kohärenz des Verhaltens. Zudem bedeutet Modularisierung und hohe Strukturierung immer auch die Festlegung auf bestimmte Funktionen. Dem gegenüber ermöglicht eine zentrale Architektur mehr Offenheit und Allgemeinheit und vermeidet architekturbedingte Einschränkungen. Insgesamt bieten Ebenenarchitekturen jedoch mehr Möglichkeiten einerseits effiziente Spezialisierungen und andererseits eine allgemeine Funktionalität miteinander zu kombinieren. Durch Modularisierung wird außerdem die Robustheit des Verhaltens erhöht, sowohl dank der Spezialisierung, als auch wegen der weitgehenden Unabhängigkeit der Ebenen voneinander. Dadurch entsteht aber auch das Problem der Kohärenz des Verhaltens, da die einzelnen Ebenen miteinander koordiniert werden müssen, während in einer zentralen Architektur alle Entscheidungen zentral getroffen werden. Eine Ebenenarchitektur ist also dank der Modularisierung und Spezialisierung insgesamt von Vorteil, sofern die Einheitlichkeit und Kohärenz der Architektur gewahrt bleibt und zumindest einige Ebenen genügend Allgemeinheit und Flexibilität besitzen.

Durch die kompetenzbasierte Ebenenauswahl und die Ausführung durch Verpflichtung der nächsttieferen Ebene einerseits sowie die zentrale Wissensbasis und die Repräsentation der procedure PoB als Operatoren zur Planung andererseits bleibt in INTERRAP ein hohes Maß an Einheitlichkeit und Kohärenz gewahrt, zumal auch bei einer Architektur mit zentraler Kontrolle wie SOAR Kohärenz keineswegs garantiert ist. Ein flexibles Verhalten und allgemeine Intelligenz können durch die Planungsebenen erreicht werden.

Die Aufteilung in Ebenen geschieht in INTERRAP vor allem nach funktionalen Gesichtspunkten. Die BBL ist für reaktives und situatives, die LPL für deliberatives und die CPL für kooperatives Verhalten zuständig. Eine Einteilung in Abstraktionsstufen ist in INTERRAP bislang jedoch nur implizit enthalten, insofern die Planung auf einem generell abstrakteren Niveau stattfindet als die Ausführung und die Reaktionen. Ein Beispiel für eine vor allem an Abstraktionsstufen orientierte Architektur gibt Radermacher (1996). Diese Architektur besteht

aus vier Ebenen, die jeweils in Sensorik und Aktorik unterteilt sind. Von unten nach oben wird die Sensorik beginnend bei sensorischen Daten über Merkmalsextraktion und symbolischer Begriffsbildung hin zu formalen Theorien zunehmend abstrakter, während die Aktorik von oben nach unten ausgehend von theoriegestütztem Problemlösen über symbolisches Planen hin zu Bewegungssequenzen und Motorik zunehmend konkreter wird. Im Vergleich dazu fehlen INTERRAP bislang genauere Spezifikationen auf der subsymbolischen Ebene unterhalb der BBL sowie Mechanismen zur zunehmenden Abstraktion von Wissen. Eine Unterteilung der Weltschnittstelle in Sensorik/Motorik einerseits und subsymbolische Verarbeitungen andererseits könnte dabei sinnvoll sein. Auch die Ansiedlung zumindest einfacher Reaktionen auf einer subsymbolischen Ebene und damit die Trennung von reaktiver und situativer Ebene ist zu erwägen, da die Gewinnung symbolischer Repräsentationen recht zeitaufwendig ist.

Bei der bislang recht willkürlichen Trennung von deliberativer und kooperativer Planung in INTERRAP besteht die Gefahr, daß die Planung auf lokaler Ebene scheitert, sobald nur minimale Interaktionen mit anderen Agenten notwendig oder sinnvoll sind und auf der kooperativen Ebene ein Plan völlig neu erstellt werden muß. Die Trennung von lokaler deliberativer und globaler kooperativer Planung ist nur dann sinnvoll, wenn die lokale Planung nicht prinzipiell von Wissen über andere Agenten ausgeschlossen ist und wenn die kooperative Planung auf einer wirklich höheren Abstraktionsstufe stattfindet, was bislang nicht gewährleistet ist. Entsprechend sollte nicht nur die Interaktion, sondern auch die Kommunikation mit anderen Agenten nicht auf die kooperative Ebene beschränkt bleiben. Andernfalls ist die lokale Planung nichts als ein Spezialfall der kooperativen, bei der nur die Möglichkeit der Interaktion mit anderen Agenten und somit willkürlich ein bestimmter Typ von Handlungsalternativen vernachlässigt wird. In diesem Fall wäre die lokale und kooperative Planung durch eine einzige gemeinsame Ebene sinnvoller zu realisieren, schon allein um Neuplanungen auf dem gleichen Abstraktionsniveau zu vermeiden. Die Unterscheidung zwischen lokaler und kooperativer Planung sollte deshalb weniger anhand des verwendeten Wissens, sondern anhand der Unterscheidung zwischen lokalen und globalen Zielen sowie einer höheren Abstraktionsstufe vorgenommen werden.

Insgesamt bietet INTERRAP als Ebenenarchitektur viele Vorzüge, eine detailliertere Erweiterung der Ebenenstruktur nach unten sowie eine klarere Abgrenzung der Funktionalität von lokaler und kooperativer Ebene erscheinen jedoch notwendig.

5.2. Vergleichende Beurteilung von INTERRAP

Ein Vergleich integrierender Architekturen stützt sich auf deren Struktur und Funktionalität. Die Struktur von INTERRAP ist vergleichsweise komplex, da die Architektur auf drei Arten gegliedert ist: zum einen durch die Aufteilung in Weltschnittstelle, Wissensbasis und Kontrolleinheit, zum zweiten durch die Unterteilung von Wissensbasis und Kontrolleinheit in drei Ebenen für reaktives, deliberatives und kooperatives Wissen und Verhalten sowie zum dritten durch die Gliederung jeder Ebene durch die Funktionen BR, SG und PS. Die Einteilung in Ebenen und deren einheitlicher struktureller Aufbau sowie die klare Kontrollstruktur zwischen den Ebenen sorgen bei aller Komplexität der Architektur für Übersichtlichkeit und Transparenz. Andererseits mangelt es bislang noch an Detailliertheit bei der Spezifikation der Komponenten der einzelnen Ebenen, so daß ein Vergleich mit den übrigen Architekturen stellenweise schwerfällt, da diese anders als INTERRAP ihren Schwerpunkt weniger bei der Integration, als im Aufbau der Komponenten setzen. Neben dem Aufbau der Architektur und der integrierten Komponenten ist vor allem der Kontrollzyklus strukturbestimmend für die Architektur, da er den genauen Ablauf der kognitiven Informationsverarbeitungsprozesse regelt. Anders als bei den übrigen Architekturen ist die Kontrolle in INTERRAP zweigeteilt, da sie einerseits zwischen den Ebenen, andererseits innerhalb jeder Ebene stattfindet. Dadurch gewinnen die Ebenen an Eigenständigkeit und somit das Gesamtverhalten an Robustheit.

Die Funktionalität von INTERRAP umfaßt reaktives, deliberatives und soziales Verhalten, das jeweils durch eine eigene Ebene realisiert wird. Durch die parallelen Ebenen entsteht eine gewisse Eigenständigkeit der verschiedenen Verhaltensweisen, die eine angemessene und zugleich voneinander unbeeinflußte Behandlung verschiedener Probleme ermöglicht, durch die Kontrollstruktur zwischen den Ebenen bleibt jedoch die Einheitlichkeit und Kohärenz des Gesamtverhaltens gewahrt. Gegenüber ACT und SOAR bietet INTERRAP den Vorteil, daß nicht nur zentrale Aspekte deliberativer Kognition, sondern zusätzlich situationsgerechtes Verhalten und soziale Interaktion mit anderen Agenten direkt in der Architektur berücksichtigt sind.

5.2.1. Vergleichende Beurteilung der Struktur von INTERRAP

Bei den verwendeten Informationsarten gibt es viele Übereinstimmungen zwischen den Architekturen. Alle verwenden für deklaratives Wissen symbolische Repräsentationen aus Eigenschaftslisten, die die Eigenschaften von Objekten als Mengen von Attribut/Wert-Paaren angeben. Auch wenn alle Architekturen dabei mehr oder weniger explizit die Verwendung von Objektklassen unterstützen, besitzt nur INTERRAP mit AKB eine formale Sprache, die eine feste Syntax zur Definition von Objekthierarchien mit Vererbung von Eigenschaften vorgibt.

Die prozeduralen Informationen der Architekturen zerfallen in Produktionen, die Situationen mit Handlungen verbinden, und Pläne aus Operatoren, die Handlungen abstrakt repräsentieren. Produktionenartige Strukturen kontrollieren den Prozeßablauf dynamisch durch Wissen statt statisch durch ein fest vorgegebenes Programm. Sie werden in INTERRAP jedoch anders als in Produktionensystemen nur als isolierte, situationsgesteuerte Aktionen und nicht als sich modular ergänzende kognitive Einzelschritte verwendet. Eine modulare Verwendung prozeduralen Wissens findet in INTERRAP lediglich auf den Planungsebenen bei der deliberativen Planung mit Planoperatoren statt, die PoB der BBL stellen in sich geschlossene Handlungsabläufe dar.

Als einzige Architektur sieht INTERRAP explizit die Repräsentation und Verwendung von Kommunikations- und Kooperationswissen vor. Dazu zählen erstens deklarative Informationen über physische und mentale Zustände anderer Agenten, zweitens kooperative Multiagentenpläne, die Handlungssequenzen einzelner Agenten und deren Synchronisation umfassen, sowie drittens Protokolle und Strategien zum Durchführen von Verhandlungen.

Als kognitive Architekturen benötigen alle Architekturen Information mit der Funktion von Wahrnehmungsdaten, Wissen, Zielen, Plänen und Intentionen. Während meist keine genauere Unterscheidung abgesehen von der zwischen Wissen und Zielen durchgeführt wird, bietet INTERRAP im Rahmen der BDI-Theorie eine klare Kategorisierung mentaler Kategorien. Dies erhöht einerseits die Transparenz der Architektur, andererseits ermöglicht es eine detailliertere Formulierung von Rationalitätsanforderungen als eine reine Wissen/Ziel-Logik (Cohen & Levesque 1990).

Bei der Speicherorganisation trennt INTERRAP anders als SOAR und ACT nicht zwischen einem Arbeitsspeicher für direkt verwendbare Information und einem Langzeitspeicher, in dem die gesamte Information permanent erhalten bleibt. Statt dessen wird in INTERRAP Wahrnehmungswissen lediglich dazu benutzt, ein konsistentes Modell der Welt aufrechtzuerhalten, wodurch Wissen über vergangene Weltzustände verloren geht. Dies ist aber für große Datenbestände nicht sehr praktikabel und übersichtlich. Ein separater Arbeitsspeicher hingegen ermöglicht die Fokussierung auf relevantes Wissen, das durch einen kontextsensitiven Assoziationsmechanismus aus dem Langzeitspeicher abgerufen wird.

Die integrierten Komponenten der Architekturen dienen jeweils bestimmten Informationsverarbeitungsprozessen. Entsprechend besitzen alle Architekturen Komponenten zur Informationsspeicherung, für die Auswahl und Planung von Handlungen sowie weitgehend eigenständige Module für Wahrnehmung und Handlungsausführung. Die Planung von Handlungen geschieht in INTERRAP anders als in ACT oder SOAR nicht zentral, sondern getrennt für reaktives, situatives, deliberatives und kooperatives Verhalten. Dadurch kann für jede dieser Verhaltensweisen der Planungsprozeß besser als bei einer zentralen Planung auf die jeweiligen Anforderungen und Möglichkeiten abgestimmt und optimiert werden. Die BBL ist für reaktive und situative Planung zuständig, die in direkter Abhängigkeit zur Umgebung stehen, wofür die

BBL als einzige Ebene von INTERRAP mit der Weltschnittstelle direkt verbunden ist. Die BBL ist auf eine schnelle und situationsabhängige Planung hin ausgelegt. Reaktionen werden durch reactor PoB direkt von der Wahrnehmung ausgelöst und sofort in Handlungen umgesetzt, sofern keine wichtigeren Handlungen mit höherer Priorität vorliegen. Die procedure PoB sind für eine situationsgerechte Umsetzung von Intentionen zuständig. Die LPL übernimmt deliberative Planungen von lokalen Handlungssequenzen, die CPL die Planung von und Verhandlungen über Multiagentenpläne. Die Art der deliberativen Planung auf diesen beiden Ebenen geschieht wie in den übrigen Architekturen durch Konstruktion von Plänen durch sukzessive Operatoranwendung auf Planzustände oder durch den Abruf fertiger Pläne aus einer Planbibliothek. Während in SOAR, ACT und PRODIGY dieser Planungsprozeß das Kernstück der Architekturen ist und entsprechend detailliert ausgeführt ist, fehlt in INTERRAP bislang ein fester Planungsalgorithmus. Dadurch ist auch die Entscheidungsfindung bei der Auswahl von Handlungen – eigentlich der zentrale Punkt jeder kognitiven Architektur – in INTERRAP unterspezifiziert. Dafür sind die Planungsebenen von INTERRAP nicht nur auf einen einzelnen Planungsvorgang hin ausgelegt, sondern auch auf die Koordinierung mehrerer Pläne, die parallel erstellt werden können, durch einen Scheduler. Sind die Planoperatoren der anderen Architekturen rein abstrakt, so sind sie in INTERRAP abstrakte Repräsentationen der procedure PoB, so daß für eine situationsgerechte Ausführung einzelner Planschritte gesorgt ist.

Der Kontrollzyklus aller Architekturen besteht in der Abfolge der Aufnahme von Wissens und dessen Analyse auf bekannte Situationen, dem Auffinden von der Situation und den Zielen angemessenen Handlungsmöglichkeiten und der Auswahl aus diesen sowie der Ausführung der ausgewählten Handlungen. Das Verhalten wird somit durch Informationen über die Umgebung, durch das Wissen über die eigenen Fertigkeiten sowie durch interne Zielsetzungen bestimmt und kann somit als kognitiv bezeichnet werden. Während in den übrigen Architekturen die Kontrolle zentral bzw. der Kontrollfluß weitgehend sequentiell geschieht, wird in INTERRAP zunächst anhand der Kompetenz eine Ebene ausgewählt, die mit einer gegebenen Situation angemessen umgehen kann. Parallel zueinander werden die einzelnen Ebenen durch einen eigenen sequentiellen Kontrollfluß bestimmt, sie interagieren untereinander durch den Austausch von Nachrichten. Die Kontrolle über die Interaktion mit der Umgebung liegt bei der BBL, so daß alle Handlungen situationsgerecht ausgeführt werden und zugleich die Möglichkeit schneller Reaktionen erhalten bleibt. Die Eigenständigkeit der Planungsebenen ermöglicht hingegen eine von Änderungen der Umgebung unbeeinflußte Planung.

5.2.2. Vergleichende Beurteilung der Funktionalität von INTERRAP

Bei der Beurteilung der Funktionalität einer Architektur sind weniger die prinziellen Möglichkeiten zu beachten, die grundsätzlich immer durch entsprechende vorgegebene Informationen erreicht werden können, sondern auf die architekturbedingten Vorgänge und auf die Art und

die Qualität, wie eine bestimmte Funktionalität realisiert wird. Deshalb ist eine explizite Berücksichtigung eines breiten Spektrums an Fähigkeiten und eine stark strukturierte Architektur wie INTERRAP von Vorteil gegenüber einer möglichst allgemein gehaltenen und zentral organisierten Architektur wie SOAR, da hier die Architektur die Informationsverarbeitungsprozesse spezifischer und damit meist auch besser unterstützt. Andererseits kann eine zu hohe Strukturierung jedoch auch die Flexibilität im Umgang mit der Information einschränken. Durch die Ebenenstruktur ermöglicht INTERRAP eine optimierte Behandlung von reaktivem, deliberativem und kooperativem Verhalten.

Die BBL von INTERRAP ist speziell auf reaktives, situationsabhängiges Verhalten hin ausgerichtet. Eine hohe Reaktivität wird durch die Einfachheit der verwendeten Datenstrukturen und des Kontrollzyklus der BBL erreicht, so daß schnell und effizient reagiert werden kann. Zentrale Architekturen wie SOAR verwenden für Reaktivität hingegen die gleichen Mechanismen wie für deliberatives Verhalten, so daß keine spezielle Anpassung der Architektur auf die spezifischen Anforderungen an Reaktivität möglich ist.

Die situative Ausführung von Intentionen durch procedure PoB ist insofern unflexibel, als procedure PoB fest vorprogrammierte und in sich geschlossene Datenstrukturen sind, die nur für eine bestimmte Menge bei der Gestaltung berücksichtigter Situationen und Probleme angemessen ihr Ziel erreichen können. Die situative Planung durch HAP in TOK zeigt eine interessante Alternative für eine zugleich effiziente und situationgesteuerte Planung, die dennoch dank modularer Teilpläne hinreichend flexibel bleibt. Durch die verschiedenen Überwachungsfunktionen zur Kontrolle der Ausführung besitzt INTERRAP gegenüber ACT und SOAR eine flexiblere Handlungskontrolle, da während der Ausführung eines procedure PoB relevante Situationen, Probleme und Ausnahmefälle explizit überwacht und gegebenenfalls behandelt werden können. Zudem ermöglicht die Trennung von abstrakter Planung und situativer Ausführung in INTERRAP ein angemessenes Verhalten in dynamischen Umgebungen, insofern deren mögliche Veränderungen nicht bei der Planung, sondern erst bei der Ausführung berücksichtigt werden brauchen. Durch diese Trennung und durch die hohe Reaktivität ist INTERRAP besonders für die Gestaltung autonomer Agenten für dynamische Umgebungen geeignet und erreicht eine Robustheit des Gesamtverhaltens, die SOAR oder ACT höchstens durch sorgfältige Programmierung oder durch die Realisierung ähnlicher Mechanismen erreichen können.

Das deliberative Verhalten besteht in INTERRAP wie in den übrigen Architekturen vor allem in zielgerichteter Problemraumsuche. Allerdings fehlt bislang eine detailliertere Spezifizierung der damit verbundenen Prozesse.

Ein weiterer Vorteil von INTERRAP gegenüber den zentralen Architekturen SOAR und ACT ist die explizite Berücksichtigung von sozialem Verhalten wie Kooperation und Kommunikation. Zwar können diese auch einfach als eine spezielle Art von Information aufgefaßt werden, die eine flexible und umfassende Architektur selbstverständlich verarbeiten kann, gerade

Kooperation und Kommunikation setzen jedoch sehr spezielles Wissen voraus. Zudem vereinfacht eine explizite Berücksichtigung auch die Gestaltung von Agenten für bestimmte Aufgabenbereiche. Andererseits sind die Kommunikationsmechanismen von INTERRAP auf Verhandlungen mit gleichartigen Agenten beschränkt. Auch fehlen soziale Aspekte wie die Bewertung von Verhaltensweisen oder individuelle Einstellungen gegenüber anderen Agenten wie sie die emotionale Komponente EM von TOK bietet.

Adaptives Verhalten wird von INTERRAP bislang überhaupt nicht unterstützt. Die einzige vorkommende Form von Lernen ist die Aktualisierung des Weltmodells und des sozialen Modells durch Wahrnehmung und Kommunikation. Es gibt jedoch weder integrierte Mechanismen zum Lernen oder Verändern prozeduraler Fertigkeiten, noch zur Verbesserung der Entscheidungsfindung.

5.2.3. Schluß

Während sich die Architekturen ACT, PRODIGY und SOAR auf zentrale Aspekte der Kognition konzentrieren, versuchen TOK und INTERRAP das gesamte kognitive Spektrum zwischen Wahrnehmung und Handlung abzudecken. Entsprechend liegen auch die Stärken von ACT, PRODIGY und SOAR vor allem bei zentralen Aspekten wie einer rationalen Entscheidungsfindung, assoziativer Speicherorganisation oder Lernen, während die Stärken von TOK und INTERRAP in der Interaktion mit der Umgebung bestehen, insbesondere durch eine hohe Reaktivität, durch eine situative Ausführung von Handlungen bei abstrakter Planung sowie dank der Berücksichtigung der Interaktion mit anderen Agenten.

Insgesamt ergeben sich die meisten Vorzüge von INTERRAP aus der Organisation in drei parallele Verhaltensebenen. Sie ermöglichen ein situationsgerechtes Verhalten in dynamischen Umgebungen, rationales Verhalten bei komplexen Problemen und die Kooperation mit anderen Agenten. Dabei ist jede Ebene speziell auf ihre Funktion hin ausgerichtet. Insbesondere die effiziente Reaktivität und die Berücksichtigung von Kooperation zeichnen INTERRAP aus.

Hingegen bestehen die meisten Schwächen von INTERRAP weniger in strukturellen Mängeln der Architektur, als in behebbaren Lücken und Unterspezifiziertheiten, die aus dem Entwicklungsstadium von INTERRAP heraus bedingt sind und an denen teilweise bereits gearbeitet wird. Dazu zählt vor allem die mangelnde Detailliertheit zentraler Prozesse wie Planen, Scheduling und Entscheidungsfindung. Eine Erweiterung um Lernen wäre innerhalb des Systems relativ einfach zu realisieren. Eine effizientere Speicherorganisation macht jedoch eine gewisse Umstrukturierung der Architektur beispielsweise durch einen separaten Arbeitsspeicher notwendig.

LITERATURVERZEICHNIS

[Anderson 1983] John R. Anderson: *The Architecture of Cognition*; Cambridge, MA: Harvard University Press, 1983

[Anderson 1988] John R. Anderson: *Kognitive Psychologie*; Heidelberg: Spektrum der Wissenschaft Verlagsgesellschaft, 1988

[Anderson 1990] John R. Anderson: *The Adaptive Character of Thought*; Hillsdale, NJ: Lawrence Erlbaum Associates, 1990

[Anderson 1993] John R. Anderson: *Rules of the Mind*; Hillsdale, NJ: Lawrence Erlbaum Associates, 1993

[Anderson & Bower 1973] John R. Anderson und Gordon Bower: *Human Associative Memory*; Washington, DC: Winston & Sons, 1973

[Barsalou 1992] Lawrence W. Barsalou: *Cognitive Psychology: an Overview for Cognitive Scientists*; Hillsdale, NJ: Lawrence Erlbaum Associates, 1992

[Barwise & Etchemendy 1989] Jon Barwise und John Etchemendy: *Model-Theoretic Semantics*; in: Posner 1989

[Bates et al. 1992] Joseph Bates, A. Bryan Loyall und W. Scott Reilly: *An Architecture for Action, Emotion, and Social Behavior*; Pittsburgh, PA: Technical Report CMU-CS-92-144, Carnegie Mellon University, Mai 1992

[Blythe & Reilly 1993] Jim Blythe und W. Scott Reilly: *Integrating Reactive and Deliberative Planning for Agents*; Pittsburgh, PA: Technical Report CMU-CS-93-155, Carnegie Mellon University, Mai 1993

[Bratman et al. 1987] M. E. Bratman, D. J. Israel und M. E. Pollack: *Toward an Architecture for Resource-Bounded Agents*; Stanford: Technical Report CSLI-87-104, Stanford University, August 1987

[Brooks 1986] Rodney A. Brooks: *A Robust Layered Control System for a Mobile Robot*; in: IEEE Journal of Robotics and Automation RA-2 (1), S. 14-23, April 1986

[Brooks 1991] Rodney A. Brooks: *Intelligence Without Representation*; in: Artificial Intelligence 47, S. 139-159, 1991

[Carbonell et al. 1991] Jaime Carbonell, Oren Etzioni, Yolanda Gil, Robert Joseph, Craig Knoblock, Steve Minton und Manuela Veloso: *PRODIGY: An Integrated Architecture for Planning and Learning*; in: Laird 1991, S. 51-55

[Carbonell et al. 1992] Jaime Carbonell, Jim Blythe, Oren Etzioni, Yolanda Gil, Robert Joseph, Dan Kahn, Craig Knoblock, Steve Minton, Alicia Pérez, Scott Reilly, Manuela Veloso und Xuemei Wang: *PRODIGY 4.0: The Manual and Tutorial*; Pittsburgh, PA: Technical Report CMU-CS-92-150, Carnegie Mellon University, Juni 1992

[Cohen & Levesque 1990] Philip R. Cohen und Hector J. Levesque: *Intention Is Choice with Commitment*; Artificial Intelligence, 42(3), 1990

[Cummins 1994] Robert Cummins: *Interpretational Semantics*; in: Stich & Warfield 1994

[Dennett 1971] Daniel C. Dennett: *Intentional Systems*; in: Haugeland 1981

[Dennett 1978] Daniel C. Dennett: *Brainstorms*; Montgomery, VT: Bradford Books, 1978

[Fischer et al. 1995] Klaus Fischer, Jörg P. Müller und Markus Pischel: *Unifying Control in a Layered Agent Architecture*; Saarbrücken: Technical Memo DFKI-TM-94-05, Januar 1995

[Fodor 1975] Jerry A. Fodor: *The Language of Thought*; New York: Crowell, 1975

[Gardner 1989] Howard Gardner: *Dem Denken auf der Spur*; Stuttgart: Klett-Cotta, 1989

[Görz 1995] Günther Görz (Hrsg.): *Einführung in die künstliche Intelligenz*; Bonn, Paris, Reading MA [u.a.]: Addison-Wesley, 2. Auflage 1995

[Haugeland 1978] John Haugeland: *The Nature and Plausibility of Cognitivism*; in: Haugeland 1981

[Haugeland 1981] John Haugeland (ed): *Mind Design*; Cambridge CA: MIT Press, 1981

[Haugeland 1981a] John Haugeland: *Semantic Engines*; in: Haugeland 1981

[Laird et al. 1987] John E. Laird, Allen Newell und Paul S. Rosenbloom: *SOAR: Architecture for General Intelligence*; in: Artificial Intelligence 33, S. 2-63, 1987

[Laird 1991] John E. Laird: *Special Section on Integrated Cognitive Architectures*; SIGART Bulletin 2 (4), S. 12-135, 1991

[Lakoff 1987] George Lakoff: *Woman, Fire, and Dangerous Things*; Chicago: The University of Chicago Press, 1987

[Loyall & Bates 1991] A. Bryan Loyall und Joseph Bates: *HAP: A Reactive, Adaptive Architecture for Agents*; Pittsburgh, PA: Technical Report CMU-CS-91-147, Carnegie Mellon University, Juni 1991

[Loyall & Bates 1996] A. Bryan Loyall und Joseph Bates: *Personality-Rich Believable Agents That Use Language*; Paper submitted to AAAI-96

[Michon & Akyürek 1992] John A. Michon und Aladin Akyürek (eds): *SOAR: A Cognitive Architecture in Perspective*; Dordrecht: Kluwer Academic Publishers, 1992

[Minsky 1986] Marvin Minsky: *The Society of Mind*; New York: Simon and Schuster, 1986

[Müller 1995] Jörg P. Müller: *Analyzing the Behavior of Interacting Agents – an Architectural Perspective*; in: International Journal of Intelligent and Cooperative Information Systems Vol. 3, No. 1 (1994)

[Müller 1996] Jörg P. Müller: *An Architecture for Dynamically Interacting Agents*; Saarbrücken: Dissertation, Draft Version, 1996

[Müller & Pischel 1993] Jörg P. Müller und Markus Pischel: *The Agent Architecture INTERRAP: Concept and Application*; Saarbrücken: Research Report DFKI-RR-93-26, Juni 1993

[Müller et al. 1995] Jörg P. Müller, Markus Pischel und Michael Thiel: *Modeling Reactive Behaviour in Vertically Layered Agent Architectures*; in: M. J. Wooldridge und N. R. Jennings (eds), Intelligent Agents – Theories, Architectures, and Languages, volume 890: Lecture Notes in AI; Springer, Januar 1995

[Neisser 1974] Ulric Neisser: *Kognitive Psychologie*; Stuttgart: Ernst Klett Verlag, 1974

[Neisser 1979] Ulric Neisser: *Kognition und Wirklichkeit*; Stuttgart: Klett-Cotta, 1979

[Newell & Simon 1976] Allen Newell und Herbert A. Simon: *Computer Science as Empirical Inquiry: Symbols and Search*; in: Haugeland 1981

[Newell et al. 1989] Allen Newell, Paul S. Rosenbloom und John E. Laird: *Symbolic Architectures for Cognition*; in: Posner 1989

[Newell 1990] Allen Newell: *Unified Theories of Cognition*; Cambridge, MA: Harvard University Press, 1990

[Newell 1992] Allen Newell: *Précise of Unified theories of cognition*; in: Behavioral and Brain Sciences 15:3

[Newell 1992a] Allen Newell: *Unified Theories of Cognition and the Role of SOAR*; in: Michon & Akyürek 1992

[Posner 1989] Michael I. Posner (ed): *Foundations of Cognitive Science*; Cambridge, MA: MIT Press, 1989

[Putnam 1981] Hilary Putnam: *Reason, Truth, and History*; Cambridge: Cambridge University Press, 1981

[Radermacher 1996] F. J. Radermacher: *Cognition in Systems*; in: Cybernetics and Systems: An International Journal, 27, S. 1-41, 1996

[Rao & Georgeff 1991] A. S. Rao und M. P. Georgeff: *Modeling Agents within a BDI-Architecture*; in: Proc. of the 2nd International Conference on Principles of Knowledge Representation and Reasoning, S. 473-484

[Rosenbloom et al. 1991] Paul S. Rosenbloom, John E. Laird, Allen Newell und Robert McCarl: *A Preliminary Analysis of the SOAR Architecture as a Basis for General Intelligence*; Artificial Intelligence 47, S. 289-325, 1991

[Rumelhart 1989] David E. Rumelhart: *The Architecture of Mind: A Connectionist Approach*; in: Posner 1989

[Simon & Kaplan 1989] Herbert A. Simon und Craig A. Caplan: *Foundations of Cognitive Science*; in: Posner 1989

[Stich & Warfield 1994] Stephen P. Stich und Ted A. Warfield (eds); *Mental Representation*; Cambridge, MA: Blackwell, 1994

[Veloso et al. 1995] Manuela Veloso, Jaime Carbonell, Alicia Pérez, Daniel Borrajo, Eugene Fink und Jim Blythe: *Integrating Planning and Learning: The PRODIGY Architecture*; in: Journal of Experimental and Theoretical Artificial Intelligence, 7 (1), 1995

[Weiser 1995] Thomas Weiser: *AKB: Assertional Knowledge Base*; interner Bericht

[Wessells 1984] Michael G. Wessells: *Kognitive Psychologie*; New York: Harper & Row, 1984

ANHANG: TABELLEN

Anforderung	Funktionalität	Charakteristika
Verwendbarkeit in realen Situationen	flexibles Verhalten in detaillierten, reichhaltigen, komplexen und dynamischen Umgebungen in Echtzeit	detaillierte Wahrnehmung, große Menge an Wissen, flexibles Bewegungssystem
Reaktives Verhalten	Reaktion auf dringende und unvorhergesehene Ereignisse	schnelle, der Situation angemessene Reaktion
Deliberatives Verhalten	Erreichen von vorgegebenen Aufgaben und eigenen Zielen	Ziele, zweckgerichtetes Handeln, Entscheidungen, Planen
Adaptives Verhalten	Anpassung an Umgebung und deren Veränderungen	angepaßte Konstruktion, Erwerb neuen Wissens, Lernen aus Beispielen, Beobachtung, Erfahrung und Fehlern
Sozialverhalten	Interaktion mit anderen kognitiven Systemen	Kommunikation, Kooperation, Koordinierung von Handlungen

Tabelle 5: Anforderungen an das kognitive Spektrum kognitiver Architekturen

Anforderung	Funktionalität	Charakteristika
Rationalität	konsistentes, effektives und möglichst optimales Verhalten	logische Schlüsse, effizientes, zweckgerichtetes Planen
Robustheit	geringe Fehleranfälligkeit	Erkennen von Fehlern und Gefahren, schnelle Reaktionen, Umgang mit unvollständigem und unsicherem Wissen
Flexibilität	neuartige Lösungen für bekannte Aufgaben	verschiedene Arten von Repräsentationen und Strategien
Vielseitigkeit	Lösungen für neuartige Aufgaben	Flexibilität, Kreativität, Lernfähigkeit
Universalität	breites Spektrum an möglichen Aufgaben und Umgebungen	Berechnungsvollständigkeit

Tabelle 6: Anforderungen an das Verhaltensspektrum kognitiver Architekturen

Architektur	INTERRAP	ACT	TOK	PRODIGY	SOAR
deklaratives Wissen	AKB: Objekte und Konzepte, Eigenschaften und Relationen	Chunk: Eigenschaftsliste mit Typ (IS-A)	Attribut/Wert-Paare	PDL: Objekte und Typen, Prädikatenlogik	Eigenschaftslisten aus Attribut/Wert-Paaren
prozedurales Wissen	reactor PoB, procedure PoB, Planoperatoren, Einzelagentenpläne, Multiagentenpläne, Verhandlungsprotokolle	Produktionen: "Bedingung → Aktion", Analogien	Produktionen: "Vorbedingung → Kontext, Plan", Plan: Handlungen, Teilziele	Planoperatoren, Inferenzregeln, Kontrollregeln, Beispielpläne	Planoperatoren, Präferenzen, Produktionen: Kodierung und Dekodierung
Abstraktionsstufen	Begriffstaxonomie, geschichtete Wissens-basis: WM/MM/SM, abstrakte Planung	hierarchische Strukturierung der Chunks	Pläne bestehen aus Teilplänen (Zielen)	Typenhierarchie, Abstraktions-ebenen für Planoperatoren	nur implizit
funktionale Rolle	BDI: Wissen, Ziele, Absichten, Situationen, Optionen, Pläne, Protokolle, Strategien	deklaratives Wissen, prozedurale Fertigkeiten, Ziele, Aktivierung	Wissen, Pläne, Ziele, Emotionen, Standards, Einstellungen	Domänenwissen, Kontrollwissen, Pläne, Ziele	Wissen, Ziele

Tabelle 7: Die Informationsarten der Architekturen im Überblick

Architektur	INTERRAP	ACT	TOK	PRODIGY	SOAR
Langfristiger Speicher	geschichtete Datenbank: Faktenwissen, Planbibliotheken	deklarativer Speicher für Chunks, prozeduraler Speicher für Produktionen	ISM, Planspeicher	zentraler Speicher als Schnittstelle für Module	Produktionenspeicher
Speicher für kurzfristigen Abruf	–	Arbeitsspeicher: aktivierter Teil des deklarativen Speichers	–	–	zentraler temporärer Arbeitsspeicher: Datenbus für Module
Organisation und Strukturierung	Unterteilung in Weltmodell, mentales Modell, soziales Modell	deklarativer Speicher: assoziatives Netz aus Chunks	–	–	–
Speicherzugriff und Assoziation	expliziter Zugriff durch Datenbankdienste, Überwachungsdienste, ebenenabhängiger Zugriff	Basisaktivierung und Aktivierungsausbreitung der Chunks, Produktionenstärke	–	–	assoziativer Zugriff durch Produktionen, Bedingungen beschreiben Kontext

Tabelle 8: Die Informationsspeicher der Architekturen im Überblick

Architektur	INTERRAP	ACT	TOK	PRODIGY	SOAR
Wahrnehmung	Sensoren der Weltschnitt-stelle, Aktualisieren der Wissensbasis	sensorische Information in Arbeitsspeicher	Aktualisieren von ISM und APT	–	Kodierung sensorischer Information in Arbeitsspeicher
Handeln	Aktoren der Weltschnittstelle, Ausführen von Intentionen	motorische Befehle in Aktionsteil von Produktionen	direkte Ausführung bei Auswahl aus APT	–	Dekodierung motorischer Befehle aus Arbeitsspeicher
Deklaratives Lernen	–	Erfahrung: Anpassen von Basisaktivierung und assoziativer Stärke	–	–	Data Chunking
Prozedurales Lernen	–	Übung: Anpassen der Produktionenstärke, Prozeduralisierung, statistische Abstimmung	–	Lernmodule für Domänenwissen und Kontrollwissen	Chunking: neue Produktionen werden aus Auflösungen von Sackgassen abgeleitet
Problemlösen und Planen	Einzelagenten- und Multiagentenplanung, Planbibliotheken	zielgerichtete Anwendung von Produktionen, Aufstellen von Teilzielen, Problemlösen per Analogie	situatives Erweitern des APT, Aufruf von PRODIGY	Rückwärtsverket-tung und simulierte Ausführung von Operatoren	zielgerichtete Suche in Problemraum durch Operatoranwendung und Auflösen von Sackgassen
Logisches Schließen	durch reactor PoB (knowledge modifier)	Deduktion durch Produktionen, Induktion durch Analogien	–	Deduktion durch Inferenzregeln und EBL	Deduktion durch Produktionen, Induktion durch Chunking
Entscheiden	statisch: Auswahl der PoB durch Prioritäten, Bewertungsfunktion für Pläne, Scheduling	dynamisch: Produktionen-auswahl durch aktivierungs-abhängige Instantiierung und Bewertung	statisch: Spezifität, Prioritäten dynamisch: durch Emotionen	dynamisch: Auswahl durch Kontrollregeln, Planbewertung	dynamisch: Auswahl durch Präferenzen

Tabelle 9: Die Informationsverarbeitungsprozesse der Architekturen im Überblick

Architektur	INTERRAP	ACT	TOK	PRODIGY	SOAR
Integrierte Komponenten	Weltschnittstelle, geschichtete Wissensbasis, geschichtete Kontrolleinheit	Produktionensystem, deklarativer Speicher	Wahrnehmung (TOK: Sensoren, ISM), reaktive Planung (HAP), Emotionen (EM), Sprache (GILDA), deliberative Planung (PRODIGY), Lernmodule (PRODIGY)		Problemraumsuche, Arbeitsspeicher, Langzeitspeicher, Chunking, Sensorik, Motorik
Integration der Komponenten	Kommunikation zwischen Ebenen, Speicherzugriff	Produktionensystem operiert auf deklarativem Speicher	Zugriff von HAP und EM auf ISM, Beeinflussung von HAP durch EM, Aufruf von PRODIGY durch HAP		Arbeitsspeicher als Datenbus
Kontrollstruktur	kompetenzgesteuert, situationsgesteuert	aktivierungsgesteuert, kontextgesteuert, zielgesteuert	situationsgesteuert, emotionsgesteuert	zielgesteuert, wissensgesteuert	problemgesteuert, kontextgesteuert, wissensgesteuert
Kontrollfluß	bottom-up: Planungsanforderung top-down: Verpflichtung zur Ausführung	Aktivierungsausbreitung	Kontrolle durch HAP		zentral
Kontrollzyklus	Wissen aktualisieren (BR) → Situation erkennen (SG) → planen, entscheiden (PS) → ausführen	Produktionenbedingung instantiieren → Produktion bewerten → Produktionenaktion ausführen	ISM aktualisieren → APT aktualisieren → Ziel auswählen → Plan ausführen	Operatorauswahl → Operator-anwendung	Wissenssuche → Entscheidung → Ausführung

Tabelle 10: Die Struktur der Architekturen im Überblick

Architektur	INTERRAP	ACT	TOK	PRODIGY	SOAR
Intendiertes Spektrum	reaktives, deliberatives und soziales Verhalten	menschliche kognitive Fertigkeiten	autome Agenten, glaubwürdige Charaktere	intelligente Planung durch Lernen	allgemeine Intelligenz
Reaktives Verhalten	BBL: reactor PoB, situative Ausführung der procedure PoB/Pläne	–	HAP: situative Planinterpretation, eigenständige Dämonen		schnelles Problemlösen und Agieren durch reine Wissensuche
Deliberatives Verhalten	zielgerichtete Planung	Produktionenauswahl ist zielabhängig	PRODIGY: zielgesteuerte Problemraumsuche		zielgesteuerte Problemraumsuche
Adaptives Verhalten	–	statistisches Lernen	–	Lernmodule	Chunking
Soziales Verhalten	Kooperation, Kommunikation, Multiagentenplanung	–	Sprache (GLINDA), Emotionen (EM), Moral (EM)	–	–

Tabelle 11: Das kognitive Spektrum der Architekturen im Überblick

Architektur	INTERRAP	ACT	TOK	PRODIGY	SOAR
Rationalität	BDI, Planung, Planbewertung	Kostenminimierung durch Assoziation, Produktionenauswahl durch Bewertung	situative Planung	zielgerichtete, wissensgesteuerte, korrekte Planung, Planbewertung	problemgesteuerte Planung, assoziative Wissenssuche
Robustheit	parallele Ebenen, überwachte Ausführung	–	überwachte Ausführung	abstrakte Planung	–
Flexibilität	situative Planausführung	modulare Produktionen	situative Planerstellung	dynamisches Kontrollwissen	dynamische Kontrolle durch Präferenzen, Lernen durch Chunking
Vielseitigkeit	Kombination von reaktivem, deliberativem und kooperativem Verhalten			Modularität von Domänen- und Kontrollwissen	informationsintensive Architektur

Tabelle 12: Das Verhaltensspektrum der Architekturen im Überblick